МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ОНЛАЙН-ОГОЛОШЕНЬ ДЛЯ КУПІВЛІ ТА ПРОДАЖУ ТОВАРІВ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Андрій ГОРБУНОВ**

«__» _____ 2026 р.

Керівник роботи _____ **Катерина АНТІПОВА**
PhD, доцентка(б. в. з.)

«__» _____2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший(бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
Євген ДАВИДЕНКО
_____
16 травня 2026р.

ЗАВДАННЯ
на кваліфікаційну бакалаврську роботу здобувача
Горбунова Андрія Валерійовича

1) Тема кваліфікаційної роботи «Вебзастосунок онлайн-оголошень для купівлі та продажу товарів» затверджена наказом ректора ЧНУ ім. Петра Могили №349 від «26 грудня» 2025р.

2) Строк представлення кваліфікаційної роботи «30 травня» 2026р.

3) Очікуваний результат: працездатний вебзастосунок з функціоналом створення, перегляду та адміністрування оголошень, система авторизації користувачів, пошук та фільтрація товарів.

4) Перелік питань, що підлягають розробці:
   1) аналіз предметної області та існуючих рішень на ринку онлайн-торгівлі;
   2) проєктування архітектури бази даних для зберігання даних про користувачів, товари та категорії;

3) розробка серверної частини для обробки запитів;
4) розробка клієнтської частини з адаптивним дизайном;
5) реалізація системи безпеки та розмежування прав доступу;
6) тестування розробленого програмного продукту.

5) Перелік графічних матеріалів:
   1) діаграма прецедентів;
   2) діаграма класів;
   3) дизайн інтерфейсу користувача;
   4) скріншоти готового вебзастосунку;
   5) презентація.

6) Консультанти:

Консультант	Кафедра(організація)	Частина роботи

16 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Вебзастосунок онлайн-оголошень для купівлі та продажу товарів

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	15.02.2026	16.02.2026	Виконано
2	Огляд літератури за темою роботи	16.02.2026	18.02.2026	Виконано
3	Складання календарного плану КБР	18.02.2026	19.02.2026	Виконано
4	Аналіз предметної області	19.02.2026	20.02.2026	Виконано
5	Розробка проєктних рішень	20.02.2026	22.02.2026	Виконано
6	Моделювання та конструювання ПЗ	22.02.2026	27.02.2026	Виконано
7	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва	27.02.2026	27.03.2026	Виконано
8	Відгук керівника КБР	03.05.2026	03.05.2026	Виконано
9	Оформлення КБР та презентації	04.05.2026	18.05.2026	Виконано
10	Попередній захист	26.05.2026	26.05.2026	Виконано
11	Рецензування	09.06.2026	09.06.2026	Виконано
12	Завершення оформлення КБР та презентації	12.06.2026	12.06.2026	Виконано
13	Захист кваліфікаційної роботи	22.06.2026	22.06.2026	

Здобувач _____ **Андрій ГОРБУНОВ**

16 лютого 2026 р.

Керівник роботи _____ **Катерина АНТІПОВА**

PhD, доцентка(б. в. з.)

16 лютого 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзастосунок онлайн-оголошень для купівлі та продажу товарів»

Здобувач 408 гр.: Горбунов Андрій

Керівник: PhD, доцентка (б. в. з.) Антіпова Катерина

Актуальність теми: у сучасних умовах цифровізації економіки попит на зручні інструменти для реалізації товарів залишається стабільно високим. Створення власного рішення дозволяє впровадити специфічні алгоритми фільтрації, гнучку систему категорій та інтеграцію з локальними сервісами, що є критично важливим для нішевих ринків або закритих спільнот.

Об’єкт роботи: процеси купівлі-продажу товарів у мережі Інтернет за допомогою спеціалізованих вебплатформ.

Предмет роботи: інструменти та архітектурні рішення, що використовуються для розробки та розгортання вебзастосунку онлайн-оголошень.

