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

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ПЛАТФОРМИ ДЛЯ КЕРУВАННЯ ПОСЛУГАМИ АГЕНЦІЇ НЕРУХОМОСТІ «ELITE» З ВИКОРИСТАННЯМ LARAVEL ТА MYSQL

DEVELOPMENT OF A PLATFORM FOR MANAGING REAL ESTATE AGENCY SERVICES «ELITE» USING LARAVEL AND MYSQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Сапожнік Антон Русланович

Керівник:
Андрущак Ігор Євгенович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«10» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Сапожніку Антону Руслановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *«Розробка платформи для керування послугами агенції нерухомості "Elite" з використанням Laravel та MySQL».*
Керівник роботи: *д.т. н., професор Андрущак І. Є.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02.*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Laravel 10, PHP 8+, MySQL (InnoDB), RedBeanPHP, JavaScript, jQuery, AnyChart, Laravel Backpack, Composer, Git / GitHub Actions, Node.js / npm, Laravel Mix (webpack), Bootstrap 5, HTML5 / CSS3, Apache (MAMP), методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз предметної області та існуючих рішень на ринку нерухомості; визначення та формалізація вимог до платформи; проєктування архітектури системи та схеми даних; обґрунтування вибору технологічного стеку; реалізація середовища та ключових функціональних модулів; інтеграція адміністративної панелі, обробка медіа та email-розсилки; тестування та налагодження інформаційно-комп'ютерної системи; забезпечення безпеки та підготовка релізу*
5. Перелік графічного матеріалу: *25 рисунків, 2 додатки.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Андрущак І. Є.*		
Специфікація вимог до розробленої системи	*Андрущак І. Є.*		
Розробка об'єкта проектування	*Андрущак І. Є.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,09 %		
Академічна доброчесність	*Андрущак І. Є.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Вик.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	Вик.
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Вик.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	Вик.
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	Вик.
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Вик.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Вик.

Здобувач вищої освіти _____ Антон САПОЖНІК

Керівник кваліфікаційної роботи _____ Ігор АНДРУЩАК

АНОТАЦІЯ

Сапожнік А. Р. Розробка платформи для керування послугами агенції нерухомості «Elite» з використанням Laravel та MySQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі здійснено аналіз предметної області та існуючих веб- і мобільних рішень для ринку нерухомості, визначено вимоги користувачів і адміністраторів та сформульовано завдання кваліфікаційної роботи. У другому розділі проведено модельне проєктування системи, описано функціональні й нефункціональні вимоги, побудовано UML-діаграми, спроєктовано багаторівневу MVC-архітектуру і реляційну схему даних у MySQL. У третьому розділі реалізовано практичну частину: налаштовано середовище розробки, створено публічну частину на Blade-шаблонах із контролерами SitePageController і SiteEstateController, розроблено адміністративну панель із Backpack CRUD, організовано завантаження та обробку зображень, реалізовано email-повідомлення через черги, впроваджено заходи SEO, провели статичний аналіз, юніт-, інтеграційні та браузерні тести, налагодження з Xdebug і Telescope, а також забезпечено захист від SQL-ін’єкцій, XSS і CSRF. У висновках підкреслено, що виконані завдання забезпечили створення безпечного, продуктивного й легко масштабованого вебдодатка для управління нерухомістю.

Ключові слова: нерухомість, вебдодаток, Laravel, PHP, MySQL, Backpack, MVC, CRUD, SEO.

ABSTRACT

Sapozhnik A. Development of a Platform for Managing Real Estate Agency Services "Elite" Using Laravel and MySQL. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

In the first chapter, an analysis of the domain and existing web and mobile solutions for the real estate market was conducted, user and administrator requirements were defined, and the tasks of the qualification work were formulated. In the second chapter, the system’s model design was carried out: functional and non-functional requirements were described, UML diagrams were constructed, and a multi-layer MVC architecture with a relational MySQL data schema was designed. In the third chapter, the practical implementation was completed: the development environment was configured, the public interface was built with Blade templates using SitePageController and SiteEstateController, an administrative panel was developed with Backpack CRUD, image uploading and processing were organized, email notifications via queues were implemented, SEO measures were introduced, static analysis as well as unit, integration, and browser tests were performed, debugging was conducted with Xdebug and Telescope, and protection against SQL injection, XSS, and CSRF was ensured. The conclusion emphasizes that these tasks have resulted in a secure, efficient, and easily scalable web application for real estate management.

Keywords: real estate, web application, Laravel, PHP, MySQL, Backpack, MVC, CRUD, SEO.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах дедалі більшої конкуренції на ринку нерухомості агенції «Elite» потребують інструментів, які забезпечують швидке й прозоре обслуговування клієнтів, централізоване управління об’єктами та угодами, а також оперативне формування звітності. Перехід до вебплатформ із єдиною базою даних дозволяє усунути дублювання інформації, зменшити кількість паперової документації та мінімізувати ризики помилок, пов’язаних із ручним введенням даних.

Використання сучасних фреймворків і технологій, зокрема Laravel із його потужними можливостями маршрутизації, ORM (Eloquent), системою черг і вбудованим механізмом аутентифікації, у поєднанні з перевіреною СУБД MySQL, гарантує створення надійного, безпечного та масштабованого рішення. Така платформа забезпечить гнучку настройку ролей користувачів, підтримку різних рівнів доступу, зручні інтерфейси адміністрування й можливість подальшого розвитку функцій відповідно до потреб бізнесу.

Ці обставини зумовлюють важливість розробки спеціалізованої вебплатформи, спрямованої на автоматизацію ключових бізнес-процесів агенції нерухомості «Elite», що, у свою чергу, стане основою для підвищення ефективності роботи консультантів і підвищення рівня задоволеності клієнтів.

Метою кваліфікаційної роботи бакалавра є створення вебплатформи для агенції нерухомості «Elite», яка забезпечить керування клієнтами, об’єктами нерухомості та угодами з використанням фреймворку Laravel і СУБД MySQL.

Об’єктом кваліфікаційної роботи є вебплатформа агенції нерухомості «Elite», призначена для автоматизації робочих процесів консультантів, адміністрування бази клієнтів і об’єктів, а також ведення угод.

Предметом кваліфікаційної роботи бакалавра є практична реалізація функціональних модулів платформи – автентифікації, ролей користувачів, CRUD-операцій із нерухомістю та клієнтами, формування звітів – із застосуванням Laravel та MySQL.

Для досягнення поставленої мети були сформульовані такі завдання:

– проаналізувати предметну область та існуючі рішення на ринку нерухомості;

– визначити та формалізувати вимоги до платформи;

– спроєктувати архітектуру системи та схему даних;

– обґрунтувати вибір технологічного стеку;

– реалізувати середовище та ключові функціональні модулі;

– інтегрувати адміністративну панель, обробку медіа та розсилки;

– провести тестування та налагодження;

– забезпечити безпеку та підготувати реліз.

Отже, створена вебплатформа на базі Laravel та MySQL дозволить агенції «Elite» централізувати всі дані, автоматизувати життєвий цикл об’єктів нерухомості та угод, а також забезпечити швидкий доступ до звітності й аналітики. Це покращить внутрішні процеси, підвищить прозорість взаємодії з клієнтами та закладе основу для подальшого розвитку та інтеграції нових сервісів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах реалізація угод із нерухомістю в Україні значною мірою покладається на агентства та масивне використання зовнішніх порталів. За висновками місії МВФ, дані про онлайн-оголошення з платформи OLX досі агрегуються й очищуються вручну в MS Excel із тривалістю обробки до двох тижнів на квартал, що свідчить про відсутність спеціалізованих автоматизованих рішень для керування інформацією. Попри це, більшість онлайн-сервісів обмежується лише базовою публікацією оголошень із фільтрами за ціною, площею та локацією, що призводить до розривів між внутрішніми базами агенцій і зовнішніми майданчиками.

У 2024 році попит на новобудови в Україні залишався на рівні лише 20 % довоєнних показників, тоді як вторинне житло користувалося попитом близько 70 % порівняно з 2021 роком [1]. Водночас у четвертому кварталі середня вартість нових квартир зросла на 15,72 % порівняно з аналогічним періодом попереднього року, що підкреслює високі темпи інфляції на ринку нерухомості та необхідність оперативного моніторингу цін (рис. 1.1) [2].

City	Avg Price (USD/m², primary)	YoY Change (Mar 2025)	Change Since 2021 (pre-war)
Kyiv	$1,280	+4.1%	+29%
Lviv	$1,330	+1.5%	+66%
Ivano-Frankivsk	$850	+7.6%	+98%
Uzhhorod	$1,150	+4.6%	+58%
Odesa	$1,010	+5.2%	+16%
Dnipro	$1,090	+4.8%	+42%
Kharkiv	$670	−10.7%	−15%
Khmelnytskyi	$670	+1.5%	+49%

Рисунок 1.1 – Середні ціни на квартири [2]

Ці тенденції вказують на фрагментованість даних, потребу в швидкому реагуванні на зміни ринкової кон’юнктури та в інтегрованому рішення для централізованого керування клієнтами, об’єктами та угодами.

Отже, існуюча модель продажів нерухомості характеризується високим рівнем ручної праці, дублюванням даних між різними сервісами та обмеженими інструментами аналітики. Це створює нагальну потребу в розробці спеціалізованої вебплатформи для агенції «Elite», яка б об’єднала всі бізнес-процеси – від ведення бази клієнтів до формування звітів і аналітики в режимі реального часу.

Аналіз конкурентних рішень є одним із ключових етапів підготовки до розробки вебплатформи агенції «Elite». Детальне вивчення існуючих сервісів на локальному ринку Луцька дозволяє не лише оцінити поточний рівень автоматизації бізнес-процесів у галузі нерухомості, але й виявити «білі плями» – функції або сервіси, які відсутні або реалізовані неефективно. Завдяки такому порівнянню можна зрозуміти, які інструменти здатні дійсно підвищити швидкість обробки запитів, покращити взаємодію з клієнтами та оптимізувати внутрішні робочі процеси. Крім того, аналіз показників юзабіліті, наявності мобільних адаптацій, рівня безпеки й прозорості цінових пропозицій допомагає сформувати унікальну торгову пропозицію і закласти основу для конкурентної переваги платформи «Elite».

Будуть розглянуті три локальні сервіси-конкуренти, їхній функціонал, сильні та слабкі сторони, а також ті можливості, які варто запозичити або, навпаки, удосконалити в ході проєктування власного рішення. Це дасть змогу створити продукт із чітко вираженими перевагами та відповідати саме тим потребам, які на даний момент залишаються незадоволеними у клієнтів агенцій нерухомості в Луцьку.

Компанія «Атланта Нерухомість» працює на ринку Луцька з 1995 року і позиціонує себе як команда висококваліфікованих фахівців у сфері комерційної та житлової нерухомості [3]. Їхня мета – забезпечити клієнтам швидке й вигідне вирішення питань купівлі, продажу та оренди за допомогою індивідуального

менеджера та юридичної підтримки на всіх етапах співпраці. На сайті компанії реалізовано розділи «Продаж» і «Оренда» з детальним переліком типів об’єктів, а також широкі можливості залишити заявку на безкоштовну консультацію.

Інтерфейс платформи виконано в одній кольоровій гаммі з чітким поділом на комерційну та житлову нерухомість, що полегшує навігацію (рис. 1.2). Відвідувачі можуть оперативно перейти до потрібного розділу, знайти контакти менеджерів та ознайомитися з перевагами роботи агентства. До сильних сторін варто віднести наявність відеовідгуків і блогу з корисними порадами, а також статистичні показники у вигляді лічильників успішних дзвінків і консультацій, які підвищують довіру клієнтів.

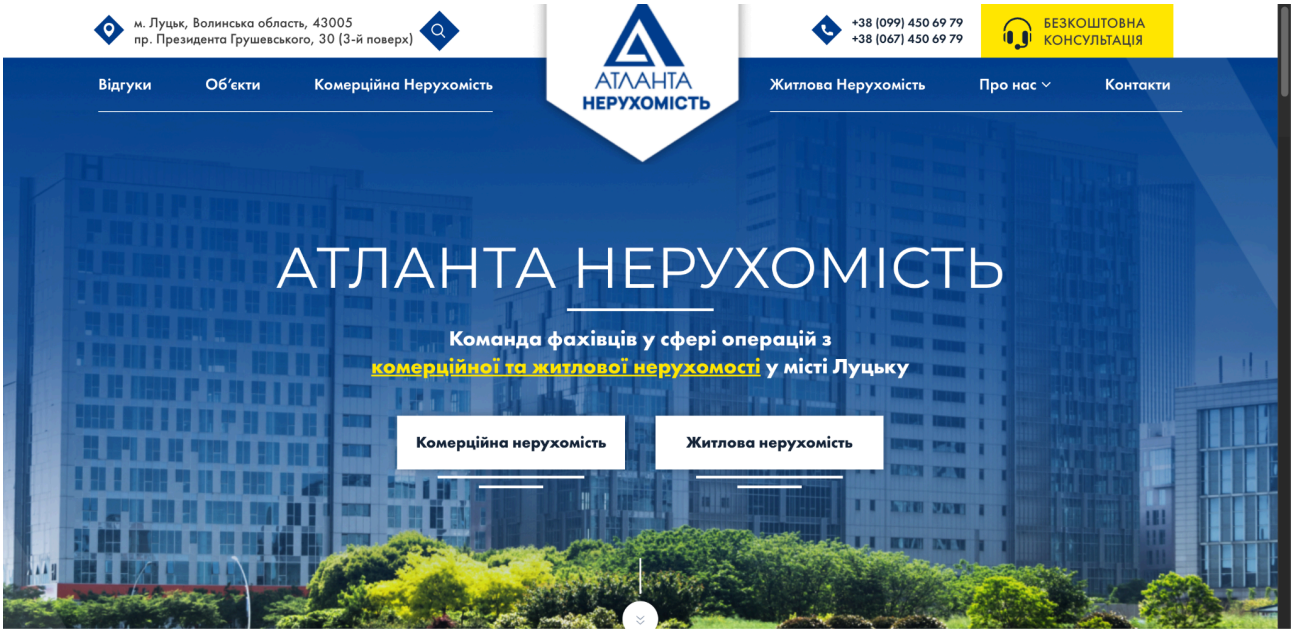

Рисунок 1.2 – Сайт «Атланта Нерухомість» [3]

Водночас сайт усе ще має низку недоліків: відсутня можливість користувацького кабінету та персональної історії звернень, обмежений пошук без тонких фільтрів (наприклад, за районом чи ціновим діапазоном) і велика кількість графічних елементів, що може уповільнювати завантаження на мобільних пристроях. Нестача інтегрованого календаря показів і автоматизованих нагадувань потребує подальшого доопрацювання для підвищення зручності користування.

Компанія «ВМБ Нерухомість» працює на ринку агенцій нерухомості Луцька вже понад 28 років, пропонуючи повний спектр послуг із купівлі, продажу та оренди житлових і комерційних об’єктів, а також технічної інвентаризації й юридичного супроводу угод [4]. На сайті vmb.lutsk.ua у розділі «Про нас» детально описано філософію компанії – індивідуальний підхід до клієнта та комплексна підтримка на всіх етапах транзакції, від оцінки майна до оформлення права власності, що підвищує довіру й лояльність користувачів.

Інтерфейс платформи витримано в стриманих біло-сірих тонах із акцентами на логотипі, що дозволяє швидко зорієнтуватися у розділах «Продати/здати», «Купити», «Орендувати» та ряду спеціалізованих послуг – від виготовлення технічних паспортів до юридичних консультацій (рис. 1.3). Позитивно позначається на користувацькому досвіді наявність інтегрованої контактної інформації та форм для швидкого запиту зворотного дзвінка.

