

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:

«Веб-орієнтована система пошуку нерухомості»

Виконав студент групи ПЗ-21-2
ПАШКЕВИЧ Віталій Валерійович

Керівник роботи:
ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2025

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

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПАШКЕВИЧ Віталій Валерійович**

Керівник роботи: **ЛОБАНЧИКОВА Надія Миколаївна**

Тема роботи: **«Веб-орієнтована система пошуку нерухомості»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: Об'єктом дослідження є процеси розробки та проектування веб-орієнтованих систем. Предметом дослідження є моделі, методи та засоби розробки веб-орієнтованої системи пошуку нерухомості.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Лобанчикова Н.М.	17.02.2025р.	17.02.2025р.
2	Лобанчикова Н.М.	18.03.2025р.	18.03.2025р.
3	Лобанчикова Н.М.	14.04.2025р.	14.04.2025р.

РЕФЕРАТ

Випускна робота бакалавра складається з програмного продукту «Веб-орієнтована система пошуку нерухомості» та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить текстову частину, викладену на 73 сторінках друкованого тексту та має 3 додатки. Список використаних джерел містить 20 найменувань. У роботі наведено 24 рисунки та 12 таблиць.

Метою цієї роботи є створення веб-орієнтованої системи для пошуку нерухомості для підвищення зручності та оперативності отримання актуальної інформації про доступні об'єкти купівлі, продажу та оренди. Основне завдання полягає в розробці функціональної платформи, що дозволить здійснювати пошук об'єктів нерухомості за різними критеріями, забезпечуючи високу швидкість обробки запитів та інтуїтивно зрозумілий інтерфейс.

У межах проекту розроблено веб-орієнтовану систему для пошуку нерухомості, яка враховує сучасні тенденції та потреби як кінцевих користувачів, так і ринку. Для вирішення поставлених завдань використано сучасні технології, зокрема Docker, Node.js, MySQL, React, MapTiler. Веб-орієнтована система має чітко продуману архітектуру, а також зрозумілий користувацький інтерфейс з інтерактивною картою нерухомості, що дозволяє зручно виконувати пошук, фільтрацію та управління оголошеннями.

КЛЮЧОВІ СЛОВА: ВЕБСИСТЕМА, НЕРУХОМІСТЬ, КАРТА, ОГолошення, база даних.

					ІПЗ.КР.Б-121-25-ІПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Веб-орієнтована система пошуку нерухомості	Літ.	Арк.	Аркуші
Розроб.		Пашкевич В.В.						
Керівник		Лобанчикова Н.М.					3	73
Рецензент		Сфремов Ю.М.				Житомирська політехніка, група ІПЗ-21-2		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of the software product “Web-Based Real Estate Search System” and an accompanying explanatory note. The explanatory note comprises 73 pages of printed text and includes 3 appendices. The list of references contains 20 sources. The work also features 24 illustrations and 12 tables.

The objective of this thesis is to develop a modern web-based system for real estate search that provides users with fast, convenient, and efficient access to up-to-date information on available properties for purchase, sale, and rent. The primary goal is to create a functional platform that enables real estate searches based on various criteria, ensuring high query processing speed and an intuitive user interface.

As part of the project, a web-based real estate search system was developed that takes into account modern trends and the needs of both end users and the market. To solve the defined tasks, modern technologies were used, including Docker, Node.js, MySQL, React, and MapTiler. The web-based system features a well-thought-out architecture and a clear user interface with an interactive real estate map that enables convenient search, filtering, and listing management.

KEYWORDS: WEB SYSTEM, REAL ESTATE, MAP, LISTINGS, DATABASE

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ПОШУКУ НЕРУХОМОСТІ.....	9
1.1. Опис та постановка задачі.....	9
1.2. Аналіз функціональних характеристик існуючих сервісів-аналогів	12
1.3. Вибір і обґрунтування архітектури та інструментальних засобів реалізації програмного продукту	17
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ПОШУКУ НЕРУХОМОСТІ	22
2.1. Визначення варіантів використання системи	22
2.2. Розробка бази даних веб-системи.....	25
2.3. Реалізація веб-орієнтованої системи пошуку нерухомості	31
Висновки до другого розділу	37
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ОРІЄНТОВАНОЮ СИСТЕМОЮ ПОШУКУ НЕРУХОМОСТІ	39
3.1. Порядок встановлення та налаштування системних параметрів	39
3.2. Структура інтерфейсу та порядок взаємодії з веб-орієнтованою системою пошуку нерухомості.....	41
3.3. Тестування роботи розробленої веб-орієнтованої системи.....	50
Висновки до третього розділу.....	53
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ.....	59

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – application programming interface

HTTP – HyperText Transfer Protocol, протокол передачі гіпертекстових документів

HTTPS – HyperText Transfer Protocol Secure, безпечний протокол передачі гіпертексту

NFR – non-functional requirements, нефункціональні вимоги

SPA – single-page application, односторінковий додаток

UX/UI – User Interface / User Experience, користувацький інтерфейс / користувацький досвід

СУБД – система управління базами даних

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми: Ринок нерухомості є важливою складовою економічної системи, що характеризується високою динамікою розвитку та значною конкуренцією. В умовах цифровізації суспільства особливої актуальності набуває впровадження інноваційних веб-орієнтованих інформаційних систем, які забезпечують оперативний доступ до актуальної інформації щодо об'єктів нерухомості.

Традиційні методи пошуку нерухомості, зокрема через друковані засоби масової інформації або агентства, поступово втрачають свою ефективність через обмеженість доступу до інформації та повільність процесів. Натомість веб-орієнтовані системи надають користувачам можливість швидкого і зручного отримання актуальних даних про об'єкти нерухомості, здійснення пошуку за різноманітними критеріями та забезпечують інтерактивну взаємодію між продавцями і покупцями.

Незважаючи на наявність великої кількості онлайн-платформ, існує потреба у розробці систем, що відзначаються підвищеною функціональністю, адаптивністю до потреб різних категорій користувачів та високим рівнем зручності використання. Це зумовлює необхідність створення нових рішень, які б відповідали сучасним вимогам до якості веб-сервісів та сприяли підвищенню ефективності ринку нерухомості.

Таким чином, розробка веб-орієнтованої системи пошуку нерухомості є актуальним завданням, що має важливе значення для підвищення ефективності інформаційних процесів у сфері нерухомості та забезпечення більш якісного обслуговування користувачів.

Метою роботи є розробка веб-орієнтованої системи пошуку нерухомості, для підвищення ефективності, швидкості та зручності доступу користувачів до актуальної інформації щодо об'єктів купівлі, продажу та оренди нерухомості.

Для досягнення поставленої мети необхідним є вирішення наступних задач:

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- провести аналіз функціоналу існуючих сервісів-аналогів;
- здійснити вибір архітектури вебдодатку;
- обґрунтувати вибір інструментальних засобів реалізації програмного забезпечення;
- побудувати моделі веб-орієнтованої системи пошуку нерухомості;
- розробити базу даних веб-орієнтованої системи;
- провести програмну реалізацію та тестування запропонованої системи;
- розробити рекомендації щодо налаштування, особливостей розгортання та експлуатації запропонованої вебсистеми.

Об’єктом дослідження є процеси розробки та проектування веб-орієнтованих систем.

Предметом дослідження є моделі, методи та засоби розробки веб-орієнтованої системи пошуку нерухомості.

Основним завданням є створення функціональної платформи, яка дозволить здійснювати пошук за різноманітними критеріями, забезпечуватиме високу швидкість обробки запитів та інтуїтивно зрозумілий інтерфейс.

Випускна кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел із 20 найменувань (2 стор.), 3-х додатків, 12-ти таблиць та 24 ілюстрацій по тексту. Повний обсяг роботи становить 73 сторінки, з них 58 сторінок основного тексту.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ПОШУКУ НЕРУХОМОСТІ

1.1. Опис та постановка задачі

Ефективна організація інформаційних процесів на ринку нерухомості набуває особливого значення, зважаючи на активний розвиток цифрових технологій та діджиталізації. Та, попри наявність широкого спектра веб-сервісів для пошуку об'єктів нерухомості, більшість існуючих платформ не завжди відповідає сучасним запитам користувачів.

Багато існуючих рішень не враховують специфічних вимог окремих ринків нерухомості або потреб певних категорій користувачів, що створює додаткові труднощі при пошуку необхідних об'єктів. В умовах високої конкуренції та постійного зростання обсягів інформації виникає потреба у створенні більш гнучкої та масштабованої веб-орієнтованої системи, яка здатна ефективно адаптуватися до змінних умов ринку й потреб споживачів.

Відповідно до цього, у процесі розробки системи необхідно вирішити такі основні задачі:

- визначити основні функціональні і нефункціональні вимоги, дослідження потреб цільової аудиторії та аналіз існуючих аналогічних рішень на ринку;
- розробка логічної структури та проєктування архітектури веб-додатку, визначення необхідного стеку технологій;
- реалізація інтуїтивно зрозумілого, адаптивного інтерфейсу для зручної взаємодії із системою;
- реалізація роботи з базою даних та обробки запитів користувачів;
- тестування усіх компонентів, перевірка оптимізації швидкості завантаження сторінок, виявлення помилок та їх усунення.

Усе це є важливими етапами для отримання в результаті якісної та надійної

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

веб-орієнтованої системи пошуку нерухомості, яка повністю відповідатиме очікуванням користувачів та сучасним вимогам ринку.

Розроблювана веб-система пошуку житла має реалізувати низку ключових функціональних можливостей, спрямованих на забезпечення повноцінної взаємодії користувачів із платформою. Основними задачами, які повинна вирішувати система, є:

- організація функціоналу створення оголошень: забезпечення можливості для зареєстрованих користувачів створювати оголошення про оренду або продаж нерухомості через спеціалізовану сторінку. При створенні оголошення користувач заповнює форму із введенням основних характеристик об'єкта (тип нерухомості, площа, ціна, номер квартири, опис, фотографії тощо) та вказує його розташування за допомогою інтерактивної карти;

- підтримка різних типів оголошень: надання можливості класифікації оголошень за типами (наприклад, оренда, продаж, житлова чи комерційна нерухомість) з метою полегшення подальшого пошуку та фільтрації інформації;

- додавання переваг до оголошення: реалізація механізму, що дозволяє користувачам зазначати додаткові характеристики об'єкта нерухомості (наприклад, наявність паркінгу, ремонту, меблів тощо);

- інтерактивна робота з картою: відображення розташування об'єктів на інтерактивній карті із зазначенням вартості;

- механізм скарг на оголошення: впровадження функціоналу, що дозволяє користувачам надсилати скарги на недостовірні або небажані оголошення для подальшого їхнього розгляду адміністрацією системи;

- адміністративна панель керування: створення окремого інтерфейсу для адміністраторів системи, що дозволяє здійснювати управління користувачами, модерацію оголошень, розгляд скарг, а також роботу із забудовниками та іншими зацікавленими сторонами.

Кожна із перелічених задач спрямована на забезпечення максимальної функціональності системи, підвищення її юзабіліті та створення комфортного

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

середовища для всіх категорій користувачів: як фізичних осіб, так і представників агентств нерухомості. Також для досягнення цих цілей необхідно врахувати нефункціональні вимоги (табл. 1.1). Нефункціональні вимоги (NFR) також відомі як характеристики якості або вимоги до якості послуг. Хоча NFR зазвичай асоціюються з технічними рішеннями, вони зазвичай доповнюють функціональні вимоги, встановлюючи обмеження для цих вимог або описуючи саме такі характеристики якості, якими має володіти готове рішення.

Таблиця 1.1

Нефункціональні вимоги до веб-орієнтованої системи

№	Назва вимоги	Опис
1	Продуктивність	Система має швидко обробляти запити та завантажувати сторінки навіть при великому обсязі даних.
2	Масштабованість	Архітектура повинна підтримувати подальше розширення функціоналу та кількості користувачів.
3	Надійність	Гарантувати цілісність і збереження даних у разі збоїв.
4	Безпека	Захист персональних даних та запобігання несанкціонованому доступу.
5	Адаптивність	Коректне відображення інтерфейсу на різних пристроях і роздільних здатностях екранів.
6	Юзабіліті	Інтуїтивно зрозумілий і зручний інтерфейс для користувачів різного рівня навичок та підготовки.

На основі сформульованих функціональних та нефункціональних вимог можна зробити висновок, що розроблювана веб-орієнтована система пошуку нерухомості повинна не лише надавати базові можливості для створення, перегляду та взаємодії з оголошеннями, але й забезпечувати високі стандарти конкурентоспроможного продукту: зручності використання, безпеки, продуктивності та масштабованості.

1.2. Аналіз функціональних характеристик існуючих сервісів-аналогів

Огляд та порівняння функціональних можливостей існуючих аналогів є невід’ємним етапом при проектуванні будь-якого нового програмного продукту. Вивчення схожих вже існуючих систем на ринку, дає змогу визначити найкращі практики, виявити переваги та недоліки конкурентних рішень, а також зрозуміти, які саме аспекти є найбільш важливими для майбутньої цільової аудиторії. Такий аналіз дозволяє не лише уникнути повторення типових помилок, що вже існують, а й створити новий продукт, який матиме безумовні переваги завдяки врахуванню досвіду, отриманого іншими.

Особливого значення цей етап набуває при створенні веб-орієнтованої системи пошуку нерухомості. Дана предметна область характеризується високим рівнем конкуренції, різноманітністю сервісів та значним попитом на максимально комфортні рішення серед користувачів. Саме тому ретельне порівняння функціональних характеристик сервісів-аналогів є необхідною умовою для створення якісного програмного продукту.

Першим аналогом для детального огляду та аналізу було обрано Lun.ua (рис. 1.1).

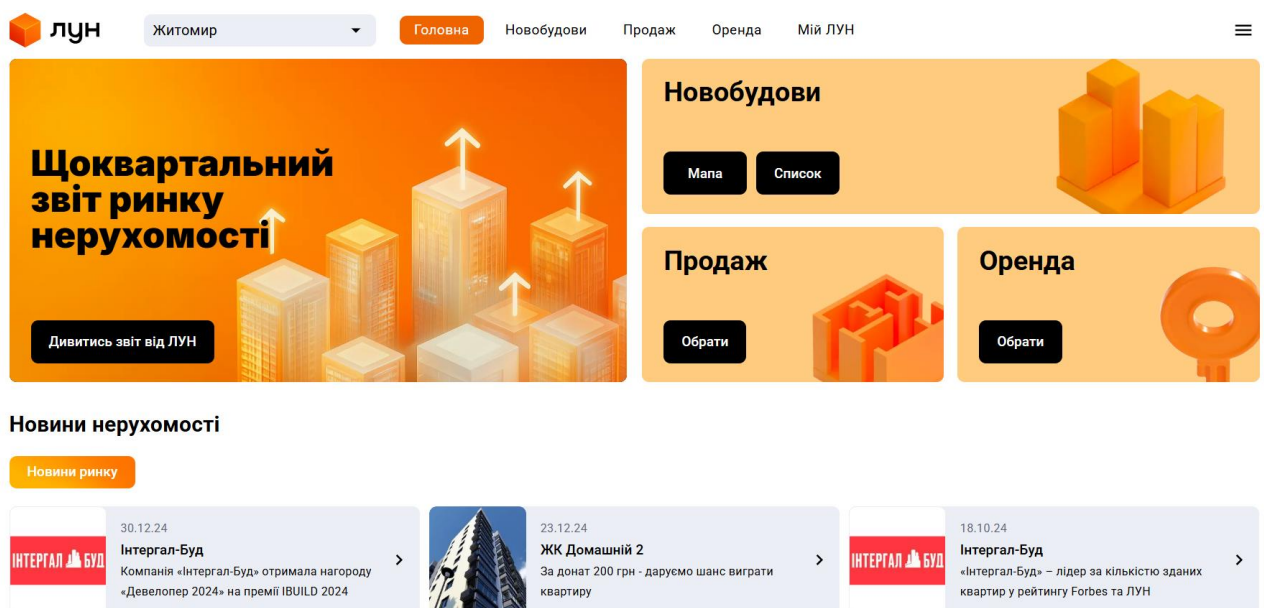


Рис. 1.1. Головна сторінка веб-сервісу ЛУН

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

ЛУН є одним із провідних українських онлайн-сервісів, який спеціалізується на пошуку нерухомості для купівлі та оренди [5]. Платформа дозволяє користувачам зручно переглядати пропозиції житла та комерційних об'єктів, що представлені на інтерактивних картах українських міст, та швидко знаходити необхідні об'єкти відповідно до заданих критеріїв.

Функціонал сервісу включає широкий спектр можливостей для фільтрації та сортування нерухомості за типом об'єкта (квартира, будинок, офіс тощо), бюджетом, площею приміщень, типом та роком будівництва, географічним розташуванням (місто, район), а також тривалістю оренди (довгострокова чи подорожня). Особливістю Lun.ua є наявність спеціального розділу порівняння житлових комплексів первинного ринку, де користувачі можуть отримати інформацію про ціни, забудовників, характеристики житла, документи та стан будівництва.

Окрім функції пошуку та фільтрації, Lun.ua надає користувачам аналітичні можливості завдяки розділу «ЛУН Статистика» (рис. 1.2), який містить регулярно оновлювані дані щодо стану ринку нерухомості у столиці та інших містах України. Завдяки цьому потенційні покупці та продавці можуть оперативно оцінити поточні ціни, проаналізувати тенденції ринку та порівняти свою нерухомість з аналогічними пропозиціями за різними параметрами, такими як ціна, місце розташування чи додаткові переваги об'єктів.

