МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему:

«Обґрунтування вибору технологій клієнт-серверної розробки веб-додатку для агенства нерухомості»

Виконав: студент 6 курсу
спеціальності 126
«Інформаційні системи та технології»

Хемич Р.Й.
(прізвище та ініціали)

Керівник: Желєзняк А.М.
(прізвище та ініціали)

ДУБЛЯНИ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти другий (магістерський)
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Хемич Роман Йосипович
(прізвище, ім’я, по батькові)

1. Тема роботи *«*Обґрунтування вибору технологій клієнт-серверної розробки веб-додатку для агенства нерухомості*»*
керівник роботи *к. е н., доцент., Желєзняк А.М.*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП № 616/к-с від 12.09.2024

2. Строк подання студентом роботи 2.12.2024 р.

3. Вихідні дані: *аналітичні дані роботи та характеристика об’єкту дослідження, опис бібліотек мов програмування, науково-технічна і довідкова література.*

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз стану питання в теорії та практиці та постановка завдання
2. Обґрунтування, вибір та реалізація інструментарію вирішення задачі
3. Результати вирішення задачі
4. Охорона праці та безпека в надзвичайних ситуаціях
5. Визначення ефективності
Висновки
Бібліографічний список

5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 5	*Желєзняк А.М., доцент кафедри інформаційних технологій*			
4	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*			

7. Дата видачі завдання *1.03.2024*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Отримання завдання. Вивчення рекомендованої літератури по темі роботи. Написання першого розділу*	*01.03.2024 – 31.05.2024*	
2	*Проектування та опис технічного завдання, обґрунтування та вибір інструментарію реалізації проекту (написання другого розділу).*	*01.06.2024 – 31.08.2024*	
3	*Програмна реалізація поставленого завдання (написання третього розділу)*	*01.09.2024 – 18.10.2024*	
4	*Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»*	*19.10.2024 – 04.11.2024*	
5	*Оцінка ефективності поставленого завдання (виконання п'ятого розділу)*	*05.11.2024 – 18.11.2024*	
6	*Завершення оформлення основної частини, написання висновків та підготовка презентаційного матеріалу*	*19.11.2024 – 02.12.2024*	
7	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*03.12.2024 – 16.12.2024*	

Студент _____ *Хемич Р.Й.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Желєзняк А.М.*
(підпис) (прізвище та ініціали)

УДК 004.775:347.2

Обґрунтування вибору технологій клієнт-серверної розробки веб-додатку для агентства нерухомості.

Хемич Р.Й. Кафедра інформаційних технологій - Дубляни, Львівський НУП, 2024.

Кваліфікаційна робота: 45 с. текст. част., 19 рис., 3 табл., 22 джерела, 4 додатки

Наведено теоретичні основи технологій клієнт-серверної розробки для створення веб-додатків. Проведено огляд технологій та мов програмування, які користуються попитом при розробці таких додатків.

Обґрунтовано перспективи застосування клієнт-серверних технологій в розробці веб-додатків для агентств нерухомості для вдосконалення процесу відбору та презентації кінцевому користувачеві – потенційному клієнту.

Запропоновано методи покращення продуктивності клієнт-серверного коду

Здійснено обґрунтування та вибір стеку технологій для розробки веб-додатку, визначені їх переваги та недоліки. Обґрунтована потреба в застосуванні інших бібліотек. Проаналізовано перспективи розвитку додатків в даній сфері застосування.

Здійснено аналіз травматичних ситуацій при виконанні різних робіт у сфері використанням комп’ютерної техніки, викладено питання охорони праці.

Ключові слова: веб-додаток, веб-програмування, клієнт-серверна архітектура.

ЗМІСТ

ВСТУП

Щороку інтернет-технології все більше стають частиною нашого щоденного життя. У сучасних умовах інтернет-ресурси дозволяють швидко отримувати різноманітні послуги — від освіти та медичних консультацій до банківських і державних сервісів — прямо з дому.

Метою цієї кваліфікаційної роботи є вивчення й обґрунтування ефективних технологій для створення та тестування клієнт-серверних веб-додатків у сфері нерухомості. Веб-додатки для агентств нерухомості можуть стати ключовим інструментом для зручного пошуку житла, перегляду оголошень та швидкого зв’язку з ріелторами. Такі додатки також допомагають автоматизувати процеси, що спрощує роботу агентів і робить послуги агентства доступнішими для більшої кількості користувачів.

Тема створення веб-додатку для агентства нерухомості є актуальною, оскільки сучасний ринок нерухомості потребує інноваційних цифрових рішень для автоматизації бізнес-процесів та поліпшення взаємодії з клієнтами. Готові платформи не завжди враховують специфічні потреби окремих агентств і їхніх клієнтів, що створює попит на розробку індивідуальних рішень. Такий підхід дозволяє забезпечити унікальні конкурентні переваги, гнучкість і масштабованість, що є важливими для ефективного функціонування агентства на динамічному ринку.

В ході виконання кваліфікаційної роботи були поставлені наступні завдання:

1. Дослідити теоретичні та методологічні аспекти, основні концепції та підходи вибору клієнт-серверної архітектури при побудові веб-додатків.
2. Розглянути основні технології розробки веб-додатків агентств нерухомості.
3. Здійснити вибір інструментів, технологій та мов програмування для вирішення поставленої задачі.

4. Проаналізувати стан охорони праці та безпеки в надзвичайних ситуаціях для ІТ-компаній.

5. Оцінити ефективність інструментів розробки веб-додатків агентств нерухомості.

Загалом дипломна робота складається з п’яти розділів. В першому розділі досліджені теоретичні аспекти вибору клієнт-серверної архітектури при побудові веб-додатків. Проаналізовано сучасні рішення щодо автоматизації роботи агентств нерухомості. У другому розділі розглянули та обрали основні технології для розробки клієнтської та серверної частини. Також ми спроектували структуру додатку та взаємодії компонентів. Третій розділ присвячено реалізації поставленої задачі та висвітлено результати роботи. У четвертому розділі висвітлені питання охорони праці. В п’ятому розділі провели оцінку ефективності використаних інструментів розробки.

Наукова новизна даної кваліфікаційної роботи полягає у розробці індивідуального підходу до створення веб-додатку для агентства нерухомості, який враховує сучасні технологічні тенденції, особливості українського ринку нерухомості та специфічні потреби клієнтів і агентств. Дослідження включає аналіз існуючих рішень, огляд основних недоліків та переваг, огляд можливості впровадження автоматизації бізнес-процесів, персоналізований користувацький досвід і підвищення рівня безпеки даних. Це дозволяє запропонувати унікальне рішення з можливістю масштабування, яке відповідатиме сучасним вимогам бізнесу.

Під час практичної підготовки та виконання кваліфікаційної роботи результати були апробовані на Міжнародній студентській науковій конференції «Студентська молодь і науковий прогрес в АПК», матеріали якої були опубліковані у вигляді тез:

1. Хемич Р.Й. Огляд технологій клієнт-серверної розробки. Тези доповідей Міжнародного студентського наукового форуму «Студентська молодь і науковий прогрес» 4-6 жовтня 2024 р., м.Львів / Львів.нац. ун-т природ. Львів, 2024.С.348.

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ПИТАННЯ В ТЕОРІЇ ТА ПРАКТИЦІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Теоретичні аспекти та підходи до вибору клієнт-серверної архітектури при розробці веб-додатків

Клієнт-серверна модель [1] є основою для побудови багатьох веб-додатків і передбачає розподіл процесів між двома основними компонентами: клієнтом і сервером. У цій архітектурі клієнт (користувацький інтерфейс) надсилає запити на сервер, який обробляє ці запити, виконує необхідні операції і повертає клієнту потрібні дані або відповіді. Такий підхід сприяє розподілу завдань, спрощує роботу системи та дозволяє використовувати різні платформи для клієнтів і серверів. Це полегшує управління ресурсами та забезпечує більш раціональне використання обчислювальної потужності. Це полегшує розподілення ресурсів та забезпечує більш ефективне використання обчислювальної потужності.

На рисунку 1.1 показано загальну структуру клієнт-серверної архітектури.

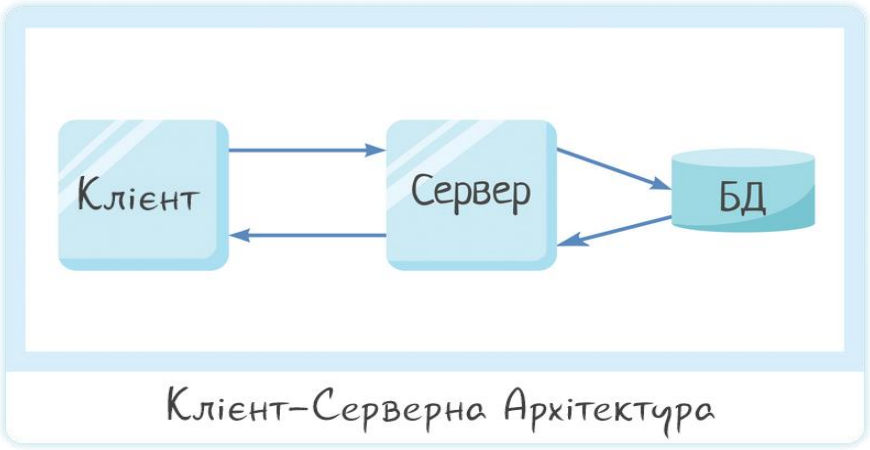


Рисунок 1.1 – Структура клієнт-серверної архітектури

Клієнт-серверна архітектура [2] широко використовується в різних галузях, таких як веб-розробка, управління базами даних, обмін

повідомленнями, розробка ігор тощо. Ця модель дозволяє створювати масштабовані та продуктивні системи, які задовольняють різні потреби користувачів. До того ж, клієнт-серверна архітектура підвищує безпеку системи, адже сервер забезпечує контроль доступу до ресурсів і може встановлювати права користувачів, централізовано керуючи захистом даних і функціоналом. Це є важливим для галузей [3], де критично важлива конфіденційність і захист від несанкціонованого доступу. До того ж, клієнт-серверні додатки можуть оновлюватися централізовано, що спрощує керування версіями та впровадження нових функцій, роблячи такі системи більш гнучкими та легкими в адмініструванні.

Отже можемо підсумувати область відповідальності клієнта і сервера:

Сервер відповідає за:

- зберігання, доступ, захист і резервне копіювання даних;
- обробка клієнтського запиту;
- відправлення результату (відповіді) клієнту.

Клієнт відповідає за:

- надання користувальницького інтерфейсу;
- формулювання запиту до сервера і його відправка;
- отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

Клієнт-серверна архітектура може мати кілька типів, кожен з яких має свої особливості, переваги та недоліки [4]. Вибір оптимального типу залежить від специфіки веб-додатку, кількості користувачів, складності функціоналу та вимог до продуктивності. Можна виділити такі основні типи клієнт-серверної архітектури :

- монолітна архітектура
- дворівнева архітектура
- трирівнева архітектура
- мікросервісна архітектура

Монолітна архітектура підходить для невеликих веб-додатків або систем із невисокими вимогами до продуктивності оскільки всі компоненти системи, включаючи інтерфейс, бізнес-логіку та базу даних, об'єднані в єдину структуру, яка функціонує як один блок. З одного боку це дає простоту розробки та легкість в тестуванні і впровадженні, проте з іншого боку таку систему важко масштабувати.

Дворівнева архітектура, яка складається з клієнта та сервера, теж може бути корисною при розробці невеликих проектів, оскільки потребує мінімальних ресурсів і є зручною для швидкої розробки. Як і монолітна цей вид архітектури не позбавлений проблем з масштабованістю та проблем, які з'являться при навантаженні на веб-додаток.

Трирівнева архітектура підходить для веб-додатків середньої складності з потребою у швидкому доступі до даних, які мають стабільну структуру, наприклад, базові системи для огляду об’єктів нерухомості. Вона складається з трьох основних шарів: інтерфейсу користувача (клієнт), сервера(-ів) та бази даних, часто додатково встановлюють балансувальника, задача якого вирішити на який з серверів (якщо їх кілька) слати запит. Схема даної архітектури зображена на рисунку 1.2.