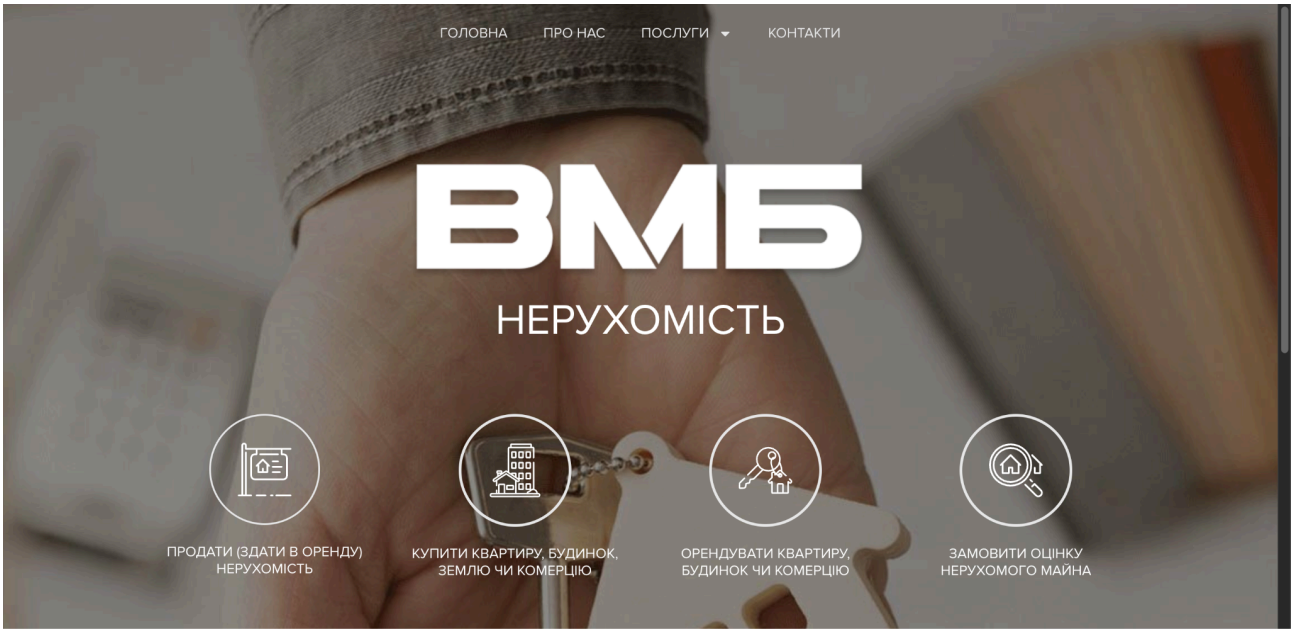

Рисунок 1.3 – Сайт «ВМБ Нерухомість» [4]

Разом з тим у сайту є недоліки: по-перше, відсутність пошуку за ключовими параметрами (район, метраж, ціна), що змушує клієнта вручну переглядати довгі списки об’єктів; по-друге, відсутність особистого кабінету, де можна було б зберігати обрані об’єкти й статуси переговорів. Крім того, низька

оптимізація зображень та велика кількість JavaScript-скриптів уповільнюють завантаження сторінки на мобільних пристроях із повільним інтернетом. Ці моменти свідчать про потенціал для впровадження більш гнучких інструментів фільтрації та персоналізації інтерфейсу в майбутньому.

Компанія «Еверест Нерухомість» позиціонує себе як «вершина можливостей у світі нерухомості Луцька», пропонуючи клієнтам індивідуальний підхід до кожної угоди від первинних консультацій до повного юридичного супроводу [5]. На головній сторінці відразу кидається в очі фірмовий слоган та пошукова панель, яка дозволяє відфільтрувати об’єкти за типом (квартира, будинок, котедж, ділянка, офіс), мікрорайоном і кількістю кімнат, що створює відчуття персоналізованого сервісу (рис. 1.4). Далі йде блок із рекомендованими об’єктами – кожен із фотографією, короткими характеристиками (метраж, кількість санвузлів, кімнат) та вартістю, що допомагає клієнту швидко зорієнтуватися в асортименті.

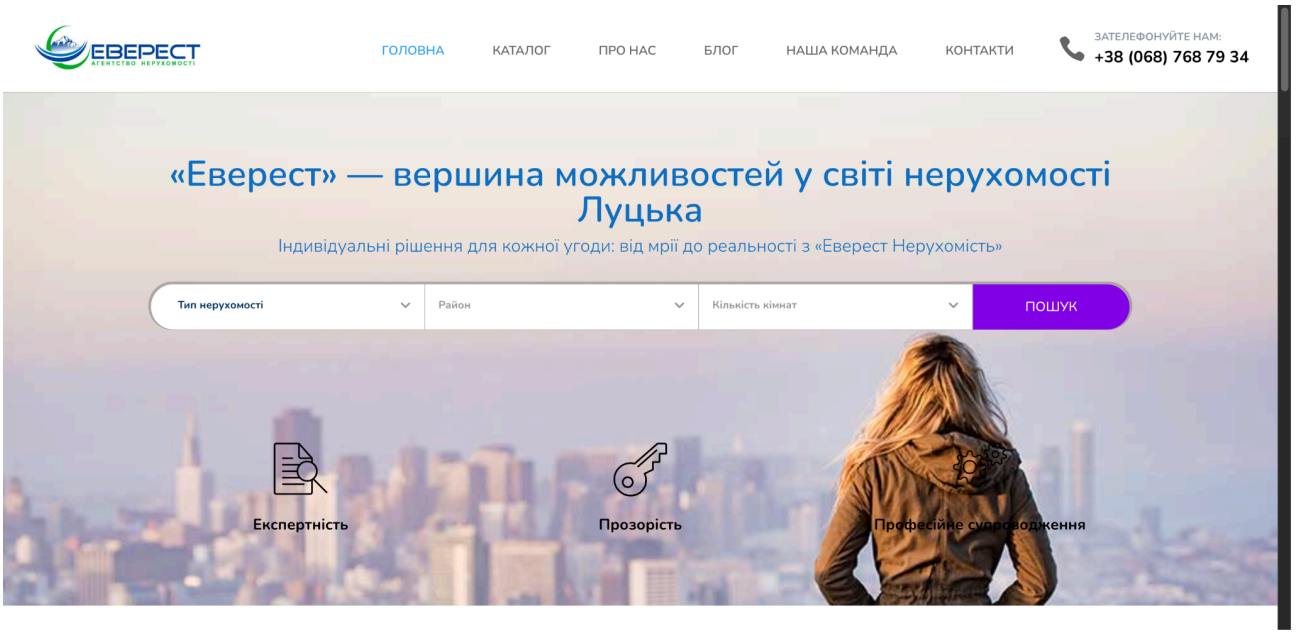

Рисунок 1.4 – Сайт «Еверест Нерухомість» [5]

До переваг платформи належать лаконічний і зрозумілий інтерфейс у фірмових кольорах агенції, додатковий розділ із описом кожного району Луцька, який сприяє обізнаності покупця, а також секція «Наша команда» з формою

швидкого звернення до конкретного менеджера. Блок «Блог» із корисними порадами та «Про нас» створюють відчуття надійності й професіоналізму.

Водночас сайт має обмежені можливості персоналізації: немає користувацького кабінету для збереження обраних об’єктів чи історії звернень, а пошук задовольняє лише базові запити без можливості вказати ціновий діапазон або додаткові параметри (стан ремонту, наявність меблів тощо). Відсутність інтеграції з картою та автоматизованих нагадувань про перегляди також ускладнює планування зустрічей. Крім того, через велику кількість зображень та скриптів сторінки іноді завантажуються з затримкою на мобільних пристроях, що може негативно вплинути на користувацький досвід .

Служба нерухомості «Волинь» позиціонує себе як локальний лідер із понад десятирічним досвідом роботи в Луцьку та його передмісті, надаючи повний спектр послуг із купівлі, продажу та оренди житлової нерухомості [6]. З першого погляду на головній сторінці користувача зустрічає широкий набір фільтрів: операція (купівля/оренда), категорія об’єктів (квартири, кімнати, будинки), кількість кімнат, поверх і навіть загальна кількість поверхів у будинку (рис. 1.5).

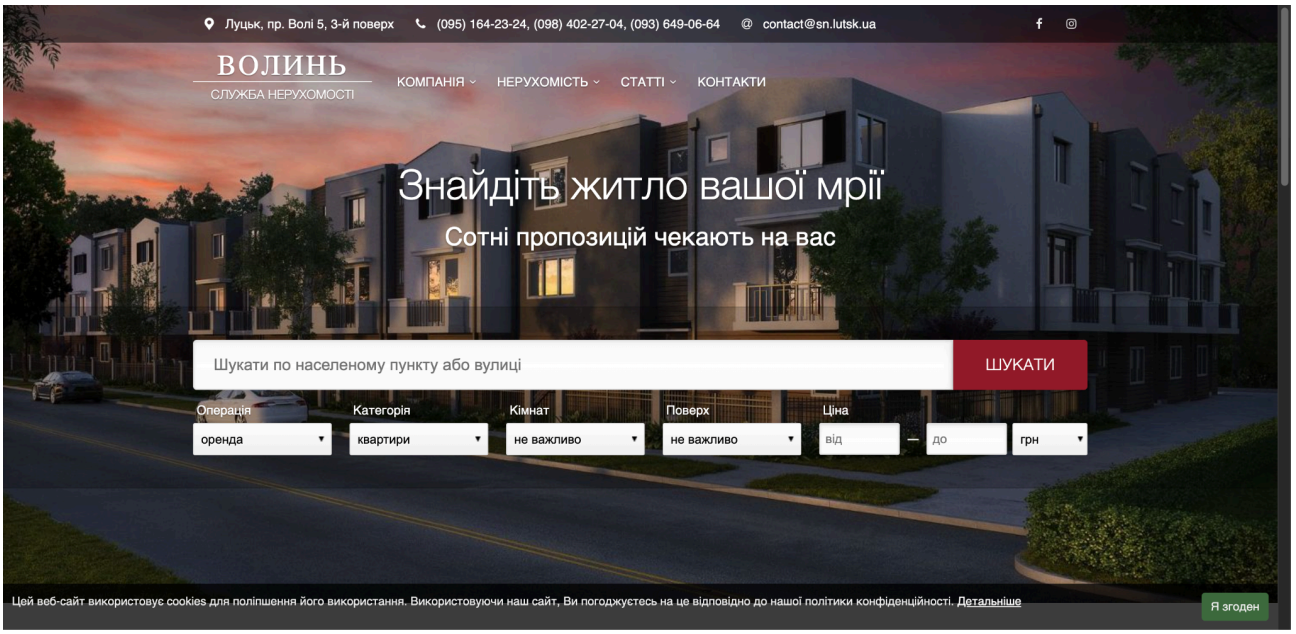

Рисунок 1.5 – Сайт служби нерухомості «Волинь» [6]

Цей рівень деталізації дає змогу швидко відсіювати нерелевантні пропозиції, не вдаючись до довгого перегляду списків, та миттєво знайти оптимальні варіанти житла.

Низка розділів «Останні об’єкти» та «Останні статті» демонструє свіжі пропозиції й аналітичний контент, що утримує увагу користувача й стимулює довіру до агенції. На сайті також інтегровані актуальні курси валют, що важливо для клієнтів, які порівнюють ціни в різних валютах. Присутність блоків «Про компанію» та «Контакти» з фотографіями і прямими телефонами менеджерів створює відчуття прозорості та доступності сервісу.

Водночас платформа не пропонує особистого кабінету: немає можливості зберігати вибрані пропозиції, слідкувати за статусом запитів або отримувати нагадування про перегляди. Відсутня інтеграція з картами, що ускладнює візуальне оцінювання розташування об’єктів, а також календар для планування оглядів нерухомості. Додатково, хоча фільтри й детальні, бракує можливості задати ціновий діапазон у валюті та відсортувати за датою додавання оголошення, що могло б підвищити гнучкість пошуку для користувачів зі специфічними вимогами.

Проведений аналіз показав, що локальні сервіси нерухомості в Луцьку пропонують базовий функціонал: розділи «Продаж»/«Оренда», первинні фільтри за типом об’єкта, кількістю кімнат і площею, а також контактні форми й контентні блоки (блоги, відгуки). Водночас жоден із конкурентів не забезпечує персоналізованого досвіду: відсутні облікові записи з історією звернень, гнучкі багатопараметричні фільтри (ціна, стан ремонту, локалізація на карті) та автоматизовані інструменти планування (календар показів, нагадування). Поряд із цим помітні проблеми з продуктивністю на мобільних пристроях через велику кількість графічних елементів і скриптів. Отже, розроблювана платформа «Elite» повинна поєднати сильні сторони існуючих рішень (прозорий UI, корисний контент, оперативний зворотний зв’язок) з розширеними можливостями персоналізації, поглибленою фільтрацією, інтегрованою картою та засобами автоматизації клієнтського шляху.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи бакалавра є створення вебплатформи для агенції нерухомості «Elite», яка забезпечить керування клієнтами, об’єктами нерухомості та угодами з використанням фреймворку Laravel і СУБД MySQL.

Для досягнення поставленої мети в роботі було сформульовано такі завдання:

– проаналізувати предметну область та існуючі рішення на ринку нерухомості;

– визначити та формалізувати вимоги до платформи;

– спроєктувати архітектуру системи та схему даних;

– обґрунтувати вибір технологічного стеку;

– реалізувати середовище та ключові функціональні модулі;

– інтегрувати адміністративну панель, обробку медіа та розсилки;

– провести тестування та налагодження;

– забезпечити безпеку та підготувати реліз.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ ПЛАТФОРМИ ДЛЯ КЕРУВАННЯ ПОСЛУГАМИ АГЕНЦІЇ НЕРУХОМОСТІ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Успішна розробка будь-якої програмної системи починається з чіткого та всебічного визначення вимог. Саме на цьому етапі встановлюються межі та склад функціоналу, уточнюються очікувані характеристики продуктивності, надійності та безпеки, а також оцінюються потреби кінцевих користувачів і бізнес-процеси, які система покликана автоматизувати. Недостатньо деталізовані або невідповідні реальним потребам вимоги призводять до «роздуття» обсягу робіт, зростання витрат на доробки та низької якості кінцевого продукту. Тому аналіз та формалізація функціональних і нефункціональних вимог є ключовим кроком, який гарантує, що подальше проектування архітектури, моделювання даних і вибір технологій будуть максимально відповідають поставленим цілям і обмеженням проєкту.

На основі потреб агенції були виокремлені ключові функціональні та нефункціональні вимоги до платформи «Elite». Функціональні вимоги охоплюють керування статичними сторінками, повноцінний CRUD-інтерфейс для об’єктів нерухомості, зручні інтерфейси фронтенду з фільтрацією та деталізацією пропозицій, а також механізми збирання зворотного зв’язку від клієнтів. Нефункціональні включають рівні продуктивності, надійності, безпеки та доступності, що гарантують високу якість роботи системи у різних умовах та відповідність сучасним стандартам.

Функціональні вимоги.

1. Динамічне керування статичними сторінками:

– створення, редагування та видалення інформаційних розділів (наприклад, «Про нас», «Контакти») через адмін-панель із WYSIWYG-редактором;

– збереження структури меню сайту на основі порядку записів у таблиці pages.

2. Керування об’єктами нерухомості:

– можливість додавати нові квартири з атрибутами: назва, опис, ціна, площа, кількість кімнат, адреса, геокоординати;

– додавання кількох фотографій до кожного об’єкта з автоматичним створенням прев’ю;

– зміна порядку відображення та активація/деактивація об’єктів для публікації на фронтенді.

3. Публікація інформації на фронтенді:

– відображення на головній сторінці вибраних об’єктів із короткою інформацією та фото;

– сторінка списку квартир із фільтрами за ціною, площею та кількістю кімнат;

– сторінка деталізації об’єкта з галереєю, картою і формою «Залишити заявку»;

– автоматичне SEO-генерування метаданих (<title>, <meta description>) на основі полів у БД.

4. Контактна форма:

– надсилання повідомлень із фронтенду на email агенції з полями: ім’я, телефон, e-mail, повідомлення;

– збереження звернень у базі для подальшої обробки адміністраторами.

5. Адміністративний доступ:

– автентифікація через вбудований Laravel Auth або OAuth-плагін Backpack;

– розмежування прав доступу: лише адміністратори можуть редагувати сутності.

Нефункціональні вимоги.

1. Продуктивність та масштабованість:

– час відповіді сервера на запит фронтенду не більше 200 мс під навантаженням до 50 одночасних користувачів;

– підтримка горизонтального масштабування вебсервера (статичні файли віддаються через CDN).

2. Надійність та доступність:

– гарантований час безвідмовної роботи (uptime) ≥ 99,5 %;

– резервне копіювання бази MySQL щонайменше раз на добу.

3. Безпека:

– захист від SQL-ін’єкцій через використання Eloquent ORM та перевірку вхідних даних;

– CSRF-захист для всіх форм (middleware Laravel);

– сховище паролів із використанням bcrypt;

– HTTPS-шифрування для всіх сторінок.