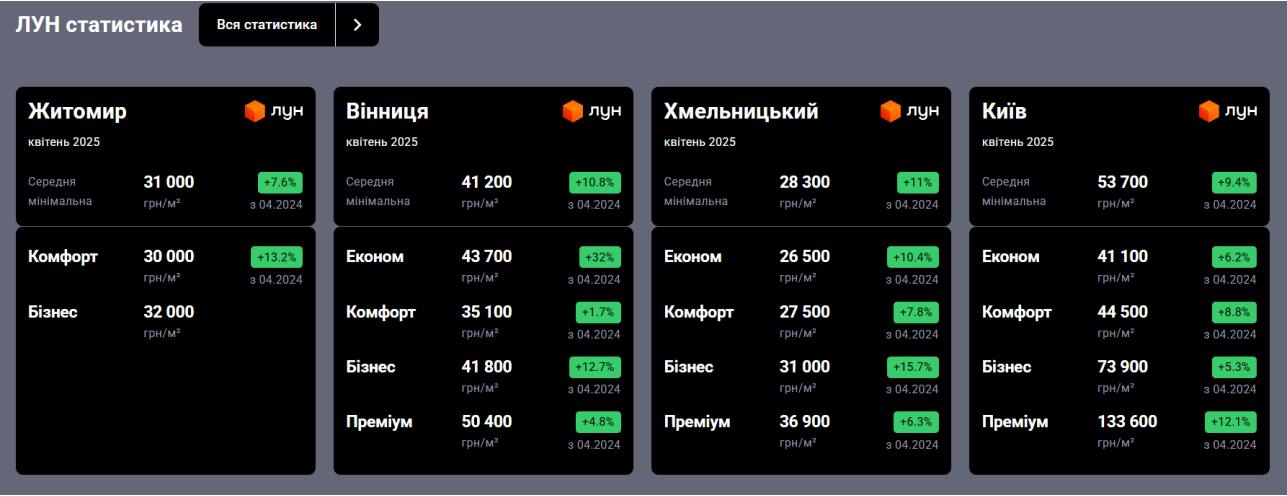


Рис. 1.2. Розділ «ЛУН Статистика»

Наступний програмний продукт, що вже існує та активно використовується на ринку – DIM.ria.com (рис. 1.3) [10]. Це популярний український онлайн-ресурс, що спеціалізується на купівлі, продажу та оренді нерухомості, а також є одним із лідерів серед тематичних майданчиків у результатах видачі Google. Попри велику кількість інформації, оголошень і рекламних матеріалів, сайт має зручний та логічно структурований інтерфейс, який дозволяє легко орієнтуватись користувачам та ефективно здійснювати пошук відповідних пропозицій.

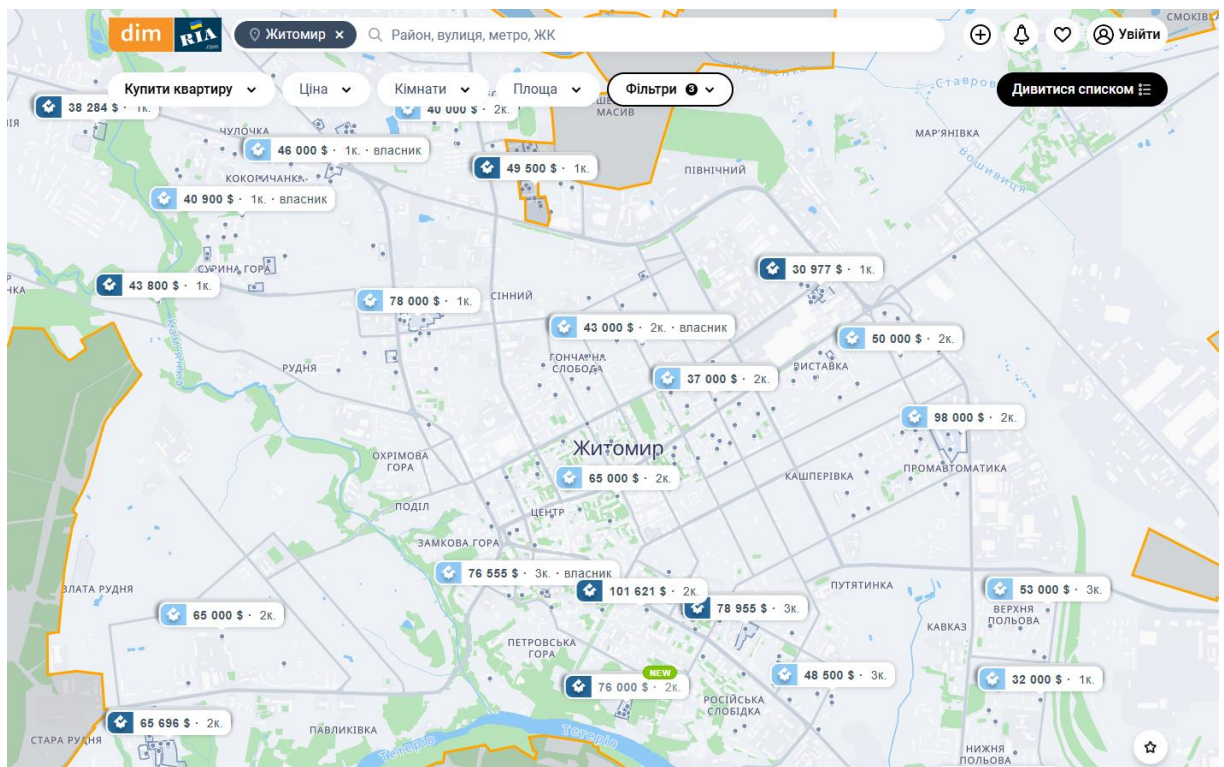


Рис. 1.3. Перегляд нерухомості на мапі в сервісі DIM.ria.com

Головною особливістю DIM.ria.com є акцент на перевірності інформації про нерухомість. Додатково ресурс надає можливість користувачам завантажувати панорамні фотознімки об'єктів, що значно підвищує інформативність оголошень та рівень довіри потенційних покупців.

Функціонал платформи дозволяє швидко і зручно розміщувати власні оголошення, а також знаходити цікаві варіанти житла як на первинному ринку (новобудови), так і вторинному. Завдяки такому підходу DIM.ria.com заслужено

входить до рейтингу найкращих сайтів із продажу нерухомості, надаючи не лише широкий вибір житла, а й максимально перевірену та надійну інформацію для користувачів.

Також не менш популярним сервісом в Україні є OLX (рис 1.4). OLX - нерухомість – один із найбільш відомих та відвідуваних онлайн-сервісів України, що спеціалізується переважно на продажі та купівлі нерухомості вторинного ринку [6]. Платформа щомісяця залучає понад 60 мільйонів користувачів, пропонуючи доступ до більш ніж 200 тисяч активних оголошень, які охоплюють житло, земельні ділянки, гаражі, офіси та інші типи нерухомості.

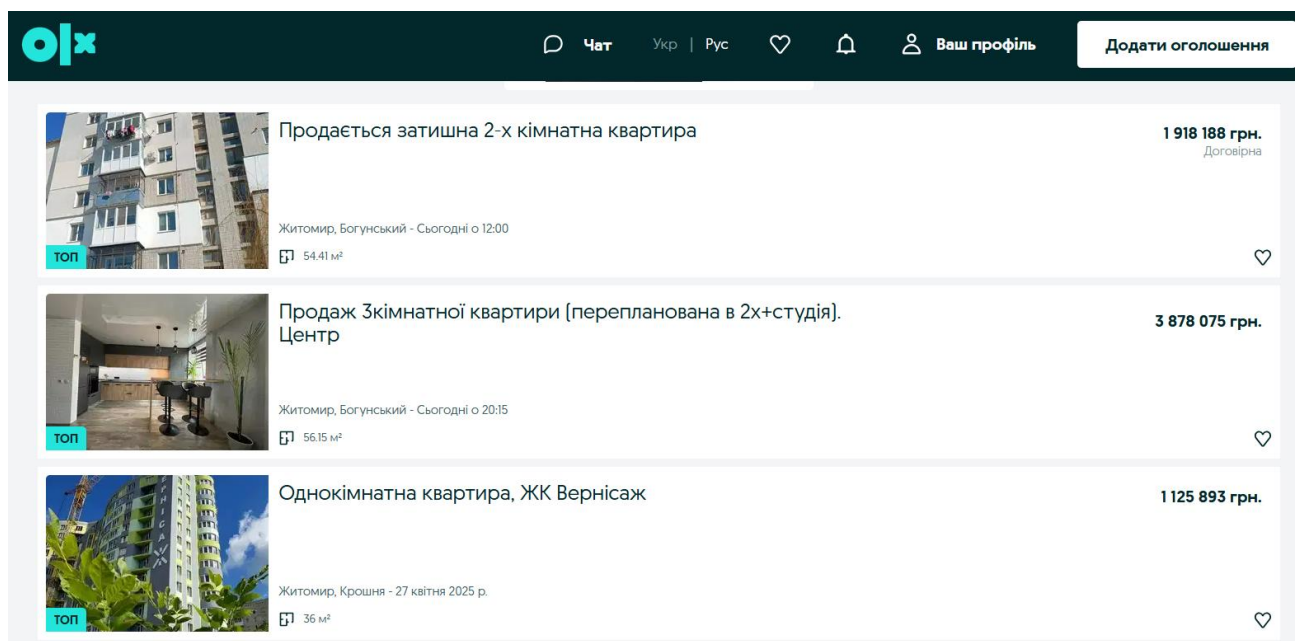


Рис. 1.4 Перегляд відфільтрованих оголошень на сайті OLX-нерухомість

Однією з ключових переваг сервісу OLX є зручність користування та адаптивність до потреб користувачів. Веб-сайт автоматично визначає місцезнаходження користувача і надає актуальні, так звані «гарячі» пропозиції відповідно до регіону пошуку. Інтерфейс ресурсу має базові пошукові фільтри, що дозволяють здійснювати швидкий пошук за областю, типом об'єкта та іншими критеріями, завдяки чому процес пошуку стає максимально комфортним і швидким.

Особливістю OLX є те, що цей ресурс активно використовують власники

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

нерухомості, завдяки чому покупці мають змогу напряму спілкуватися з власниками та уникати посередників, що дозволяє знизити витрати на комісію ріелторів.

Незважаючи на значні переваги та широкі можливості розглянутих сервісів (Lun.ua, DIM.ria.com та OLX-нерухомість), детальний аналіз їхніх особливостей дозволяє виявити певні недоліки, які впливають на ефективність взаємодії користувачів із системою та загальний рівень задоволеності відвідувачів веб-сайтів. Серед найбільш помітних спільних недоліків можна виокремити наступні: перевантаженість інтерфейсів рекламною інформацією (особливо характерно для DIM.ria.com та OLX), що може ускладнювати пошук та сприйняття корисної інформації користувачами, які прагнуть швидко знайти необхідні оголошення; відсутність або недостатній розвиток інтерактивних функцій комунікації між користувачами: сервіси не завжди забезпечують зручні та швидкі канали для прямого спілкування між продавцем і покупцем у режимі реального часу; недостатньо глибока інтеграція картографічних сервісів (крім Lun.ua): у деяких сервісах об'єкти представлені лише умовно на карті без додаткових зручних інструментів, таких як позначення найближчої інфраструктури, транспортних вузлів чи об'єктів соціальної інфраструктури; обмеженість аналітичних інструментів: хоча Lun.ua надає певні статистичні дані, інші сервіси (DIM.ria.com, OLX) приділяють недостатньо уваги збору та аналізу інформації про тенденції ринку, що може бути важливим для тих, хто прагне ухвалювати рішення на основі обґрунтованих аналітичних даних.

Враховуючи ці недоліки та вищезазначені аспекти, розробка власної веб-орієнтованої системи пошуку нерухомості надає можливість створити продукт, що відповідатиме сучасним вимогам.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3. Вибір і обґрунтування архітектури та інструментальних засобів реалізації програмного продукту

Основою архітектури розроблюваної веб-орієнтованої системи пошуку нерухомості є модель клієнт-серверної взаємодії з використанням контейнеризованих веб-сервісів на базі технології Docker. Такий підхід відповідає сучасним тенденціям у проєктуванні складних інформаційних систем, де пріоритетними стають гнучкість, масштабованість, розподіленість і висока доступність компонентів [13].

Контейнеризація за допомогою Docker дозволяє розгортати окремі компоненти системи як незалежні сервіси, кожен з яких працює у власному середовищі, ізольованому від інших. Застосування контейнерної архітектури істотно спрощує процеси розгортання, оновлення та масштабування системи, забезпечуючи відтворюваність середовищ розробки, тестування та продуктивної експлуатації. Однією з ключових переваг використання Docker є можливість швидкого перенесення додатку між різними хостами без необхідності адаптації до особливостей операційної системи або конфігурації середовища, що суттєво підвищує портативність та стійкість рішення [19].

Користувачі взаємодіють із системою через веб-інтерфейс, реалізований за допомогою React. Запити від клієнтської частини спрямовуються до веб-сервера Nginx, який виконує роль зворотного проксі-сервера. Він приймає запити HTTP/HTTPS та маршрутизує їх до відповідного сервісу. Завдяки використанню Nginx значно підвищується загальна пропускна здатність системи та знижується час відповіді на запити користувачів, що є критичним фактором для зручності роботи з веб-орієнтованими платформами.

Запити, що потребують обробки бізнес-логіки (наприклад, пошук оголошень, додавання коментарів, обробка скарг), пересилаються до бекенд-сервісу на Node.js. Бекенд, у свою чергу, звертається до бази даних MySQL для зчитування або запису структурованої інформації. База даних MySQL є центральним сховищем структурованої інформації про користувачів,

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

оголошення, скарги, коментарі та інші сутності системи. Її вибір обумовлений високою надійністю, широкими можливостями оптимізації запитів та підтримкою складних реляційних моделей даних.

У разі потреби обробки або завантаження медіафайлів сервер взаємодіє з об'єктним сховищем Minio. Використання об'єктного сховища дозволяє розвантажити основну базу даних та підвищити загальну продуктивність системи при роботі з медіа-контентом.

Кожен компонент системи працює в окремому Docker-контейнері, що дозволяє досягти високого рівня ізоляції, керованості та масштабованості.

Загалом архітектура системи побудована відповідно до принципів мікросервісної архітектури (Microservices Architecture), де кожен компонент виконує окрему роль і може бути незалежно розгорнутий, масштабований або оновлений. Такий підхід сприяє підвищенню надійності, гнучкості й адаптивності системи до змін навантаження або змін у бізнес-вимогах. Реалізована архітектура забезпечує високу доступність додатку, ефективне управління ресурсами, легкість обслуговування та оновлення.

Вибір інструментальних засобів для реалізації веб-орієнтованої системи пошуку нерухомості базувався на принципах забезпечення високої продуктивності, зручності розробки й подальшої підтримки програмного продукту. Підбір стеку технологій орієнтувався на відповідність сучасним тенденціям у сфері веб-розробки та вимогам до побудови надійних, адаптивних і безпечних систем [16].

У якості середовища розгортання було обрано Docker як інструмент контейнеризації додатку. Застосування контейнерів дозволяє створювати стандартизоване ізольоване середовище для кожного компонента системи, забезпечуючи незалежність їхньої роботи та полегшуючи процеси масштабування й оновлення.

Для обробки HTTP-запитів, розподілу навантаження та забезпечення безперебійного доступу користувачів до ресурсу було обрано веб-сервер Nginx.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

Висока ефективність обробки великої кількості одночасних з'єднань та здатність працювати як зворотний проксі-сервер робить Nginx оптимальним рішенням для забезпечення стабільної роботи системи.

Використання Node.js дає змогу створювати високопродуктивні серверні додатки, здатні обслуговувати велику кількість одночасних запитів із мінімальними затримками.

Як основний засіб зберігання структурованих даних була використана реляційна база даних MySQL. Вибір цієї системи управління базами даних (СУБД) зумовлений її зрілістю, надійністю, високою продуктивністю, підтримкою складних транзакційних операцій та широкими можливостями масштабування. Крім того, MySQL має розвинену екосистему інструментів для резервного копіювання, відновлення та оптимізації продуктивності бази даних.

Для оптимізації процесу розробки, збірки та гарячого перезавантаження модулів було використано інструмент Vite. Завдяки своїй архітектурі на основі ES-модулів Vite забезпечує надзвичайно швидкий початок середовища для розробки, що значно підвищує ефективність розробки та тестування інтерфейсу користувача.

Інтерфейсні компоненти створені із застосуванням бібліотеки Ant Design, яка пропонує великий набір готових до використання елементів з високим рівнем якості дизайну. Це дозволяє прискорити розробку інтерфейсних рішень та забезпечити єдину стилістичну концепцію усього застосунку, дотримуючись сучасних принципів User Interface / User Experience (UX/UI) дизайну.

Організація навігації між сторінками веб-застосунку реалізована за допомогою бібліотеки React Router. Вона забезпечує гнучке й ефективне керування маршрутами у single-page application (SPA-додатку), що сприяє покращенню загального користувацького досвіду завдяки мінімізації перезавантажень сторінок і безперервній роботі інтерфейсу.