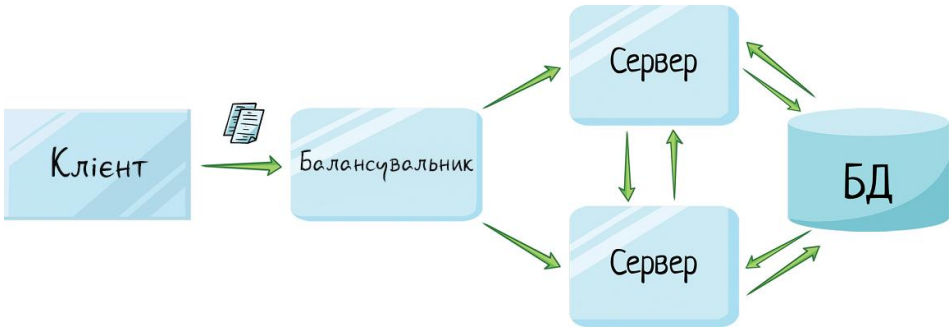


Рисунок 1.2 – Трирівнева архітектура клієнт-серверних додатків

Мікросервісна архітектура [5] є повною протилежністю до монолітної архітектури і який передбачає розділення додатку на декілька незалежних,

спеціалізованих сервісів. Основним недоліком даної архітектури є складність налаштування, що потребує додаткових ресурсів на адміністрування.

Мікросервіси – це невеликі, автономні сервіси, які виконують специфічні функції додатка та спілкуються між собою за допомогою API. Схема мікросервісної архітектури зображена на рисунку 1.3.

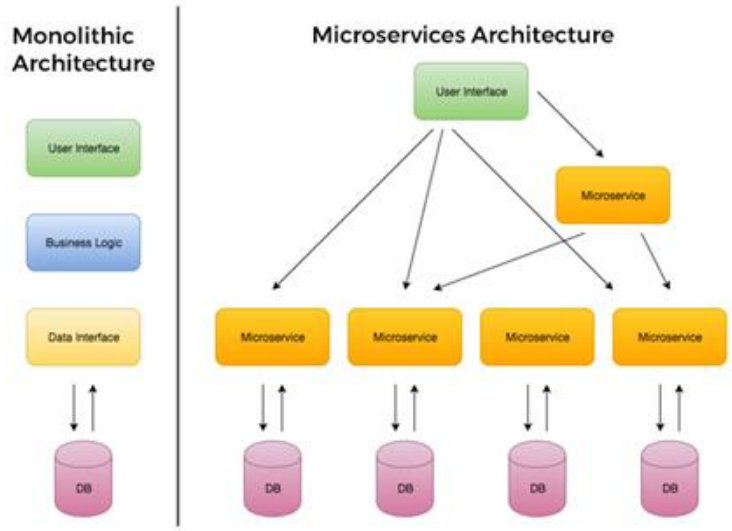


Рисунок 1.3 – Приклад мікросервісної архітектури застосунку

До переваг можна сміло віднести високу масштабованість, можливість незалежного оновлення кожного сервісу, краща стійкість до збоїв. Проте даний вид архітектури також не позбавлений і недоліків, а саме більшої складності управління, потреби у спеціалізованій інфраструктурі для обробки мікросервісів.

1.2 Особливості застосування технологій клієнт-серверної архітектури

В сучасному світі цифрових технологій клієнт-серверна архітектура стала невід'ємною частиною нашого повсякденного життя, хоча більшість користувачів навіть не замислюються про її існування. Щодня мільйони людей користуються веб-сайтами, мобільними додатками та іншими цифровими сервісами, які працюють саме за цим принципом. Про це також написав

Мартін Фаулер у своїй фундаментальній праці "Patterns of Enterprise Application Architecture" [6], де він підкреслює, що саме ця архітектурна модель стала основою для розвитку сучасного інтернету та цифрових комунікацій.

Історично склалося так, що розвиток клієнт-серверної архітектури тісно пов'язаний з еволюцією самого інтернету. На початку свого становлення веб-додатки були досить простими, з мінімальною інтерактивністю та обмеженим функціоналом. Еріх Гамма та його колеги у своїй роботі "Design Patterns" [7] описують цікаву трансформацію - від простих статичних сторінок до складних багаторівневих систем, які ми використовуємо сьогодні. Кожен етап цієї трансформації додавав нові можливості та ускладнював взаємозв'язки між компонентами. На рисунку 1.4 представлено загальну схему, яка ілюструє ключові елементи клієнт-серверної архітектури та демонструє, як відбувається обмін даними між різними рівнями системи.

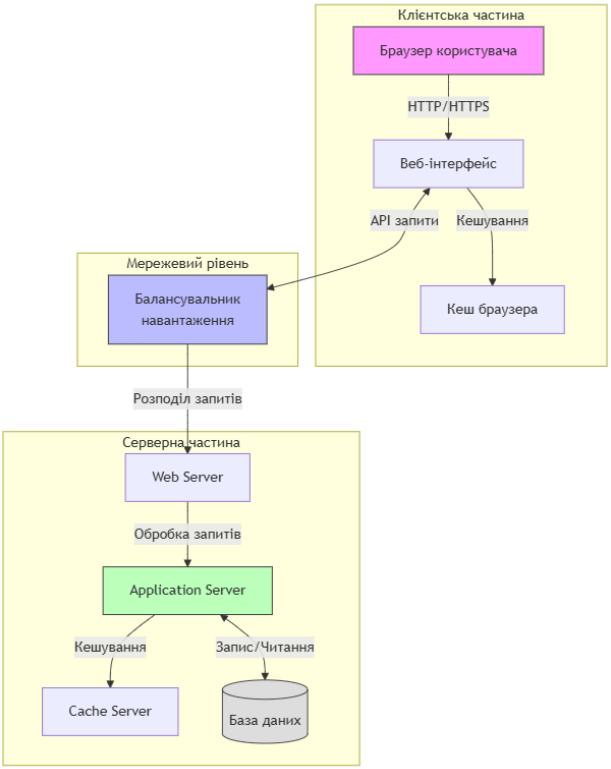


Рисунок 1.4 – Схема взаємодії компонентів та потоків даних у веб-додатку

Революційним також можна відзначити і інший підхід до організації обміну даними між клієнтом та сервером, а саме архітектурний стиль REST, який розробив Рой Філдінг. У своїй дисертації він описував REST як простий та елегантний спосіб організації взаємодії між компонентами системи. З того часу він став одним із найбільш широко використовуваних підходів для створення веб інтерфейсів API. Варто зазначити, що REST - це не просто технічний протокол, а цілісна філософія проектування розподілених систем, що базується на простих, але потужних принципах.

REST API — це один із найпопулярніших типів API [8], який дозволяє здійснювати обмін даними між клієнтом і сервером за допомогою стандартних HTTP-методів, таких як GET, POST, PUT і DELETE. REST API забезпечує простий і зрозумілий механізм для реалізації автентифікації та доступу до ресурсів (рисунок 1.5). Кожен такий метод виконує чітко визначену дію: GET отримує дані, POST створює нові, PUT оновлює існуючі, а DELETE видаляє ресурси. Простота та передбачуваність REST API робить його надзвичайно зручним для розробників, дозволяючи легко інтегрувати різні системи та сервіси. Наприклад, з використанням протоколу OAuth 2.0 через REST API можна реалізувати безпечну аутентифікацію, що дозволяє користувачам надавати доступ до своїх даних без необхідності розкривати облікові дані.

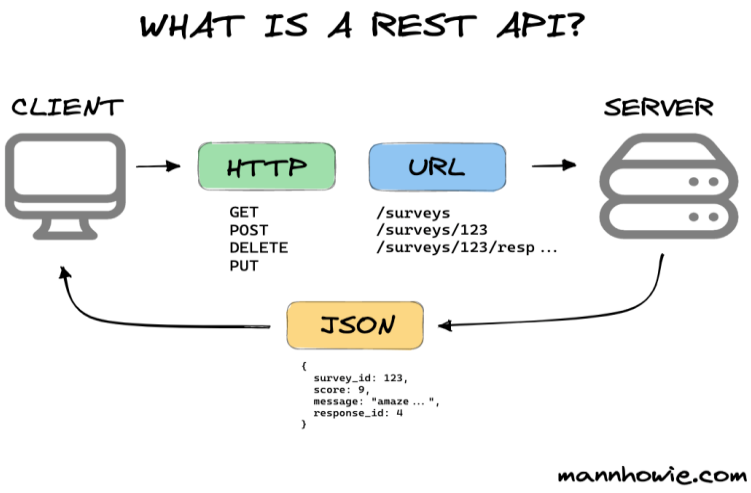


Рисунок 1.5 – Схема роботи REST API.

Варто зазначити, що REST API використовує стандартні HTTP-коди статусу для комунікації результатів запиту. Наприклад:

- 200 OK - успішний запит
- 400 Bad Request - помилка у запиті
- 401 Unauthorized - проблеми з автентифікацією
- 404 Not Found - ресурс не знайдено

Завдяки цим особливостям, REST API став фактичним стандартом для побудови сучасних веб-сервісів та мікросервісних архітектур.

Іншим типом API, що набирає популярність є GraphQL. Він надає гнучкість у виборі даних, які необхідно отримати, що дозволяє зменшити обсяг запитів і покращити продуктивність. GraphQL [9] може також бути використаний для реалізації автентифікації та безпеки даних, використовуючи токени, такі як JSON Web Tokens (JWT), для підтвердження особи користувача та контролю доступу до ресурсів.

Для автентифікації користувачів часто використовують JWT, які забезпечують безпечний механізм для передачі інформації про аутентифікацію. JWT дозволяє клієнтам підтверджувати свою особу без повторного введення паролів, що підвищує зручність та безпеку використання веб-додатка.

Основною відмінністю між цими двома широко відомими API [10], як показано на рисунку 1.6, є додатковий шар резолвер - GraphQL Server, що піклується про створення API та обробку запитів від клієнта та надсилає відповідну відповідь. Сервер GraphQL – це місце, де розробник серверної частини проектуватиме та створюватиме схему GraphQL, API та функції резолвера. Функція резолвера розпізнає значення для типу або поля в схемі GraphQL. Він може повертати об’єкти або скаляри.

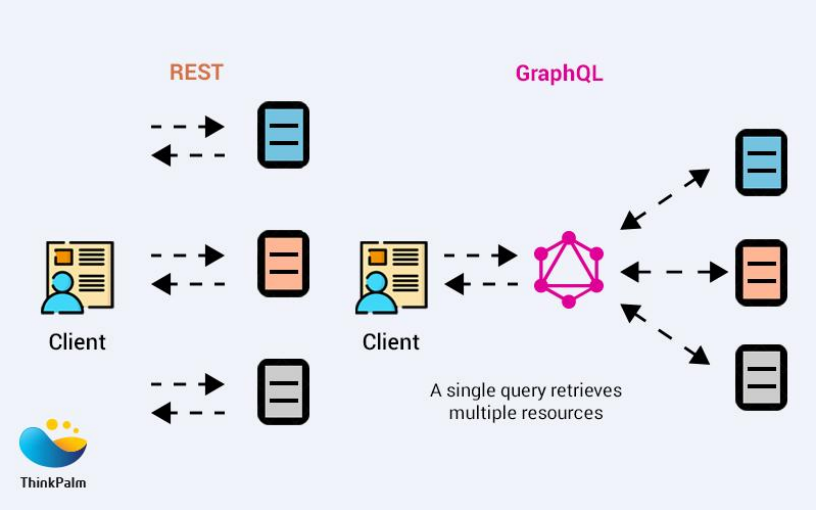


Рисунок 1.6 – Порівняння структури роботи REST API і GraphQL API.

Ще однією революційною технологією стала поява технології WebSocket. Вона змінила наше розуміння взаємодії між клієнтом та сервером. Якщо раніше комунікація була одно направленою (клієнт запитує - сервер відповідає), то WebSocket [11] дозволив створювати по-справжньому інтерактивні додатки з миттєвим оновленням даних. Це особливо важливо для таких додатків як онлайн-чати, біржові термінали чи системи моніторингу в реальному часі.

Питання безпеки в клієнт-серверних системах заслуговує особливої уваги. Брюс Шнайєр, якого часто називають "гуру криптографії", у своїй книзі "Applied Cryptography" наголошує на тому, що безпека - це не продукт, а процес. Він порівнює захист інформаційних систем з будівництвом фортеці, де кожен елемент повинен бути ретельно продуманий та укріплений. Особливо цікавим є його погляд на те, що найслабшою ланкою в системі безпеки часто є не технічні засоби, а людський фактор. Для посилення безпеки часто використовують інструменти шифрування, такі як SSL/TLS, які забезпечують захищений зв'язок між браузером користувача та сервером. Ці протоколи шифрують дані під час передачі, запобігаючи їх перехопленню зловмисниками.

У контексті користувацького інтерфейсу варто згадати революційні ідеї Ітана Маркотта щодо відзивчивого дизайну. Його підхід змінив спосіб, яким

ми думаємо про веб-дизайн. Замість створення окремих версій сайту для різних пристроїв, він запропонував концепцію "адаптивного" дизайну, який адаптується під будь-який розмір екрану. Це особливо актуально сьогодні, коли кількість різних пристроїв для доступу до інтернету постійно зростає.