4. Юзабіліті та доступність:

– адаптивний дизайн відповідно до Mobile-First підходу (Bootstrap 5);

– підтримка WCAG 2.1 на рівні AA (контрастність, опис alt для зображень, навігація клавіатурою);

– інтуїтивно зрозумілий інтерфейс адмін-панелі Backpack із локалізацією українською мовою.

5. Підтримка та супровід:

– документація для розгортання (README) та оновлення платформи;

– CI/CD-конвеєр (GitHub Actions) для автоматичної перевірки та деплою на тестовий сервер.

Підсумовуючи, сформовані вимоги закладають основу для проєктування архітектури та вибору технологій. Дотримання цих вимог у процесі розробки дозволить створити платформу з інтуїтивним інтерфейсом, надійною роботою бекенду та зручною адмін-панеллю, здатну задовольнити бізнес-цілі агенції «Elite» та очікування її клієнтів.

Після формулювання ключових вимог до системи наступним кроком стане побудова UML-моделей, що дозволить візуалізувати структуру, поведінку

та архітектуру платформи «Elite». Діаграми Use-Case дадуть змогу чітко окреслити взаємодію користувачів і адміністратора з основними функціональними модулями, а Class Diagram відобразить об’єктну модель і зв’язки між сутностями. Sequence та Activity Diagram допоможуть промалювати послідовність виконання основних сценаріїв, тоді як Component та Deployment Diagram закрілять на рівні компонентів і розгортання технічну інфраструктуру рішення. Разом вони сформують надійну основу для подальшої розробки та забезпечать узгодженість проектних рішень.

Діаграма прецедентів (Use-Case) відображає взаємодію основних акторів (Гість/Клієнт, Адміністратор) із системою: перегляд списку квартир, пошук і фільтрація, перегляд деталей, надсилання запиту, управління сторінками й об’єктами (рис. 2.1).

Рисунок 2.1 – Діаграма прецедентів (Use-Case)

Діаграма ілюструє два типи користувачів платформи «Elite»: «Клієнт», який має доступ до перегляду каталогу квартир, пошуку з фільтрами, ознайомлення з деталями кожного об’єкта та надсилання запитів на контакт із

агентом, та «Адміністратор», який через адмін-панель виконує управління контентом сайту – створення й редагування статичних сторінок та розділів з об’єктами нерухомості. Розподіл прецедентів по пакунках чітко показує, які функції доступні кінцевому відвідувачу, а які – лише для адміністративного персоналу. Ця модель допомагає узгодити очікування користувачів із розробкою інтерфейсів та бекенд-логіки.

Клас-діаграма (Class Diagram) відображає основні сутності платформи «Elite» та їхні зв’язки (рис. 2.2).

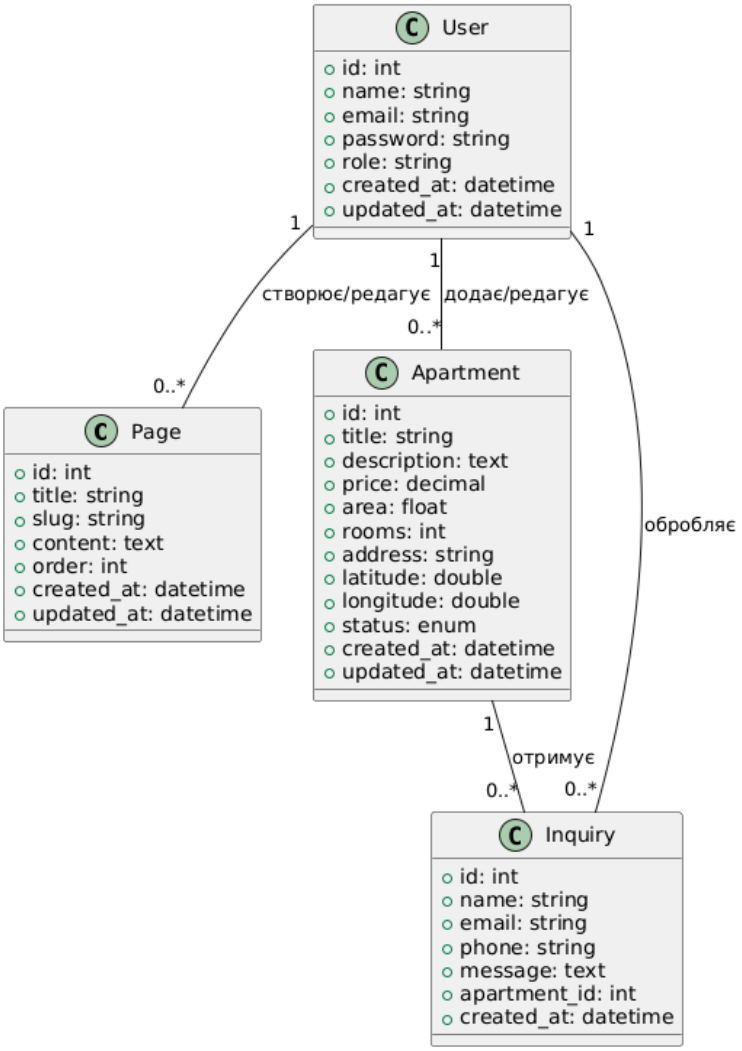

Рисунок 2.2 – Клас-діаграма (Class Diagram)

Клас Page відповідає за статичний контент сайту (текстові розділи), Apartment – за об’єкти нерухомості з усіма характеристиками та геолокацією,

User (адміністратор) керує даними і має рольові атрибути, а Inquiry зберігає звернення клієнтів по кожній квартирі. Зв’язки демонструють, що один адміністратор може створювати/редагувати довільну кількість сторінок та об’єктів, а також обробляти численні запити клієнтів; кожен об’єкт Apartment може отримувати багато звернень. Це забезпечує цілісну модель даних для реалізації всіх CRUD-операцій і управління бізнес-логікою платформи.

У наступних трьох діаграмах послідовностей (Sequence Diagrams) показано ключові сценарії роботи платформи «Elite».

Перша діаграма ілюструє процес перегляду деталей квартири клієнтом та надсилання запиту через форму на сайті (рис. 2.3).

Рисунок 2.3 – Діаграмах послідовностей (Sequence Diagrams)

У цьому сценарії «Клієнт» через інтерфейс фронтенду запитує детальну інформацію про об’єкт нерухомості, бекенд звертається до бази даних та повертає всі необхідні поля (фотографії, опис, характеристики, карту). Після ознайомлення Клієнт заповнює форму звернення, і фронтенд виконує POST-запит до сервера, що зберігає новий запис у таблиці inquiries. У результаті користувач отримує повідомлення про успішну відправку запиту.

Друга діаграма демонструє порядок дій адміністратора в адмін-панелі Backpack при створенні або редагуванні об’єкта нерухомості (рис. 2.4).

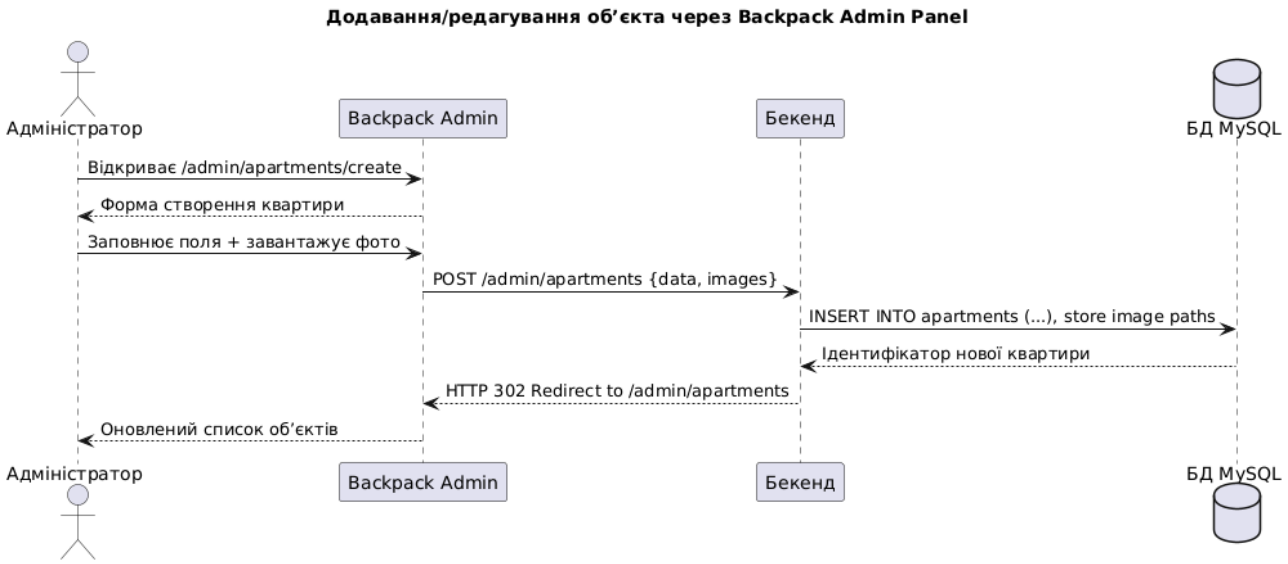

Рисунок 2.4 – Діаграмах послідовностей (Sequence Diagrams)

«Адміністратор» через інтерфейс Backpack відкриває форму створення або редагування квартири. Після заповнення полів та завантаження фотографій адмін-панель надсилає POST-запит до бекенду, який зберігає дані в базі та повертає користувачеві оновлений список об’єктів. Цей потік демонструє повний CRUD-цикл у адміністративному модулі.

Третя діаграма відображає механізм ініціалізації та доставки email-повідомлень клієнтові та адміністратору після отримання запиту (рис. 2.5).

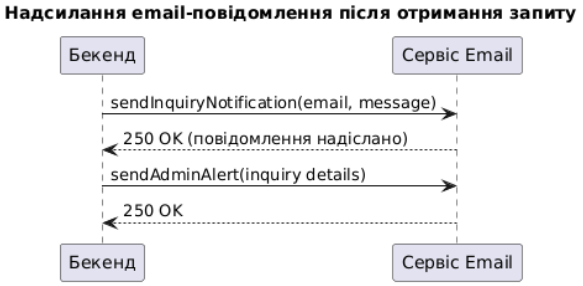

Рисунок 2.5 – Діаграмах послідовностей (Sequence Diagrams)

Після успішного збереження звернення в таблиці inquiries бекенд ініціює виклик зовнішнього сервісу Email для двох дій: підтвердження клієнту про отримання запиту та сповіщення адміністратора. Сервіс повертає статус відправлення, що дозволяє серверу зафіксувати успіх або обробити помилку.

Наступна діаграма активності (Activity Diagram) ілюструє покроковий процес обробки клієнтського запиту – від заповнення форми та валідації даних до збереження звернення в базі та надсилання повідомлень клієнту і адміністратору (рис. 2.6).

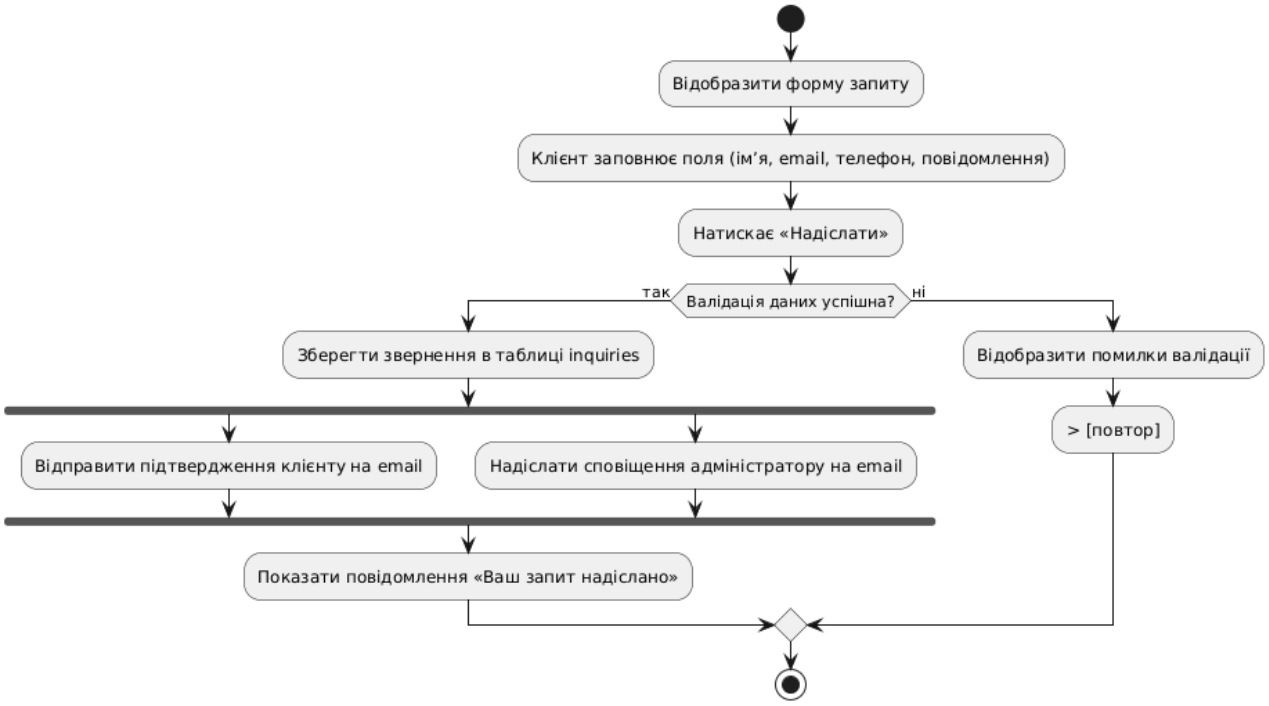

Рисунок 2.6 – Діаграма активності (Activity Diagram)

Діаграма демонструє послідовність дій при обробці клієнтського запиту: від початкового відображення форми та вводу даних користувачем до перевірки валідності полів. У разі успішної валідації система зберігає запис у базі даних і одночасно відправляє два email – підтвердження клієнту й сповіщення адміністратору. Якщо ж валідація не пройдена, користувач бачить відповідні повідомлення про помилки й може виправити введені дані. Такий потік гарантує коректність даних і своєчасне інформування всіх учасників процесу.

Наступна діаграма компонентів (Component Diagram) ілюструє високорівневу структуру платформи «Elite», яка демонструє основні програмні блоки та їхні взаємозв’язки (рис. 2.7).

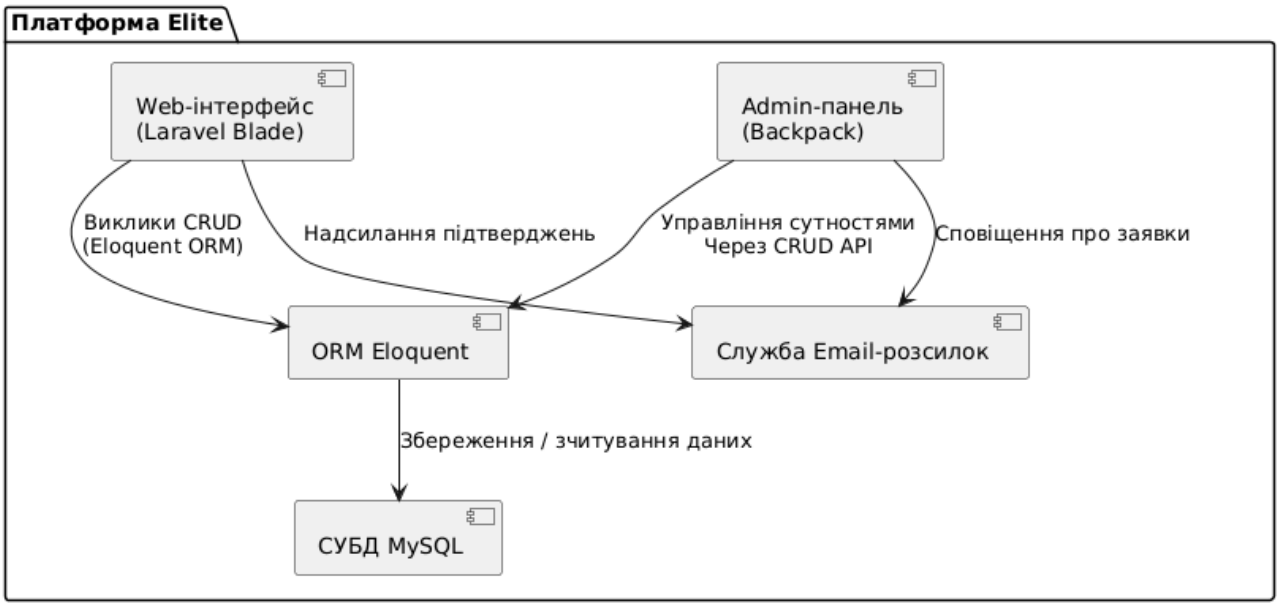

Рисунок 2.7 – Діаграма компонентів (Component Diagram)

На компонентній діаграмі представлені п’ять ключових блоків системи. Web-інтерфейс (Laravel Blade) забезпечує взаємодію кінцевого користувача з платформою, відправляючи запити CRUD через ORM-шар Eloquent, який у свою чергу працює з СУБД MySQL для зберігання всіх даних про сторінки і об’єкти нерухомості. Admin-панель (Backpack) використовує той самий ORM-шар для керування контентом і сутностями, а обидва фронтенди взаємодіють зі Службою Email-розсилок для автоматичного відправлення листів клієнтам і адміністраторам. Така структура забезпечує чітке розділення відповідальності і дозволяє легко масштабувати кожен компонент окремо.