Інтеграція картографічних можливостей здійснена за допомогою сервісу Maptiler, який надає можливість працювати з інтерактивними картами,

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

розміщувати мітки об'єктів нерухомості та інтегрувати геолокаційні дані без значних витрат на власну інфраструктуру карт.

Для реалізації асинхронної взаємодії клієнтської частини із сервером використовується бібліотека Axios, яка надає простий і гнучкий application programming interface (API) для обробки HTTP-запитів та відповідей.

Покращення візуальної складової інтерфейсу забезпечено за допомогою бібліотек React Icons для використання іконок та React Slick для реалізації слайдерів, що значно підвищує зручність та естетичну привабливість застосунку.

Таким чином, обраний стек технологій забезпечує розробку сучасного, продуктивного та приємного у користуванні веб-сервісу пошуку нерухомості, який відповідає актуальним вимогам та тенденціям ринку.

Висновки до першого розділу

Розглянуто та описано ключові аспекти розробки веб-орієнтованої системи пошуку нерухомості. На початковому етапі було визначено актуальність створення власного веб-сервісу, яка зумовлена недостатнім рівнем зручності, гнучкості та функціональності існуючих рішень на ринку. Аналіз ринку та конкурентних платформ, таких як Lun.ua, DIM.ria.com та OLX-нерухомість, дозволив виявити як переваги, так і недоліки, які важливо враховувати при створенні власного продукту. Серед типових недоліків конкурентних платформ було зазначено перевантаження інтерфейсів рекламними матеріалами, недостатню персоналізацію контенту та недостатню інтеграцію сучасних картографічних і аналітичних сервісів.

Проведений аналіз конкурентних аналогів дозволив сформулювати конкретні вимоги до системи, як функціональні (розміщення та управління оголошеннями, фільтрація пошукових запитів, можливість коментування та подання скарг, інтеграція інтерактивних карт), так і нефункціональні (висока продуктивність, надійність, масштабованість, безпека, адаптивність інтерфейсу, зручність користування). Обґрунтування необхідності реалізації цих вимог

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

пояснюється потребою в наданні якісних послуг, зручності використання платформи для різних категорій користувачів та забезпеченні конкурентних переваг на сучасному ринку нерухомості.

Також було детально розглянуто й обґрунтовано вибір архітектури системи та стеку технологій, які забезпечують можливість ефективного вирішення поставлених завдань. Основою архітектури обрано контейнеризований підхід із використанням Docker. Вибір технологій для серверної частини (Node.js, MySQL, Minio) та фронтенд-частини (React, Vite, Ant Design, Redux Toolkit, React Router, Tanstack Table, Maptiler, React Icons, React Slick), а також використання веб-сервера Nginx є обґрунтованим та відповідає сучасним вимогам до побудови ефективних веб-систем.

Завдяки обраному технологічному стеку та правильно спроектованій архітектурі планується, що система матиме високу продуктивність та стабільність роботи. Використання найкращих практик у проєктуванні архітектури та інтерфейсів забезпечує якісне виконання функціональних і нефункціональних вимог, що дозволить створити конкурентоспроможний веб-сервіс на ринку нерухомості.

Таким чином, після детального опису найголовніших аспектів у першому розділі можна переходити до безпосередньої реалізації описаної веб-системи, забезпечуючи чітке розуміння технічних особливостей реалізації проєкту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ПОШУКУ НЕРУХОМОСТІ

2.1. Визначення варіантів використання системи

Проектування та розробка веб-системи пошуку нерухомості передбачає чітке розуміння сценаріїв взаємодії користувачів із майбутнім програмним продуктом. Варіанти використання (use cases) є стандартним способом опису функціональних можливостей системи з точки зору кінцевих користувачів [7]. Це дозволяє чітко визначити межі та особливості функціональності майбутньої системи, забезпечує її відповідність реальним потребам та очікуванням користувачів, а також допомагає уникнути помилок при розробці.

Веб-орієнтована система пошуку нерухомості має кілька ключових ролей користувачів, кожна з яких має свої специфічні задачі та потреби. Основні варіанти використання системи:

Варіант використання 1: Реєстрація та авторизація користувачів

Користувач (покупець, орендодавець чи ріелтор) може створити обліковий запис у системі, надаючи базову інформацію про себе та проходячи процес авторизації для подальшого доступу до повного функціоналу платформи.

Варіант використання 2: Пошук та фільтрація оголошень

Користувач має можливість здійснювати пошук об'єктів нерухомості відповідно до заданих критеріїв (тип нерухомості, ціна, площа, район, близькі об'єкти інфраструктури). Пошук здійснюється за допомогою зручних фільтрів, результатом чого є перелік відповідних оголошень, що представлені у вигляді списку або відображаються на інтерактивній карті.

Варіант використання 3: Перегляд детальної інформації оголошення

Користувач може переглядати повну інформацію щодо конкретного об'єкта нерухомості, включаючи опис, фотографії, ціну, додаткові характеристики, місце розташування на карті, близькі місця та іншу релевантну інформацію.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Варіант використання 4: Створення та публікація оголошень

Користувач (продавець або орендодавець) має можливість розмістити власне оголошення про нерухомість. Під час створення оголошення користувач вказує основні параметри об'єкта (тип, площу, ціну тощо), додає опис і фотографії, обирає розташування нерухомості на інтерактивній карті та вказує додаткові переваги або характеристики.

Варіант використання 5: Додавання оголошень до обраного

Користувач має змогу зберігати оголошення, які зацікавили його, в особистий список «Обране» для зручного подальшого доступу до них без необхідності повторного пошуку.

Варіант використання 6: Коментування оголошень

Користувачі можуть залишати коментарі під оголошеннями, уточнюючи інформацію, ставлячи запитання або висловлюючи думки щодо об'єкта нерухомості. Це сприяє більш активній взаємодії між учасниками платформи.

Варіант використання 7: Подання скарг на оголошення

Користувачі мають можливість подавати скарги на оголошення, які містять недостовірну інформацію, порушують правила використання сайту або є шахрайськими. Скарги розглядаються адміністрацією системи.

Варіант використання 8: Адміністрування системи

Адміністратор системи здійснює управління всіма користувачами та оголошеннями, проводить модерацію коментарів і скарг, може блокувати або видаляти оголошення, аналізує активність користувачів для забезпечення стабільної роботи платформи.

Детальний опис наведених сценаріїв використання дозволяє системно підійти до розробки веб-орієнтованої системи, забезпечуючи повноту реалізації необхідного функціоналу, високий рівень комфорту користування [3].

Користувач (zareestrovaniy vidviduvach, poкупець, продавець, орендодавець) є однією з ключових ролей у веб-орієнтованій системі пошуку нерухомості. Нижче наведена діаграма варіантів використання (рис. 2.1).

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

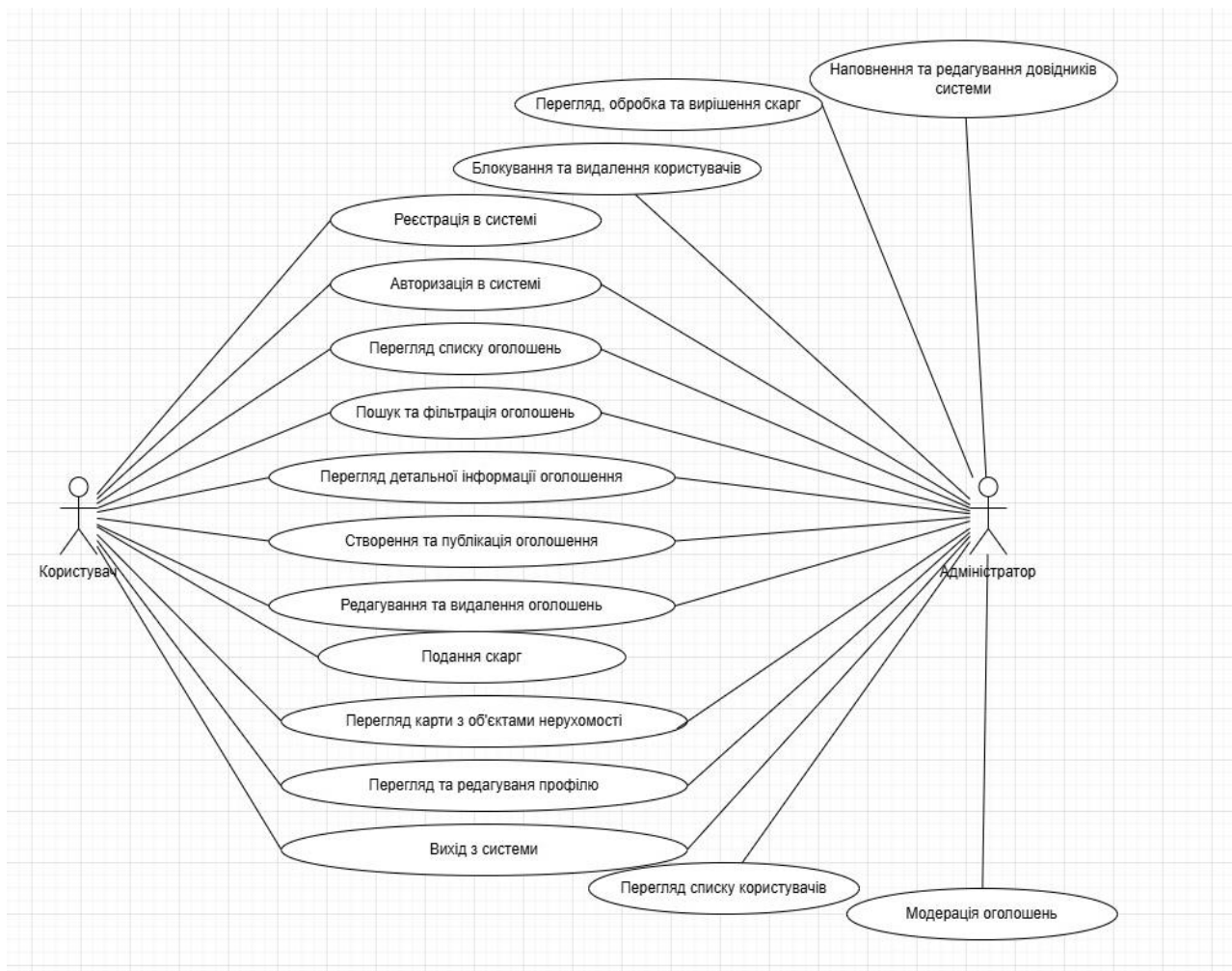


Рис. 2.1 Діаграма варіантів використання для користувача та адміністратора системи

Діаграма прецедентів використання відображає основні сценарії взаємодії користувача з веб-орієнтованою системою пошуку нерухомості. Користувач взаємодіє із системою через низку базових операцій, серед яких реєстрація та авторизація в системі, перегляд списку оголошень, здійснення пошуку та фільтрації за заданими параметрами, перегляд детальної інформації про об'єкти нерухомості, створення, редагування та видалення власних оголошень.

Дана діаграма чітко ілюструє основні сценарії взаємодії користувача із системою та охоплює всі ключові процеси, що забезпечують повний цикл роботи платформи для пошуку та розміщення нерухомості.

Адміністратор є іншою ключовою роллю у веб-системі пошуку нерухомості. На діаграмі прецедентів використання представлено основні функції адміністратора у веб-орієнтованій системі пошуку нерухомості. Адміністратор здійснює авторизацію, переглядає та управляє профілями користувачів, блокує чи видаляє користувачів, модерує оголошення, обробляє скарги та контролює інформацію на карті об'єктів, наповнює та керує довідниками системи. Діаграма варіантів використання демонструє ключові адміністративні дії, спрямовані на підтримку порядку, безпеки та актуальності даних у системі.

Наведені use case діаграми ілюструють основні функції системи з точки зору кінцевих користувачів та адміністративного персоналу, що дозволяє чітко структурувати вимоги до функціональних можливостей розроблюваної системи. Цей підхід є необхідним для визначення меж відповідальності, забезпечення якісної реалізації функціоналу та прозорості роботи сервісу.

2.2. Розробка бази даних веб-системи

База даних є одним із ключових компонентів будь-якої інформаційної системи, оскільки вона забезпечує зберігання, упорядкування, швидкий доступ та ефективну обробку великого обсягу даних [14].

У процесі розробки веб-орієнтованої системи пошуку нерухомості особливу увагу було приділено створенню логічно обґрунтованої, структурованої та масштабованої моделі бази даних (рис. 2.2), яка б відповідала функціональним вимогам.

При проектуванні бази даних враховувалися особливості предметної області, необхідність підтримки складних взаємозв'язків між різними сутностями системи (користувачами, оголошеннями, розташуванням об'єктів, перевагами, скаргами та іншими даними) та можливість подальшого розширення функціоналу.

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

Структура бази даних була спроектована таким чином, щоб забезпечити цілісність інформації, мінімізувати надлишковість даних і оптимізувати основні операції читання та запису.

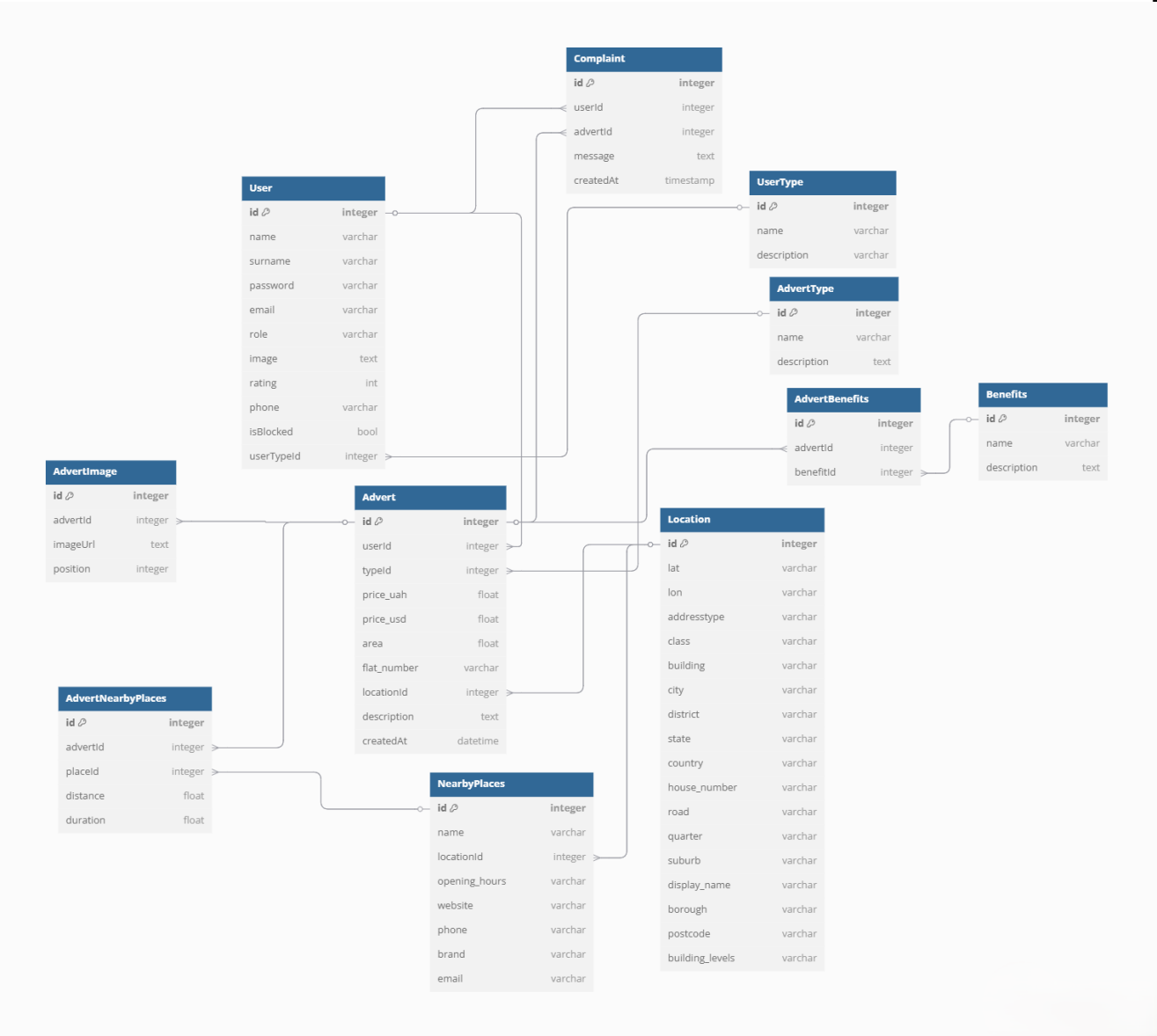


Рис. 2.2. Спроектowana схема бази даних системи пошуку нерухомості

Таблиця UserType (табл. 2.1) містить інформацію про типи користувачів системи. Вона визначає класифікацію користувачів за ролями, що дозволяє гнучко розподіляти права доступу та функціональні можливості в залежності від їхнього типу.

Таблиця 2.1

Таблиця UserType

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
name	varchar(255)	-	-	назва типу користувача
description	varchar(255)	-	-	опис

Таблиця User (табл. 2.2) зберігає основні дані про зареєстрованих користувачів платформи, включаючи персональні дані, контактну інформацію, роль у системі, а також рейтинг та статус блокування користувача.