Мікросервісна архітектура, детально описана Семом Ньюменом, стала логічним розвитком клієнт-серверного підходу. Ньюмен порівнює традиційні монолітні додатки з величезними круїзними лайнерами - вони потужні, але неповороткі. Натомість, мікросервіси він порівнює з флотилією маленьких човнів - кожен відповідає за свою функцію, і разом вони утворюють гнучку та адаптивну систему.

Контейнеризація, яку активно просуває Келсі Хайтауер, додала новий вимір до розгортання клієнт-серверних додатків. Використання контейнерів можна порівняти з революцією, яку здійснили стандартизовані транспортні контейнери в логістиці - вони забезпечили універсальний спосіб упаковки та транспортування вантажів. Аналогічно, Docker-контейнери забезпечують універсальне середовище для запуску додатків.

Підсумовуючи, можна сказати, що клієнт-серверна архітектура продовжує еволюціонувати, адаптуючись до нових вимог та можливостей. Роберт Мартін у своїй книзі "Clean Architecture" [12] підкреслює важливість гнучкості архітектури, її здатності до змін, адже єдине, що залишається незмінним у світі технологій - це постійні зміни. Розуміння принципів та особливостей клієнт-серверної архітектури дозволяє створювати системи, які не тільки відповідають сучасним вимогам, але й готові до викликів майбутнього.

1.3. Огляд існуючих рішень у сфері автоматизації роботи агентств нерухомості

Виходячи з потреб ринку, можемо узагальнити, що веб-додаток повинен забезпечувати можливість організовувати каталог об'єктів з можливістю

додавання, редагування, видалення об’єктів з інтеграцією карт для локації, особистий кабінет користувача для збереження обраних об’єктів і перегляду історії заявок, система пошуку з фільтрацією за параметрами об’єктів, аналітика та звітність для відстеження ефективності, а також система сповіщень для інформування про нові об'єкти та важливі події (підписка).

Український ринок нерухомості за останні роки зазнав суттєвої цифрової трансформації. Стрімкий розвиток інформаційних технологій спонукав появу потужних онлайн-платформ, які кардинально змінили підхід до пошуку та продажу нерухомості. Серед найбільш популярних CRM систем для агентств нерухомості в Україні є LUN.UA, DOM.RIA, OLX Нерухомість, , кожна з яких має свої унікальні особливості та підходи до вирішення потреб користувачів.

Широковідома LUN.UA більш підходить для висвітлення даних про об’єкти, ніж на автоматизацію роботи агентства. Платформа позиціонує себе як сервіс, орієнтований насамперед на первинний ринок нерухомості і відрізняється своїм інноваційним підходом до візуалізації об'єктів нерухомості. Хоча зараз почали розробку частини і для вторинного ринку. Варто відзначити систему фільтрації новобудов за такими параметрами, як розтермінування від забудовника так і наявність державні програми кредитування, таких як “ЄОселя” та “ЄВідновлення” (рисунок 1.7).

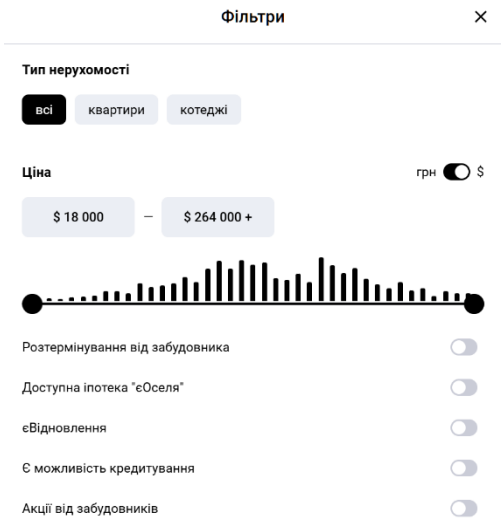


Рисунок 1.7 - Приклад системи фільтрації за програмами кредитування.

Розробники приділили особливу увагу якості фотоматеріалів та зручності перегляду планувань квартир. Цікавим рішенням стала можливість віртуального туру житловими комплексами, що особливо актуально в умовах, коли особистий перегляд об'єктів може бути ускладнений. На рисунку 1.8 показано 3D-візуалізацію квартири. Клієнтську частина сайту пропрацьовано добре, є зручна система фільтрів, інтуїтивний і зручний інтерфейс, який спрощує пошук об'єктів за різними параметрами, такими як район, ціна, площа. Також всі об’єкти можна вивести на карті, що дозволяє аналізувати кількість оголошень і ціни в певному районі. Також є перевірка оголошень на достовірність фото та інших даних, що підвищує довіру користувачів.

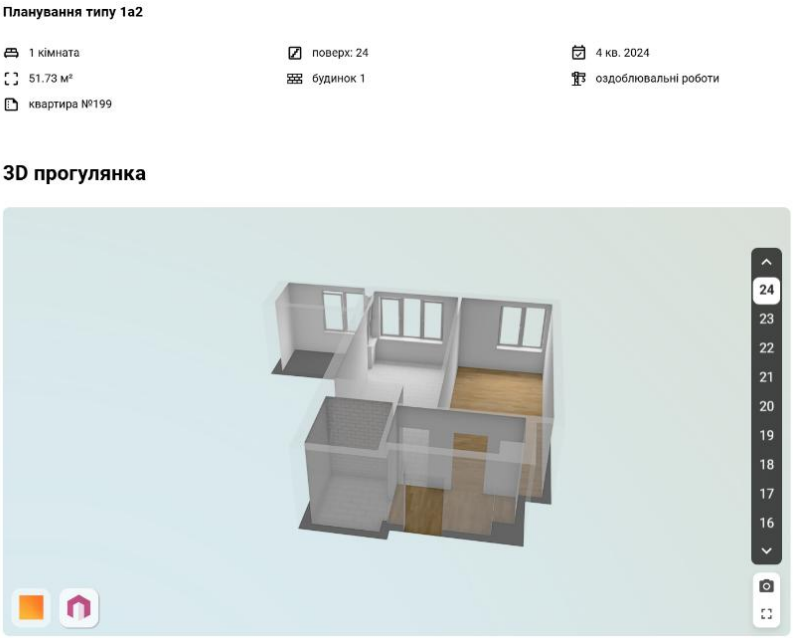


Рисунок 1.8 – 3D-візуалізацію квартири

DOM.RIA має розширений набір інструментів для агентств, включаючи CRM-функції для управління об’єктами та клієнтами, що полегшує ведення бізнесу (рисунок 1.9). Це дозволяє працювати з об’єктами нерухомості комплексно, обмінюючись даними між вашою системою і їх CRM системою за допомогою XML фідів, які звісно потрібно буде ще налаштувати з сового боку.

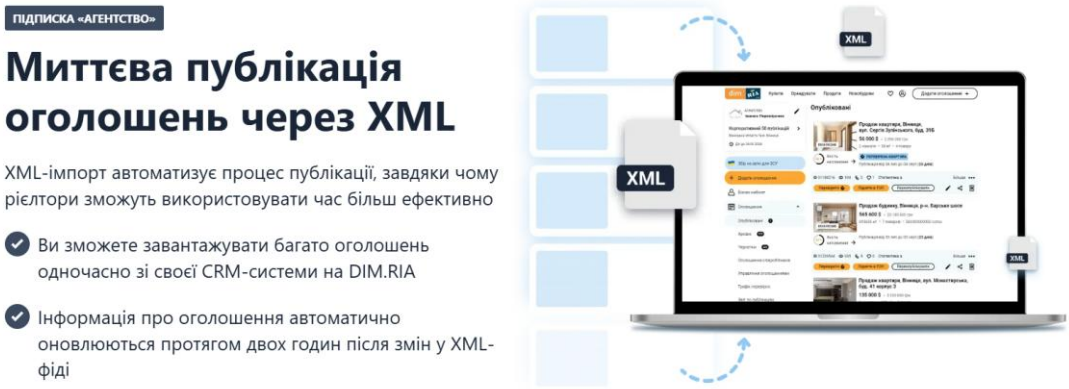


Рисунок 1.9 – Одна з переваг кабінету ріелтора чи агентства нерухомості на Dom.ria

На відміну від свого конкурента, охоплює значно ширший спектр ринку нерухомості, включаючи як первинний, так і вторинний сегменти. Має зручну систему фільтрів і налаштування пошуку, які дозволяють користувачам швидко знаходити потрібні пропозиції. Користувачі мають змогу налаштовувати десятки параметрів, починаючи від базових, таких як ціна та розташування, і закінчуючи специфічними характеристиками житла (рисунок 1.10).

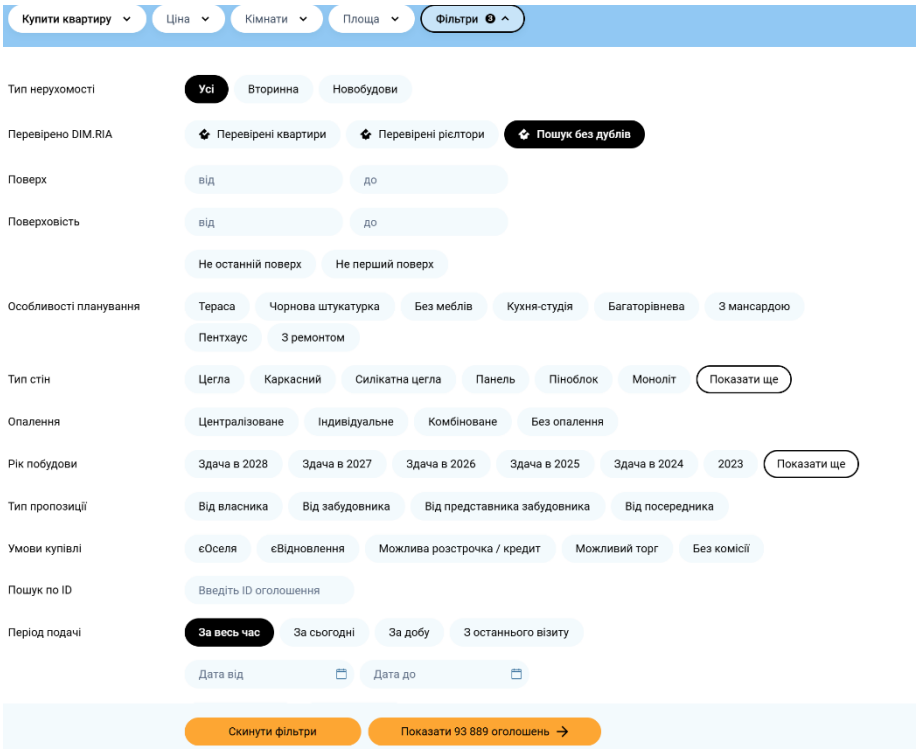


Рисунок 1.10 – Система фільтрації DOM.RIA

Останнім відомим рішенням, яке користується популярністю є OLX Нерухомість, що представляє дещо інший підхід до організації ринку нерухомості онлайн. Будучи частиною більшого маркетплейсу, сервіс пропонує простий і швидкий процес публікації оголошень, що дозволяє агентствам і приватним особам швидко розміщувати свої об'єкти (рисунок 1.11). Також варто відмітити широку аудиторію та кількість користувачів, які є потенційними клієнтами. Для швидшого просування оголошень є внутрішня платна реклама, що допомагає виділити пропозиції та привернути більше уваги.

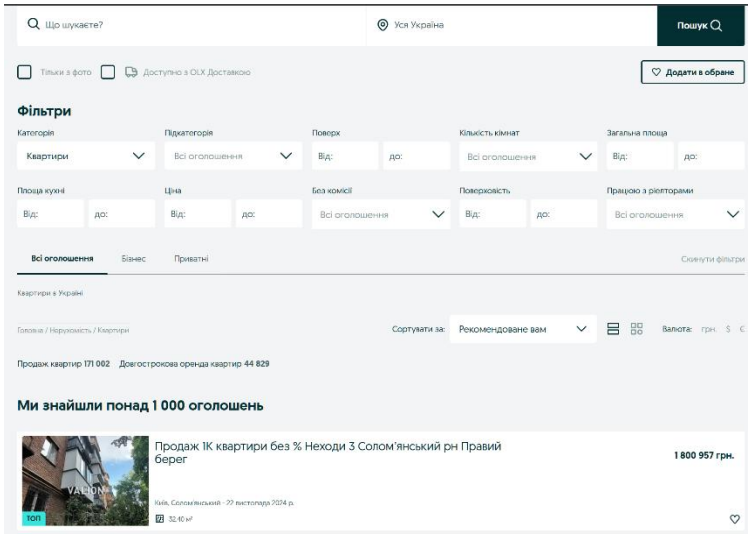


Рисунок 1.11 – Сторінка оголошень OLX Нерухомість

Цікавим технологічним рішенням, яке використовують всі три платформи, є інтеграція з картографічними сервісами. Проте кожен сервіс реалізував це по-своєму - LUN.UA зробив акцент на відображенні інфраструктури навколо житлових комплексів, DOM.RIA надає можливість оцінити транспортну доступність об'єктів, а OLX пропонує базову геолокацію об'єктів нерухомості.