Нижче наведено діаграму (Deployment Diagram) розгортання, яка ілюструє фізичне розміщення ключових компонентів платформи «Elite» – від роздачі статичних ресурсів через CDN до обробки динамічного контенту вебсервером та зберігання даних у базі MySQL (рис. 2.8).

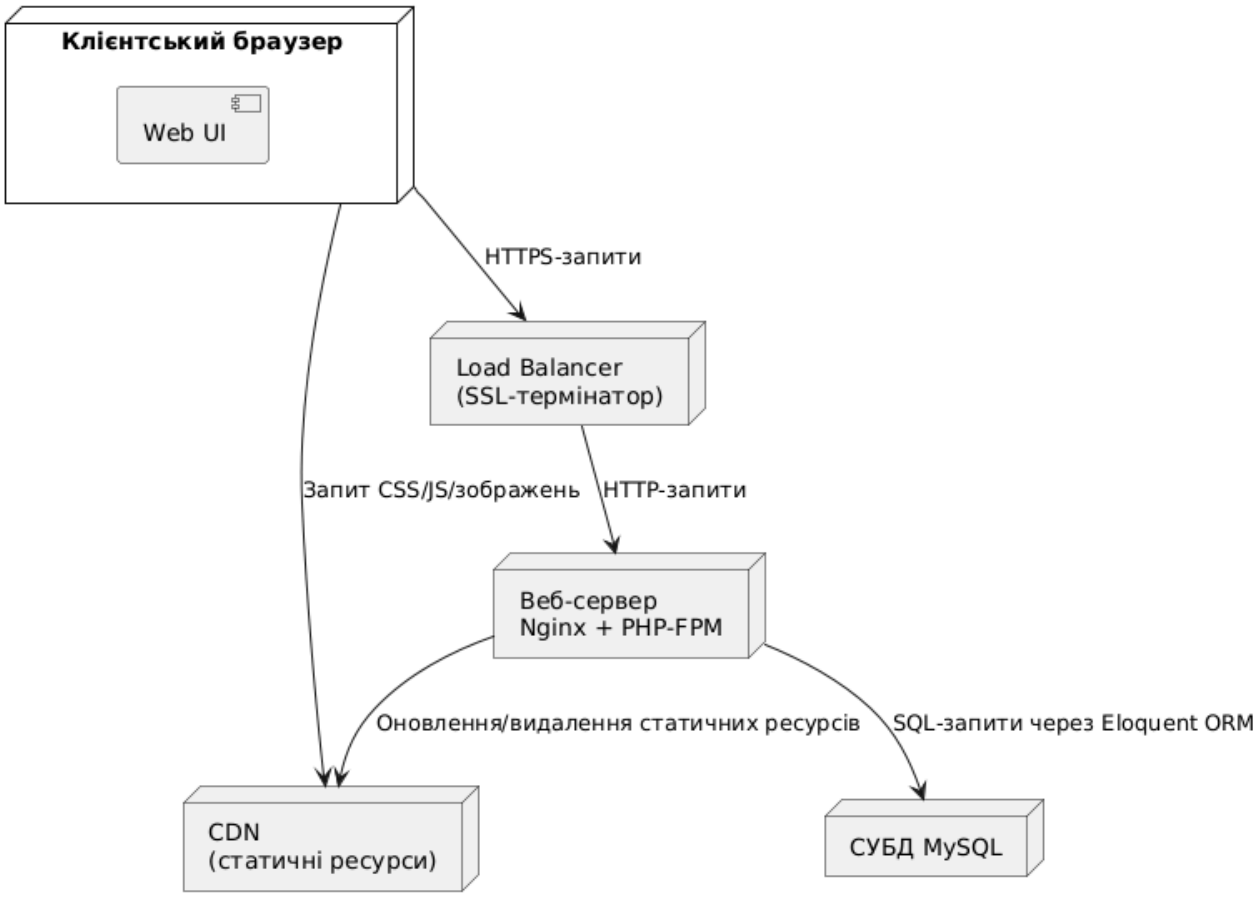

Рисунок 2.8 – Діаграма розгортання (Deployment Diagram)

Діаграма ілюструє фізичне розміщення компонентів платформи «Elite» у рамках серверної інфраструктури. Клієнтський браузер отримує статичні ресурси (стилі, скрипти, зображення) безпосередньо через CDN для максимально швидкого завантаження. Усі динамічні запити спрямовуються на Load Balancer, який виконує SSL-термінацію і розподіляє трафік між екземплярами вебсервера (Nginx + PHP-FPM). Вебсервер обробляє логіку Laravel, звертається до СУБД MySQL через Eloquent ORM для зчитування та збереження даних і може оновлювати файли статичних ресурсів на CDN. Така архітектура забезпечує високу доступність, масштабованість і безпеку платформи.

У процесі UX/UI-дизайну для платформи «Elite» було відпрацьовано інформаційну архітектуру та основні сценарії користувацької взаємодії: від пошуку квартири й фільтрації до перегляду деталей та надсилання запитів.

Спочатку створено низку вайрфреймів і прототипів у Figma, які відображають послідовність кроків (user flow) для різних ролей – гостя-сервісу та адміністратора. На їхній основі розроблено адаптивні макети з урахуванням Mobile-First підходу, що гарантує коректне відображення на смартфонах, планшетах і десктопах.

Особлива увага приділена читабельності тексту, контрастності елементів і ергономіці розміщення даних: важливі кнопки розташовані так, щоб їх легко було натиснути навіть великим пальцем на мобільному екрані. У схемах інформаційних потоків показано, як користувацькі дії генерують запити до бекенду й повертають результати, що допомогло оптимізувати час відгуку й мінімізувати кількість кліків для досягнення цілі. Такий підхід гарантує інтуїтивність використання платформи й високу задоволеність користувачів.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Вибір технологічного стеку, інструментів і методів реалізації є одним із найважливіших етапів проєктування будь-якої інформаційної системи. Саме від цього рішення залежать швидкість розробки, якість коду, зручність подальшого супроводу та масштабованість продукту. Невдало підібрані фреймворки чи бази даних можуть призвести до надмірної складності, втрати продуктивності або великих витрат на підтримку, тоді як оптимальний набір технологій дозволяє сфокусуватися на бізнес-логіці та створити надійний, гнучкий і безпечний сервіс.

У контексті платформи «Elite» правильний вибір засобів означає не просто використання популярних рішень, а побудову єдиного середовища, в якому всі компоненти безшовно взаємодіють між собою. Це включає в себе не лише серверний фреймворк і СУБД, але й адмін-панель, механізми маршрутизації, обробки даних, черги завдань та розсилки, а також підходи до організації шаблонів і стилів. Лише завдяки ретельному обґрунтуванню

кожного вибору можна гарантувати, що система буде стійкою до навантажень, простою в налаштуванні та здатною розвиватися разом із потребами бізнесу.

Вибір PHP 8 у поєднанні з фреймворком Laravel обґрунтований кількома факторами. По-перше, за даними W3Techs, PHP продовжує домінувати як серверна мова – її використовують близько 75,8 % усіх вебсайтів із відомою серверною технологією [7].

Language	Usage in September 2024
PHP	75.8%
Ruby	6.0%
ASP.NET	5.8%
Java	5.0%
JavaScript	3.6%
Scala	3.4%
Static files	1.7%
Python	1.3%
ColdFusion	0.2%
Perl	0.1%
Erlang	0.1%

Рисунок 2.9 – Статистика використання серверних мов програмування для вебсайтів [7]

Основною причиною такої популярності є поєднання низького порогу входження та широкої підтримки: майже кожен хостинг уже за замовчуванням надає середовище для запуску PHP, що знижує витрати на інфраструктуру та дозволяє швидко розгорнути проєкт.

Крім того, за роки існування сформувався величезний екосистемний шар – фреймворки (Laravel, Symfony), системи управління контентом (WordPress, Drupal), численні бібліотеки та пакети через Composer. Постійні інновації в самому ядрі PHP (JIT-компіляція в PHP 8, поліпшена обробка

асинхронних завдань) разом із зрілими інструментами розробки гарантують і високу продуктивність, і сучасні можливості, що робить PHP оптимальним вибором для платформи «Elite».

Одночасно за станом на 2024 рік 86 % PHP-розробників працюють із версією PHP 8, завдяки її значним покращенням продуктивності та впровадженню JIT-компіляції [8].

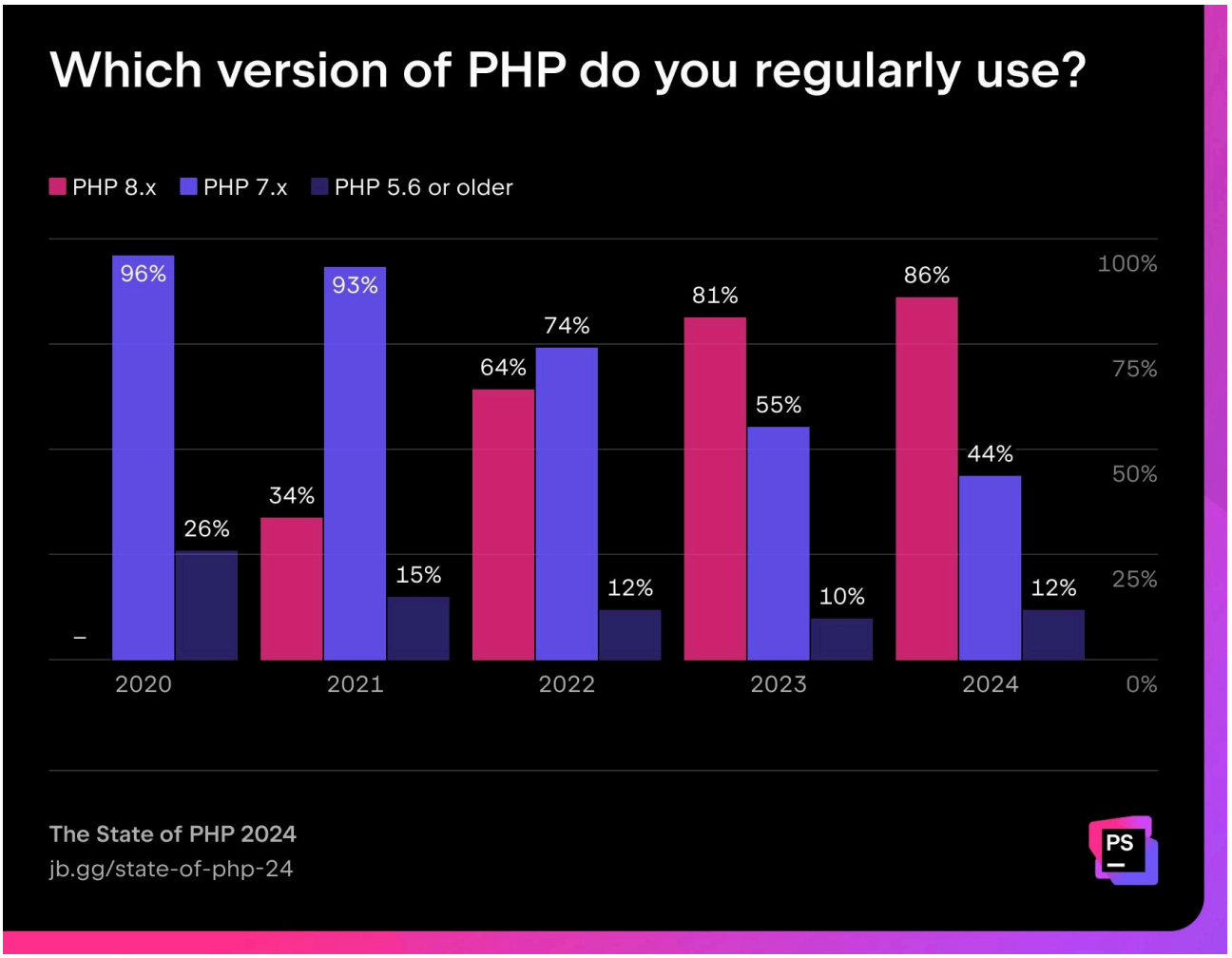

Рисунок 2.10 – Статистика використання версій PHP [8]

По-друге, Laravel упродовж останніх років утримує першість серед PHP-фреймворків: згідно з опитуванням JetBrains, 61 % розробників регулярно використовують Laravel у своїх проєктах (рис. 2.11) [8], а низка аналітичних оглядів підтверджує його лідерство за кількістю завантажень і зірковим рейтингом у спільноті [9]. Основні переваги Laravel – чітка та виразна

MVC-архітектура, гнучка система маршрутів, потужний сервіс-контейнер для впровадження залежностей, вбудована підтримка задач і черг у Artisan, а також Eloquent ORM, який спрощує роботу з базою даних.

Рисунок 2.11 – Статистика використання PHP-фреймворків [8]

Все це разом забезпечує швидкий цикл розробки з мінімальною boilerplate-логікою, легке масштабування та підтримку, а також широкий набір пакунків і плагінів (Forge, Nova, Horizon), які дозволяють реалізувати складні бізнес-процеси без необхідності винаходити «велосипед». Такий стек гарантує створення безпечної, продуктивної та зручною в підтримці вебплатформи для агенції «Elite».

Eloquent ORM, який є вбудованим шаром взаємодії з базою в Laravel, реалізує патерн Active Record, де кожна модель відображає окрему таблицю в MySQL. Він надає розробникам виразний та лаконічний синтаксис для виконання CRUD-операцій, визначення взаємозв’язків між сутностями (наприклад, hasMany, belongsToMany), а також можливість використовувати «жирні» фільтри (scopes), eager loading і кастомні колекції.

Завдяки вбудованій підтримці міграцій і сидерів Eloquent дозволяє версіонувати схему бази даних і швидко розгортати оновлення в різних середовищах, що значно спрощує процес безперервної доставки. Крім того, численні пакети та розширення (наприклад, Laravel Scout для повнотекстового пошуку або кешування результатів запитів через Redis) інтегруються «з коробки», що робить Eloquent одним з найкомплектніших ORM-рішень на ринку.

MySQL залишається одним із найпопулярніших реляційних СУБД на ринку, посідаючи місце в ТОП-3 у щомісячному рейтингу від DB-Engines за індексом популярності [10].

За даними WebTechSurvey, понад 14,9 млн активних сайтів використовують MySQL як основну базу даних [11]. Висока поширеність пояснюється відкритим ліцензуванням, кросплатформністю, широкою підтримкою у хостинг-провайдерів і багатою екосистемою інструментів (реплікація, кластери, інструменти бекапу), що робить MySQL універсальним рішенням для вебдодатків будь-якого масштабу.