Мета роботи: проєктування та розробка масштабованого вебзастосунку розміщення та пошуку онлайн-оголошень, який спрощує взаємодію між користувачами.

У вступі обґрунтовано актуальність теми розробки платформи онлайн-оголошень, визначено об'єкт, предмет, мету та перелік завдань дослідження.

Перший розділ присвячений системному аналізу предметної області електронної комерції та порівняльному огляду існуючих на ринку рішень для виявлення їхніх недоліків.

У другому розділі сформовано специфікацію вимог до програмного забезпечення, розроблено функціональні моделі системи та обґрунтовано вибір технологічного стеку.

Третій розділ присвячений архітектурному проєктуванню платформи, у межах якого розроблено концептуальні та логічні схеми реляційної бази даних, а також побудовано комплекс структурних UML моделей, зокрема діаграми класів та розгортання.

У четвертому розділі наведено особливості програмної реалізації ключових модулів серверної та клієнтської частин застосунку, представлено результати функціонального тестування, проведено аналіз швидкодії системи при обробці запитів, а також розроблено детальне керівництво користувача.

У висновках підбито підсумки виконаної інженерної роботи та оцінено ступінь досягнення поставленої мети.

Розроблений продукт може бути впроваджений як самостійний комерційний майданчик для локального ринку.

КБР викладена на 64 сторінки, вона містить 4 розділи, 20 ілюстрацій, 2 таблиці, 16 джерел в переліку посилань.

Ключові слова: вебзастосунок, електронна комерція, дошка оголошень, клієнт-серверна архітектура, бази даних, Laravel, React.

ABSTRACT

of the Bachelor`s Thesis

“Online classifieds web application for buying and selling goods”

Student of group 408: Horbunov Andrii

Supervisor: PhD, associate professor Antipova Katerina

Relevance of the topic: in the modern conditions of economic digitalization, the demand for convenient tools for selling goods remains consistently high. Creating a custom solution allows for the implementation of specific filtering algorithms, a flexible category system, and integration with local services, which is critically important for niche markets or closed communities.

Object of development: the process of buying and selling goods on the Internet using specialized web platforms.

Subject of development: methods, tools, and architectural solutions used for the development and deployment of an online classifieds web application.

Purpose of the work: design and development of a scalable web application for posting and searching online classifieds, which provides automation of the buying and selling process between users.

The introduction justifies the relevance of developing an online classifieds platform and defines the object, subject, purpose, and list of tasks for the study.

The first chapter is devoted to a systematic analysis of the subject area of e-commerce and a comparative review of existing market solutions to identify their shortcomings.

The second chapter formulates the software requirements specification, develops functional models of the system, and justifies the choice of the technology stack.

The third chapter is devoted to the architectural design of the platform, which includes the development of conceptual and logical diagrams for the relational database, as well as the creation of a set of structural UML models, including class diagrams and deployment diagrams.

The fourth chapter describes the specifics of the software implementation of the key modules of the application’s server and client components, presents the results of

functional testing, analyzes the system’s performance when processing queries, and provides a detailed user manual.

The conclusions summarize the engineering work performed and assess the degree to which the set goal has been achieved.

The developed product can be implemented as an independent commercial platform for the local market.

The bachelor's thesis is presented on 64 pages, it contains 4 sections, 20 illustrations, 2 tables, 16 sources in the list of references.

Keywords: web application, e-commerce, classifieds board, client-server architecture, databases, Laravel, React.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

B2C	business to customer
C2C	customer to customer
CSS	cascading style sheets
HTML	hypertext markup language
JS	JavaScript
MPA	multiple page application
MVC	model-view-controller
SPA	single page application
SQL	structured query language
UI	user interface
UML	unified modeling language
UX	user experience

ВСТУП

У сучасних умовах цифровізації економіки попит на зручні інструменти для реалізації товарів залишається стабільно високим. Створення власного рішення дозволяє впровадити специфічні алгоритми фільтрації, гнучку систему категорій та інтеграцію з локальними сервісами, що є критично важливим для нішевих ринків або закритих спільнот.