Також заслуговує уваги і те, що кожна з платформ реалізувала мобільні версії сайтів чи розробили окремі додатки, оскільки в сучасних умовах понад 70% користувачів здійснюють пошук нерухомості зі смартфонів . DOM.RIA розробила потужний нативний мобільний додаток, доступний як для iOS, так і для Android платформ. У додатку реалізовано можливість швидкого пошуку

об'єктів поблизу за допомогою геолокації, а також систему push-сповіщень про появу нових об'єктів за заданими критеріями. LUN.UA обрав інший підхід, зосередившись на розробці адаптивної веб-версії сайту. Їхнє рішення вирізняється особливою увагою до швидкості завантаження та оптимізації зображень для мобільних пристроїв, що забезпечує миттєвий доступ до основної інформації навіть при повільному мобільному інтернеті. OLX Нерухомість, як частина екосистеми OLX, інтегрована в основний мобільний додаток платформи. Це рішення має свої переваги, оскільки користувачі отримують доступ до всіх категорій товарів та послуг в одному додатку. Додаток пропонує функцію швидкого створення оголошень з можливістю завантаження фотографій не тільки з галереї, а і безпосередньо з камери смартфона (рисунок 1.12), що особливо зручно для ріелторів під час огляду об'єктів.

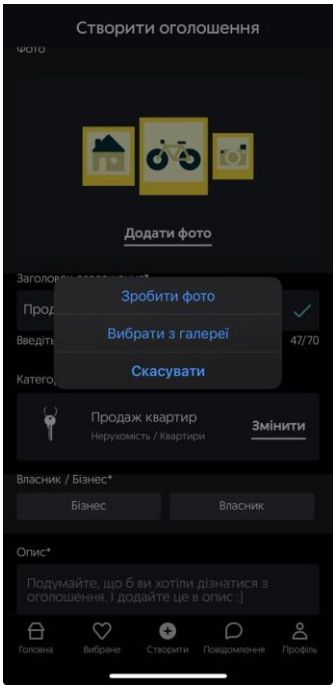


Рисунок 1.12 – Вигляд мобільного додатку Olx при створенні оголошення

Отже підсумувавши особливості користувацького досвіду на кожній платформі можемо виділити наступне:

- DOM.RIA пропонує найбільш розвинений функціонал для професійних учасників ринку, включаючи CRM-систему для ріелторів та агентств.
- LUN.UA створює преміальний досвід взаємодії з платформою, орієнтуючись на сегмент покупців новобудов.
- OLX, у свою чергу, робить ставку на простоту використання та широке охоплення аудиторії.

Також всі три платформи приділяють особливу увагу оптимізації роботи з фото контентом на мобільних пристроях, використовуючи технології стиснення зображень без втрати якості та прогресивного завантаження галерей. Це особливо важливо для сфери нерухомості, де якісні фотографії відіграють ключову роль у прийнятті рішення про перегляд об'єкта.

РОЗДІЛ 2.
ОБҐРУНТУВАННЯ, ВИБІР ТА РЕАЛІЗАЦІЯ ІНСТРУМЕНТАРІЮ ВИРІШЕННЯ ЗАДАЧІ

2.1. Вибір архітектури клієнт-серверної взаємодії

Для більшості сучасних веб-додатків клієнт-серверна архітектура є основою моделлю, і агентства нерухомості не є винятком, оскільки вона дозволяє ефективно розподіляти обробку даних і ресурси між клієнтською та серверною частинами, що в свою чергу надає можливості ефективно обробляти велику кількість запитів і об’єми даних. При застосуванні такого типу архітектури ми можемо застосувати розподілену обробку запитів, це коли клієнт (клієнтська частина) відповідає за відображення інформації та взаємодію з користувачем, в той час як сервер (серверна частина) відповідає за надання, обробку і збереження даних, а також забезпечує необхідну логіку для функціонування системи.

У сфері агентств нерухомості важливою перевагою клієнт-серверної архітектури є можливість масштабування і обробки великих обсягів інформації, таких як дані про об'єкти нерухомості, користувачів, транзакції та інші ресурси, що зберігатимуться в БД (базі даних). Отже, можемо виділити таку особливість, як необхідність одночасної обробки, як структурованої інформації, так і різноманітного медіа контенту. Наприклад, для одного об'єкта нерухомості потрібно зберігати не лише базові характеристики, такі як площу чи кількість кімнат, але й значний обсяг фотографій, планів приміщень, відеоматеріалів та документації. Останнім часом також все більшого поширення набувають інтерактивні 3D-тури об'єктами, що створює додаткові вимоги до архітектури системи. Варто відзначити, що крім основних даних будуть також і похідні дані, такі як статистика переглядів, запитів, підписок та іншої аналітичної інформації, що дозволяє проводити маркетингові комунікації.

Розглядаючи особливості архітектурних рішень, варто звернути увагу на необхідність забезпечення надійного та розподіленого зберігання даних. Досвід показує, що найефективнішим підходом є комбінування різних типів сховищ. При цьому важливо враховувати, що об'єм даних у сфері нерухомості постійно зростає, тому архітектура повинна бути достатньо гнучкою для масштабування.

Важливою складовою архітектури є також безпека даних. Враховуючи, що через систему проходять конфіденційні дані клієнтів та можливі відомості про фінансові операції чи можливості клієнтів щодо таких, захист інформації стає критично важливим аспектом. Практика показує, що найбільш вразливими місцями часто стають канали передачі даних між клієнтом та сервером, тому варто приділяти увагу шифруванню та захисту від несанкціонованого доступу.

Таблиця 2.1. – Порівняльна таблиця архітектурних підходів

Критерій	Монолітна архітектура	Трирівнева архітектура	Мікросервісна архітектура
Складність розробки	Низька	Середня	Висока
Масштабованість	Обмежена	Хороша	Відмінна
Надійність	Середня	Висока	Висока
Швидкість розробки	Висока	Середня	Низька
Вартість розробки	Низька	Середня	Висока
Гнучкість змін	Низька	Середня	Висока
Продуктивність	Висока	Висока	Середня
Складність підтримки	Низька	Середня	Висока

На основі проведеного аналізу та специфіки проєкту, оптимальним вибором для веб-додатку агентства нерухомості є трирівнева архітектура з REST API. Це рішення забезпечує оптимальний баланс між складністю розробки, можливостями масштабування та вартістю реалізації.

Трирівнева архітектура (рисунок 2.1) складатиметься з:

1. Рівень представлення (клієнтський рівень):
   - Відповідає за користувацький інтерфейс
   - Забезпечує адаптивність під різні пристрої
   - Реалізує клієнтську логіку та валідацію даних
2. Рівень бізнес-логіки (серверний рівень):
   - Обробляє запити клієнта через REST API
   - Реалізує бізнес-процеси та правила
   - Забезпечує інтеграцію з зовнішніми сервісами
3. Рівень даних:
   - Відповідає за зберігання та управління даними
   - Забезпечує цілісність та безпеку даних
   - Оптимізує запити до бази даних

Важливим аспектом обраної архітектури є впровадження REST API, що надає такі переваги:

- Стандартизований інтерфейс для взаємодії між клієнтом та сервером
- Можливість легкого масштабування та модифікації окремих компонентів
- Простота інтеграції з мобільними додатками та сторонніми сервісами
- Кешування даних для оптимізації продуктивності

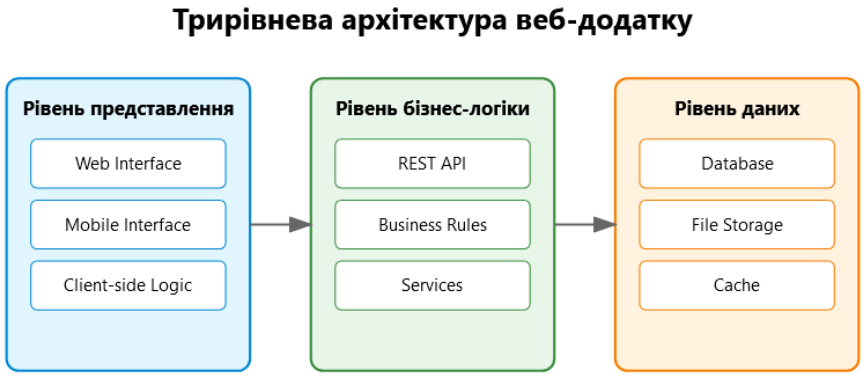


Рисунок 2.1. – Трирівнева архітектура для веб-додатку агентства нерухомості.

Таке архітектурне рішення забезпечить необхідну гнучкість для подальшого розвитку системи та дозволить ефективно масштабувати окремі компоненти залежно від навантаження. Важливо відзначити, що обрана архітектура також спрощує процес розробки та тестування, оскільки кожен рівень можна розробляти та тестувати незалежно.

2.2. Огляд та вибір технологій для розробки клієнтської та серверної частини

Фронтенд — це частина веб-додатку, з якою безпосередньо взаємодіє користувач. Його завдання — забезпечити зручний, інтуїтивний та швидкий інтерфейс, через який можна отримати доступ до всіх функцій додатка. У випадку з додатком для агентства нерухомості фронтенд повинен надавати можливість швидко знаходити потрібні об'єкти, переглядати їх деталі та зв’язуватися з агентами.

Більшість веб-додатків клієнскої частини побудовані з використанням трьох технологій, а саме зв’язки HTML + CSS + JavaScript (рисунок 2.2).

Рисунок 2.2 – Основний набір технологій для Front-end розробки

HTML (HyperText Markup Language) - розмітка, яка визначає структуру сторінок і дозволяє браузерам правильно відображати контент. HTML забезпечує каркас для інших фронтенд-технологій [13].

Для стилізації сторінок застосовують CSS (Cascading Style Sheets), а щоб це все дивилось гарно і красиво на різних пристроях (телефони, монітори,

планшети, тощо.) застосовують інструменти для адаптивного дизайну, такі як Bootstrap.

JavaScript використовують для надання інтерфейсам (сторінці чи конкретному блоку на ній) динамічності, що покращує взаємодію з користувачем. Тут розробники обирають сучасні такі, як React, Vue.js і Angular, хоча для простих речей ще часто використовують JQuery.

Всі ці технологій і фреймворки мають місце при розробці додатку для веб-додатку агентства нерухомості, проте думаю що HTML (Blade) + Bootstrap + Vue.js [14] буде достатньо. Для реалізації простих задач з динамічності будемо також використовувати JQuery [15] та інші JS бібліотеки.

Жоден середній чи більший веб-застосунок не обходиться без серверної частини – її прийнято називати бекенд. Простіше кажучи, бекенд - це все, що виконуєтьтся на сервері і спрямоване на реалізацію логіки ресурсу. Саме завдяки бекенду веб-застосунок може зберігати дані, обробляти запити користувачів та надавати відповідні результати. Без цього рівня серверної частини забезпечення інтерактивності та динамічності веб-додатку було б неможливим. Цей весь процес прихований від очей користувача, так як відбувається за межами його браузера чи комп’ютера.

Сучасні бекенд-розробники можуть обирати серед ряду популярних мов програмування та фреймворків. PHP (зокрема, фреймворк Laravel), Node.js, Python (з Django та Flask (рисунок 2.3) , оскільки вони забезпечують масштабованість, зручність обробки запитів і безпеку.

Рисунок 2.3 – Популярні мови програмування для Back-end розробки

У випадку додатка для агентства нерухомості бекенд забезпечує функціонал для обробки запитів, управління базою даних об’єктів, фільтрів, профілів користувачів та обробки транзакцій і доцільними буде використання технологій PHP [16] чи Python. Охарактеризуємо недоліки та переваги даних мов програмування :

PHP — це популярна мова для веб-розробки, яка широко використовується для створення сайтів і веб-додатків. Вона легко інтегрується з різними веб-серверами, такими як Apache та Nginx, і добре працює з базами даних, наприклад, MySQL. Один із головних плюсів PHP — це наявність потужних фреймворків, таких як Laravel , які допомагають швидко розробляти складні проєкти. Проте у PHP є й недоліки. Іноді код може стати важким для підтримки, особливо без дотримання хороших практик. Також, хоча PHP [17] постійно розвивається, деякі сучасні можливості інших мов можуть бути відсутні. З появою PHP 8.0, PHP намагається еволюціонувати від чистої серверної мови сценаріїв до мови програмування загального призначення.