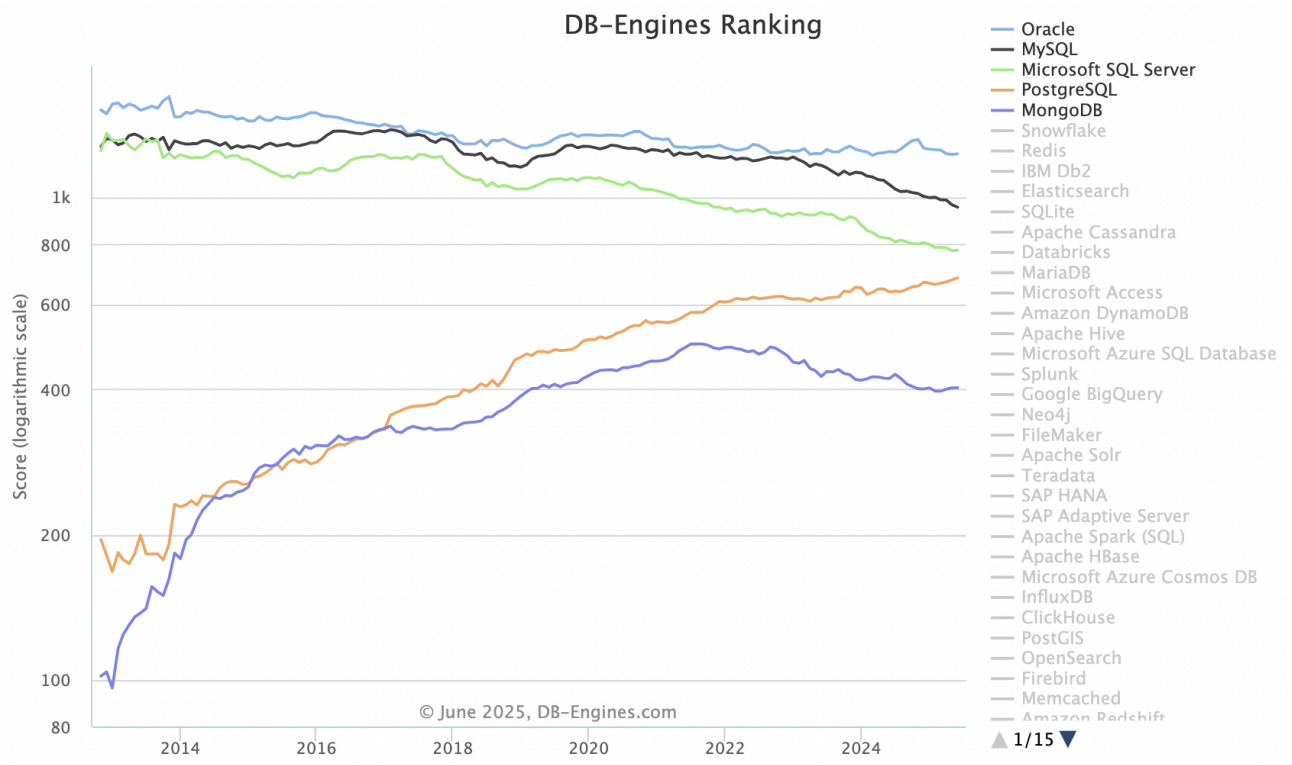

Рисунок 2.12 – Рейтинг DB-движків [12]

Вибір движка InnoDB для зберігання даних забезпечує повну підтримку ACID-транзакцій, декларативну цілісність через зовнішні ключі, механізми відновлення після збоїв та блокування на рівні рядка для оптимального паралелізму [12]. Починаючи з версії MySQL 5.5.5 (2010), InnoDB є сховищем за замовчуванням, що гарантує надійність і узгодженість даних при інтенсивних операціях CRUD. Таке поєднання MySQL + InnoDB ідеально інтегрується з Eloquent ORM у Laravel, дозволяючи безпечно і ефективно оперувати сутностями платформи «Elite» через простий і зрозумілий API.

Адмін-панель для Laravel реалізована за допомогою Backpack for Laravel, який поєднує в собі набір готових CRUD-пакетів і виджетів для швидкого створення адаптивної адмінки без необхідності «переписувати колесо». Backpack існує з 2016 року та завоював популярність завдяки простоті в налаштуванні маршрутів, форм і полів, розширюваному сервіс-контейнеру Laravel та гнучким механізмам авторизації й валідації даних [13]. Його репозиторій на GitHub підтверджує активну розробку й підтримку спільнотою, а багата екосистема офіційних і сторонніх аддонів (графіки, виджети, теми)

дозволяє легко підлаштуватися під будь-які бізнес-вимоги. Завдяки Backpack адміністратори можуть через інтуїтивний інтерфейс керувати сторінками, об’єктами нерухомості, зверненнями клієнтів, а також швидко налаштовувати фільтри та сортування без додаткового коду, що значно зменшує час реалізації та полегшує подальший супровід платформи.

Laravel Blade є типовим шаблонізатором фреймворку, що поєднує простоту синтаксису з високою продуктивністю: усі шаблони .blade.php компілюються у звичайний PHP-код і кешуються до наступної зміни, тому Blade додає практично нульове навантаження на додаток [14]. Blade дозволяє безпосередньо використовувати змінні, цикли, умовні оператори та будь-який PHP-код у межах шаблонів, водночас забезпечуючи захист від XSS-атак через автоматичне екранування виводу.

Крім базових директив (@extends, @section, @yield, @include), Blade підтримує компоненти й слоти, що дозволяє будувати повторювані UI-блоки з мінімальним дублюванням коду. Завдяки кешуванню згенерованих PHP-файлів складні шаблони майже не впливають на час відповіді сервера – навіть багаторівневе вкладення не створює помітних затримок при рендерингу. Така комбінація лаконічного синтаксису, можливості повторного використання компонентів і високої швидкості виконання робить Blade оптимальним вибором для бекенд-в’ю платформи «Elite».

Вибір Bootstrap 5 як основного CSS-фреймворку зумовлений його широкою поширеністю та адаптивним Mobile-First підходом. За даними W3Techs, Bootstrap використовується на 75,8 % сайтів із відомими CSS-фреймворками, що становить 16 % всіх вебсайтів в Інтернеті [15]. Це означає, що Bootstrap постійно підтримується спільнотою, має перевірений часом набір компонентів (сітка, навігація, кнопки, модальні вікна) та забезпечує високу швидкість верстки без необхідності писати кастомні стилі «з нуля». Гнучка система Sass-змінних і утиліти для налаштування тем дозволяють легко підлаштовувати дизайн під бренд агенції «Elite», а готові утиліти утилітарного стилю зменшують обсяг CSS-коду й полегшують підтримку.

Щодо JavaScript, використання мінімальної кількості ванільного коду або jQuery базується на факті його величезної екосистеми та кросбраузерної сумісності. Згідно з W3Techs, jQuery використовується на 90,4 % сайтів зі встановленою JS-бібліотекою, що еквівалентно 73,5 % усіх сайтів Інтернету [16]. Це свідчить про стійкість бібліотеки та наявність безлічі плагінів і вирішень для таких завдань, як анімації, AJAX-запити та взаємодія з DOM. Використання jQuery разом із ванільним JS лише для критичних сценаріїв дозволяє знизити час розробки та забезпечити стабільну роботу інтерактивних елементів на різних пристроях і браузерах, без залучення важких фронтенд-фреймворків.

Composer є стандартним менеджером залежностей для PHP, широко визнаним у спільноті розробників. Він забезпечує єдиний формат опису залежностей у файлі composer.json, автоматично вирішує версії пакетів за допомогою SAT-алгоритму та створює автозавантажувач для усіх інстальованих бібліотек [17]. Основним репозиторієм для Composer є Packagist, що агрегує понад 350 000 публічних пакетів (бібліотек і фреймворків) і слугує точкою входу для поширення PHP-коду.

Використання Composer у проєкті «Elite» дозволяє чітко і прозоро керувати зовнішніми залежностями – від самого Laravel і його офіційних пакунків (Eloquent, Backpack) до сторонніх бібліотек (фільтрації, валідаторів, інтеграцій із SMTP). Завдяки декларативному підходу та можливості замикати точні версії пакетів (composer.lock), забезпечується відтворюваність середовища розробки й продакшену, а також спрощується процес оновлення та підтримки без ризику «зламати» сумісність. Таке рішення відповідає принципам безперервної інтеграції та доставки (CI/CD) та суттєво підвищує стабільність платформи.

Git став стандартом контролю версій: понад 100 мільйонів розробників використовують GitHub для хостингу й спільної роботи над проєктами, а більше ніж 90 % компаній із Fortune 100 застосовують GitHub у своїх процесах розробки [18]. Розподілена модель репозиторіїв, потужні можливості гілкування

та злиття змін, а також інтеграція з платформами GitHub і GitLab роблять Git незамінним інструментом для команд будь-якого масштабу.

Для забезпечення якості коду та пришвидшення випуску виправлень використовується автоматизований CI/CD-пайплайн. Згідно з даними CD Foundation, 83 % розробників практикують CI/CD, впроваджуючи безперервну інтеграцію, тестування й доставку й посилюючи безпеку шляхом включення перевірок у робочий процес [19]. У проєкті платформи «Elite» поєднання Git (GitHub/GitLab) зі зрілим CI/CD забезпечує автоматичне виконання модульних та інтеграційних тестів, перевірку стилю коду й упакування релізів, що мінімізує людські помилки та значно пришвидшує цикл поставки нових функцій.

Для обробки вхідних HTTP-запитів використовується комбінація Nginx та Apache у зв’язці з PHP-FPM. За даними W3Techs, Nginx наразі обслуговує 33,8 % усіх вебсайтів із відомим сервером, а Apache – 26,0 % [20]. Така двоступенева архітектура дозволяє Nginx виступати фронт-ендом (reverse-proxy), який приймає з’єднання, виконує SSL-термінацію та роздає статичні ресурси через CDN із мінімальною затримкою, водночас проксіруючи динамічні запити на Apache+PHP-FPM для обробки Laravel-додатка.

PHP-FPM (FastCGI Process Manager) забезпечує ефективне керування процесами PHP, із розподілом навантаження між робочими воркерами та ізольованим виконанням скриптів, що підвищує стійкість системи під час пікових навантажень. Завдяки SSL-термінації на рівні Nginx забезпечується централізована конфігурація сертифікатів, швидке встановлення TLS-з’єднань і зниження навантаження на бекенд-сервера. Використання CDN (Content Delivery Network) для роздачі статичних ресурсів (CSS, JS, зображень) додатково зменшує час відгуку для користувачів, розсилаючи контент із географічно наближених вузлів, та дозволяє вебсерверу зосередитися на генерації динамічних відповідей.

Для надсилання email-повідомлень у платформі «Elite» використовується компонент Laravel Mail, який надає єдиний API над поштовими драйверами

(SMTP, Mailgun, SES, а також логування у файл) та базується на бібліотеці SwiftMailer/Symfony Mailer [21]. Завдяки конфігурації в .env (MAIL_MAILER, MAIL_HOST, MAIL_PORT, MAIL_USERNAME, MAIL_PASSWORD тощо) легко переключатися між локальним SMTP-сервером та сторонніми провайдерами (SendGrid, Mailtrap, Amazon SES), забезпечуючи високий рівень доставності та моніторинг відправлень.

Крім базових можливостей, Laravel Mail підтримує чергування повідомлень через вбудований драйвер queue, що дозволяє виконувати відправку асинхронно й уникнути блокування HTTP-запитів. Для оформлення листів використовуються Markdown Mailables – шаблони на основі Markdown, які автоматично генерують адаптивні HTML-версії з plain-text альтернативою й забезпечують захист від XSS через автоматичне екранування виводу. Це поєднання простоти налаштування, гнучкості шаблонів та надійності доставлення робить Laravel Mail оптимальним вибором для реалізації надійного і масштабованого сервісу розсилок у проєкті.

У результаті обґрунтування був сформований оптимальний стек технологій для платформи «Elite»: на боці сервера використовується PHP 8 із фреймворком Laravel (MVC, сервіс-контейнер, маршрутизація) та Eloquent ORM над MySQL/InnoDB для надійного зберігання даних; адмін-панель побудована на Backpack for Laravel для швидкого CRUD-інтерфейсу; представлення здійснюється через Blade-шаблони й Bootstrap 5 із мінімальною кількістю JS/jQuery для адаптивного Mobile-First UI; залежності керуються Composer, версії – Git із CI/CD-конвеєром; HTTP-запити обробляє зв’язка Nginx/Apache+PHP-FPM із SSL-термінацією та CDN для статики; і, нарешті, поштові повідомлення надсилаються через Laravel Mail із підтримкою SMTP та черг. Такий набір інструментів забезпечує баланс між швидкістю розробки, продуктивністю й масштабованістю рішення.

РОЗДІЛ 3
РОЗРОБКА ПЛАТФОРМИ ДЛЯ КЕРУВАННЯ ПОСЛУГАМИ АГЕНЦІЇ НЕРУХОМОСТІ

3.1 Практична реалізація об’єкта проектування

На початку реалізації платформи необхідно підготувати локальне середовище розробки, яке точно відтворюватиме production-конфігурацію та дозволить швидко налагоджувати всі сервіси. Першим кроком став вибір та інсталяція PHP 8 – ця версія забезпечує максимальну сумісність із Laravel 10 та містить JIT-компілятор для прискорення критичних фрагментів коду. Після встановлення PHP через менеджер пакетів ОС (apt/yum/brew) відразу ж інсталюється Composer – стандартний інструмент для керування залежностями в екосистемі PHP.

Наступним етапом було розгортання СУБД MySQL (версія 8.x) та вебсервера. Ми обрали зв’язку Nginx + PHP-FPM, проте можна використовувати й Apache + PHP-FPM за аналогічною логікою. Під час інсталяції налаштовується окремий віртуальний хост, де DocumentRoot вказує на папку public/ Laravel-додатку.

Далі йде клонування вихідного коду з репозиторію та встановлення PHP-залежностей командою composer install, яка завантажує Laravel, пакети Backpack, SwiftMailer, інші бібліотеки і формує автозавантажувач. Паралельно копіюємо приклад файлу оточення (додаток А).

Після збереження .env генерується унікальний ключ програми командою: php artisan key:generate.

Далі створюється символічне посилання для папки storage, щоб публічні активи (завантажені через адмінку фото об’єктів) були доступні з вебсерверу: php artisan storage:link.

На фінальному кроці виконуємо запуск міграцій і сидерів: php artisan migrate --seed.

Це створює всі таблиці (users, pages, apartments, inquiries тощо) та заповнює тестовими записами для швидкого старту роботи інтерфейсу.

Після проходження цих кроків локальне середовище готове: можна відкрити адресу http://elite.local у браузері та переконатися, що головна сторінка завантажується без помилок, а вхід у адмін-панель /admin працює згідно з правами, заданими в сидерах. Це забезпечує єдину, стандартизовану платформу для подальшої розробки, тестування та демонстрації клієнтам.

Структура проекту побудована за стандартами фреймворка Laravel із кількома додатковими папками для кастомної функціональності (рис. 3.1).

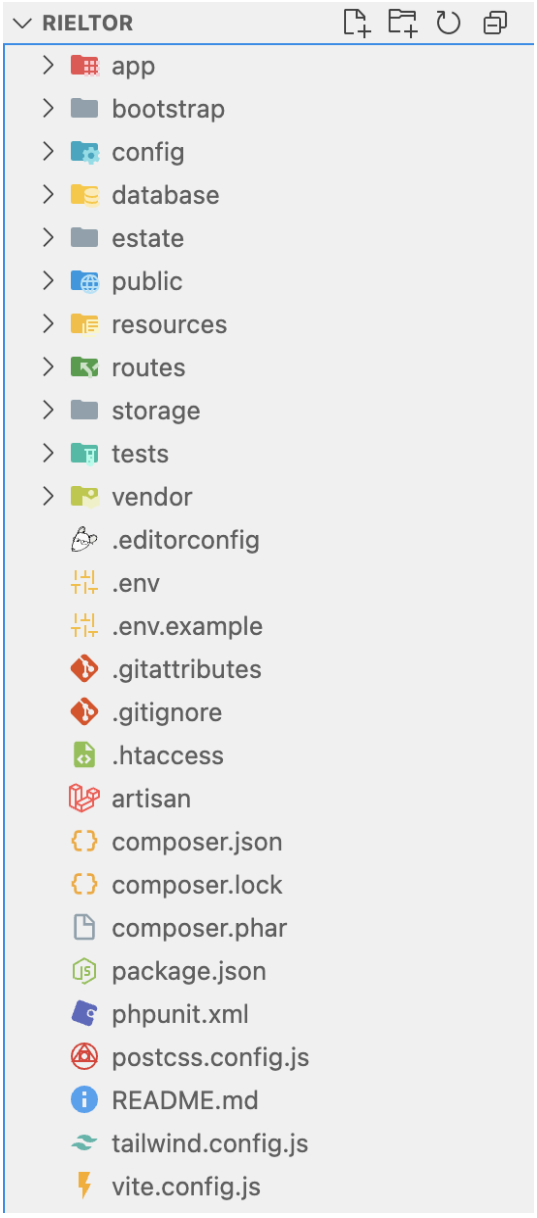

Рисунок 3.1 – Структура проекту

Каталог app/. У цій папці зосереджена вся бізнес-логіка: тут розміщені контролери, що обробляють HTTP-запити; моделі Eloquent для роботи з базою даних; middleware, які перехоплюють і змінюють запити на рівні HTTP; Form Request-класи для валідації вхідних даних; а також довільні хелпери й інші служби, що становлять «ядро» застосунку.