Таблиця 2.2

Таблиця User

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
name	varchar(255)	-	-	ім'я користувача
surname	varchar(255)	-	-	прізвище користувача
password	varchar(255)	-	-	пароль
email	varchar(255)	-	-	електронна пошта
image	text	-	-	картинка профілю
rating	int	-	-	рейтинг
phone	varchar(255)	-	-	номер телефону
isBlocked	bool	-	-	чи заблокований користувач
userId	integer	-	+	ідентифікатор типу користувача

Таблиця Benefits (табл. 2.3) використовується для зберігання переліку переваг, які можуть бути вказані в оголошеннях про нерухомість, наприклад, наявність ремонту, паркінгу або меблів.

Таблиця 2.3

Таблиця Benefits

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
name	varchar(255)	-	-	назва переваги
description	varchar(255)	-	-	опис

Таблиця AdvertBenefits (табл. 2.4) реалізує зв'язок між оголошеннями та перевагами, що дозволяє прикріплювати до кожного оголошення певний набір переваг, які покращують привабливість об'єкта для потенційних покупців чи орендарів.

Таблиця 2.4

Таблиця AdvertBenefits

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
advertId	integer	-	+	ідентифікатор оголошення
benefitId	integer	-	+	ідентифікатор переваги

Таблиця AdvertImage (табл. 2.5) зберігає інформацію про зображення, які додаються до оголошень. Вона дозволяє прикріплювати до об'єкта нерухомості декілька фотографій із зазначенням порядку їхнього відображення.

Таблиця 2.5

Таблиця AdvertImage

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
advertId	integer	-	+	ідентифікатор оголошення
imageUrl	text	-	-	шлях до файлу зображення
position	integer	-	-	порядок відображення

Таблиця AdvertType (табл. 2.6) містить довідкову інформацію про типи оголошень, наприклад: продаж, оренда, оренда подовово. Це дає можливість класифікувати всі оголошення у системі за їх призначенням.

Таблиця 2.6

Таблиця AdvertType

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
name	varchar(255)	-	-	тип оголошення
description	text	-	-	опис

Таблиця Complaint (табл. 2.7) використовується для зберігання скарг користувачів на оголошення, що дозволяє адміністрації системи оперативно реагувати на можливі порушення правил користування сервісом.

Таблиця 2.7

Таблиця Complaint

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
advertId	integer	-	+	ідентифікатор оголошення
userId	integer	-	+	ідентифікатор користувача
message	text	-	-	текст скарги
createdAt	timestamp	-	-	дата та час створення скарги

Таблиця AdvertNearbyPlaces (табл. 2.8) забезпечує зберігання інформації про об'єкти інфраструктури, що знаходяться поряд із об'єктами нерухомості, із зазначенням відстані та тривалості маршруту.

Таблиця 2.8

Таблиця AdvertNearbyPlaces

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
advertId	integer	-	+	ідентифікатор оголошення
placeId	integer	-	+	ідентифікатор місця
distance	float	-	-	відстань
duration	float	-	-	тривалість

Таблиця Location (табл. 2.9) містить детальну інформацію про географічне розташування об'єктів нерухомості, включаючи широту, довготу, адресу, місто, район, країну, поштовий індекс та інші характеристики об'єкта.

Таблиця 2.9

Таблиця Location

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
lat	varchar(255)	-	-	широта
lon	varchar(255)	-	-	довгота
addresstype	varchar(255)	-	-	адреса

Назва	Тип даних	ПК	ЗК	Опис поля
class	varchar(255)	-	-	клас
building	varchar(255)	-	-	будівля
city	varchar(255)	-	-	місто
district	varchar(255)	-	-	район
state	varchar(255)	-	-	область
country	varchar(255)	-	-	країна
house_number	varchar(255)	-	-	номер будинку
road	varchar(255)	-	-	дорога
quarter	varchar(255)	-	-	квартал
suburb	varchar(255)	-	-	передмістя
display_name	varchar(255)	-	-	назва для відображення
borough	varchar(255)	-	-	адміністративний район
postcode	varchar(255)	-	-	поштовий індекс
building_levels	varchar(255)	-	-	кількість поверхів

Таблиця NearbyPlaces (табл. 2.10) зберігає дані про найближчі об'єкти інфраструктури, такі як магазини, школи, лікарні, разом із контактною інформацією та розташуванням, що дозволяє користувачам краще оцінити привабливість об'єкта нерухомості.

Таблиця 2.10

Таблиця NearbyPlaces

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
advertId	integer	-	+	ідентифікатор оголошення
name	varchar(255)	-	-	назва
locationId	integer	-	+	ідентифікатор розташування
opening_hours	varchar(255)	-	-	години роботи
website	varchar(255)	-	-	посилання на сайт
phone	varchar(255)	-	-	номер телефону
brand	varchar(255)	-	-	назва бренду
email	varchar(255)	-	-	електронна пошта

Таблиця Advert (табл. 2.11) є основною у системі та містить інформацію про всі оголошення, розміщені користувачами. Вона включає дані про продавця

або орендодавця, тип об'єкта, ціну, площу, опис, а також прив'язку до місця розташування.

Таблиця 2.11

Таблиця Advert

Назва	Тип даних	ПК	ЗК	Опис поля
id	integer	+	-	унікальний ідентифікатор запису
userId	integer	-	+	ідентифікатор користувача
typeId	integer	-	+	ідентифікатор типу оголошення
price_uah	float	-	-	ціна у гривнях
price_usd	float	-	-	ціна у доларах
area	float	-	-	площа
flat_number	varchar(255)	-	-	номер квартири
locationId	integer	-	+	ідентифікатор розташування
description	text	-	-	опис
createdAt	datetime	-	-	Дата створення

Спроектована база даних повністю охоплює функціональні потреби веборієнтованої системи пошуку нерухомості та відповідає її логіці роботи. Кожна сутність — користувачі, оголошення, переваги, розташування, об'єкти інфраструктури, типи оголошень, скарги — реалізована через окремі таблиці з чітко визначеними зв'язками за допомогою зовнішніх ключів.

Особливу роль відіграють таблиці Location і NearbyPlaces, які забезпечують просторову прив'язку оголошень до геолокації та дозволяють інтегрувати додаткову інформацію з картографічних сервісів.

2.3. Реалізація веб-орієнтованої системи пошуку нерухомості

Одним із найважливіших функціональних модулів розробленої системи є інтеграція з відкритими геолокаційними сервісами для отримання детальної інформації про об'єкти нерухомості та навколишню інфраструктуру. Для цього було створено спеціальний сервісний клас OSMDataFetcher, який забезпечує взаємодію із зовнішніми API такими як Nominatim, Overpass API та OSRM. Розробку даного модуля можна поділити на два основних етапи.

1. Реалізація отримання адресних та будівельних даних

На першому етапі була реалізована можливість отримання детальної адресної інформації та додаткових характеристик будівель за допомогою координат (широти та довготи). Для цього було використано два відкриті API:

- Nominatim API для зворотного геокодування координат у текстову адресу [17];
- Overpass API для отримання специфічних характеристик найближчих будівель або об'єктів інфраструктури [18].

Відповідна частина коду реалізована в методі getAddressDetails:

```
async getAddressDetails(lat, lon) {
  try {
    const nominatimResponse = await axios.get(`${this.nominatimUrl}`, {
      params: {
        format: 'json',
        lat: lat,
        lon: lon,
        addressdetails: 1
      },
      headers: {
        'User-Agent': 'NodeJS OSM Data Fetcher'
      }
    });
  }
};
```

Метод надсилає запит до Nominatim, обробляє відповідь та додатково звертається до методу getBuildingDetails, що дозволяє збагачувати отриману інформацію даними про найближчі будівлі.

```
const buildingDetails = await this.getBuildingDetails(lat, lon);

return {
  ...nominatimResponse.data,
  buildingDetails
};
} catch (error) {
  console.error('Error fetching address details:', error.message);
  throw error;
}
```

Отримання будівельних даних здійснюється через спеціально сформований запит до Overpass API:

```
async getBuildingDetails(lat, lon) {
  const radius = 10;
  const query = `
    [out:json][timeout:25];
    (
      way(around:${radius},${lat},${lon})["building"];
    )
```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        relation(around:${radius},${lat},${lon})["building"];
        way(around:${radius},${lat},${lon})["amenity"];
        relation(around:${radius},${lat},${lon})["amenity"];
    );
    out body;
>;
    out skel qt;
`;

```

Даний запит дозволяє шукати будівлі та об'єкти інфраструктури в радіусі 10 метрів навколо заданої координати. У відповіді обирається перший об'єкт, що має теговані властивості.

```

try {
  const response = await axios.post(this.overpassUrl, query, {
    headers: {
      'Content-Type': 'text/plain',
    }
  });

  const buildingData = response.data.elements.find(element =>
    element.tags
  );

  return buildingData ? buildingData.tags : null;
} catch (error) {
  console.error('Error fetching building details:', error.message);
  return null;
}

```

У разі виникнення помилок усі виклики API супроводжуються обробкою винятків і логуванням у консоль, що забезпечує надійність роботи модуля. Таким чином, метод getAddressDetails надає об'єднану інформацію про адресу та атрибути будівлі, яка у подальшому використовується при створенні або редагуванні оголошення.

2. Реалізація отримання маршрутної інформації

Другим етапом розробки функціональності стало створення можливості побудови маршрутів між об'єктами на основі географічних координат. Для цього використовувався сервіс OSRM.

Метод getRoute виконує побудову пішохідного маршруту між двома точками:

```

async getRoute(startLat, startLon, endLat, endLon) {
  try {
    const routeResponse = await

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

```

axios.get(`${this.osrmUrl}/${startLon},${startLat};${endLon},${endLat}`, {
  params: {
    overview: 'full',
    geometries: 'geojson'
  }
});

return routeResponse.data.routes[0];
} catch (error) {
  console.error('Error fetching route:', error.message);
  throw error;
}
}

```

При виклику даного методу формується запит до OSRM API з передачею координат початку та кінця маршруту. Використання параметрів `overview=full` та `geometries=geojson` дозволяє отримати повну геометрію маршруту у форматі GeoJSON, що полегшує його подальшу інтеграцію із картографічним сервісом на стороні клієнта.

Також одним із важливих аспектів роботи веб-орієнтованої системи пошуку нерухомості є надання користувачам інформації про об'єкти інфраструктури, розташовані поблизу об'єкта нерухомості. З метою автоматизації збору та збереження таких даних була реалізована функція `createNearbyPlace`, яка об'єднує декілька важливих етапів обробки даних.

Основна логіка реалізована у наступному фрагменті коду:

```

const createNearbyPlace = async (body) => {
  try {
    const { lat, lon } = body;
    const location = await locationRepository.createLocation(lat, lon);
    const data = await OSMDataFetcher.getJsonData(lat, lon);

```

Функція починається із розпакування координат широти та довготи, які передаються через тіло запиту. Отримані координати використовуються для створення нового запису в таблиці `Location` за допомогою репозиторію `locationRepository`. Після створення запису система за допомогою виклику `OSMDataFetcher.getJsonData(lat, lon)` звертається до зовнішніх геолокаційних API для отримання розширеної інформації про об'єкт поблизу — включно з режимом роботи, контактною інформацією та іншими характеристиками.

					ІПЗ.КР.Б-121-25-ІЗ	Арк. 34
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const nearbyPlaceData = {
  locationId: location.id,
  opening_hours: data.buildingDetails?.opening_hours || null,
  website: data.buildingDetails?.website || null,
  phone: data.buildingDetails?.phone || null,
  brand: data.buildingDetails?.brand || data.address?.amenity ||
null,
  email: data.buildingDetails?.email || null,
  name: data.name || data.address?.amenity || null,
};

return await NearbyPlaces.create(nearbyPlaceData);
} catch (error) {
  throw new Error('Не вдалося створити місце поблизу: ' + error.message);
}
};

```

Далі формується об'єкт `nearbyPlaceData`, який агрегує отримані дані. Логіка побудована таким чином, що якщо деякі дані відсутні у відповіді, вони замінюються на `null` або беруться із альтернативних джерел відповіді (`buildingDetails` або `address`). Це дозволяє уникнути помилок при збереженні запису, а також підтримувати цілісність структури даних.

Не менш складною частиною системи у реалізації стало створення інтерактивної карти нерухомості. Інтерактивна карта забезпечує візуалізацію розташування об'єктів і підвищує зручність взаємодії користувачів із платформою. Для її реалізації було використано `MapTiler SDK`, який дозволяє створювати інтегровані веб-карти з розширеним функціоналом.

Базову ініціалізацію карти реалізовано у компоненті `Map`, який під час завантаження налаштовує параметри центру, початкового рівня масштабування, мовних налаштувань та меж області перегляду:

```

map.current = new maptilersdk.Map({
  container: mapContainer.current,
  center: [31.490773812277094, 49.01605198088264],
  zoom: 5.5,
  language: maptilersdk.Language.UKRAINIAN,
  maxBounds: [
    [21.79545514866079, 43.424418976677686],
    [42.82825058830223, 52.70793206338687],
  ],
  ...options
});

```

Після завантаження викликається функція onLoad, яка передає екземпляр карти для подальшої роботи у компоненті AdvertsMap. У цьому компоненті при завантаженні карти здійснюється підключення джерела даних типу GeoJSON, що містить інформацію про розташування об'єктів нерухомості. Функція formatSourceFeatures перетворює масив оголошень у відповідний формат:

```
export function formatSourceFeatures(data) {
  const features = data.map(({ id, locationForAdvert: { lat, lon } }) => ({
    id,
    type: 'Feature',
    geometry: {
      type: 'Point',
      coordinates: [lon, lat]
    },
  }));

  return {
    type: 'FeatureCollection',
    features: features
  }
}
```

Далі відбувається візуалізація об'єктів у вигляді кругових маркерів із динамічно змінними властивостями стилізації залежно від стану об'єкта:

```
mapInstance.addLayer({
  id: 'property',
  type: 'circle',
  source: 'property',
  paint: {
    "circle-radius": {
      stops: [[8, 2], [13, 3], [16, 6]]
    },
    "circle-color": [
      "case",
      ["==", ["feature-state", "active"], true],
      "#F4801A",
      "white"
    ],
    "circle-stroke-width": [
      "case",
      ["any",
        ["==", ["feature-state", "hover"], true],
        ["==", ["feature-state", "active"], true]
      ],
      3,
      1
    ],
    "circle-stroke-color": "#F4801A"
  }
});
```

Таким чином, об'єкти можуть змінювати свій вигляд під час взаємодії — наприклад, змінювати колір або межі при активації або наведенні. Також була реалізована логіка створення власних маркерів для кожного об'єкта нерухомості за допомогою `useMapMarkers` (створення HTML-елемента із зазначенням вартості об'єкта):

```
const marker = document.createElement('div');
marker.className = 'property-marker';
marker.textContent = formatCurrency(price_usd);
marker.dataset.markerId = id;
```

Далі маркер додається на карту:

```
const markerInstance = new Marker({
  element: marker,
  anchor: "bottom",
  offset: [0, -5]
}).setLngLat(coordinates)
.addTo(map);
```

Таким чином, кожен об'єкт отримує свій власний маркер із відображенням ціни без необхідності додаткової обробки на клієнтській стороні. Розроблена інтерактивна карта забезпечує динамічну візуалізацію об'єктів нерухомості, підтримує обробку кліків, наведення та вибору об'єктів (`handlePropertyHover`, `handleMapClick` та `handlePropertyClick`), що значно покращує користувацький досвід та відповідає сучасним стандартам розробки вебсистем.

Висновки до другого розділу

Здійснено комплексне проєктування веб-орієнтованої системи пошуку нерухомості, що включало визначення основних варіантів використання, побудову логічної структури бази даних та розробку ключових функціональних модулів.

Аналіз варіантів використання дозволив системно окреслити взаємодію користувачів із системою. Було визначено основні сценарії роботи для різних категорій користувачів, включаючи реєстрацію, пошук та фільтрацію оголошень, перегляд детальної інформації, створення та керування оголошеннями, збереження об'єктів до обраного, коментування, подання скарг і

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

адміністрування платформи. Створені діаграми варіантів використання забезпечили чітку структурування функціональних можливостей.

У процесі проектування бази даних було визначено основні сутності та їх взаємозв'язки відповідно до предметної області. Спроектована структура забезпечує підтримку цілісності даних, що є необхідною умовою для ефективної роботи системи в умовах обробки великих обсягів інформації.

Особливу увагу приділено реалізації модулів інтеграції з геолокаційними сервісами. Розроблений спеціалізований клас забезпечує отримання актуальної інформації про місцезнаходження об'єктів, їх характеристики, а також можливість побудови маршрутів. Це дозволяє автоматизувати процес збору даних та забезпечити високу актуальність і повноту інформації в системі.

Реалізація інтерактивної карти із застосуванням засобів MapTiler SDK значно розширила функціональні можливості системи. Було забезпечено візуалізацію об'єктів нерухомості, побудову динамічних маркерів, інтеграцію функцій вибору, наведення та перегляду об'єктів без перезавантаження сторінки. Реалізовані механізми взаємодії користувачів із картою відповідають сучасним вимогам до інтерфейсів веб-застосунків.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ОРІЄНТОВАНОЮ СИСТЕМОЮ ПОШУКУ НЕРУХОМОСТІ

3.1. Порядок встановлення та налаштування системних параметрів

Процес встановлення та налаштування системних параметрів є невід’ємним кроком у підготовці до запуску веб-орієнтованої системи пошуку нерухомості. Система базується на сучасних вебтехнологіях, серед яких Docker, Nginx, Node.js, MinIO, MySQL, а також фреймворк React у поєднанні з Vite, Redux Toolkit, React Router, Tanstack Table, Ant Design, MapTiler, React Icons та React Slick.

Процес розгортання розробленої веб-орієнтованої системи пошуку нерухомості на новому сервері складається з кількох послідовних етапів, які забезпечують її стабільну роботу в робочому середовищі. Цей процес охоплює підготовку сервера, налаштування змінних середовища, розгортання необхідних сервісів, конфігурацію бази даних та інтеграцію зовнішніх API.

На першому етапі потрібно підготувати сервер до роботи, встановивши Docker та Docker Compose, які є основою для контейнеризації компонентів системи. Після цього слід завантажити вихідний код системи та розмістити його в окремій директорії на сервері.

Docker Compose — це зручний інструмент, який спрощує роботу з додатками, що складаються з кількох контейнерів Docker. Він дозволяє описувати та керувати групою взаємопов'язаних сервісів, використовуючи один конфігураційний YAML-файл.

Наступний крок – це створення конфігураційних файлів середовища, які є важливими для налаштування роботи вебсистеми в різних середовищах (розробка, тестування, продакшн). У цих файлах ми визначаємо ключові параметри, такі як доступ до зовнішніх API (наприклад, ключі MapTiler для інтеграції інтерактивних карт), облікові дані для MySQL (користувач, пароль, назва бази, порт підключення), а також конфігурацію адрес і портів для

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

внутрішніх сервісів (наприклад, Node.js API, MinIO, Nginx). Крім того, тут можуть бути й інші змінні середовища, які впливають на логіку роботи системи (рівень логування, налаштування кешування, параметри безпеки). Використання цих конфігураційних файлів дозволяє швидко змінювати налаштування системи без необхідності редагувати програмний код, що значно спрощує процес розгортання та підтримки вебдодатку.

Конфігураційний файл `docker-compose.yml` є центральним елементом процесу розгортання системи, що містить набір директив, необхідних для налаштування роботи багатоконтейнерних застосунків. Ці директиви визначають, які саме контейнери будуть запущені, їхню взаємодію, способи зберігання даних, а також розподіл системних ресурсів.

Основними директивами, що використовуються при конфігурації вебсистеми, є:

- `version` — яка визначає версію синтаксису Docker Compose;
- `services` — перелік сервісів, що складають програмний комплекс;
- `image` — зазначає контейнерний образ, який використовується;
- `build` — дозволяє створювати контейнери на основі інструкцій з Dockerfile;
- `ports` — відкриває доступ до контейнерів через конкретні порти;
- `environment` — задає необхідні змінні середовища для контейнерів;
- `volumes` — дозволяє зберігати інформацію поза межами контейнерів для забезпечення її доступності після перезапусків;
- `networks` — налаштовує внутрішню комунікацію між контейнерами;
- `depends_on` — визначає порядок запуску контейнерів, забезпечуючи коректну послідовність старту сервісів у системі.

Після цього необхідно налаштувати інтеграцію із зовнішніми сервісами. При розробці веб-орієнтованої системи пошуку нерухомості був використаний сервіс MapTiler для роботи інтерактивної карти. Для її підключення слід отримати API-ключ сервісу MapTiler. Для цього спершу потрібно

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

zareestruvatisia na saiti maptiler.com [12]. Otrimаний ключ потрібно підставити у змінну середовища VITE_MAPTILER_API_KEY у файлі .env каталогу frontend.

Після конфігурації змінних середовища потрібно виконати розгортання системи за допомогою Docker Compose. Для цього використовується спеціальний скрипт запуску recreate.sh, який знаходиться у кореневій директорії проєкту. Перший запуск здійснюється двічі, що дозволяє повністю ініціалізувати контейнери всіх необхідних сервісів (Node.js API, фронтенд із React, Nginx як зворотний проксі-сервер, база даних MySQL та файлове сховище MinIO). Подальші перезапуски або оновлення системи здійснюються однократним запуском цього ж скрипта.

Після успішного запуску контейнерів необхідно провести налаштування бази даних, виконавши усі прописані міграції. Ця операція створить необхідні таблиці та загальну структуру бази даних відповідно до розробленої та описаної раніше схеми, забезпечуючи коректну взаємодію всіх компонентів системи.

3.2. Структура інтерфейсу та порядок взаємодії з веб-орієнтованою системою пошуку нерухомості

Інтерфейс користувача – це один з найважливіших аспектів, який визначає успіх веб-орієнтованих систем. Він безпосередньо впливає на зручність і ефективність взаємодії користувачів із наявним функціоналом системи. Якщо інтерфейс розроблений правильно та легким для розуміння, це не лише підвищує загальну задоволеність користувачів, але й зменшує ймовірність помилок під час роботи з платформою, а також, що не менш важливо, скорочує час для виконання основних операцій.

З огляду на специфіку сфери нерухомості інтерфейс системи має забезпечувати швидкий доступ до потрібної інформації, зручні механізми фільтрації, пошуку та перегляду оголошень.

Коли користувач вперше заходить у вебсистему, він потрапляє на головну сторінку сайту (рис. 3.1), яка слугує початковою точкою для взаємодії з

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

ключовими функціями платформи. На цій сторінці розташована навігаційна панель з основними тематичними розділами, такими як «Купівля», «Оренда» та «Новобудови». Користувач може легко перейти до цих розділів, вибираючи той, що найбільше відповідає його інтересам і потребам. Додатково, на головній сторінці можна знайти останні оголошення, які згруповані в секції «Останні пропозиції». Це дозволяє швидко ознайомитися з новими актуальними пропозиціями на ринку нерухомості.

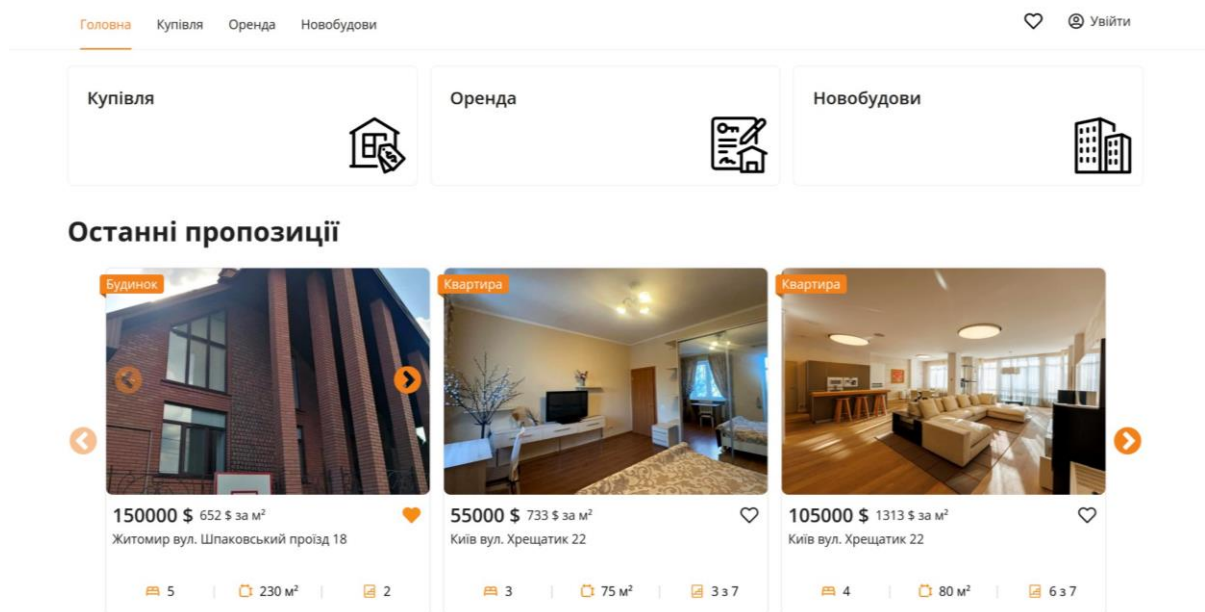


Рис. 3.1 Головна сторінка вебсистеми пошуку нерухомості

Після переходу до розділу «Купівля нерухомості» користувач отримує доступ до спеціалізованої сторінки, де представлений перелік актуальних оголошень щодо продажу різних об'єктів нерухомості (рис. 3.2).

Для підвищення зручності та наочності пошуку інформації вище над списком оголошень відображається інтерактивна карта. На цій карті розміщено маркери у вигляді позначок, кожна з яких відповідає конкретному об'єкту нерухомості та містить інформацію про його вартість.

Маркери дозволяють користувачу візуально оцінити розташування нерухомості, її близькість до інших важливих об'єктів інфраструктури та загальний рівень цін у певних районах чи місцевостях.

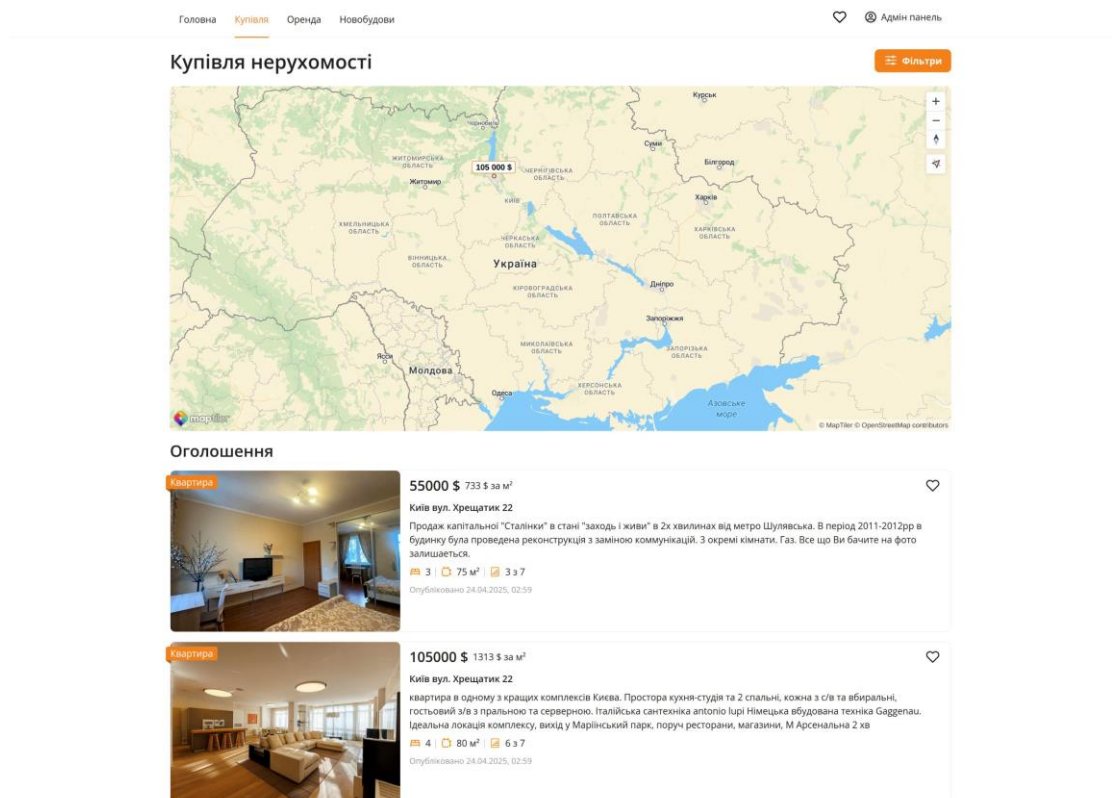


Рис. 3.2. Сторінка розділу «Купівля нерухомості»

Кожен маркер на карті є інтерактивним: при кліку на нього користувач може переглянути коротку інформацію про об'єкт (ціна, адреса, площа, поверх і тд) або ж перейти до повного опису. (рис. 3.3)

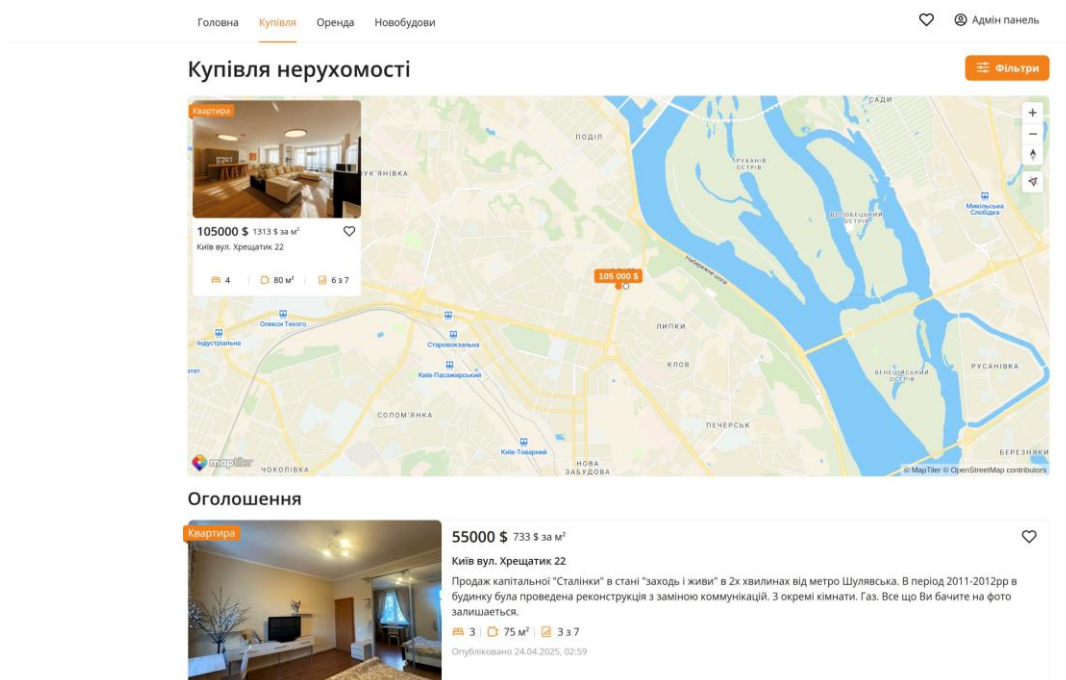


Рис. 3.3. Перегляд основної інформації про маркер на карті

Для кращої ефективності пошуку та врахування індивідуальних потреб та побажань покупців був створений набір пошукових фільтрів (рис. 3.4). Серед усіх наявних оголошень користувачі можуть легко звузити список, обираючи з різних критеріїв: тип нерухомості (квартира, будинок, комерційне приміщення тощо), місто, ціновий діапазон, кількість кімнат чи загальна площа об'єкта. Як тільки змінюється будь-який параметр, список оголошень оновлюється, показуючи лише ті пропозиції, що відповідають обраним критеріям.

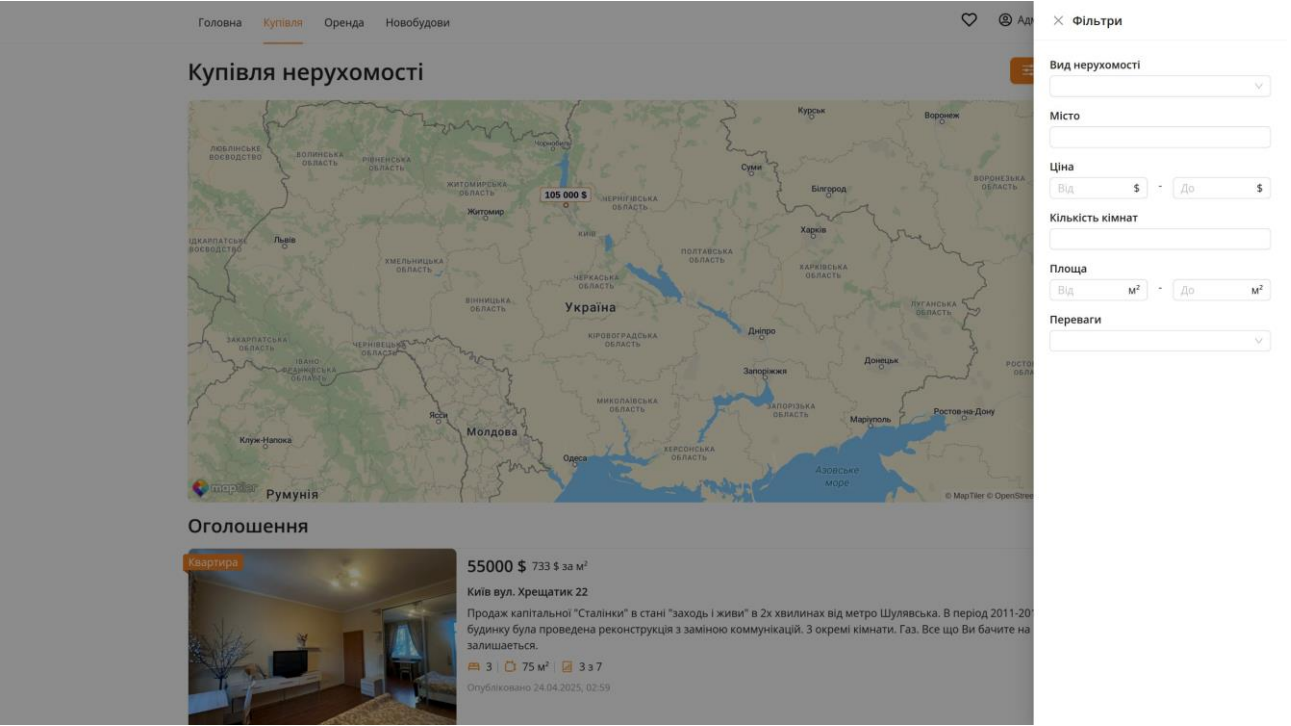


Рис. 3.4 Застосування фільтрів до списку оголошень

Після вибору оголошення, що зацікавило, користувач переходить на сторінку детальної інформації (рис. 3.5), де зосереджені всі ключові дані про об'єкт нерухомості. Окрім основного опису, тут доступні фотогалерея, що демонструє інтер'єр та екстер'єр, а також параметри житла: загальна площа, поверх, кількість кімнат, наявність балкону чи паркінгу тощо. Додатково представлено перелік переваг об'єкта, які можуть включати якісний ремонт, вид із вікон або близькість до транспортних вузлів. Карта дозволяє проаналізувати точне розташування нерухомості, а контактні дані власника чи рієлтора забезпечують швидкий зв'язок для уточнення нюансів або організації перегляду.

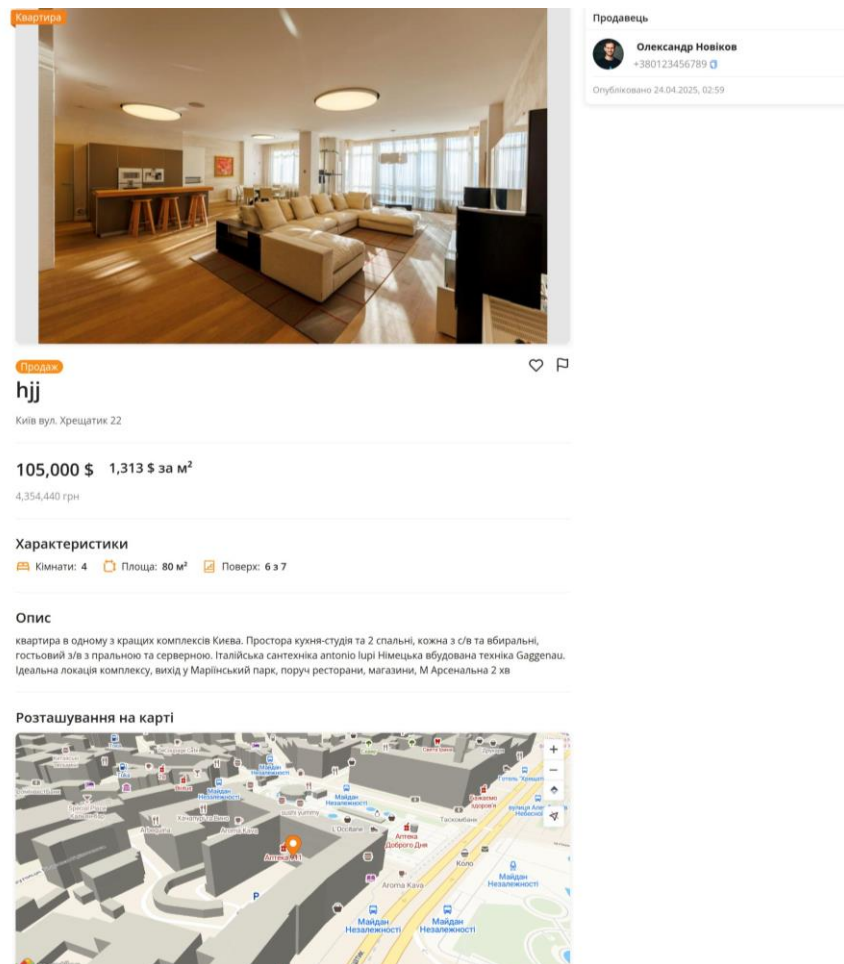


Рис. 3.5 Сторінка детальної інформації про об'єкт нерухомості

У випадку виявлення користувачем невідповідностей в оголошенні, що суперечить загальноприйнятим правилам, передбачено механізм подання скарги на конкретне оголошення. Після натискання відповідної кнопки відкривається спливаюче вікно із формою зворотного зв'язку (рис. 3.6), у якій можна детально описати причину претензії. Користувачу пропонується ввести стислий текст повідомлення, що пояснює виявлену проблему—наприклад, хибні дані про площу, невідповідність зображень чи порушення етичних стандартів.

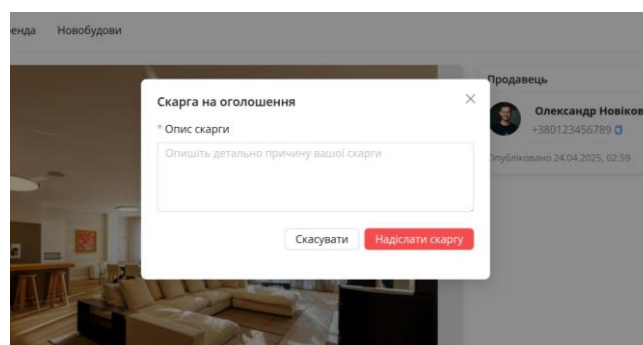


Рис. 3.6. Форма відправки скарги на оголошення

Надіслана скарга автоматично потрапляє до черги модерації (рис. 3.7), де адміністратор проводить перевірку наведених фактів та вживає необхідних заходів для усунення недоліків або видалення оголошення. Це гарантує підтримку якості контенту та довіри користувачів до платформи.

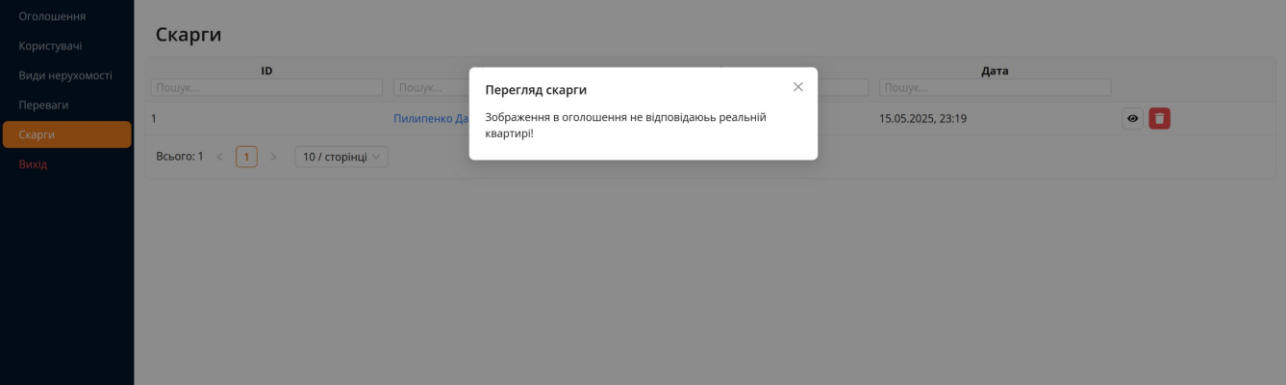


Рис. 3.7. Сторінка перегляду надісланих скарг

У разі підтвердження порушення правил публікації або виявлення неправдивої інформації адміністратор може видалити відповідне оголошення з платформи. Якщо на одного користувача (на його оголошення) надходить велика кількість скарг, що свідчить про систематичне порушення, адміністратор має право заблокувати обліковий запис цього користувача (рис. 3.8).

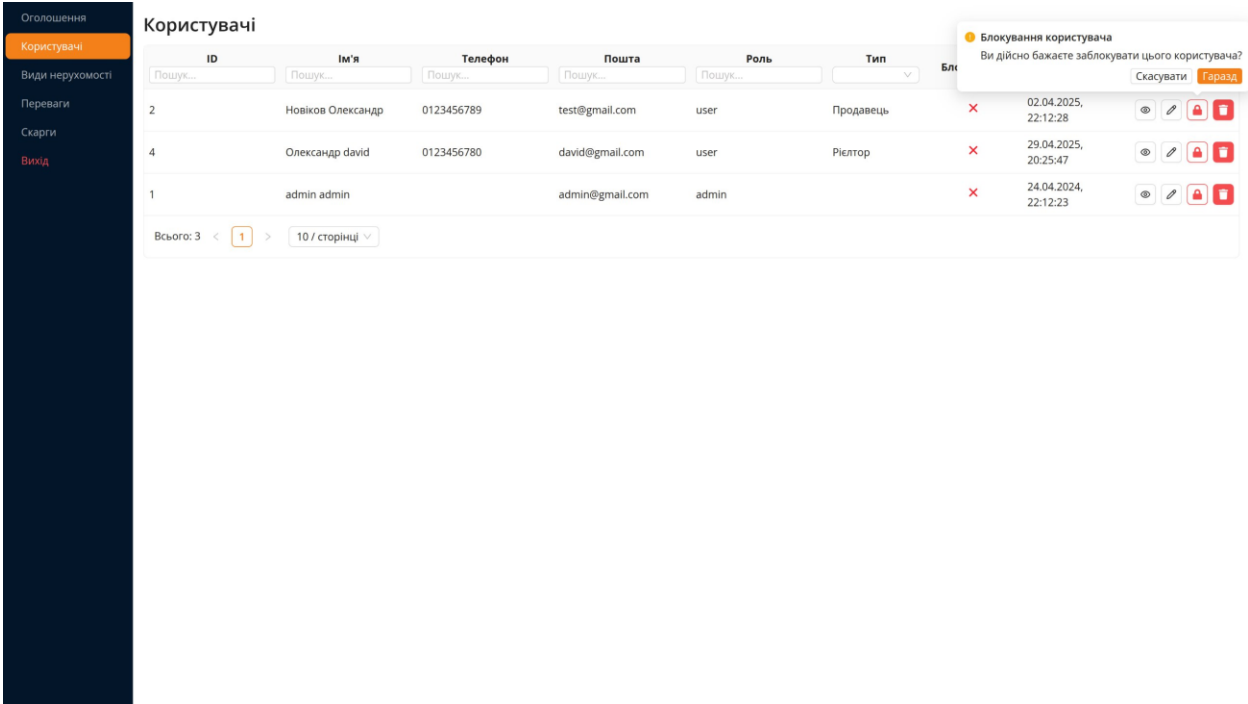


Рис. 3.8. Адміністративна панель керування користувачами

У разі зацікавленості користувача певним оголошенням передбачена можливість його збереження для пізнішого перегляду. Для цього поряд із кожним оголошенням розміщено кнопку «Зберегти» (іконку сердечка), при натисканні на яку оголошення автоматично додається до розділу «Збережені оголошення» (рис. 3.9) в особистому профілі. Користувач може у будь-який момент перейти до цього розділу, щоб швидко відкрити та переглянути всі відкладені пропозиції без необхідності повторного пошуку. Збережені оголошення відображаються у вигляді списку з можливістю видалення непотрібних записів або переходу до їх детального перегляду.

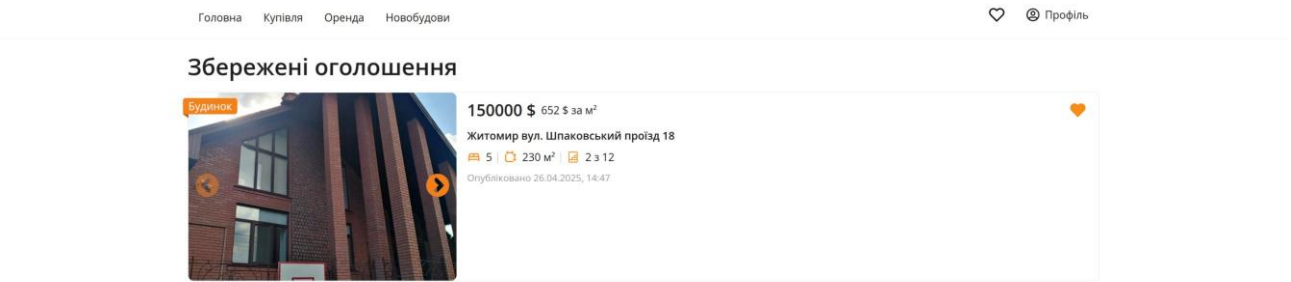


Рис. 3.9. Сторінка зі збереженими оголошеннями

Для доступу до цих функціональних можливостей платформи — збереження оголошень, подання скарг, публікації власних оголошень — користувач мусить пройти процедуру реєстрації та авторизації. Процес реєстрації передбачає заповнення відповідної форми (рис. 3.10), до якої входять обов’язкові поля: ім’я, прізвище, адреса електронної пошти, пароль та номер телефону. Додатково користувач повинен вказати свою роль на платформі, що визначає набір прав доступу і доступний функціонал у його профілі.