Об'єкт роботи: Процес купівлі-продажу товарів у мережі Інтернет за допомогою спеціалізованих вебплатформ.

Предмет роботи: Методи, інструменти та архітектурні рішення, що використовуються для розробки та розгортання вебзастосунку онлайн-оголошень.

Мета роботи: проєктування та розробка масштабованого вебзастосунку розміщення та пошуку онлайн-оголошень, який спрощує взаємодію між користувачами.

Науково-практичне значення: Розроблений вебзастосунок забезпечує високу швидкість взаємодії між користувачами завдяки використанню сучасного стеку технологій (Laravel та React). Практична цінність роботи полягає у:

1) масштабованості. Архітектура системи дозволяє легко додавати нові типи товарів та функціональні модулі без переписування коду;

2) ефективній комунікації. Впровадження системи сповіщень у реальному часі значно скорочує час відгуку на оголошення;

3) користувацькому досвіді. Створення адаптивного та інтуїтивно зрозумілого інтерфейсу, що мінімізує кількість кроків від реєстрації до публікації товару;

4) оптимізації бізнес-процесів. Автоматизація модерації та зручна система фільтрації підвищують релевантність пошуку для покупця.

Завдання:

1) проаналізувати існуючі рішення та виявити їхні переваги й недоліки;

2) сформулювати специфікацію технічних вимог до системи, включаючи функціональні та нефункціональні аспекти;

3) здійснити моделювання архітектури системи;

4) виконати проєктування структури бази даних для зберігання категорій товарів, оголошень та даних користувачів;

5) розробити функціонал реєстрації, автентифікації та особистого кабінету користувача;

6) реалізувати систему пошуку з фільтрацією за параметрами;

7) створити інтерфейс для подання оголошень з можливістю завантаження мультимедійних файлів;

8) налаштувати систему сповіщень між користувачами;

9) провести комплексне тестування для перевірки стабільності роботи та безпеки даних.

1 АНАЛІЗ СФЕРИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1 Аналіз предметної області та тенденцій розвитку систем електронної комерції

Швидкий розвиток інформаційно-комунікаційних технологій та загальна інтеграція мережі Інтернет у повсякденне життя кардинально змінили ведення бізнесу та здійснення торговельних операцій. Електронна комерція вже не є чимось новим, а навпаки, стала звичайним явищем. У цьому контексті, особливої актуальності набуває аналіз та вдосконалення платформ для розміщення онлайн-оголошень, які забезпечують пряму комунікацію між продавцями та покупцями.

Предметна область онлайн-оголошень охоплює кілька ключових бізнес-моделей:

– покупець-покупець (C2C) – взаємодія між приватними особами, де одна людина виступає продавцем власних вживаних або нових речей, а інша – покупцем. Ця модель є основною, оскільки вимагає мінімального порогу входу для користувачів та відсутності складних бюрократичних процедур;

– бізнес-покупець (B2C) – використання дошок оголошень малим та середнім бізнесом як додаткового або основного каналу збуту товарів і послуг. Такі користувачі створюють більший обсяг контенту і потребують розширених інструментів для управління оголошеннями, відстеження статистики переглядів та комунікації з клієнтами.