Python, у свою чергу, відомий своєю простотою і читабельністю, що дозволяє швидко писати і розуміти код. Він є гнучким, адже може використовуватися не тільки для веб-розробки, а й для аналізу даних, машинного навчання та інших задач. Python [18] має багато корисних бібліотек і фреймворків, таких як Django і Flask, які спрощують створення веб-додатків. Проте Python може бути повільнішим, ніж PHP, і вимагати більше ресурсів для обробки запитів. Крім того, не всі хостинги добре підтримують Python, що може ускладнити розгортання проєктів.

У підсумку, вибір між PHP і Python залежить від специфіки вашого проєкту. PHP добре підходить для швидкої розробки веб-сайтів, а Python — для більш складних завдань, де важлива гнучкість і простота підтримки.

При розробці веб-додатку для агентства нерухомості надзвичайно важливим є етап вибору бази даних, оскільки саме вона відповідає за зберігання, управління та доступ до інформації про нерухомість, користувачів

та угоди. Вибір між реляційними та нереляційними базами даних залежить від вимог проєкту та способу роботи з даними.

Виділяють два основні типи баз даних при розробці веб-застосунків – рисунок 2.4.

Рисунок 2.4 – Типи баз даних веб-додатків.

Реляційні бази даних, такі як MySQL чи PostgreSQL [19], є популярним варіантом для додатків, які потребують чітко структурованих даних з визначеними зв’язками. Вони підтримують виконання складних запитів і транзакцій, що дозволяє ефективно обробляти дані про об'єкти, ціни та користувачів. У випадку агентства нерухомості реляційна база даних може полегшити доступ до аналітики та звітності, а також забезпечити управління інформацією про угоди та транзакції.

MySQL відома своєю швидкістю та простотою у використанні. Вона легко налаштовується і добре інтегрується з різними платформами, що робить її популярним вибором для веб-додатків. MySQL також має великий обсяг документації та підтримку спільноти, що спрощує процес навчання та вирішення проблем. Проте MySQL має свої обмеження. Вона не завжди добре справляється з складними запитами, особливо коли йдеться про великі обсяги даних.

PostgreSQL же відома своєю гнучкістю та потужністю. Вона підтримує складні типи даних і дозволяє виконувати складні запити, що робить її

відмінним вибором для проєктів, які вимагають високої продуктивності та складної аналітики. Однак, через свою потужність PostgreSQL може бути трохи складнішою в налаштуванні та використанні, особливо для новачків. Її продуктивність може бути нижчою в простих запитах.

На противагу цьому, нереляційні бази даних, такі як MongoDB, можуть бути вигідними у проєктах, де дані часто змінюються або мають непостійні структури. Вони дозволяють зберігати дані у форматі JSON, що спрощує їх обробку і масштабування. Якщо веб-додаток потребує роботи з великими обсягами інформації або високої швидкості зберігання та доступу, нереляційна база може виявитися більш доцільною. Частково ми можемо використовували її для зберігання наборів для фільтрів.

Для розробки веб-додатку агентства нерухомості оптимальним вибором буде MySQL, оскільки тут ми маємо чітко структуровані дані, такі як ціни, характеристики об’єктів, тощо.

2.3. Визначення перспектив застосування клієнт-серверних технологій в розробці веб-додатків для агентств нерухомості

Сучасний ринок нерухомості стрімко розвивається в цифровому напрямку, і технологічні інновації відіграють все більшу роль у цьому процесі. Проаналізуємо основні тенденції та перспективи розвитку клієнт-серверних технологій у контексті веб-додатків для агентств нерухомості.

Віртуальна та доповнена реальність у сфері нерухомості стає одним з найперспективніших напрямків. Технології VR та AR дозволяють створювати віртуальні тури об'єктами нерухомості, що особливо актуально в умовах, коли фізичний перегляд може бути ускладнений. З точки зору клієнт-серверної архітектури, це створює нові виклики щодо обробки та передачі великих обсягів даних. Перспективним рішенням є використання прогресивного завантаження 3D-моделей та оптимізація потокової передачі даних.

Штучний інтелект та машинне навчання відкривають нові можливості для персоналізації користувацького досвіду. Наприклад, системи рекомендацій, що базуються на аналізі поведінки користувача, можуть автоматично підбирати найбільш релевантні об'єкти нерухомості. З технічної точки зору, це вимагає впровадження мікросервісів для обробки аналітичних даних та інтеграції з AI-моделями.

Варто додати загальну діаграму перспективних напрямків розвитку клієнт-серверних технологій (рисунок 2.5) та коротко охарактеризувати її.

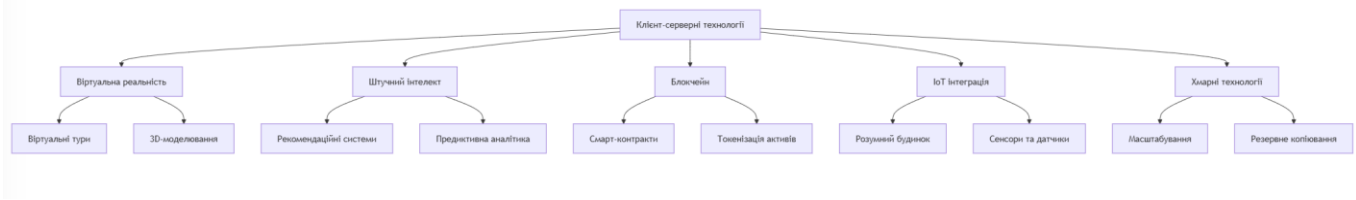


Рисунок 2.5. Перспективні напрямки розвитку клієнт-серверних технологій

Інтеграція з технологіями Інтернету речей (IoT) стає все більш актуальною. Розумні будинки та автоматизовані системи управління нерухомістю вимагають створення надійних API для взаємодії з різноманітними пристроями та датчиками. Це створює потребу в розробці спеціалізованих протоколів обміну даними та забезпеченні безпеки IoT-комунікацій.

Блокчейн-технології можуть революціонізувати процеси укладання угод та верифікації документів у сфері нерухомості. Впровадження смарт-контрактів та токенізація активів вимагатимуть модифікації існуючих архітектурних рішень для забезпечення взаємодії з блокчейн-мережами.

Розвиток хмарних технологій відкриває нові можливості для масштабування та оптимізації роботи веб-додатків. Використання сервісів типу Infrastructure as a Service (IaaS) та Platform as a Service (PaaS) дозволяє гнучко управляти ресурсами та забезпечувати високу доступність сервісів.

З точки зору користувацького інтерфейсу, перспективним напрямком є розвиток Progressive Web Applications (PWA). Ця технологія дозволяє створювати веб-додатки, які за функціональністю наближаються до нативних мобільних додатків, при цьому зберігаючи переваги веб-технологій.

Важливим аспектом розвитку є вдосконалення систем аналітики та збору даних. Використання Big Data технологій дозволить агентствам нерухомості краще розуміти потреби клієнтів та оптимізувати свої бізнес-процеси. Це вимагає впровадження спеціалізованих сховищ даних та систем обробки великих масивів інформації.

Перспективним напрямком є також розвиток систем автоматизації маркетингу та CRM-систем, інтегрованих з веб-додатками агентств нерухомості. Це дозволить підвищити ефективність роботи з клієнтами та автоматизувати рутинні процеси.

Таблиця 2.2. Технологічні тренди та їх вплив

Технологічний тренд	Вплив на архітектуру	Необхідні адаптації
VR/AR технології	Збільшення навантаження на передачу даних	Оптимізація CDN, кешування, стиснення даних
AI/ML	Потреба в обробці великих масивів даних	Впровадження спеціалізованих мікросервісів
Блокчейн	Необхідність інтеграції з блокчейн-мережами	Розробка спеціальних API та протоколів
IoT	Збільшення кількості підключених пристроїв	Масштабування серверної інфраструктури
PWA	Нові вимоги до клієнтської частини	Оптимізація офлайн-функціональності
Big Data	Необхідність аналізу великих обсягів даних	Впровадження систем аналітики

Таким чином, розвиток клієнт-серверних технологій у сфері нерухомості спрямований на створення більш інтелектуальних, автоматизованих та персоналізованих рішень. Успішна реалізація цих перспективних напрямків вимагає комплексного підходу до проектування архітектури та вибору технологій, з урахуванням як поточних потреб, так і майбутніх викликів галузі.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ ВИРІШЕННЯ ЗАДАЧІ

3.1. Огляд результатів реалізації серверної частини

Було розроблено трирівневу клієнт-серверну архітектуру на базі Laravel, що забезпечує можливість реалізації структурованого MVC (Model-View-Controller) додатку, високу масштабованість та розділення відповідальностей між компонентами. Варто відзначити також те, що Laravel надає з коробки Eloquent ORM [20] для моделей, blade шаблонізатор для views, вбудовані механізми валідації, підтримку REST API та багато інших можливостей. Основними компонентами можна виділити:

- Model: Робота з даними, бізнес-логіка
- Controller: Обробка HTTP-запитів
- View: Представлення (рендеринг) інтерфейсу

Архітектура додатку передбачає також впровадження додаткових шарів безпеки та кешування для оптимізації продуктивності системи. Крім того, розроблена структура дозволяє легко нарощувати функціональність та інтегрувати нові модулі без суттєвої перебудови існуючого коду.

На рисунку 3.1 зображена загальна структура проєкту.

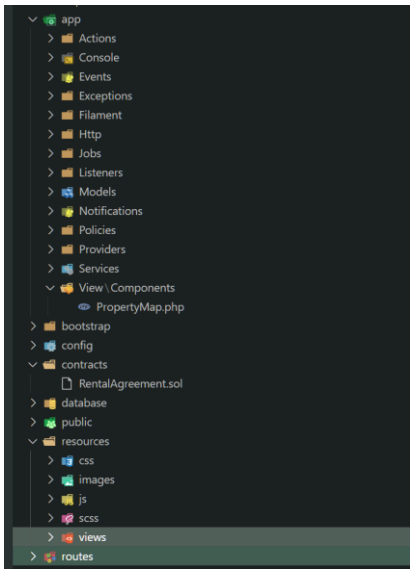


Рисунок 3.1. Загальна структура проєкту

Другим важливим аспектом серверної частини є База даних . При проєктуванні бази даних було дотримано принципів нормалізації та забезпечено цілісність даних через використання реляційної моделі MySQL. Архітектура бази передбачає гнучку систему зв'язків між сутностями, що дозволяє ефективно зберігати та опрацьовувати інформацію про різні типи нерухомості з мінімальною надлишковістю даних. Представляти весь список таблиць і їх зв’язків в даній роботі ми не будемо, проте прикріпимо ER діаграму основних таблиць (рисунок 3.2) та покажемо лістинг міграції для створення основної таблиці об’єктів нерухомості (додаток А).

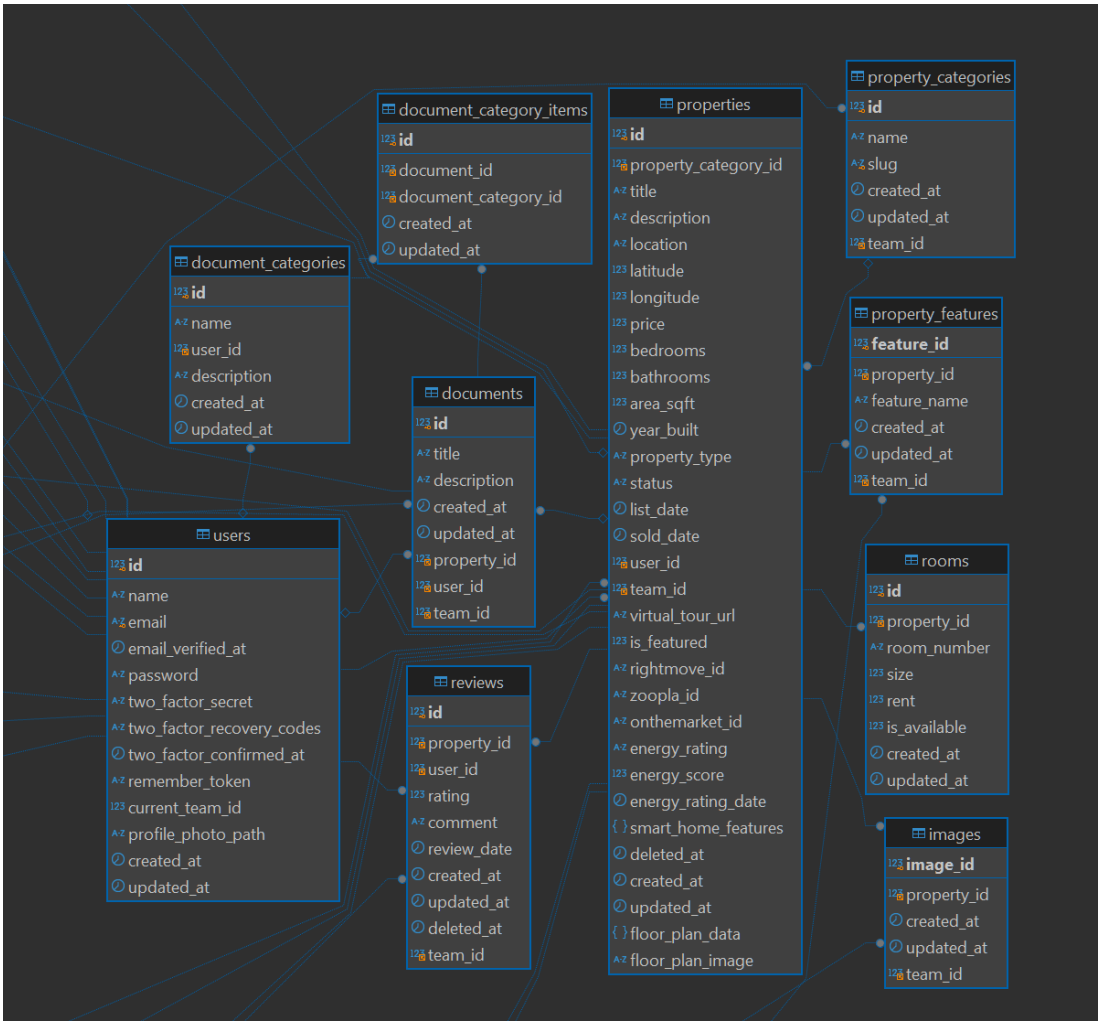


Рисунок 3.2. ER діаграма основних таблиць веб-додатку агентства нерухомості

Третьою важливою складовою додатку є REST API, де ми реалізувати авторизацію користувача API за допомогою пакету Laravel Sanctum, що