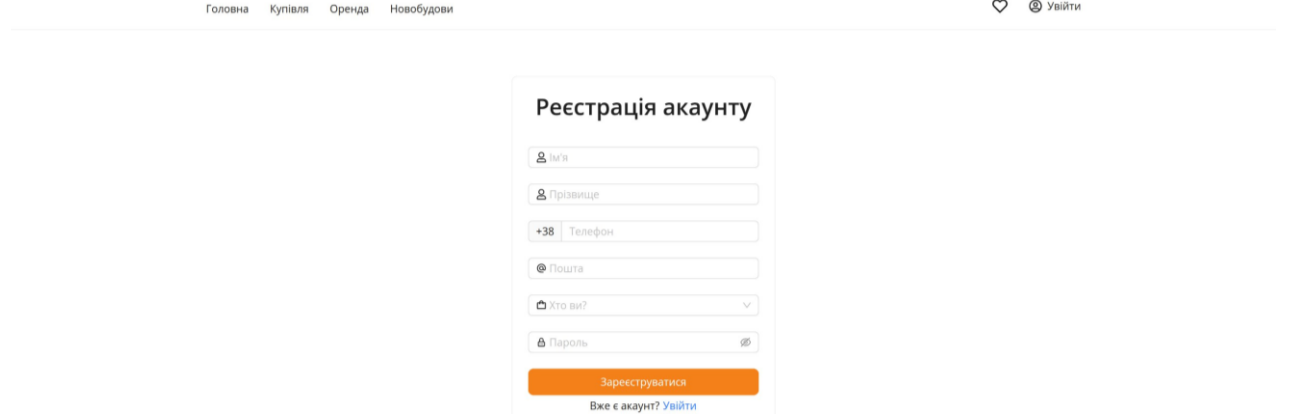


Рис. 3.10. Форма реєстрації нового користувача

У разі вибору під час реєстрації ролі «продавець нерухомості» або «орендодавець» користувачу відкривається набір інструментів для управління власними оголошеннями. Після авторизації в цій ролі він отримує можливість створювати нові записи про об'єкти нерухомості, вказуючи всі необхідні параметри — тип об'єкта, площу, ціну, фотографії та геолокацію на карті.

Додатково передбачено редагування вже опублікованих оголошень з оновленням інформації (рис. 3.11) та можливість їхнього видалення. Такий функціонал дозволяє власникам нерухомості оперативно керувати своїми пропозиціями, контролювати актуальність опису та ціноутворення.

Рис. 3.11. Сторінка редагування оголошення

Кожен зареєстрований користувач, незалежно від наданої йому ролі (покупець, продавець, орендодавець чи адмін), має можливість перейти до власного профілю (рис. 3.12) та оновити персональні дані. На сторінці профілю передбачено форми редагування базової інформації — ім'я, прізвище, електронну пошту, номер телефону — а також зміни пароля.

3.3. Тестування роботи розробленої веб-орієнтованої системи

Після розробки та отримання вже готової веб-орієнтованої системи пошуку нерухомості необхідно також провести тестування усіх функціональних можливостей, які в майбутньому буде надано користувачам. Це, зазвичай, робиться для того, щоб перевірити продукт на наявність помилок та побачити, як він поводить себе у тій чи іншій ситуації.

З цією метою було проведено детальне мануальне тестування основних сценаріїв взаємодії користувачів з вебсайтом. Ручне тестування вебдодатків володіє низкою переваг, які не завжди можуть бути забезпечені автоматизованими інструментами. По-перше, тестувальники здатні оперативно адаптуватися та перевіряти ті ділянки функціоналу, які не були охоплені сценаріями автоматизованих тестів, що істотно підвищує загальне тестове покриття і дозволяє виявити приховані дефекти.

По-друге, людина може помічати нетипові або непередбачені помилки, які важко формалізувати заздалегідь, оскільки автотести не здатні «імпровізувати». А також, лише людина може скористатися власним здобутим досвідом та інтуїцією для оцінки ергономіки програми на рівні, доступному лише живому користувачу.

Процедура тестування веб-системи передбачала ретельну перевірку механізмів валідації форм як на рівні клієнтської, так і на рівні серверної частини. Зокрема, під час реєстрації користувача здійснено контроль обов'язкових полів: якщо хоча б одне із них лишається незаповненим або містить некоректний формат (наприклад, неправильно введену електронну адресу), система негайно відображає детальні повідомлення про помилку (рис. 3.15) й блокує надсилання запиту до сервера. Такий підхід запобігає виникненню помилок у серверному оточенні, пов'язаних із недостатністю даних, необхідних для успішного завершення операції.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Реєстрація акаунту

 ім'я

ім'я обов'язкове

 Прізвище

Прізвище обов'язкове

+38 Телефон

Телефон обов'язковий

@ Пошта

Пошта обов'язкова

 Хто ви?

Виберіть хто ви

 Пароль

Пароль обов'язковий

Зареєструватися

Вже є акаунт? [Увійти](#)

Рис. 3.15. Виведення повідомлень про обов'язкові поля при реєстрації

Аналогічний механізм валідації використовувався і при створенні оголошень: перед тим, як дані потраплять на сервер, система перевіряє, чи заповнені всі обов'язкові поля та чи відповідають вони правильному формату. Якщо користувач пропустив будь-який з необхідних параметрів або ввів значення, що виходять за межі допустимого для даного поля, йому буде показано чітке повідомлення про помилку, а відправка форми буде заблокована до тих пір, поки неточності не будуть виправлені. Такий підхід забезпечує, що всі важливі дані будуть коректно передані на сервер, а це допомагає уникнути помилок під час обробки на серверній стороні.

Олександр Новіков

test@gmail.com

Мої оголошення

Персональні дані

Зміна пароля

Вихід

Створення оголошення

* Заголовок

Заголовок обов'язковий

* Вид пропозиції

Оберть вид пропозиції

*

Оберть вид пропозиції

* Ціна

Введіть ціну

\$

Ціна обов'язкова

* Місце знаходження

Пошук

Місце знаходження обов'язкове

Опис

Опис до оголошення...

Рис. 3.16. Перевірка введених даних під час створення оголошення

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

Під час тестування механізму авторизації також перевірялася стійкість системи до некоректних даних від користувача. Зокрема, вводилися випадки зі зміненими символами в електронній адресі, неправильними комбінаціями логіну та пароля або повністю відсутніми полями, після чого відправлявся запит на сервер. У відповідь система одразу відображала інформативне повідомлення про помилку («Пошту або пароль вказано неправильно») і блокувала доступ до розділів, що недоступні для неавторизованих користувачів, без здійснення подальших запитів до бази даних.

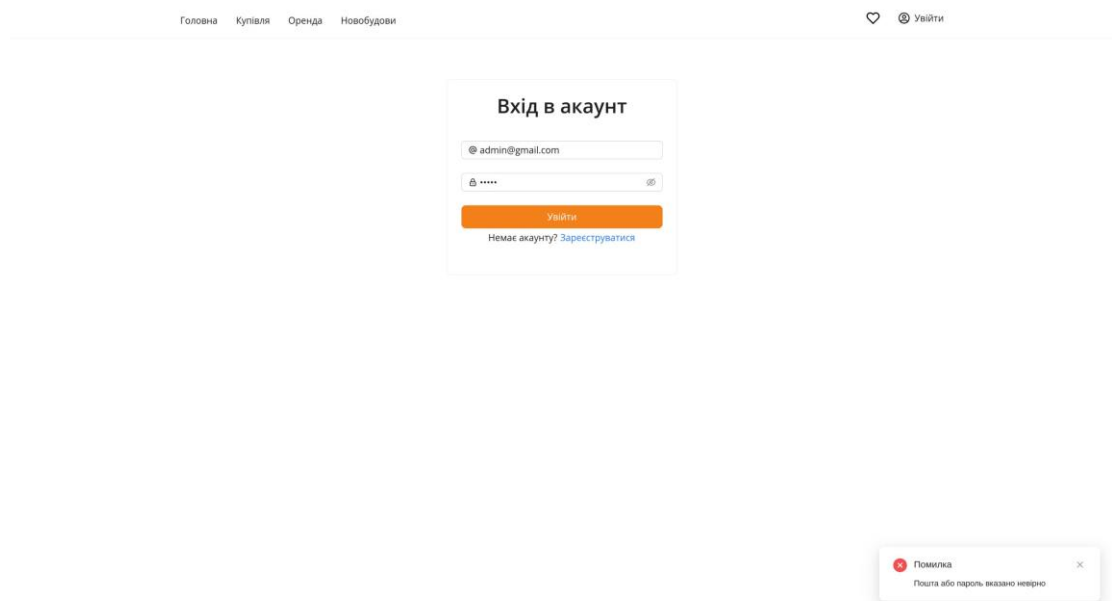


Рис. 3.17. Відображення помилки під час авторизації

Крім того, під час тестування було перевірено, чи є механізм підтвердження перед виконанням важливих операцій, таких як, наприклад, видалення записів.

Коли хтось з користувачів намагається видалити оголошення або будь-який інший важливий елемент, система показує модальне вікно з запитом підтвердження («Ви дійсно хочете видалити цей запис?») (рис. 3.18), що вимагає від користувача свідомого підтвердження дії.

Це перешкоджає помилковому видаленню наявної інформації та зменшення витрат часу на відновлення та повторне введення видаленої інформації на сайт.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

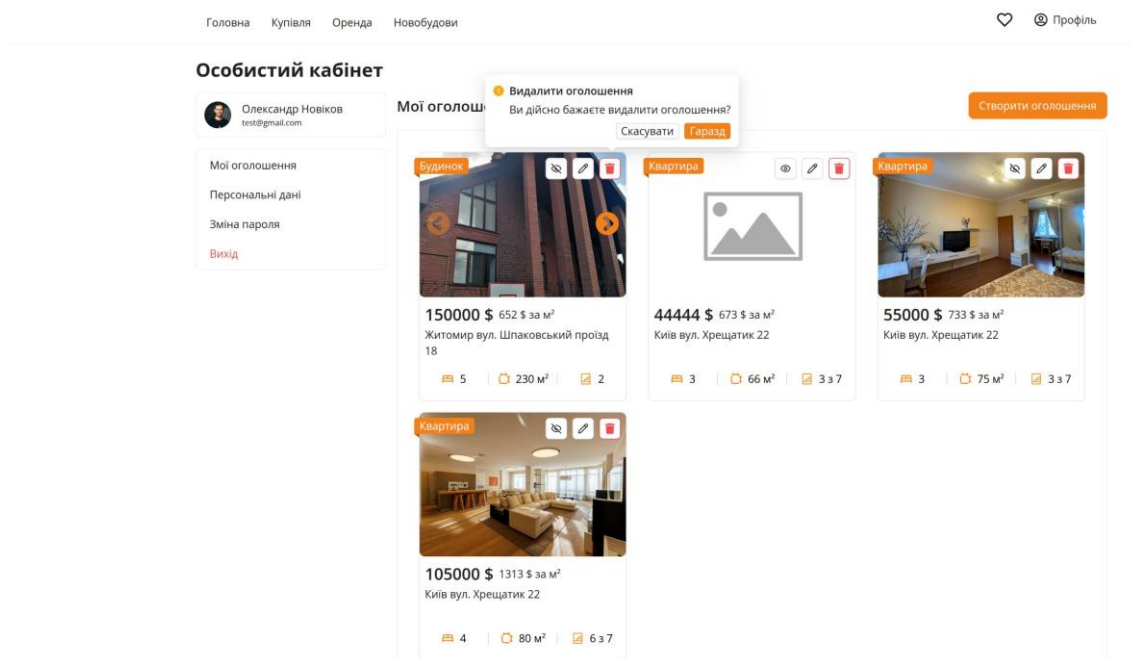


Рис. 3.18. Підтвердження видалення оголошення

Отже, завдяки здійсненому тестуванню системи пошуку нерухомості, було виявлено та виправлено знайдені недоліки. В результаті отримано покращений вебдодаток з підвищеною надійністю та стабільністю роботи, і також покращено інтерфейс для зручнішого досвіду користування системою в майбутньому.

Висновки до третього розділу

У третьому розділі було детально описано процес розгортання, налаштування та тестування розробленої веб-орієнтованої системи пошуку нерухомості. Використання контейнеризації з технологією Docker і Docker Compose значно спростило налаштування середовища, допомагаючи уникнути проблем, пов'язаних з несумісністю конфігурацій на різних серверах. Ключовим етапом стала конфігурація змінних середовища, яка включала налаштування доступу до бази даних, інтеграцію сервісу карт MapTiler, а також параметри взаємодії між компонентами (Node.js, MySQL, MinIO, React).

У процесі реалізації інтерфейсу значну увагу було приділено його ергономіці та зручності для майбутнього кінцевого користувача. Структуроване розташування основних елементів навігації, чітке виокремлення розділів («Купівля», «Оренда», «Новобудови») та наявність інтерактивної карти з

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

можливістю візуального перегляду оголошень об'єктів нерухомості дозволяють користувачам швидко знаходити потрібну інформацію. Окрім цього, інтерфейс включає широкі функціональні можливості: збереження вподобаних оголошень, надсилання скарг на фотографії / описи оголошення, створення й управління власними оголошеннями, а також управління профілем користувача. Особливістю є наявність адміністративної панелі, що надає адміністратору можливості контролювати та актуалізувати довідники і оперативно реагувати на дії та надіслані скарги користувачів.

Не менш важливим аспектом для гарантії стабільності роботи вебсистеми стало ретельне мануальне тестування. У рамках тестування було перевірено правильність функціонування механізмів валідації введення даних на клієнтському та серверному рівнях, зокрема при реєстрації, авторизації та створенні оголошень. Проведені перевірки підтвердили, що система коректно реагує на пропуск обов'язкових полів, а також забезпечує своєчасне виведення інформативних повідомлень про недоліки у інформації та помилки користувачам. Також було перевірено механізм підтвердження важливих дій (наприклад, видалення записів), що знижує ймовірність помилок і забезпечує збереження інформації в системі.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

1. Розроблено веб-орієнтовану систему для пошуку нерухомості, яка враховує сучасні вимоги ринку та особливості взаємодії кінцевих користувачів. Глибоке дослідження конкурентних платформ допомогло виявити основні недоліки існуючих сервісів, такі як недостатня гнучкість інтерфейсів, обмежений функціонал, слабка інтеграція картографічних технологій та надмірна кількість рекламного контенту. На основі цього був складений список функціональних і нефункціональних вимог, які забезпечують конкурентоспроможність, зручність використання, високу продуктивність і масштабованість запропонованого рішення.

2. При проектуванні системи були проаналізовано та обґрунтовано вибір архітектурних рішень та технологічного стеку, які включають сучасні інструменти та фреймворки: Docker, Node.js, MySQL, MinIO для серверної частини, та React, Vite, Redux Toolkit, Ant Design, React Router, Tanstack Table, MapTiler для клієнтської частини. Використання контейнеризованого підходу на базі Docker дозволило досягнути високої та легкої переносимості та спростити процес розгортання. Це було підтверджено на практиці під час налаштування системи на сервері.

3. Розроблена структура інтерфейсу забезпечує швидкий та інтуїтивно зрозумілий доступ до ключових функцій системи, таких як пошук, фільтрація оголошень, створення та управління записами, взаємодія з інтерактивною картою та механізми адміністрування платформи. Впровадження інтерактивної карти із маркерами нерухомості значно покращує досвід користувачів завдяки візуальній наочності представленої нерухомості та динамічній взаємодії з маркерами без перезавантаження сторінки.

4. Проведено перевірку якості розробленої вебсистеми шляхом проведення комплексного мануального тестування. У результаті тестування підтверджено коректність роботи механізмів валідації введених даних на рівні як клієнта, так і сервера, успішно відпрацьовано обробку помилкових дій користувача під час

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

реєстрації, авторизації та створення оголошень. Крім того, реалізовано систему підтвердження деяких важливих операцій, таких як видалення записів, що дозволяє уникнути випадкового знищення інформації.

Таким чином, створена веб-орієнтована система пошуку нерухомості вирізняється стабільністю роботи, високою продуктивністю, зручним інтерфейсом та повною відповідністю вказаним вимогам і очікуванням кінцевих користувачів.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Безверхий, О. Ефективність застосування бібліотеки React. / Безверхий О., Куценко О. // Інформаційні технології та суспільство, 2 (4). – 2022. – С. 13–19
2. Вивчення React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>
3. Застосування техніки UML «Діаграма варіантів використання» (Use Case Diagram) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ba.in.ua/2024/02/23/zastosuvannya-tehniky-uml-diagrama-variantiv-vykorystannya-use-case-diagram/>
4. Золотухіна О.А. Якість та тестування інформаційних систем / О.А. Золотухіна, О.В. Негоденко, С.Ю. Резник, С.Я. Разіна // Навчальний посібник підготовлено до друку для самостійної роботи студентів вищих навчальних закладів. Київ: ННІТ ДУТ, 2020. – 128 с – Режим доступу до посібника: <https://duikt.edu.ua/ua/lib/1/category/2127/view/2177>
5. ЛУН [Електронний ресурс] – Режим доступу до ресурсу: <https://lun.ua/>
6. OLX [Електронний ресурс] – Режим доступу до ресурсу: <https://www.olx.ua/uk/nedvizhimost/>
7. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>
8. Building a Full-Stack Application with React and Node.js — Best Practices and Tips [Електронний ресурс]. – Режим доступу до ресурсу: <https://medium.com/@nagarjunmallesh/building-a-full-stack-application-with-react-and-node-js-best-practices-and-tips-f9877494fab2>.
9. Costa, L.F. Testing JavaScript Applications / Lucas Fernandes da Costa – Manning Publications, 2021. – 384 с. – ISBN: 9781617297910.
10. DIM.RIA [Електронний ресурс] – Режим доступу до ресурсу: <https://dom.ria.com/uk/>

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

11. How PropTech Real Estate is Changing the Industry [Електронний ресурс]. – Режим доступу до ресурсу: <https://blog.dormakaba.com/how-proptech-real-estate-is-changing-the-industry/>.

12. MapTiler. Documentation Portal [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.maptiler.com/>

13. Miell, I. Docker in Practice, Second Edition [Електронний ресурс] – Режим доступу до ресурсу: <https://www.manning.com/books/docker-in-practice-second-edition>

14. MySQL for Developers — PlanetScale [Електронний ресурс]. – Режим доступу до ресурсу: <https://planetscale.com/learn/courses/mysql-for-developers>.

15. MySQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/mysql-tutorial-excerpt/5.7/en/>

16. Node.js v24.0.2 documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>

17. Nominatim 5.1.0 Manual [Електронний ресурс] – Режим доступу до ресурсу: <https://nominatim.org/release-docs/latest/api/Overview/>

18. Overpass Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://osm-queries.ldodds.com/tutorial/index.html>

19. Poulton, N. The Docker Workshop: Learn how to use Docker containers effectively to speed up the development process – Packt Publishing, Birmingham, UK, 2023 – 792 p.

20. W3Schools. SQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.com/sql/>.

ДОДАТКИ

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізація карти з об'єктами нерухомості