Аналіз сучасного стану ринку електронної комерції показує, що користувачі висувають дедалі вищі вимоги до якості програмного забезпечення класифайдів. Сучасний маркетплейс або дошка оголошень – це вже не просто статична вебсторінка з переліком товарів, а складна інформаційна система, яка повинна відповідати ряду критичних критеріїв:

– висока продуктивність. Час завантаження сторінки та відгуку інтерфейсу безпосередньо впливає на задоволеність користувачів. Зростання

частки мобільного трафіку вимагає оптимізації передачі даних та впровадження архітектурних рішень;

– інтерактивність та комунікація в реальному часі. Сучасні користувачі очікують можливості миттєвого зв'язку. Наявність вбудованого чату з функціями повідомлень у реальному часі стала стандартом, замінивши традиційну електронну пошту чи телефонні дзвінки;

– гнучкий пошук та персоналізація. Наявність величезних масивів даних вимагає реалізації складних алгоритмів пошуку, фільтрації за множиною параметрів та, що особливо важливо, географічної локації. Система повинна вміти швидко відсіювати нерелевантні результати та пропонувати товари, що знаходяться поблизу покупця;

– довіра та безпека. Оскільки класифайди часто використовуються для C2C транзакцій, де рівень довіри між незнайомими людьми є низьким, програмне забезпечення повинно включати системи рейтингів продавців та відгуків, що допомагають формувати безпечне середовище.

Незважаючи на наявність на ринку великих, сформованих платформ, аналіз предметної області виявляє ряд проблем. Існуючі системи-гіганти часто побудовані на застарілих архітектурах, що робить їх неповороткими та важкими для масштабування. Перевантаженість інтерфейсів рекламою, повільна робота вебверсій на слабких пристроях та відсутність оптимізації клієнт-серверної взаємодії створюють простір для розробки нових, більш оптимізованих рішень.

1.2 Аналіз архітектурних підходів до побудови вебзастосунків електронної комерції

Проєктування сучасних систем електронної комерції вимагає детального аналізу існуючих архітектурних патернів. Історично більшість класифайдів створювалися за моделлю багатосторінкових застосунків. У цій архітектурі кожна дія користувача призводить до відправки запиту на сервер, який генерує нову HTML-сторінку цілком і повертає її браузеру. Хоча цей підхід є надійним і добре

індексується пошуковими системами, він має суттєвий недолік – надмірне споживання мережевих ресурсів та низьку швидкість оновлення інтерфейсу, що негативно впливає на користувацький досвід.

У відповідь на це стандартом для нових проєктів стала архітектура односторінкових застосунків. У моделі SPA сервер віддає HTML-каркас лише один раз при першому завантаженні застосунку. Усі подальші взаємодії користувача з інтерфейсом призводять лише до асинхронного завантаження необхідних даних у форматі JSON. Клієнтський фреймворк самостійно перемальовує лише ті частини сторінки, які змінилися, залишаючи незмінними навігаційні панелі, хедери та футери.

Переваги SPA-архітектури для платформи онлайн-оголошень:

- мінімізація трафіку. Сервер передає лише чисті дані, а не важку HTML-розмітку, що критично важливо для користувачів мобільного інтернету;
- плавність інтерфейсу. Переходи між сторінками оголошень відбуваються миттєво, без візуального блимання екрану;
- зменшення навантаження на сервер. Сервер баз даних фокусується виключно на обробці бізнес-логіки та формуванні API-відповідей, перекладаючи завдання рендерингу на пристрій клієнта.

Аналіз цих архітектурних підходів підтверджує, що для проєктування платформи «MarketFlow» застосування SPA у зв'язці з компонентним підходом є найбільш раціональним рішенням, яке забезпечить конкурентну перевагу над застарілими MPA системами.

1.3 Огляд та порівняльний аналіз існуючих програмних рішень

Для формування обґрунтованих вимог до платформи та вибору оптимальних шляхів її реалізації було проведено комплексний аналіз існуючих на ринку рішень у сфері електронної комерції та онлайн-класифайдів. Сучасний ринок України та світу представлений низкою потужних систем, серед яких безперечними лідерами є платформи OLX, eBay та Prom.ua. Незважаючи на їхню

популярність, кожна з цих систем має свої архітектурні та функціональні особливості, які обмежують їх ефективність у певних сценаріях використання.