дозволяє кожному користувачеві вашої програми генерувати кілька маркерів API для свого облікового запису. Це дозволить в майбутньому інтегрувати зовнішні API сервіси з різними видами авторизації, тощо. Покажемо на рисунку 3.3, відповідь з локального сервера розробки авторизації, отримання токена для подальшої роботи з API, отриманого за допомогою програми емуляції запитів Postman. Наразі ми реалізували можливість отримати список об’єктів нерухомості і дані ризикованості конкретного об’єкту. Список доступних api routes викладено в лістингу додатку Б.

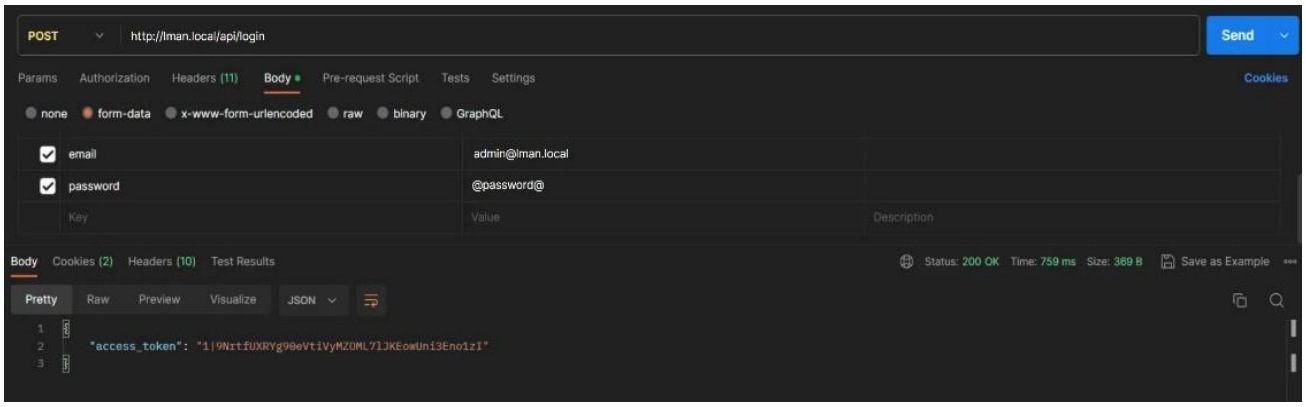


Рисунок 3.3 – Аутентифікація по API з метою отримання токену.

Доречним буде також показати лістинг методу getRisks класу PropertyRiskController, що повертає дані про ризикованість об’єкту :

```
public function getRisks($propertyId)
  {
    $property = Property::findOrFail($propertyId);

    $risks = [
      [
        'title' => 'Юридична чистота документів',
        'description' => 'Перевірка наявності обтяжень та судових спорів',
        'level' => $this->checkDocumentLegality($property)
      ],
      [
        'title' => 'Реєстраційний стан',
```

```
            'description' => 'Перевірка коректності реєстрації нерухомості',
            'level' => $this->checkRegistrationStatus($property)
        ],
        // Додаткові перевірки
    ];

    return response()->json($risks);
}
```

Рівні ризиків встановлюються в ручному режимі ріелтором, на основі документації з різних кадастрів і реєстрів. В майбутньому, коли буде змога реалізувати обмін даними по API з даними реєстрами цей процес можна автоматизувати.

3.2. Огляд результатів реалізації клієнтської частини

Клієнтська частина веб-додатку агентства нерухомості розроблена з використанням Blade шаблонізації та Bootstrap [21] для стилізації відображення різних елементів таких як форми та кнопки (рисунок 3.4) . Всі файли відображень (view) рознесені по відповідних папках компонентів, що забезпечує високу продуктивність та зручність розробки. Отож ми комплексно підійшли до побудови інтерфейсу додатку, добившись швидкого рендерингу сторінок та легкої масштабованості функціоналу.

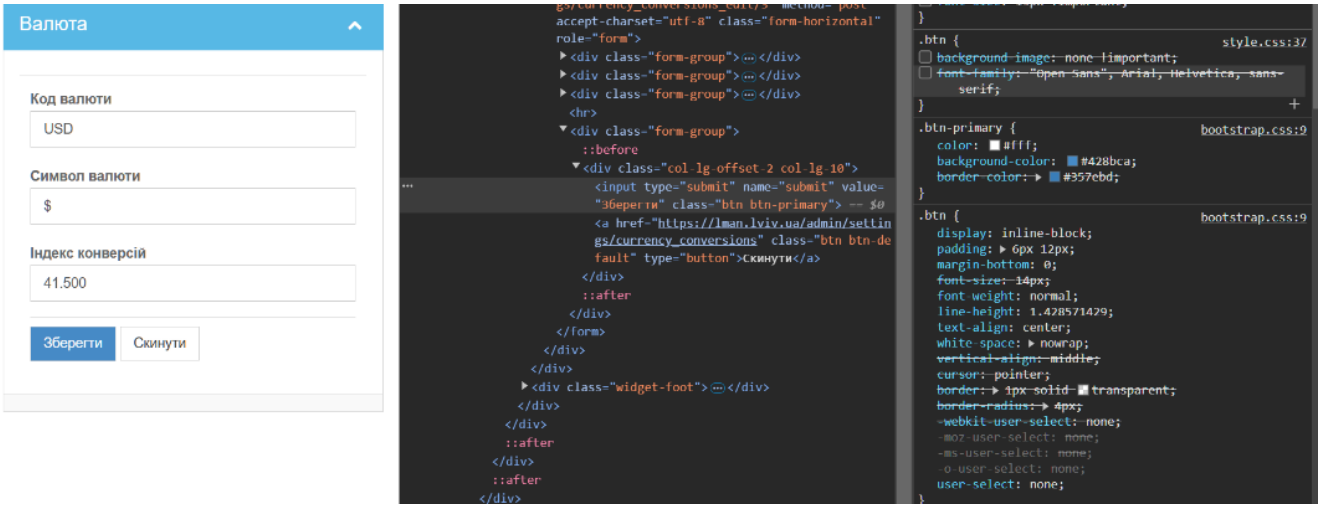


Рисунок 3.4. Приклад застосування Bootstrap

Ознайомитись з загальним виглядом панелі адміністрування можна в додатку В. Як для адміністративної частини так і для клієнтської (фронтенд) частини Bootstrap надав можливість створити Responsive [22](резиновий, під різні девайси) дизайн, що дозволяє охопити більшу кількість користувачів та надати їм якісно згруповану інформацію. Повно розмірне відображення клієнтської частини можна побачити в додатку Г, де відображена головна сторінка, а також сторінка списку нерухомості. На сторінці списку нерухомості можемо побачити зручну систему фільтрації об’єктів, яку можна налаштувати з панелі керування окремо для кожного типу нерухомості (продаж, оренда чи новобудова). На головній сторінці основний акцент зроблено на відображення об’єктів нерухомості на карті, а також відображення деталей при кліку на маркері – в попап (popup) вікні (рисунок 3.5) . Також крім основної інформації, такої як ціна, площа та поверх – є посилання на сторінку повного опису даного оголошення, що є зручно з користувацького огляду.

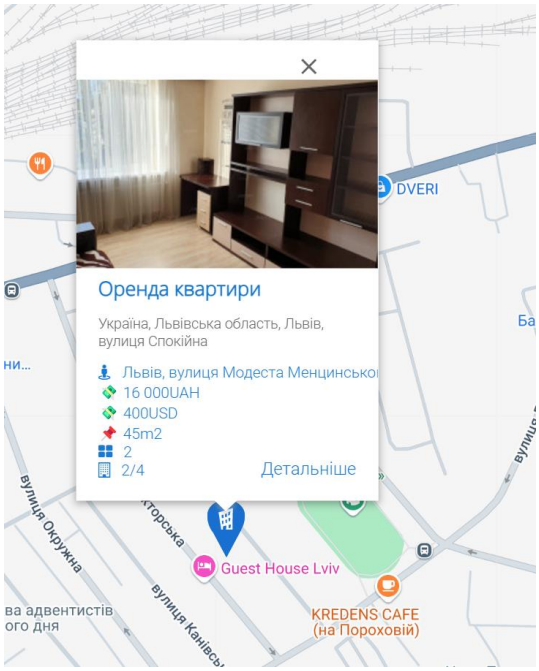


Рисунок 3.5 Динамічне відображення об’єкту на карті.

Оглядаючи сторінку самого об’єкту можна відзначити зручну галерею та додатковий функціонал для зв’язку з ріелтором, друку чи збереження даних про об’єкт нерухомості в форматі PDF (рисунок 3.6). Також для

зручності користувача додатково було реалізовано кнопку, яка дозволяє показати даний об’єкт на карті.

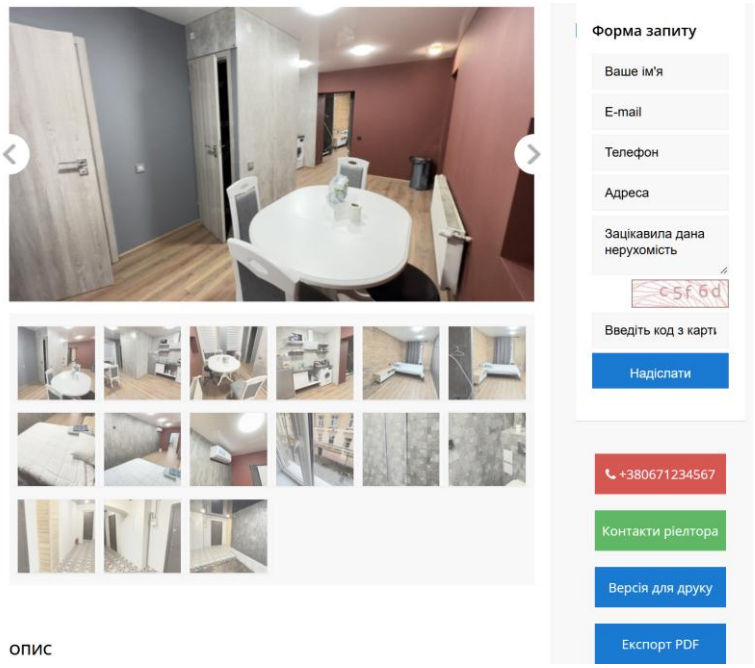


Рисунок 3.6. Функціонал сторінки деталей про об’єкт нерухомості

Підсумовуючи огляд клієнтської частини, можемо виділити лаконічність подання інформації про об’єкти нерухомості, що з точки юзабіліті повинне принести свої плоди.

3.3. Перспективи покращення продуктивності клієнт-серверного коду.