```

const ITEMS_PER_PAGE = 20;

export function AdvertsMap() {
  const screen = useBreakpoint();
  const [openModal, setOpenModal] = useState(false);
  const [selectedRealtyId, setSelectedRealtyId] = useState(null);
  const { searchParams, updateSearchParams } = useSearchParams();

  const map = useRef(null);
  const { removeAllMarkers, createMarkers } = useMapMarkers();
  const hoveredPointId = useRef(null);
  const activePointId = useRef(null);

  const { data: advertsResponse, isLoading, isFetching } = useSearchAdvertsQuery(
    { ...searchParams, limit: ITEMS_PER_PAGE },
    { refetchOnMountOrArgChange: true }
  );

  const { data: advert, isFetching: loadingAdvert } = useGetAdvertByIdQuery(
    selectedRealtyId,
    { skip: !selectedRealtyId, refetchOnMountOrArgChange: true }
  );

  const { total = 0, adverts = [] } = useMemo(() => advertsResponse || {},
    [advertsResponse]);

  const updateFeatureState = (id, states) => {
    Object.entries(states).forEach(([key, value]) => {
      map.current.setFeatureState(
        { source: 'property', id },
        { [key]: value }
      );
    });
  };

  const selectRealty = (id, coordinates) => {
    const selectedMarker = document.querySelector('.property-marker--selected');
    const marker = document.querySelector(`.property-marker[data-marker-id="${id}"]`);

    selectedMarker?.classList?.remove('property-marker--selected');

    if(activePointId.current !== null) {
      updateFeatureState(activePointId.current, { active: false });
    }

    activePointId.current = id;
    setSelectedRealtyId(id);

    if (!screen.md) setOpenModal(true);
  };

```

					ІПЗ.КР.Б-121-25-ІЗ	Арк. 60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

marker.classList.add('property-marker--selected');
updateFeatureState(id, { active: true });

if (coordinates) {
  map.current.flyTo({ center: coordinates });
}
};

const unselectRealty = () => {
  const selectedMarker = document.querySelector('.property-marker--selected');
  selectedMarker?.classList?.remove('property-marker--selected');

  updateFeatureState(activePointId.current, { active: false });
  activePointId.current = null;
  setSelectedRealtyId(null);
}

const handleMapClick = (e) => {
  const target = e.originalEvent.target;
  const isMarkerClick = target?.classList?.contains('property-marker');

  if (!isMarkerClick && activePointId.current !== null) {
    const features = map.current.queryRenderedFeatures(e.point, { layers:
['property'] });

    if (features.length === 0) {
      unselectRealty();

      if (!screen.md) setOpenModal(false);
    }
  }
};

const handlePropertyClick = (e) => {
  const realty = e.features[0];
  selectRealty(realty.id, [e.lngLat.lng, e.lngLat.lat]);
};

const handlePropertyHover = (e, isEntering) => {
  const realty = e.features[0];

  if (hoveredPointId.current !== null) {
    updateFeatureState(hoveredPointId.current, { hover: false });
  }

  if (isEntering) {
    hoveredPointId.current = realty.id;
    updateFeatureState(hoveredPointId.current, { hover: true });
    map.current.getCanvas().style.cursor = 'pointer';
  } else {
    hoveredPointId.current = null;
    map.current.getCanvas().style.cursor = '';
  }
};