Платформа OLX є найбільшою на ринку в Україні. Її основною перевагою є колосальна база користувачів та глибока категоризація товарів і послуг. Архітектурно OLX побудована з використанням складних розподілених систем для витримування високих навантажень.

Однак, з точки зору користувацького досвіду та швидкодії клієнтської частини, система має недоліки. Платформа не є класичним односторінковим застосунком, що призводить до необхідності повного перезавантаження сторінок при переході між категоріями або застосуванні фільтрів. Це створює додаткове навантаження на мережу та збільшує час очікування для користувача. Крім того, вбудована система обміну повідомленнями часто працює із затримками, що знижує інтерактивність спілкування між покупцем та продавцем.

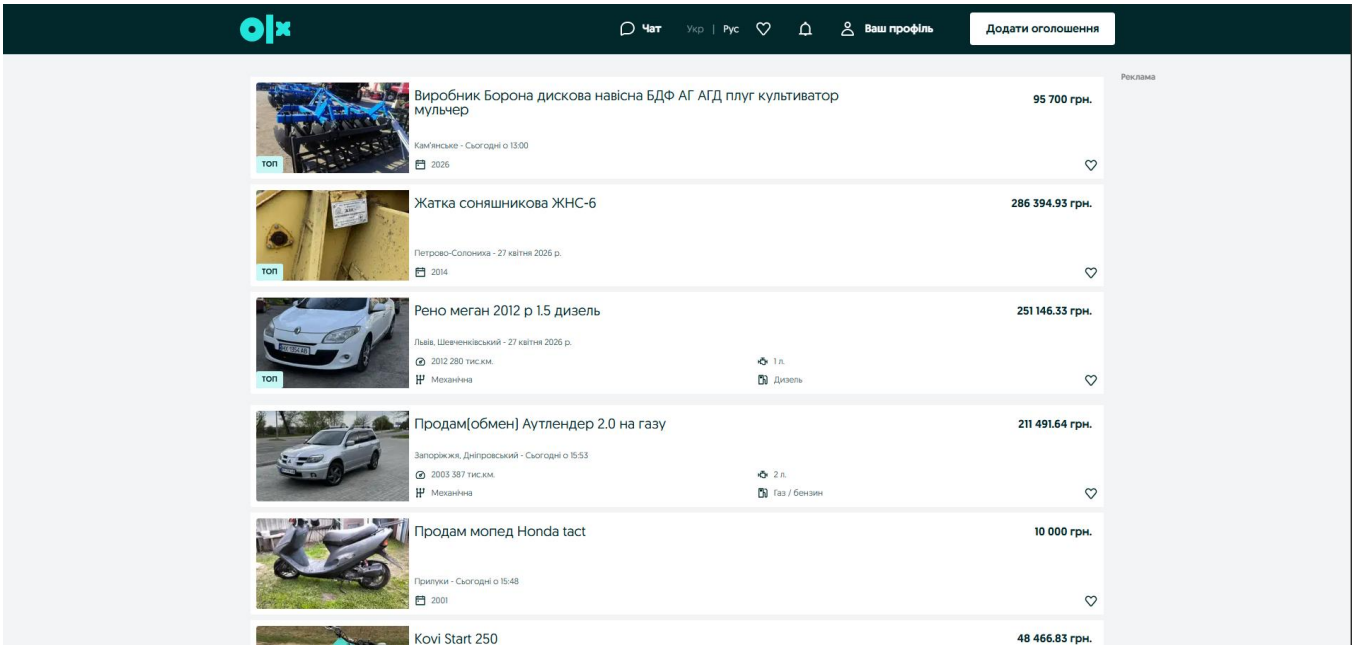


Рисунок 1.1 – Інтерфейс OLX

eBay – це світовий лідер електронної комерції, який поєднує моделі B2C та C2C, пропонуючи унікальну систему аукціонів та фіксованих цін. Перевагами eBay є високий рівень безпеки транзакцій, глобальне охоплення та потужна система захисту покупців.