Якщо серверна частину покращувати доволі просто (хоча і праце затратно), це оновлення компонентів до останньої версії і при великих навантаженнях ми можемо робити шардінги баз даних чи реплікації, то з клієнтської сторони ми можемо допомогти зменшити навантаження на серверну, переписавши чи оптимізувавши весь можливий код на обробку на стороні клієнта, тобто за допомогою компонентів Vue.js та API для отримання сирих структур даних. Однією з таких компонент можна вважати вивід блоку ризикованості про об’єкт. Покажемо лістинг файлу PropertyRisk.vue :

```
<template>
  <div class="risk-indicator" :class="getRiskClass(property.riskLevel)">
    {{ $t('property.riskLevel') }}: {{ $t(`property.risks.${property.riskLevel}`) }}
  </div>
</template>

<script setup>
  const getRiskClass = (level) => {
    const riskClasses = {
      low: "text-green-500",
      medium: "text-yellow-500",
      high: "text-red-500",
    };
    return riskClasses[level] || "";
  };
</script>
```

Як бачимо, тут ми використовуємо одну властивостей об’єкту нерухомості , яку ми отримали на попередньому кроці за допомогою методу API в форматі JSON структури, а саме property.riskLevel (рівень критичності) і тут ми динамічно оформимо відображення її ж на шаблоні, виділивши статус певний кольором.

Іншим варіантом, що також розглядається є використання пакету модуля laravel livewire, який спрощує створення динамічних інтерфейсів. Livewire рендерить початковий вихід компонента зі сторінкою, що робить його зручним для SEO. Коли відбувається взаємодія, Livewire надсилає запит AJAX серверу, який повторно відтворює компонент, після чого Livewire інтелектуально змінює DOM відповідно до отриманих змін. Прикладом можна роботи Livewire можна побачити на рисунку 3.7

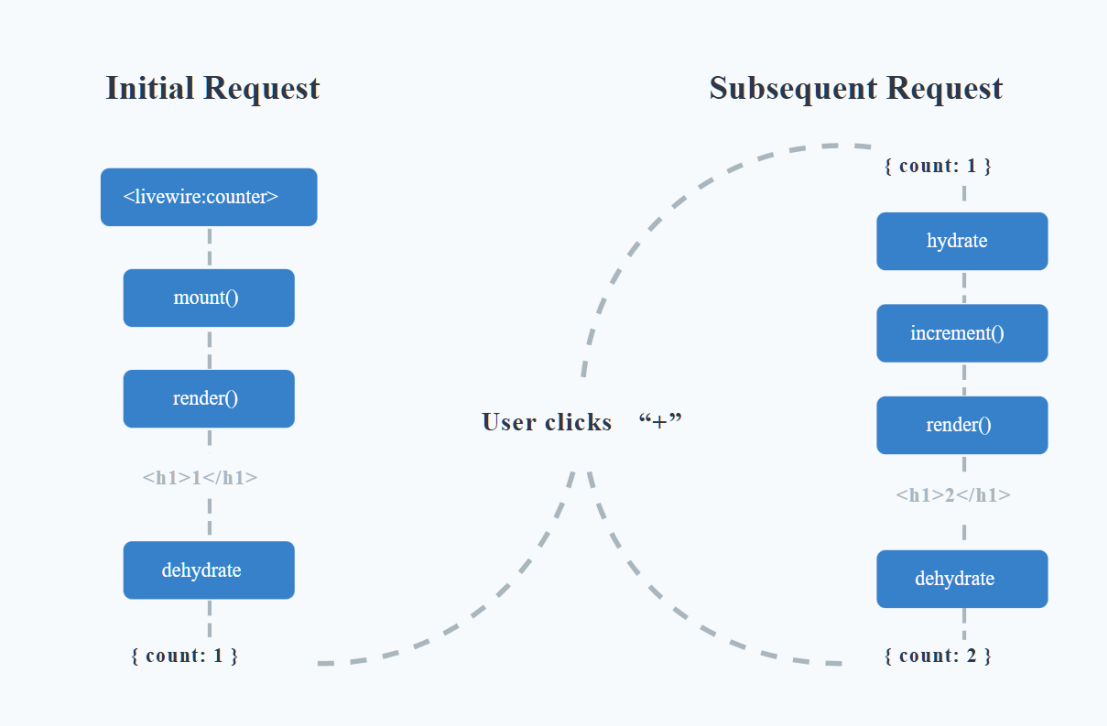


Рисунок 3.7 Принцип роботи Livewire

Також варто відмітити, що є і інші аспекти для майбутнього покращення, такі як кешування та оптимізація даних за допомогою використання Redis чи MongoDB для кешування часто запитуваних даних про нерухомість та Laravel Page Cache для статичних сторінок. Ще одним етапом оптимізації буде впровадження технології Lazy loading для відображення зображень нерухомості.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Обґрунтування можливих чинників травмонебезпечних ситуацій

У світі інформаційних технологій ризики для здоров'я та безпеки працівників мають особливу природу. На відміну від класичних виробничих середовищ, де небезпека часто пов'язана з фізичною активністю, в IT-компаніях головні загрози криються в тривалій малорухомій діяльності та постійному впливі технологічного середовища. Найбільш поширені проблеми пов'язані з тривалою роботою за комп'ютером., коли спеціаліст проводить години сидячи в нерухомій позі з монотонними рухами руками та напруженою концентрацією уваги призводять до серйозних проблем з опорно-руховим апаратом. Синдром карпального каналу, порушення постави, м'язові напруження - далеко не повний перелік професійних захворювань програмістів та інших IT-фахівців.

Технічні ризики в офісі не менш важливі. Потужна комп'ютерна техніка, складні серверні системи, мережеве обладнання створюють підвищене електричне навантаження. Короткі замикання, перепади напруги, неправильна експлуатація можуть призвести до серйозних аварійних ситуацій, включаючи пожежі.

Окрему увагу слід приділити інформаційній безпеці. В епоху цифрових технологій кібератаки, витік конфіденційної інформації, системні збої становлять не меншу загрозу, ніж фізичні пошкодження обладнання та можуть потенційно нести ризики для компанії.

Комплексний підхід до безпеки праці вимагає не лише технічних рішень, а й уваги до психологічного стану працівників. Стресові навантаження, постійне перебування в інформаційному просторі, висока відповідальність - все це створює потужне психоемоційне напруження.

Ефективна стратегія мінімізації ризиків включає кілька ключових напрямків. По-перше, це ергономіка робочого простору: правильно підібрані меблі, оптимальне освітлення, регульовані робочі місця. По-друге, технічне забезпечення: сучасні системи електробезпеки, резервування даних, антивірусний захист. По-третє, людський фактор: навчання співробітників, корпоративна культура безпеки, психологічний супровід. Важливо пам'ятати, що безпека праці - це не просто набір правил та інструкцій. Це щоденна кропітка робота кожного співробітника, менеджера, керівника компанії. Лише комплексний, усвідомлений підхід дозволить створити по-справжньому безпечне та комфортне середовище.

У результаті впровадження ефективних заходів ІТ-компанія отримує не лише захищених співробітників, а й підвищення продуктивності, лояльності персоналу, репутації надійного та турботливого роботодавця.

4.2. Умови та обставини виникнення небезпечних ситуацій та їх наслідки

Небезпечні ситуації виникають через комплекс взаємопов'язаних факторів, серед яких провідне місце займають технічні поломки, людський фактор та організаційні проблеми. Технічні несправності, зокрема відмови обладнання, короткі замикання та електричні проблеми, становлять значну загрозу для безпеки персоналу. Не менш важливим є людський фактор, який включає високу напругу та стрес через термінові проекти, недбале ставлення до техніки безпеки та ігнорування встановлених правил. Особливу увагу слід приділяти інформаційній безпеці, адже кіберзагрози та потенційні витоки конфіденційної інформації можуть мати серйозні наслідки.

Організаційні проблеми, такі як відсутність чітких процедур охорони праці, неналежне облаштування робочих місць та брак якісних інструктажів з техніки безпеки, також створюють передумови для виникнення небезпечних

ситуацій. Наслідки таких ситуацій можуть бути надзвичайно серйозними – від матеріальних втрат і травм до реальної загрози життю працівників.

Запобігання небезпечним ситуаціям вимагає комплексного підходу, який включає регулярні технічні перевірки, постійне навчання персоналу правилам безпеки, встановлення захисних пристроїв та створення сприятливої атмосфери взаєморозуміння в колективі. Тільки завдяки системній роботі з попередження та управління ризиками можливо забезпечити реальну безпеку та комфортні умови праці.

Важливо розуміти, що кожна небезпечна ситуація – це не просто випадковість, а результат комплексу невирішених проблем, які накопичуються поступово. Саме тому профілактика, постійний моніторинг та готовність до потенційних ризиків є запорукою створення безпечного робочого середовища. Комп'ютери, сервери, мережеве обладнання становлять собою потенційне джерело небезпеки, яке вимагає постійного моніторингу та профілактики. Електробезпека є одним з критичних напрямків забезпечення безпечного робочого середовища. Регулярні технічні перевірки електричної мережі, встановлення сучасних реле напруги та пристроїв захисного відключення дозволяють мінімізувати ризики виникнення аварійних ситуацій. Важливо не лише встановити технічні засоби захисту, але й навчити персонал правилам безпечного поводження з електрообладнанням.

Пожежна безпека в ІТ-офісі має свою специфіку. Потужне електронне обладнання, серверні станції, складна електропроводка - все це створює підвищені ризики загоряння. Регулярні тренінги з пожежної безпеки, навчання персоналу алгоритмам дій у надзвичайних ситуаціях стають не формальністю, а реальним інструментом порятунку життя.

Важливим чинником виникнення небезпечних ситуацій є людський фактор - недбалість, незнання елементарних правил, психологічне вигорання можуть перекреслити всі технічні заходи безпеки. Саме тому корпоративна культура, постійне навчання, створення атмосфери відповідальності та

взаємної підтримки стають найважливішими інструментами запобігання небезпечним ситуаціям.

4.3. Безпека в надзвичайних ситуаціях

Надзвичайні ситуації можуть виникати в різних формах – від природних катастроф до кібератак та воєнних конфліктів, тому розробка ефективних стратегій забезпечення безпеки стає критично важливим завданням для будь-якої організації. Відповідно до законодавства, надзвичайна ситуація розуміється як подія, що порушує нормальні умови життєдіяльності. Це можуть бути природні катастрофи, техногенні аварії, епідемії, військові конфлікти або масштабні кібератаки. Важливо розрізняти надзвичайну ситуацію та надзвичайний стан, оскільки останній є особливим правовим режимом з обмеженням конституційних прав.

Забезпечення безпеки в IT-компаніях вимагає комплексного підходу, який охоплює фізичну, технічну та кібербезпеку. Фізичний захист передбачає розробку детальних планів евакуації, обладнання приміщень системами попередження та пожежогасіння, а також створення захисних укриттів. Технічна безпека реалізується через встановлення систем безперебійного живлення, резервних комунікаційних систем та надійного захисту серверних приміщень і дата-центрів.

Особливої уваги потребує кібербезпека, яка включає постійний моніторинг потенційних загроз, захист від кібератак, навчання персоналу методам протидії соціальній інженерії та регулярне оновлення систем захисту. В умовах війни або конфлікту безпека набуває ще більшого значення і вимагає стратегічного планування дій у кризових ситуаціях.

Ключовими елементами забезпечення безпеки під час воєнних конфліктів є гарантування фізичного захисту працівників, впровадження гнучких форм роботи, збереження корпоративної комунікації та тісна співпраця з державними органами. Компанія повинна бути готова швидко

адаптуватися до мінливих обставин, забезпечуючи неперервність бізнес-процесів.

Комплексний підхід до безпеки в надзвичайних ситуаціях передбачає постійний моніторинг ризиків, розробку багаторівневих планів реагування, регулярне навчання персоналу та впровадження сучасних технологій захисту. Ефективна система безпеки дозволяє мінімізувати ризики, захистити людські та технічні ресурси, забезпечити стабільність організації в будь-яких надзвичайних обставинах.

Успіх такої системи безпеки залежить від її гнучкості, здатності до швидкої адаптації та комплексного підходу до управління ризиками. Це не лише технічне завдання, а й стратегічний напрямок розвитку компанії, який вимагає постійної уваги, аналізу та вдосконалення.

РОЗДІЛ 5
ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ

Визначення ефективності є ключовим показником успіху в будь-якій сфері діяльності та показує ефективність використання ресурсів для досягнення поставлених цілей. В контексті розробки веб-додатку для агентства нерухомості можна зазначити, що чим досягнення максимального результату з мінімальними зусиллями та ресурсами і буде ефективністю. Можемо виділити три складові ефективності – економічна (бізнес) ефективність, технічна та клієнтська.

Економічна ефективність від впровадження нашого веб-додатку для агентства нерухомості полягає в оптимізації бізнес-процесів та підвищенні загальної продуктивності діяльності агентства за рахунок суттєвого скорочення операційних витрат. Завдяки впровадженню автоматизованої системи, агентство може очікувати зменшення операційних витрат на 20-35%. Це досягається за рахунок оптимізації роботи персоналу, зменшення часу на рутинні операції та підвищення ефективності обробки клієнтських заявок. Остання в свою чергу дала додатковий економічним ефект - підвищення конверсії укладених угод, що безпосередньо впливає на фінансові показники агентства.