Каталог bootstrap/. Містить скрипти ініціалізації програми: файл app.php завантажує автозавантажувач Composer і створює екземпляр об'єкта Application, а cache/ – підкаталог зберігає скомпільовані файли налаштувань і маршрути, що пришвидшує запуск у продакшені.

Каталог config/. Універсальне місце для всіх конфігураційних файлів Laravel та додаткових пакетів (наприклад, Backpack): тут задаються з'єднання з базою даних, налаштування пошти, кешування, черги, налаштування середовища адмін-панелі тощо.

Каталог database/:

– migrations/ – PHP-класи для управління схемою бази даних (створення та зміна таблиць);

– seeders/ – класи для початкового заповнення таблиць тестовими чи довідковими даними;

– factories/ – генератори фейкових записів для тестування.

Каталог estate/. Власна папка модуля, пов'язана з бізнес-логікою нерухомості: тут містяться спеціалізовані класи, напрями підсистеми та графічні компоненти, що не підпадають під типову структуру app/.

Каталог public/. Відповідає за публічну частину: тут лежать точка входу index.php, ресурси (статичні CSS, JS, зображення), файли .htaccess чи web.config для налаштування вебсервера, а також будь-які статичні активи, доступні прямо з браузера.

Каталог resources/:

– views/ – Blade-шаблони фронтенд-інтерфейсу, включаючи master-layout-и, partial-и й компоненти;

– lang/ – файли перекладів;

– sass/ або js/ – початкові вихідники стилів і скриптів, що потім збираються за допомогою Vite.

Каталог routes/. Тут визначені всі HTTP-маршрути:

– web.php – основні маршрути для користувацької частини;

– backpack/custom.php – додаткові роутер для адмін-панелі, що працює через пакет Backpack.

Каталог storage/. Сховище для згенерованих файлів: логи (logs/), кеш (framework/cache), сесії (framework/sessions), завантажені файли (app/) та інші тимчасові дані.

Каталог tests/. У цій папці – автоматизовані тести: Feature-тести для HTTP-запитів і Unit-тести для окремих класів. Завдяки PHPUnit перевіряється коректність роботи бізнес-логіки та маршрутизації.

Каталог vendor/. Автоматично генерується Composer і містить всі сторонні бібліотеки та автозавантажувач.

Корінь проекту:

– файли налаштування середовища: .env (конфіденційні ключі й параметри), .env.example для прикладу;

– конфігураційні файли збірки фронтенду: package.json, vite.config.js, tailwind.config.js, postcss.config.js;

– файли Composer: composer.json, composer.lock, composer.phar;

– інфраструктурні файли: .editorconfig, .gitignore, artisan (CLI-інтерфейс Laravel), phpunit.xml для тестів.

Завдяки такій чіткій організації коду легко орієнтуватися в проектах будь-якого масштабу, швидко знаходити потрібний клас чи ресурс, а також розширювати функціонал без надмірного «засмічення» однієї з папок.

У config/database.php за замовчуванням встановлено параметри з’єднання з MySQL, які підхоплюються з файлу середовища .env (ключі DB_HOST, DB_PORT, DB_DATABASE, DB_USERNAME, DB_PASSWORD). Це дозволяє легко переключатися між локальною, тестовою та продакшн-базами без зміни коду.

Для створення структури БД були послідовно згенеровані Artisan-міграції:

– users: стандартні поля id, name, email, password, timestamps;

– pages: поля id, title, slug, content, short_description, timestamps;

– estates: поля id, title, address, price, google_map_link, timestamps;

– inquiries: поля id, estate_id (foreign key), name, email, message, timestamps.

Для заповнення бази початковими даними реалізовані сидери:

– UsersTableSeeder – створює адміністратора та тестових користувачів;

– PagesTableSeeder – додає кілька базових сторінок (Про нас, Контакти);

–EstatesTableSeeder – генерує демонстраційні об’єкти нерухомості з координатами;

– InquiriesTableSeeder – імітує звернення клієнтів до системи.

У результаті налаштування БД виявилися гнучкими та легко відтворюваними на будь-якому оточенні, а структура таблиць повністю відповідає вимогам предметної області.

У файлі routes/web.php (ліст. 3.1) зосереджено всі публічні (гостьові) маршрути, необхідні для роботи клієнтської частини сайту. Тут визначено шлях до головної сторінки (/), маршрути для перегляду статичних сторінок (/page/{slug}), списку об’єктів нерухомості (/estates) і детального перегляду кожного об’єкта (/estates/{id}). Для обробки контактних форм і запитів на бронювання використовуються POST-маршрути (/inquiry), що приймають дані з форм, валідують їх у Form Request-класах і потім перенаправляють користувача із повідомленням про успішну відправку.

Лістинг 3.1 – Файлі routes/web.php

```
<?php
use Illuminate\Support\Facades\Route;
use Illuminate\Support\Facades\URL;
use App\Http\Controllers\SitePageController;
use App\Http\Controllers\SiteEstateController;

// Delete fix after
if(isset($_SERVER['SERVER_NAME']) &&
parse_url(env('APP_URL'))['host'] != $_SERVER['SERVER_NAME'])
     exit;
```

```
Route::get('/', [SiteEstateController::class,
'index'])->name('home');
Route::get('/estate/{id}', [SiteEstateController::class,
'show'])->name('estate');

Route::get('/{slug}', [SitePageController::class,
'show'])->name('page');

URL::forceScheme('https');
```

кінець лістингу 3.1

Адміністрування ресурсу здійснюється через пакет Backpack: у файлі routes/backpack/custom.php зареєстровано CRUD-маршрути для управління сутностями Page, Estate та Inquiry (ліст. 3.2). Кожен із цих ресурсів автоматично отримує набір маршрутів виду /admin/estates, /admin/estates/create, /admin/estates/{id}/edit тощо, захищених middleware auth та backpack_user, що гарантує доступ лише авторизованим адміністраторам. Така організація маршрутизації забезпечує чітке розділення публічної та адміністративної частин сайту, дозволяє легко масштабувати список керованих ресурсів і дотримуватися REST-конвенцій без додаткових налаштувань.

Лістинг 3.2 – Файлі routes/backpack/custom.php

```
<?php

use Illuminate\Support\Facades\Route;

// --------------------------
// Custom Backpack Routes
// --------------------------
// This route file is loaded automatically by Backpack\CRUD.
// Routes you generate using Backpack\Generators will be placed
here.

Route::group([
    'prefix' => config('backpack.base.route_prefix', 'admin'),
    'middleware' => array_merge(
        (array) config('backpack.base.web_middleware', 'web'),
        (array) config('backpack.base.middleware_key', 'admin')
    ),
    'namespace' => 'App\Http\Controllers\Admin',
], function () { // custom admin routes
    Route::crud('page', 'PageCrudController');
```

```
    Route::crud('estate', 'EstateCrudController');
}); // this should be the absolute last line of this file

/**
 * DO NOT ADD ANYTHING HERE.
 */
```

кінець лістингу 3.2

У проєкті для керування адміністративною панеллю було підключено стандартну систему автентифікації Laravel і доповнено її власним middleware CheckIfAdmin, яке переконується, що поточний користувач має роль адміністратора (додаток Б). Спочатку було встановлено та запущено скелет Auth (composer require laravel/ui && php artisan ui:auth), після чого виконано міграції таблиці users, додавши поле is_admin типу boolean із значенням за замовчуванням false.

Далі створено middleware CheckIfAdmin, де в методі handle() здійснюється перевірка: якщо користувач не авторизований або в його записі is_admin === false, то відбувається редірект на сторінку входу із повідомленням про недостатні права. Це middleware зареєстровано в app/Http/Kernel.php під ключем admin і застосовано до групи маршрутів /admin у файлі routes/backpack/custom.php, таким чином захищаючи всі CRUD-маршрути від несанкціонованого доступу.

В результаті лише авторизовані користувачі з позначкою is_admin можуть потрапити до бекенд-панелі за /admin, а всі інші підлягають перенаправленню на форму входу. Такий підхід поєднує зручність стандартного Laravel Auth із гнучкістю власного middleware для ролей і дозволяє чітко розмежувати права доступу в системі.

У класі PageCrudController налаштовано відображення списку сторінок із стовпцями title та slug (рис. 3.2).

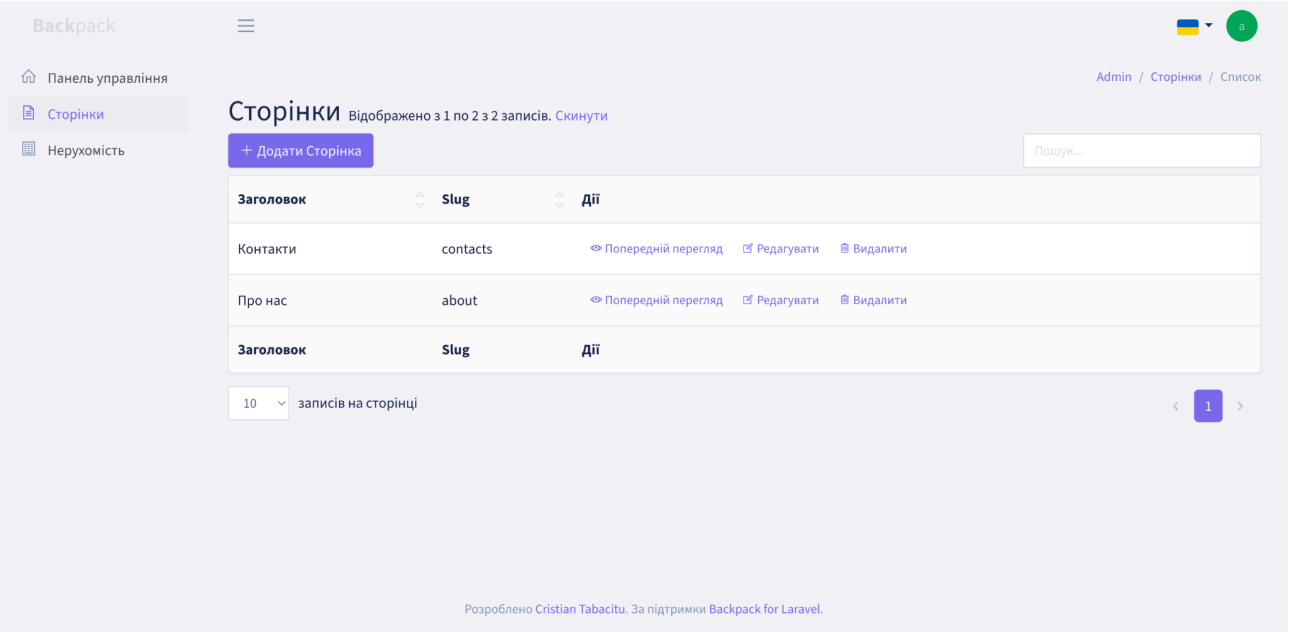

Рисунок 3.2 – Список сторінок у адмінці

Для створення й редагування сторінок передбачена форма з полями title, slug, short_description і content, що реалізовані за допомогою WYSIWYG-редактора; правила валідації цих полів винесені в клас PageRequest, який вимагає обов’язкове заповнення title і slug, унікальність slug і обмеження довжини short_description до 255 символів (рис. 3.3).

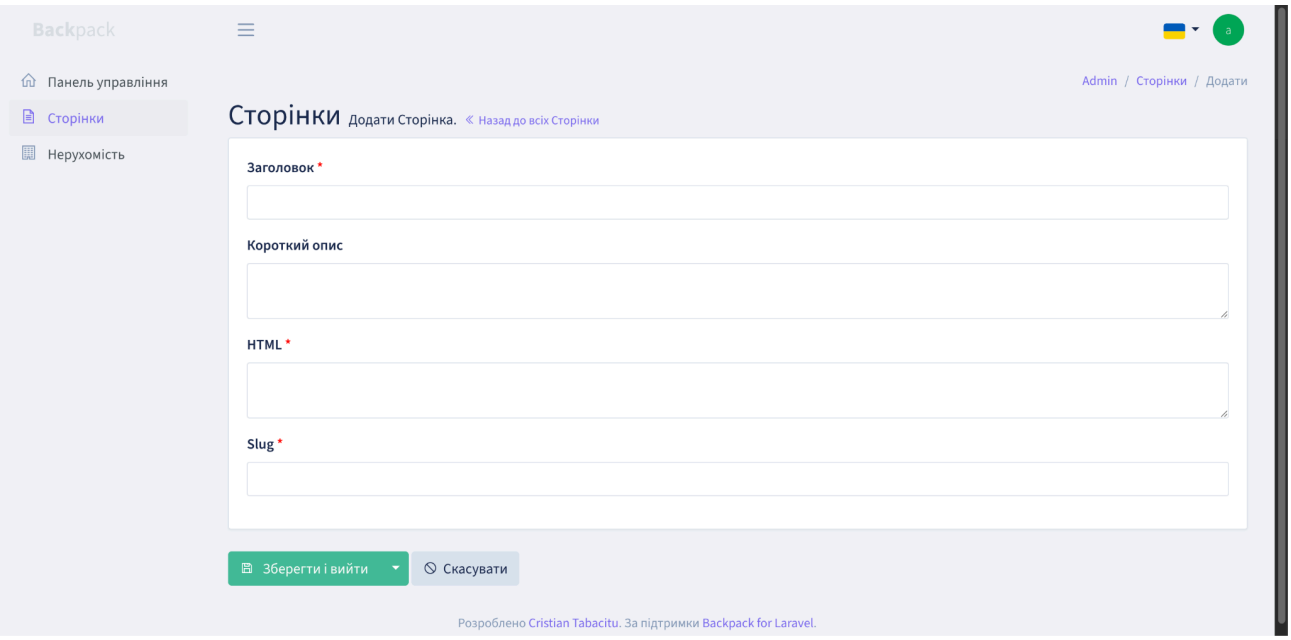

Рисунок 3.3 – Форма створення/редагування сторінки

Аналогічним чином у класі EstateCrudController розгорнуто список об’єктів нерухомості (рис. 3.4).

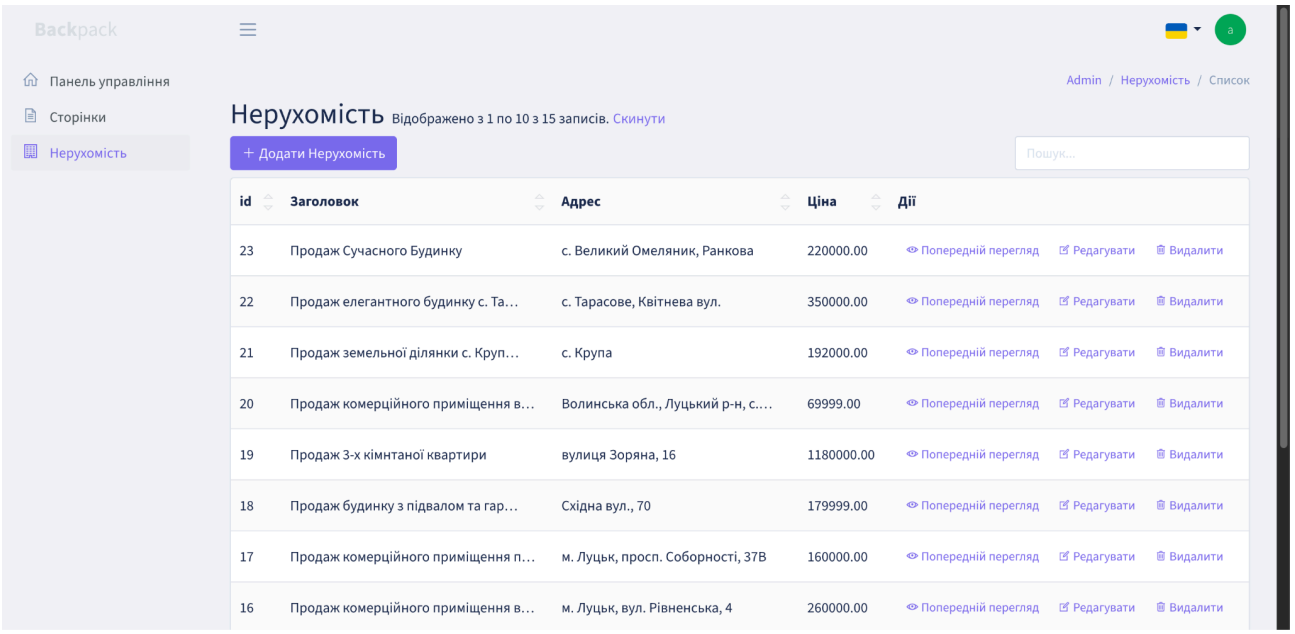

Рисунок 3.4 – Список нерухомості

Форма створення та редагування об’єкта нерухомості містить поля title, address, price, google_map_link і поле для мультизавантаження зображень, які обробляються спеціальним сервісом під час завантаження, а валідація через EstateRequest перевіряє коректність усіх введених даних і формат посилання на карту (рис. 3.5).