```

					ІПЗ.КР.Б-121-25-ІІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const handleMapLoad = (mapInstance) => {
  map.current = mapInstance;

  mapInstance.addSource('property', {
    'type': 'geojson',
    'data': formatSourceFeatures(adverts)
  });

  createMarkers(mapInstance, adverts, selectRealty);

  mapInstance.addLayer({
    id: 'property',
    type: 'circle',
    source: 'property',
    paint: {
      "circle-radius": {
        stops: [[8, 2], [13, 3], [16, 6]]
      },
      'circle-color': [
        "case",
        ["==", ["feature-state", "active"], true],
        "#F4801A",
        "white"
      ],
      "circle-stroke-width": [
        "case",
        ["any",
          ["==", ["feature-state", "hover"], true],
          ["==", ["feature-state", "active"],
        ], true]
      ],
      "circle-stroke-color": "#F4801A"
    }
  });

  mapInstance.on('click', handleMapClick);
  mapInstance.on('click', 'property', handlePropertyClick);
  mapInstance.on('mouseenter', 'property', e => handlePropertyHover(e, true));
  mapInstance.on('mouseleave', 'property', e => handlePropertyHover(e,
false));
};

const closeRealtyPopup = () => {
  setOpenModal(false);
  unselectRealty()
}

useEffect(() => {
  if (map.current) {
    removeAllMarkers();
    map.current.getSource("property").setData(formatSourceFeatures(adverts));
    createMarkers(map.current, adverts, selectRealty);
  }
});

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
  }, [adverts]));

  if (isLoading) return <Loading />;

  return (
    <Space style={{ width: "100%" }} size="middle" direction="vertical">
      <div style={{ position: "relative" }}>
        {selectedRealtyId && !loadingAdvert && (
          !screen.md ? (
            <AdvertModal
              open={openModal}
              onClose={closeRealtyPopup}
              loading={loadingAdvert}
              advert={advert}
            />
          ) : (
            <div className={styles["card-popup"]}>
              <AdvertCard link={` /adverts/${advert.id}`} advert={advert} />
            </div>
          )
        )}
      <Map
        className={styles.map}
        style={{ height: 600 }}
        onLoad={handleMapLoad} />
    </div>
    <Typography.Title
      level={3}
      style={{ marginBottom: 0 }}
    >Оголошення</Typography.Title>
    <div style={{ minHeight: 300 }}>
      {isFetching ? <Loading /> : <AdvertList adverts={adverts} />}
    </div>
    <Pagination
      onChange={(page) => updateSearchParams("page", page)}
      align="center"
      hideOnSinglePage={true}
      defaultCurrent={1}
      defaultPageSize={ITEMS_PER_PAGE}
      total={total}
    />
  </Space>
);
}

export const Map = memo(function Map(props) {
  const { onLoad, options = {}, style = {}, className } = props;
  const mapContainer = useRef(null);
  const map = useRef(null);

  useEffect(() => {
    if (map.current) return;

    map.current = new maptilersdk.Map({
      container: mapContainer.current,
      center: [31.490773812277094, 49.01605198088264],
      zoom: 5.5,
      language: maptilersdk.Language.UKRAINIAN,

```

```

    maxBounds: [
      [21.79545514866079, 43.424418976677686],
      [42.82825058830223, 52.70793206338687],
    ],
    ...options
  });

  if (onLoad) {
    map.current.on('load', () => {
      map.current.setPaintProperty("Building 3D", 'fill-extrusion-opacity',
0.9);

      onLoad(map.current)
    });
  }

  return () => {
    map.current.remove();
    map.current = null;
  }
}, [])

return (
  <div className={className} style={{ width: "100%", borderRadius: 8, ...style
}} ref={mapContainer}/>
)
})

export function useMapMarkers() {
  const markers = useRef([]);

  const removeAllMarkers = () => {
    markers.current.forEach(marker => marker.remove());
    markers.current = [];
  }

  const createMarkers = (map, adverts, onSelect) => {
    adverts.forEach(({ id, locationForAdvert, price_usd }) => {
      const { lat, lon } = locationForAdvert;
      const coordinates = [lon, lat];
      const marker = document.createElement('div');

      marker.className = 'property-marker';
      marker.textContent = formatCurrency(price_usd);
      marker.dataset.markerId = id;

      marker.addEventListener('click', () => {
        onSelect(id, coordinates);
      });

      const markerInstance = new Marker({
        element: marker,
        anchor: "bottom",
        offset: [0, -5]
      })
    })
  }
}

```

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .setLngLat(coordinates)
        .addTo(map);

    markers.current.push(markerInstance);
  });
}

return {
  markers,
  removeAllMarkers,
  createMarkers
};
}

export function formatSourceFeatures(data) {
  const features = data.map(({ id, locationForAdvert: { lat, lon } }) => ({
    id,
    type: 'Feature',
    geometry: {
      type: 'Point',
      coordinates: [lon, lat]
    },
  }));

  return {
    'type': 'FeatureCollection',
    'features': features
  }
}

export function formatCurrency(number) {
  return new
  Intl.NumberFormat("uk-UA", {
    style: "currency",
    currency: "USD",
    currencyDisplay: "narrowSymbol",
    trailingZeroDisplay: "stripIfInteger",
  }).format(number);
}

export function AdvertCard(props) {
  const { like = true, link, advert, actionSlot } = props;
  const {
    id,
    advertImages,
    price_usd,
    area,
    room,
    floor,
    propertyTypeId,
    locationForAdvert,
    advertPropertyTypeForAdvert: {
      name: propertyTypeName,
    }
  } = advert;
}

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const { building_levels } = locationForAdvert;
const locationDisplay = formatLocation(locationForAdvert);

const floorTooltip = propertyTypeId === ADVERT_PROPERTY_TYPES.HOUSE ? "Кількість поверхів" : "Поверх";
const floorDisplay = propertyTypeId === ADVERT_PROPERTY_TYPES.FLAT && building_levels ? `${floor} з ${building_levels}` : floor;

const pricePerMeter = Math.round(price_usd / area);
const imageList = advertImages?.map(({ imageUrl }) => imageUrl);

return (
  <Link to={link}>
    <div className={styles.card}>
      {actionSlot && (
        <Flex className={styles.actions} align="center" gap="small">
          {actionSlot}
        </Flex>
      )}
      <Badge.Ribbon placement="start" text={propertyTypeName}>
        <ImageCarousel images={imageList} />
      </Badge.Ribbon>
      <div className={styles["card-description"]}>
        <Flex gap="small" align="center">
          <Typography.Text strong style={{ fontSize: 24 }}>{price_usd}</Typography.Text>
          <Typography.Text>{pricePerMeter} $ за м²</Typography.Text>
          {like && <AdvertLikeBtn advertId={id}/>}
        </Flex>
        <Typography.Text style={{ display: "flex", height: 50 }}>{locationDisplay}</Typography.Text>
        <Space
          styles={{ item: { flex: 1 } }}
          style={{ width: "100%", marginTop: 16 }}
          split={<Divider className={styles.divider} type="vertical" />}
        >
          {room && (
            <Tooltip size="small" title="Кімнати" color="orange">
              <Flex justify="center" align="center" gap="small">
                <MdOutlineBed color="var(--ant-orange-6)" size={20}/>
                {room}
              </Flex>
            </Tooltip>
          )}
          <Tooltip size="small" title="Площа" color="orange">
            <Flex justify="center" align="center" gap="small">
              <RiCustomSize color="var(--ant-orange-6)" size={20}/>
              {area} м²
            </Flex>
          </Tooltip>
          {floor && (
            <Tooltip size="small" title={floorTooltip} color="orange">
              <Flex justify="center" align="center" gap="small">
                <PiStairs color="var(--ant-orange-6)" size={20}/>

```

```

        {floorDisplay}
      </Flex>
    </Tooltip>
  )}
</Space>
</div>
</div>
</Link>
)
}

export function AdvertModal({ open, onClose, advert, loading }) {
  return (
    <Modal
      className={styles.modal}
      classNames={{ content: styles["modal-content"] }}
      open={open}
      onCancel={onClose}
      loading={loading}
      footer={null}
    >
      <AdvertCard link={`/adverts/${advert.id}`} advert={advert} />
    </Modal>
  );
}

export function AdvertsFilter() {
  const [open, setOpen] = useState(false);
  const { searchParams, updateSearchParams } = useSearchParams();
  const { propertyTypeId, minPriceUsd, maxPriceUsd, room, city, minArea, maxArea,
benefits: benefitsParams } = searchParams;
  const benefitsValue = benefitsParams?.split(",").map(Number);

  const setMinPriceUsd = useDebounce(value => updateSearchParams("minPriceUsd",
value), 500);
  const setMaxPriceUsd = useDebounce(value => updateSearchParams("maxPriceUsd",
value), 500);
  const setCity = useDebounce(value => updateSearchParams("city", value), 500);
  const setRoom = useDebounce(value => updateSearchParams("room", value), 500);
  const setMinArea = useDebounce(value => updateSearchParams("minArea", value),
500);
  const setMaxArea = useDebounce(value => updateSearchParams("maxArea", value),
500);

  const { data: propertyTypes = [], isLoading: propertyTypesLoading } =
useGetAdvertPropertyTypesQuery();
  const { data: benefits = [], isLoading: benefitsLoading } =
useGetBenefitsQuery();

  const propertyTypeOptions = useMemo(() => propertyTypes.map(({ id, name }) =>
({
    value: id,
    label: name,
  })), [propertyTypes]);

  const benefitsOptions = useMemo(() => benefits.map(({ id, name }) => ({

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    value: id,
    label: name,
  })), [benefits]);

if (propertyTypesLoading || benefitsLoading) return null;

return (
  <Flex justify="end" style={{ marginBottom: 24 }}>
    <Button
      style={{ fontWeight: "bold" }}
      type="primary"
      size="large"
      icon={<IoOptions size={24} />}
      onClick={() => setOpen(true)}
    >
      Фільтри
    </Button>
    <Drawer title="Фільтри" onClose={() => setOpen(false)} open={open}>
      <Space style={{ width: "100%" }} size="middle" direction="vertical">
        <div>
          <Typography.Paragraph style={{ marginBottom: 4 }} strong>Вид
непсихомості</Typography.Paragraph>
          <Select
            style={{ width: "100%" }}
            defaultValue={propertyTypeId ? Number(propertyTypeId) : null}
            options={propertyTypeOptions}
            onSelect={value => updateSearchParams("propertyTypeId", value)}
            allowClear
          />
        </div>
        <div>
          <Typography.Paragraph style={{ marginBottom: 4 }}
strong>Місто</Typography.Paragraph>
          <Input
            style={{ width: "100%" }}
            defaultValue={city}
            onChange={e => setCity(e.target.value)}
          />
        </div>
        <div>
          <Typography.Paragraph style={{ marginBottom: 4 }}
strong>Ціна</Typography.Paragraph>
          <NumberRange
            defaultMinValue={minPriceUsd}
            defaultMaxValue={maxPriceUsd}
            setMinValue={setMinPriceUsd}
            setMaxValue={setMaxPriceUsd}
            fromInputProps={{ suffix: "$" }}
            toInputProps={{ suffix: "$" }}
          />
        </div>
        <div>
          <Typography.Paragraph style={{ marginBottom: 4 }} strong>Кількість
кімнат</Typography.Paragraph>
          <InputNumber

```

					ІПЗ.КР.Б-121-25-ІПЗ	Арк. 68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        style={{ width: "100%" }}
        min={1}
        defaultValue={room}
        onChange={setRoom}
      />
    </div>
    <div>
      <Typography.Paragraph style={{ marginBottom: 4 }}>
        strong>Площа</Typography.Paragraph>
        <NumberRange
          defaultMinValue={minArea}
          defaultMaxValue={maxArea}
          setMinValue={setMinArea}
          setMaxValue={setMaxArea}
          fromInputProps={{ suffix: "м²" }}
          toInputProps={{ suffix: "м²" }}
        />
      </div>
      <div>
        <Typography.Paragraph style={{ marginBottom: 4 }}>
          strong>Переваги</Typography.Paragraph>
          <Select
            mode="multiple"
            style={{ width: "100%" }}
            defaultValue={benefitsValue}
            options={benefitsOptions}
            onChange={value => updateSearchParams("benefits", value)}
            allowClear
          />
        </div>
      </Space>
    </Drawer>
  </Flex>
)
}

```

```

const getInformationByData = async (lat, lon) => {
  try {
    const data = await OSMDDataFetcher.getJsonData(lat, lon);
    const { address, buildingDetails, display_name, type, class: osmClass, type: osmType } = data;

    return {
      lat,
      lon,
      addresstype: data.addresstype || null,
      class: osmClass || null,
      type: type || null,
      building: buildingDetails?.building || null,
      city: address?.city || null,
      district: address?.district || null,
      state: address?.state || null,
      country: address?.country || null,
    };
  } catch (error) {
    console.error('Error fetching data:', error);
    return null;
  }
};

```

```

        house_number: address?.house_number || null,
        road: address?.road || null,
        quarter: address?.quarter || null,
        suburb: address?.suburb || null,
        display_name: display_name || null,
        borough: address?.borough || null,
        postcode: address?.postcode || null,
        building_levels: buildingDetails?.['building:levels'] || null
    };
} catch (error) {
    throw new Error('Не вдалося отримати локацію: ' + error.message);
}
};

```

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

Робота з координатами

```

const axios = require('axios');
class OSMDDataFetcher {
  constructor() {
    this.overpassUrl = 'https://overpass-api.de/api/interpreter';
    this.nominatimUrl = 'https://nominatim.openstreetmap.org/reverse';
    this.osrmUrl = 'https://router.project-osrm.org/route/v1/walking';
  }

  async getAddressDetails(lat, lon) {
    try {
      const nominatimResponse = await axios.get(`${this.nominatimUrl}`, {
        params: {
          format: 'json',
          lat: lat,
          lon: lon,
          addressdetails: 1
        },
        headers: {
          'User-Agent': 'NodeJS OSM Data Fetcher'
        }
      });

      const buildingDetails = await this.getBuildingDetails(lat, lon);

      return {
        ...nominatimResponse.data,
        buildingDetails
      };
    } catch (error) {
      console.error('Error fetching address details:', error.message);
      throw error;
    }
  }

  async getBuildingDetails(lat, lon) {
    const radius = 10;
    const query = `
      [out:json][timeout:25];
      (
        way(around:${radius},${lat},${lon})["building"];
        relation(around:${radius},${lat},${lon})["building"];
        way(around:${radius},${lat},${lon})["amenity"];
        relation(around:${radius},${lat},${lon})["amenity"];
      );
      out body;
      >;
      out skel qt;
    `;

    try {
      const response = await axios.post(this.overpassUrl, query, {

```

					ІПЗ.КР.Б-121-25-ІІЗ	Арк. 71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        headers: {
          'Content-Type': 'text/plain',
        }
      });

      const buildingData = response.data.elements.find(element =>
        element.tags
      );

      return buildingData ? buildingData.tags : null;
    } catch (error) {
      console.error('Error fetching building details:', error.message);
      return null;
    }
  }

  async getRoute(startLat, startLon, endLat, endLon) {
    try {
      const routeResponse = await
        axios.get(`${this.osrmUrl}/${startLon},${startLat};${endLon},${endLat}`, {
          params: {
            overview: 'full',
            geometries: 'geojson'
          }
        });

      return routeResponse.data.routes[0];
    } catch (error) {
      console.error('Error fetching route:', error.message);
      throw error;
    }
  }

  static async getJsonData(lat, lon) {
    const fetcher = new OSMDDataFetcher();
    return await fetcher.getAddressDetails(lat, lon);
  }
}

module.exports = OSMDDataFetcher

```

Створення місця поблизу

```
const createNearbyPlace = async (body) => {
  try {
    const { lat, lon} = body;
    const location = await locationRepository.createLocation(lat, lon);
    const data = await OSMDDataFetcher.getJsonData(lat, lon);

    const nearbyPlaceData = {
      locationId: location.id,
      opening_hours: data.buildingDetails?.opening_hours || null,
      website: data.buildingDetails?.website || null,
      phone: data.buildingDetails?.phone || null,
      brand: data.buildingDetails?.brand || data.address?.amenity || null,
      email: data.buildingDetails?.email || null,
      name: data.name || data.address?.amenity || null,
    };

    return await NearbyPlaces.create(nearbyPlaceData);
  } catch (error) {
    throw new Error('Не вдалося створити місце поблизу: ' + error.message);
  }
};
```