Технологічна ефективність є також достатньо високою оскільки наше рішення базується на сучасному стеці розробки, що включає Laravel та Vue.js. Це забезпечує високу продуктивність, масштабованість та гнучкість системи. Ключові технологічні переваги включають:

- швидку роботу клієнт-серверної взаємодії з часом відгуку до 100 мілісекунд
- здатність витримувати високі навантаження до 1000 одночасних користувачів
- надійність системи з показником відмовостійкості 99 %

Також варто відзначити, що архітектура додатку дозволяє легко впроваджувати нові функції та модифікувати існуючі без суттєвих витрат на розробку.

Оцінюючи веб-додаток з точки зору клієнтської ефективності можемо сказати, що інтуїтивно зрозумілий інтерфейс та продумані функціональні можливості спрямовані на максимальне полегшення пошуку нерухомості для користувачів. Також відображення на різних девайсах затримує більшу кількість користувачів які є потенційними клієнтами. Отже, завдяки розширеному механізму пошуку та фільтрації ми отримали скорочення часу пошуку нерухомості на 40%, за рахунок системи оцінки ризиків підвищили довіру користувачів.

Порівняно з провідними платформами нерухомості (LUN.UA, DOM.RIA, OLX), наш додаток має низку унікальних характеристик:

Таблиця 5.1. Порівняння ключових характеристик

Критерій	Наш додаток	Конкуренти
Реєстр ризиків	Повноцінний	Обмежена
Персоналізація	Висока	Обмежена
Швидкість пошуку	Висока	Середня/Низька

Підсумовуючи можемо сказати, що розроблене рішення демонструє комплексну ефективність, яка охоплює економічні, технологічні та клієнтські аспекти та має значний потенціал для подальшого вдосконалення в таких областях:

- інтеграція штучного інтелекту для прогнозування ринку нерухомості
- розширення аналітичних інструментів
- впровадження додаткових сервісів (наприклад, віртуальні огляди)

ВИСНОВКИ

Дослідження показало, що створення сучасного веб-додатку для агентства нерухомості - це складний, але захоплюючий процес, який вимагає глибокого розуміння технологій та потреб ринку. Архітектура додатку була розроблена максимально продумано. Ми свідомо обрали трирівневу модель з REST API, яка дозволяє легко розподіляти навантаження між різними компонентами системи та забезпечить гнучкість та стабільність усього додатку. Кожен архітектурний рішення було прийнято після ретельного аналізу та порівняння альтернативних підходів, що підкреслює системний характер нашої розробки.

Технологічний стек підібрано дуже ретельно. PHP з Laravel надає потужний інструментарій для серверної частини, а Bootstrap допомагає створити красивий та адаптивний дизайн. MySQL слугує надійним сховищем даних про нерухомість. Функціональність додатку виходить далеко за межі звичайного каталогу оголошень оскільки було створено повноцінну систему з авторизацією, розширеним пошуком (фільтрацією) та унікальним функціоналом оцінки ризиків нерухомості. Кожен компонент технологічного стеку було обрано не лише за його поточними можливостями, але й з огляду на потенціал майбутнього розвитку та масштабованості.

Практична цінність розробки очевидна - створено реальний інструмент, який можуть одразу впровадити агентства нерухомості. Більше того, система настільки гнучка, що її можна адаптувати під унікальні бізнес-процеси конкретної компанії. Додаток вже пройшов первинне тестування та отримав схвальні відгуки від представників ріелторської спільноти, що додатково підтверджує його ринкову затребуваність.

Особливо цікавими є перспективи розвитку проекту, а саме впровадження поглибленої аналітики ризиків та інтеграція додаткових сервісів. Також можлива інтеграція таких технологій, як віртуальна реальність чи застосування штучного інтелекту. Це означає, що додаток має потенціал

для вдосконалення та не застаріє через рік-два, а буде постійно еволюціонувати. Команда розробників вже має попередні напрацювання та бачення щодо майбутніх удосконалень, які зможуть зробити продукт ще більш конкурентоспроможним.

Загалом, ця робота демонструє: правильний вибір технологій, архітектури та підходу до розробки дозволяє створити не просто додаток, а справді корисний інструмент, який може змінити роботу цілої галузі. Наш проект є яскравим прикладом того, як сучасні технологічні рішення можуть суттєво оптимізувати та трансформувати традиційні бізнес-процеси в сфері нерухомості.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Розуміння Клієнт-Серверної Архітектури на прикладах. [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/44636/

2. Мельник І.В. Архітектура клієнт-серверних систем: Навчальний посібник. - Львів: Видавництво Львівської політехніки, 2019. - 298 с

3. Серверні WEB-технології. КПІ 2023. [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/5e3e77bd-fd03-4154-8841-dc16f26a7d1c/content

4. Web Application Architecture: Principles, Protocols and Practices / Leon Shklar, Rich Rosen. - Wiley, 2020. - 368 p.

5. Building Microservices: Designing Fine-Grained Systems / Sam Newman. - O'Reilly Media, 2021. - 280 p.

6. Patterns of Enterprise Application Architecture, Martin Fowler - Addison-Wesley Professional 2003, -560 p.

7. Designing Data-Intensive Applications / Martin Kleppmann. - O'Reilly Media, 2017. - 616 p

8. REST API Basics - 4 Things you Need to Know. [Електронний ресурс]. – Режим доступу: https://mannhowie.com/rest-api

9. REST API Design Rulebook / Mark Masse. - O'Reilly Media, 2019. - 116 p.

10. Will GraphQL Overtake REST APIs to Become The Next Big Thing in Application Programming Interface? [Електронний ресурс]. – Режим доступу: https://thinkpalm.com/blogs/will-graphql-overtake-rest-apisi-in-application-programming-interface/

11. WebSocket. [Електронний ресурс]. – Режим доступу: https://dev.to/robertobutti/websocket-with-php-4k2c

12. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin. - Pearson, 2017. - 352 p

13. Mozilla Developer Network. HTML: HyperText Markup Language. Офіційна документація. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML.

14. Vue.js Прогресивний JavaScript фреймворк. [Електронний ресурс]. - Режим доступу: https://ua.vuejs.org/.

15. Jquery. Write less, do more. [Електронний ресурс]. Режим доступу до ресурсу: https://jquery.com/

16. Огляд популярних мов програмування – Python, Java, PHP, C/C++ та інші [Електронний ресурс] – Режим доступу : https://best-work.com.ua/programming-languages/

17. PHP 7: Realistic Approach to Web Development / Branko Vranesevic. - Packt Publishing, 2018. - 412 p.

18. Пришвидшений курс Python/ Ерік Маттес - Видавництво Старого Лева, 2021 – 600 с

19. Database Systems: Design, Implementation, and Management / Carlos Coronel, Steven Morris. - Cengage Learning, 2018. - 624 p.

20. Laravel: Up & Running: A Framework for Building Modern PHP Apps / Matt Stauffer. - O'Reilly Media, 2019. - 374 p

21. Buttons in bootstrap. [Електронний ресурс]. Режим доступу до ресурсу: https://getbootstrap.com/docs/5.2/components/buttons/

22. Professional Web Developer's Guide to Responsive Design / Ben Frain. - Packt Publishing, 2020. - 340 p.

ДОДАТКИ

Додаток А

Текст вихідного коду створення таблиці Properties

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('properties', function (Blueprint $table) {
            $table->id('id');
            $table->string('title');
            $table->text('description');
            $table->string('location');
            $table->decimal('latitude', 10, 8)->nullable();
            $table->decimal('longitude', 11, 8)->nullable();
            $table->decimal('price', 10, 2);
            $table->integer('bedrooms');
            $table->integer('bathrooms');
            $table->integer('area_sqft');
            $table->year('year_built');
            $table->string('property_type');
            $table->string('status');
            $table->date('list_date')->default(now());
            $table->date('sold_date')->nullable();
            $table->unsignedBigInteger('user_id');
```

```
            $table->unsignedBigInteger('team_id')->default(1);
            $table->boolean('is_featured')->default(false);
            $table->string('rightmove_id')->nullable();
            $table->string('zoopla_id')->nullable();
            $table->string('onthemarket_id')->nullable();
            $table->string('energy_rating')->nullable();
            $table->integer('energy_score')->nullable();
            $table->date('energy_rating_date')->nullable();
            $table->json('smart_home_features')->nullable();
            $table->foreign('user_id')->references('id')->on('users');
            $table->foreign('team_id')->references('id')->on('teams');
            $table->string('virtual_tour_url')->nullable();
            $table->softDeletes();
            $table->timestamps();
            $table->json('floor_plan_data')->nullable();
            $table->string('floor_plan_image')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('properties');
    }
};
```

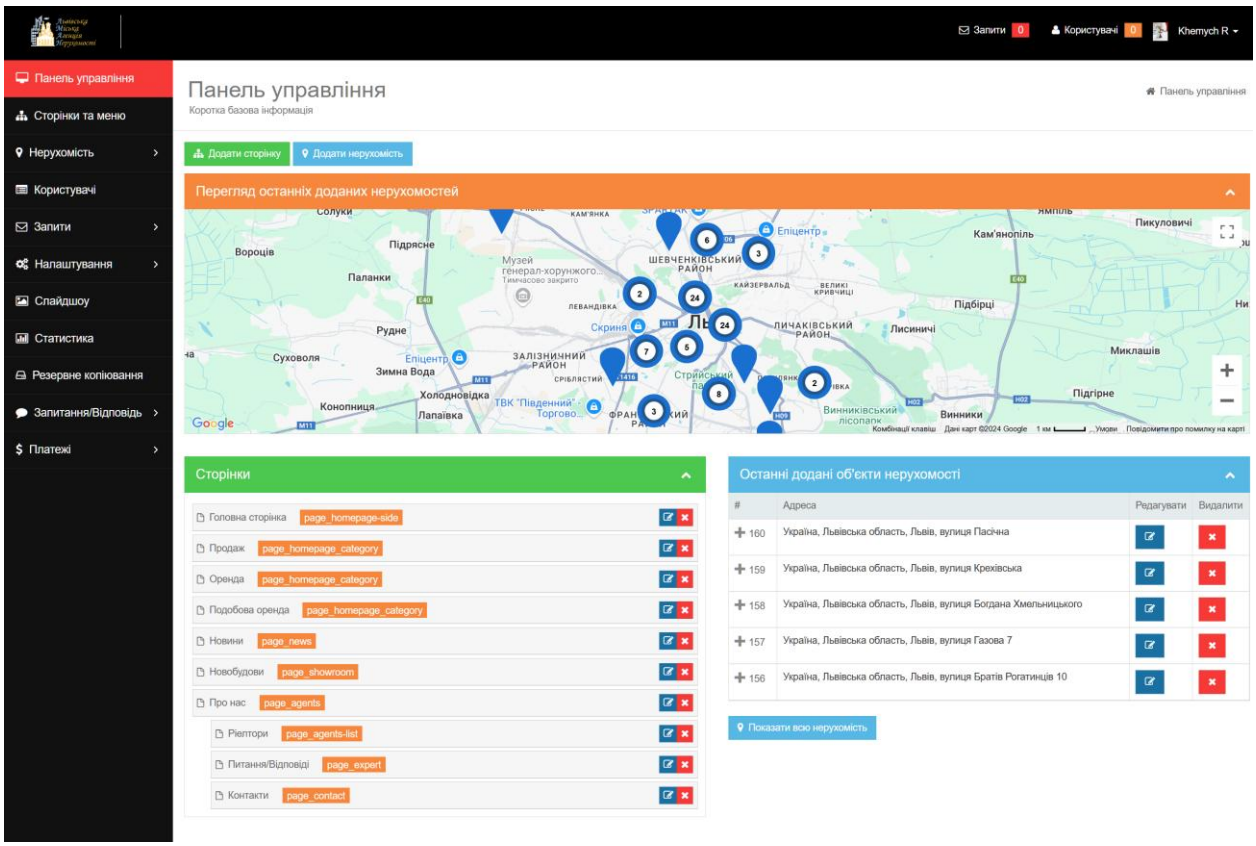
Додаток Б

Лістинг Api Routes

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PropertyController;
use App\Http\Controllers\PropertyRiskController;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

Route::middleware('auth:sanctum')->group(function () {
    Route::get('/properties', [PropertyController::class, 'index']);
    Route::get('/property-risks/{id}', [PropertyRiskController::class, 'getRisks']);

    // Define your protected API routes here
});
```

Додаток В

Вигляд панелі адміністратора



Додаток Г

Вигляд головної сторінки