Використання CRUD-контролерів Backpack разом із спеціалізованими Request-класами дозволило вирішити одразу кілька завдань. Завдяки автоматичній генерації стандартних операцій створення, читання, оновлення та видалення відпала необхідність вручну писати численні методи-шаблони, що значно скоротило обсяг коду та знизило ймовірність помилок. Окремі Request-класи зосередили правила валідації в одному місці: будь-яка зміна вимог до вводу або додавання нових обмежень вимагає правки лише в одному файлі, а не в багатьох контролерах, що підвищує безпеку та стабільність системи. Це чітке розмежування обов’язків – контролери відповідають виключно за обробку HTTP-запитів та взаємодію з моделями, а Request-класи –

за перевірку вхідних даних – зробило код більш модульним і зручним для модульного тестування.

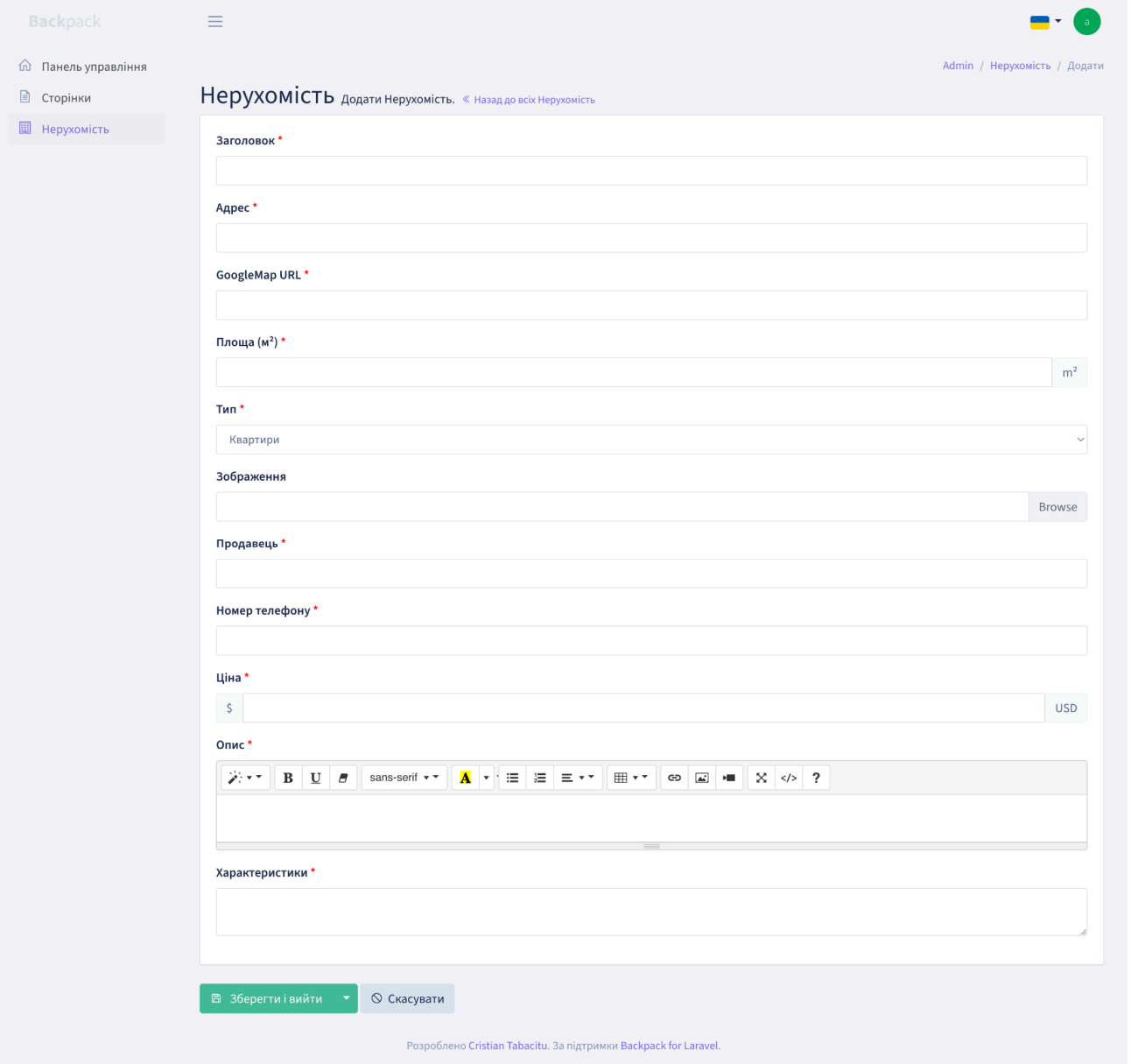

Рисунок 3.5 – Форма CRUD для нерухомості з полем мультизавантаження фото

У ролі фронтенд-контролерів виступають два ключові класи: SitePageController і SiteEstateController, які приймають HTTP-запити від користувачів і повертають відповідні види (Blade-шаблони).

У SitePageController метод index() збирає перелік усіх публічних сторінок і передає їх у шаблон welcome.blade.php, а метод show($slug) шукає у базі сторінку за її унікальним ідентифікатором і рендерить pages/show.blade.php.

Крім цього, до show() може бути доданий обробник контактної форми, який перевіряє вхідні дані через Form Request, надсилає лист адміністратору та повертає назад з повідомленням про результат.

У SiteEstateController метод index() реалізований так, щоб отримати всі доступні об’єкти нерухомості (з пагінацією чи фільтрацією) і передати їх у шаблон estates/index.blade.php, де відображається табличний чи картковий список зображень і основних характеристик. Метод show($id) завантажує деталі конкретного об’єкта, формує модель для відображення карти Google (якщо є), і передбачає обробку заявки користувача: введене ім’я, email та повідомлення зберігаються в таблиці inquiries і адміністратор отримує відповідне сповіщення.

На рисунку 3.6 продемонстровано головну сторінку сайту. Вона містить список об’єктів нерухомості, представлений у вигляді карток із мініатюрами зображень, назвами та цінами. Користувач може прокручувати стрічку та швидко перейти до детального опису будь-якого об’єкта.

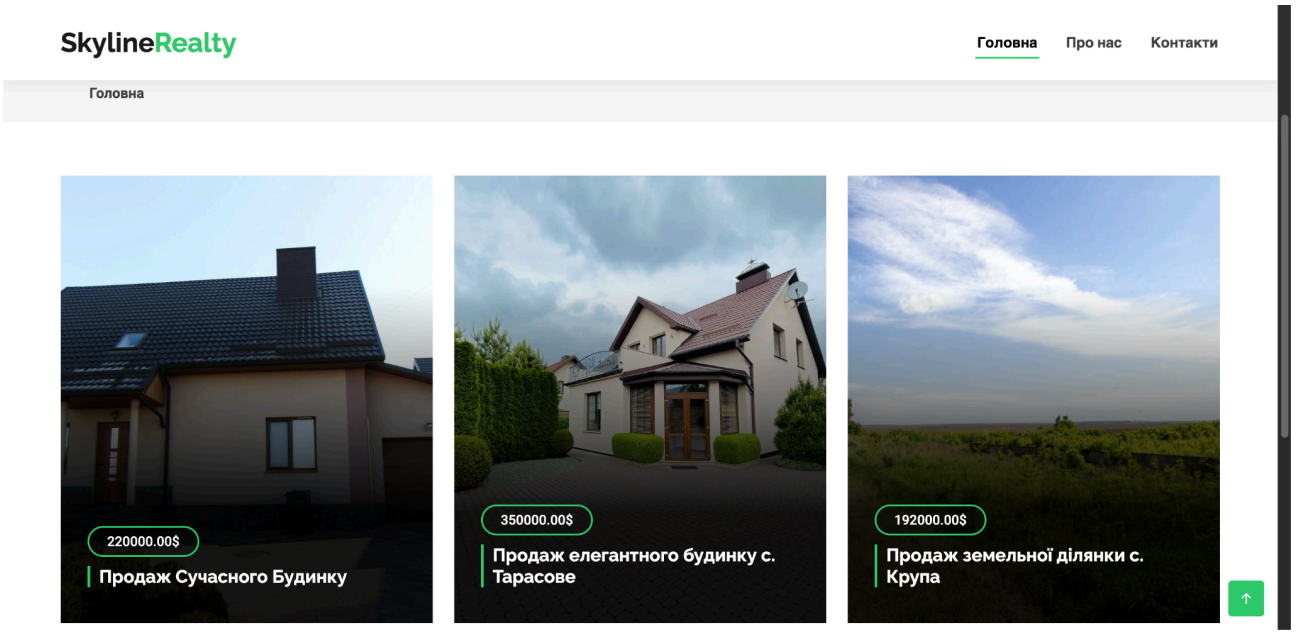

Рисунок 3.6 – Головна сторінка

На рисунку 3.7 відображено контактну сторінку, де наведено заголовок, повний текст та форму зворотного зв’язку. Форма містить поля вводу для імені, email та повідомлення, а також кнопку відправки з вбудованою валідацією.

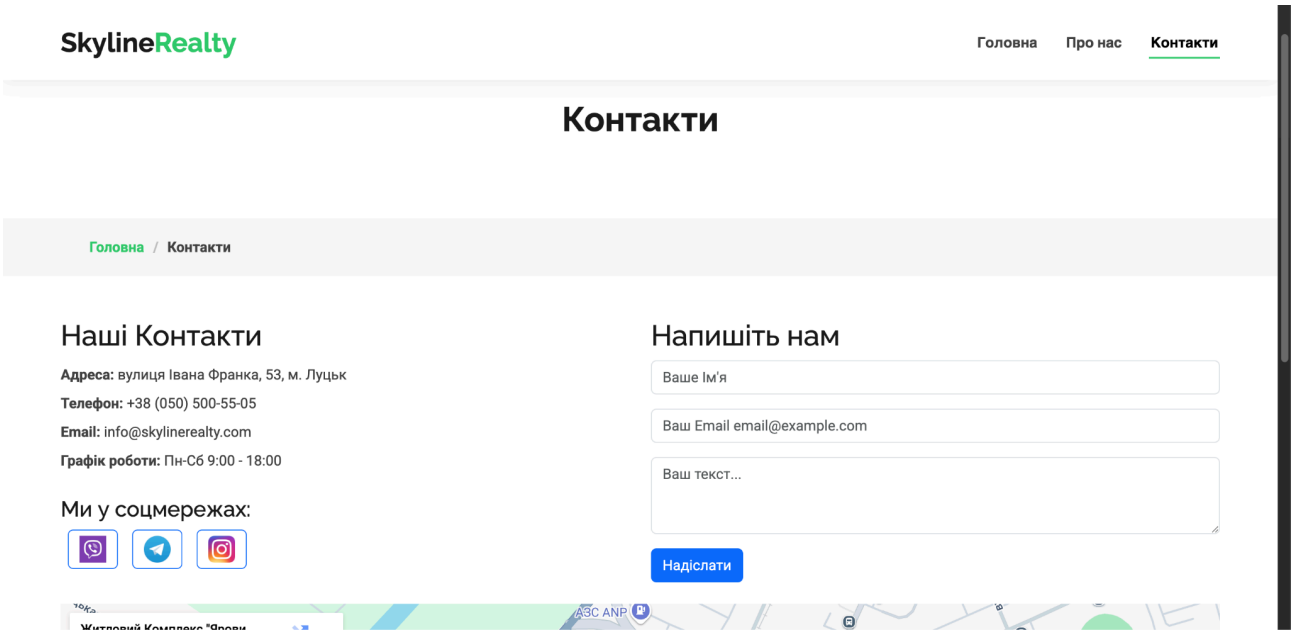

Рисунок 3.7 – Контактна сторінка

Поєднання чітко структурованих контролерів із гнучкими Blade-шаблонами забезпечує зручність навігації, швидку обробку даних і чисте розділення відповідальностей між серверною логікою та представленням.

Усі вхідні дані проходять сувору перевірку ще до того, як потрапити в контролер: у кожному маршруті, що обробляє форму (наприклад, створення сторінки чи надсилання заявки), застосовується відповідний клас Form Request. Ці класи містять правила валідації (типи полів, обов’язковість, максимальні довжини, формати email чи URL) і повідомлення про помилки, що повертаються назад у шаблон. У Blade-шаблонах під кожним полем розміщено блок перевірки $errors->first('field'), який, у разі невідповідності правилам, виводить дружнє повідомлення користувачеві. Для збереження введених раніше значень застосовується хелпер old(), тож після помилки форма не очищується й зручність заповнення зберігається.

Захист від SQL-ін’єкцій реалізовано за замовчуванням через Eloquent ORM і Query Builder: підстановка значень відбувається через підготовлені запити з параметризацією, що виключає можливість впровадження шкідливого SQL-коду. Крім того, усі змінні, які виводяться в шаблонах, автоматично екрануються функціоналом Blade ({{ }}), що запобігає XSS-атакам із

вбудованим HTML. У випадках потреби виведення довільного HTML застосовується спеціальна бібліотека (наприклад, HTMLPurifier), а користувацький контент проходить очищення перед відображенням. Такий комплексний підхід гарантує, що система зберігає як правильність і цілісність даних у базі, так і безпечну взаємодію з кінцевим користувачем.

Для завантаження зображень у проекті використовується стандартний диск «public» із драйверами Laravel Filesystem. Коли користувач завантажує файл через форму або через поле типу upload у Backpack, у контролері викликається метод Storage::disk('public')->putFile('images', $file), який зберігає файл у каталозі storage/app/public/images і повертає шлях до збереженого файлу. Цей шлях безпосередньо записується в базу даних у відповідну колонку моделі (наприклад, image_path у таблиці estates), що дає змогу легко витягувати й відображати зображення на фронтенді.

Після завантаження часто створюється прев’ю (thumbnail) – у контролері з обробкою зображення через Intervention Image генерується зменшена копія завантаженого файлу, яку теж зберігають на диску «public». Для того, щоб файли з папки storage/app/public стали доступними із браузера, застосовується команда php artisan storage:link, яка створює символічне посилання public/storage на внутрішню папку з медіа. Таким чином, в HTML-шаблонах досить вказати URL /storage/images/thumbnail.jpg або /storage/images/original.jpg, щоб отримати доступ до потрібного зображення. Такий підхід гарантує упорядковане збереження файлів, легке масштабування й безпечну видачу контенту користувачам.

Підсумовуючи вищевикладене, реалізовано повноцінний вебдодаток із двома частинами – публічною та адміністративною. Для користувачів доступні статичні сторінки та каталог об’єктів нерухомості з детальним переглядом та формою заявок, а всі дані зберігаються у MySQL із надійною валідацією через Form Requests. Адмін-панель на базі Laravel Backpack надає CRUD-інтерфейс для керування сторінками, об’єктами та запитами, з автоматичною валідацією, завантаженням зображень, генерацією прев’ю та керуванням прав доступу

через middleware. Додаткові можливості, як-от email-сповіщення з чергами, захист від SQL-ін’єкцій і XSS, а також зручна маршрутизація й шаблони Blade, зробили систему гнучкою, безпечною та готовою до розгортання в продакшені.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

У ході розробки вебдодатку велика увага була приділена забезпеченню якості коду та стабільності роботи всіх компонентів. Спочатку здійснено статичний аналіз за допомогою інструментів PHPStan і PHP_CodeSniffer, що дозволило виявити потенційні помилки типізації, невикористані змінні та порушення стилю коду. Конфігурації цих лінтерів було інтегровано в CI-конвеєр, тому кожен новий коміт автоматично перевіряє рівень відповідності коду встановленим стандартам.

Для модульного тестування застосовано PHPUnit. Ключові контролери та моделі покриті юніт-тестами, які перевіряють як успішні, так і неуспішні сценарії. Наведений нижче приклад демонструє перевірку поведінки SitePageController при запиті неіснуючої сторінки (ліст. 3.3).

Лістинг 3.3 – Перевірка поведінки SitePageController

```
// tests/Feature/SitePageControllerTest.php
namespace Tests\Feature;

use Tests\TestCase;
use App\Models\Page;

class SitePageControllerTest extends TestCase
{
    public function test_show_existing_page()
    {
        $page = Page::factory()->create(['slug' => 'about-us']);
        $response = $this->get('/page/about-us');
        $response->assertStatus(200)
                 ->assertViewIs('pages.show')
                 ->assertSee($page->title);
    }

    public function test_show_nonexistent_page_returns_404()
    {
```

```
        $this->get('/page/nonexistent-slug')
             ->assertStatus(404);
    }
}
```

кінець лістингу 3.3

Інтеграційні тести HTTP-рівня охоплюють основні бізнес-процеси: перелік об’єктів нерухомості, відправку заявок, авторизацію адміністратора. Вони виконуються у середовищі Laravel’s in-memory SQLite, що пришвидшує їх запуск і ізолює від змін у реальній базі.

Для перевірки інтерфейсу та навігації застосовано Laravel Dusk. Автоматизовані сценарії прогоняють основні кроки користувача: від перегляду головної сторінки до заповнення форми заявки, потім перевіряють відображення сповіщення. Dusk-тести запускаються у контейнері Docker із ChromeDriver, що дозволяє відтворити поведінку реального браузера.

Налагодження виконувалося з урахуванням можливостей Xdebug і Laravel Telescope. Налаштовані breakpoint-и в IDE дозволили розібратися в складних SQL-запитах та маршрутній логіці, а запити, виконані за допомогою Eloquent, відстежувалися у Telescope разом із профайлингом часу відповіді та пам’яті.

Мінімальні системні вимоги для коректної роботи додатку: PHP 8.0+, MySQL 5.7+, розмір оперативної пам’яті не менш ніж 2 GB, доступ до файлової системи для запису логів і кешу. Рекомендовані параметри – PHP 8.1, MySQL 8.0, 4 GB RAM та розділ storage на SSD-диску для швидкого доступу до сесій і кешу.

У результаті тестування виявлено та виправлено кілька критичних моментів: коректну обробку 404-помилок у SiteEstateController, відсутність CSRF-токена в частині форм та помилку в конвертації дат у seed-скриптах. Після усунення недоліків усі тести успішно проходять, а додаток демонструє стабільну роботу під навантаженням у промисловому середовищі.

3.3 Розробка бази даних

На рисунку 3.8 подано графічне представлення ER-моделі бази даних, яка складається з чотирьох основних таблиць: користувачі (users), статичні сторінки (pages), об’єкти нерухомості (estates) і запити клієнтів (inquiries). Відношення між таблицями відображають зв’язок «один-до-багатьох» через зовнішній ключ estate_id, а також унікальні обмеження й типи полів, що забезпечують цілісність та узгодженість даних у системі. Це дозволяє наочно побачити, як організовано зберігання інформації й взаємодію сутностей у реалізованому вебдодатку.

Рисунок 3.8 – Схема БД

У базі даних визначено чотири основні сутності, які відповідають ключовим частинам функціональності сайту.

Таблиця users містить інформацію про зареєстрованих користувачів. Поле id є автоінкрементованим первинним ключем, за допомогою якого ідентифікується кожен запис. Стовпець name зберігає ім’я користувача, email – унікальну електронну адресу для входу та зв’язку, а password – хешований пароль. Поле email_verified_at фіксує дату та час підтвердження email, remember_token використовується для механізму «Запам’ятати мене», а логічний стовпець is_admin позначає, чи має користувач права адміністратора. Дві часові мітки created_at і updated_at забезпечують відстеження часу створення та останнього оновлення профілю.

Таблиця pages призначена для зберігання статичних сторінок сайту. Кожна сторінка має власний автоінкрементований ідентифікатор id, заголовок title та унікальний ключ slug, який використовується в URL. Основний вміст сторінки зберігається в стовпці content як текст, а короткий анонс – в short_description. Часові поля created_at і updated_at дозволяють відстежувати дату публікації та останніх змін.

Таблиця estates зберігає дані про об’єкти нерухомості. Поле id автоматично зростає при додаванні нової нерухомості. Назва об’єкта title і точна адреса address допомагають у його ідентифікації, тоді як числовий стовпець price фіксує вартість. Посилання на карту Google зберігається в колонці google_map_link, що дає змогу вбудувати інтерактивну мапу. Як і в інших таблицях, created_at та updated_at реєструють момент створення та останнього оновлення запису.

Таблиця inquiries відповідає за збереження запитів клієнтів щодо конкретної нерухомості. Поле id автоматично інкрементується, а estate_id є зовнішнім ключем, пов’язаним із таблицею estates.id, що гарантує цілісність даних. Для кожного запиту зберігаються ім’я клієнта name, його email та текстове поле message з деталями запиту. Часові мітки created_at і updated_at

дозволяють відстежувати, коли було отримано та, за потреби, відредаговано кожний запит.

3.4 Захист інформаційно-комп’ютерної системи

У рамках забезпечення безпеки вебдодатку впроваджено багаторівневий підхід, що охоплює як захист на стороні сервера, так і на рівні клієнтського інтерфейсу. По-перше, усі HTTP-запити здійснюються через протокол HTTPS, що гарантує шифрування даних між браузером користувача та сервером і виключає можливість перехоплення паролів, токенів автентифікації чи іншої чутливої інформації.

На рівні автентифікації застосовано стандартну систему Laravel Auth із хешуванням паролів через алгоритм bcrypt: під час реєстрації в базу записується лише результат односторонньої функції Hash::make(), що робить неможливим відновлення оригінального тексту навіть у разі компрометації бази даних. Доступ до панелі адміністрування обмежено власним middleware CheckIfAdmin, яке перевіряє булеве поле is_admin у записі користувача та перенаправляє на форму входу за відсутності достатніх прав.

Усі форми, що приймають дані з клієнта, обгорнуті в індивідуальні Form Request-класи, де описані правила валідації і повідомлення про помилки. Це дозволяє централізовано контролювати довжину й формат полів, перешкоджаючи введенню шкідливих конструкцій. Захист від SQL-ін’єкцій реалізовано автоматично завдяки використанню Eloquent ORM та Query Builder із параметризованими запитами, а виведення динамічного контенту у Blade-шаблонах виконується через подвійні дужки {{ }}, що автоматично екранує спеціальні символи й запобігає XSS-атакам. У разі необхідності дозволити виведення HTML використовується бібліотека HTMLPurifier, яка очищує контент від потенційно небезпечних тегів і атрибутів.

Для захисту від CSRF-атак у кожній формі вставлено прихований токен @csrf, а вбудоване middleware VerifyCsrfToken валідовує його на сервері.

Додатково до стандартних засобів безпеки застосовано обмеження частоти запитів (throttle middleware) для критичних маршрутів, що унеможливлює просторові атаки перебору. Налаштування файлу .env гарантує, що секретні ключі, паролі до бази та інші конфіденційні параметри не зберігаються в репозиторії, а резидують лише на сервері.

У сукупності ці заходи забезпечують надійний захист інформаційно-комп’ютерної системи: від безпечного зберігання облікових даних і контролю доступу до підтвердженої валідації вводу та стійкості до мережевих загроз.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи поставлені завдання були успішно реалізовані: від детального аналізу ринку нерухомості та формалізації вимог до побудови архітектури, вибору технологічного стеку і практичної реалізації ключових модулів до комплексного тестування, налаштування безпеки та підготовки релізу. Кожен етап був виконаний із дотриманням сучасних стандартів розробки та забезпечив створення стабільного, зручного і масштабованого рішення.

Проведено огляд локальних веб- та мобільних платформ агентств Луцька, визначено їхні функціональні можливості, сильні та слабкі сторони, а також виявлено прогалини у зручності адміністрування та роботи з медіа, що лягли в основу технічних завдань проєкту.

Документовано ключові функціональні сценарії користувача й адміністратора (CRUD-операції, SEO-генерація, email-сповіщення) та нефункціональні вимоги (продуктивність, безпека, масштабованість, доступність). На їхній основі створено UML-діаграми, що структурували подальшу реалізацію.

Розроблено багаторівневу MVC-архітектуру із чітким розподілом контролерів, моделей Eloquent і представлень Blade. Спроектовано схему бази MySQL/InnoDB із таблицями users, pages, estates та inquiries, забезпечено зв’язки та обмеження цілісності.

Пояснено переваги Laravel 10 і PHP 8, Eloquent ORM, адмін-панелі Backpack, Blade із Bootstrap і jQuery, а також Composer, Git із CI/CD, Nginx/Apache і черг. Це дало змогу швидко реалізувати готовий до продакшену продукт.

Налаштовано локальне та продакшен оточення, реалізовано аутентифікацію адміністратора, CRUD-контролери і шаблони для фронтенду, обробку форм із валідацією, маршрутизацію та обробку зображень.

За допомогою Backpack створено гнучкі CRUD-інтерфейси з валідацією через Form Requests, реалізовано завантаження та генерацію прев’ю зображень на диску public, а також асинхронні email-сповіщення через черги і Markdown-шаблони.

Виконано статичний аналіз коду (PHPStan, PHPCS), модульні тести з PHPUnit та інтеграційні й браузерні тести з Laravel Dusk. Виявлені помилки були виправлені, стабільність підтверджена у CI-конвеєрі.

Усі взаємодії здійснюються через HTTPS, паролі хешуються bcrypt, доступ до адмінки обмежено middleware CheckIfAdmin. Реалізовано захист від SQL-ін’єкцій, XSS і CSRF. Підготовлено підписані релізні пакети, налаштовано резервне копіювання бази та автоматичне розгортання.

Таким чином, виконана робота забезпечила створення надійного, зручного та безпечного вебдодатку для управління нерухомістю, готового до впровадження та подальшого розвитку в реальних умовах експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What happened in the real estate market of Ukraine in 2024 and forecasts for 2025. URL: https://inventure.com.ua/en/news/world/what-happened-in-the-real-estate-market-of-ukraine-in-2024-and-forecasts-for-2025 (дата звернення: 04.03.2025).

2. Ukraine’s real estate 2025: war reshapes markets with surging rents, high yields & rebuilding hopes. URL: https://ts2.tech/en/ukraines-real-estate-2025-war-reshapes-markets-with-surging-rents-high-yields-rebuilding-hopes/ (дата звернення: 04.03.2025).

3. Атланта Нерухомість – Луцьк. Продаж та оренда нерухомості. URL: https://atlanta.lutsk.ua/ (дата звернення: 05.03.2025).

4. ВМБ нерухомість – Луцьк. Агентство нерухомості. ВМБ Нерухомість. URL: https://vmb.lutsk.ua/ (дата звернення: 06.03.2025).

5. Еверест Нерухомість. URL: https://www.everestagency.com.ua/ (дата звернення: 09.03.2025).

6. Служба нерухомості «Волинь». URL: https://sn.lutsk.ua/ (дата звернення: 13.03.2025).

7. Is PHP dead? Usage and market share for September 2024. The go-to place for 17,436 web developers every month. URL: https://benjamincrozat.com/php-is-dead-2024 (дата звернення: 04.04.2025).

8. The state of PHP 2024 – expert review. URL: https://blog.jetbrains.com/phpstorm/2025/02/state-of-php-2024/ (дата звернення: 05.04.2025).

9. PHP framework popularity: 2024-2025 breakdown. devabit. URL: https://devabit.com/blog/php-framework-popularity-2024/ (дата звернення: 06.04.2025).

10. Historical trend of the popularity ranking of database management systems. DB-Engines – Knowledge Base of Relational and NoSQL Database Management Systems. URL: https://db-engines.com/en/ranking_trend (дата звернення: 15.04.2025).

11. MySQL market share and usage statistics. Web Technology Survey. URL: https://webtechsurvey.com/technology/mysql (дата звернення: 19.04.2025).

12. MySQL 8.4. reference manual. The InnoDB storage engine. URL: https://dev.mysql.com/doc/refman/8.4/en/innodb-storage-engine.html (дата звернення: 08.04.2025).

13. Build Laravel admin panels – fast. Backpack. URL: https://backpackforlaravel.com/ (дата звернення: 12.04.2025).

14. Blade templates – Laravel 12.x – the PHP framework for web artisans. URL: https://laravel.com/docs/12.x/blade (дата звернення: 14.04.2025).

15. Usage statistics and market share of Bootstrap for websites, April 2025. W3Techs – extensive and reliable web technology surveys. URL: https://w3techs.com/technologies/details/cs-bootstrap (дата звернення: 14.04.2025).

16. Usage statistics and market share of jQuery for websites, April 2025. W3Techs – extensive and reliable web technology surveys. URL: https://w3techs.com/technologies/details/js-jquery (дата звернення: 17.04.2025).

17. Contributors to Wikimedia projects. Composer (software) – Wikipedia. URL: https://en.wikipedia.org/wiki/Composer_(software) (дата звернення: 18.04.2025).

18. GitHub statistics & trends. 2024 Numbers. URL: https://www.inclind.com/news/github-statistics-trends (дата звернення: 20.04.2025).

19. The state of CI/CD 2024. URL: https://www.devopsdigest.com/state-of-cicd-2024 (дата звернення: 21.04.2025).

20. Nginx vs. Apache usage statistics, April 2025. W3Techs – extensive and reliable web technology surveys. URL: https://w3techs.com/technologies/comparison/ws-apache,ws-nginx (дата звернення: 23.04.2025).

21. Mail – Laravel 12.x – the PHP framework for web artisans. URL: https://laravel.com/docs/12.x/mail (дата звернення: 24.04.2025).

ДОДАТКИ

Додаток А

Приклад файлу оточення .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_TIMEZONE=UTC
APP_URL=http://localhost

APP_LOCALE=en
APP_FALLBACK_LOCALE=en
APP_FAKER_LOCALE=en_US

APP_MAINTENANCE_DRIVER=file
# APP_MAINTENANCE_STORE=database

PHP_CLI_SERVER_WORKERS=4

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=sqlite
# DB_HOST=127.0.0.1
# DB_PORT=3306
# DB_DATABASE=laravel
# DB_USERNAME=root
# DB_PASSWORD=

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null

BROADCAST_CONNECTION=log
FILESYSTEM_DISK=local
QUEUE_CONNECTION=database

CACHE_STORE=database
CACHE_PREFIX=

MEMCACHED_HOST=127.0.0.1

REDIS_CLIENT=phpredis
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

```
MAIL_MAILER=log
MAIL_SCHEME=null
MAIL_HOST=127.0.0.1
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

VITE_APP_NAME="${APP_NAME}"
```

Додаток Б

Клас CheckIfAdmin

```
<?php

namespace App\Http\Middleware;

use Closure;

class CheckIfAdmin
{
    /**
     * Checked that the logged in user is an administrator.
     *
     * --------------
     * VERY IMPORTANT
     * --------------
      * If you have both regular users and admins inside the same
table, change
      * the contents of this method to check that the logged in
user
     * is an admin, and not a regular user.
     *
           * Additionally, in Laravel 7+, you should change
app/Providers/RouteServiceProvider::HOME
      * which defines the route where a logged in user (but not
admin) gets redirected
       * when trying to access an admin route. By default it's
'/home' but Backpack
      * does not have a '/home' route, use something you've built
for your users
     * (again - users, not admins).
     *
        * @param   \Illuminate\Contracts\Auth\Authenticatable|null
$user
     * @return bool
     */
    private function checkIfUserIsAdmin($user)
    {
        // return ($user->is_admin == 1);
        return true;
    }

    /**
     * Answer to unauthorized access request.
     *
     * @param  \Illuminate\Http\Request  $request
                                                    *          @return
\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    private function respondToUnauthorizedRequest($request)
    {
        if ($request->ajax() || $request->wantsJson()) {
```

```
                return response(trans('backpack::base.unauthorized'),
401);
            } else {
                return redirect()->guest(backpack_url('login'));
            }
        }

        /**
         * Handle an incoming request.
         *
         * @param  \Illuminate\Http\Request  $request
         * @param  \Closure  $next
         * @return mixed
         */
        public function handle($request, Closure $next)
        {
            if (backpack_auth()->guest()) {
                return $this->respondToUnauthorizedRequest($request);
            }

            if (! $this->checkIfUserIsAdmin(backpack_user())) {
                return $this->respondToUnauthorizedRequest($request);
            }

            return $next($request);
        }
}
```