Проте для локального ринку швидкого обміну товарами eBay є надлишково складною системою. Процес реєстрації продавця, створення оголошення та управління лотами перевантажений зайвими кроками. Інтерфейс платформи багато в чому зберіг спадковість попередніх десятиліть і є менш інтуїтивним порівняно з сучасними вебзастосунками. Комунікація між користувачами на eBay більше нагадує обмін електронними листами, а не сучасний миттєвий чат, що критично уповільнює процес обговорення деталей угоди.

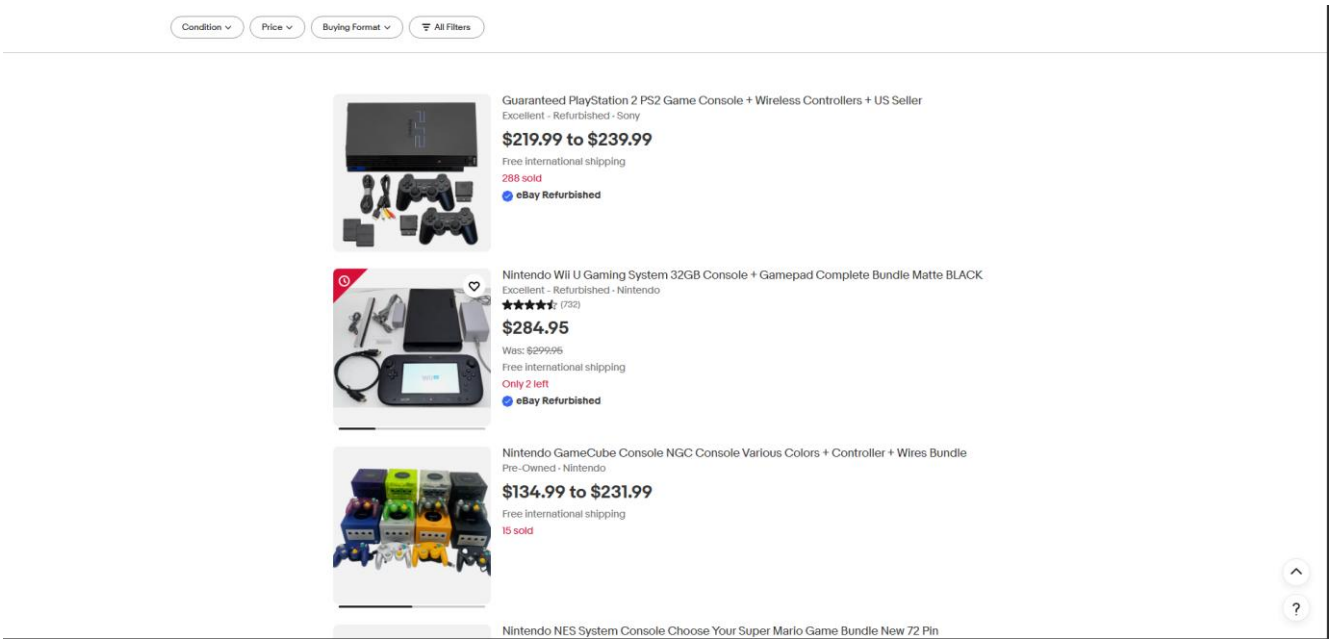


Рисунок 1.2 – Інтерфейс eBay

Prom.ua орієнтований переважно на сегмент B2C, виступаючи агрегатором інтернет-магазинів. Система пропонує потужні інструменти CRM для бізнесу, масове завантаження товарів та інтеграцію з логістичними операторами.

Водночас ця платформа абсолютно не підходить для швидкого продажу одиничних товарів приватними особами. Інтерфейс створення оголошення орієнтований на професійних продавців, що створює високий поріг входу для звичайного користувача.

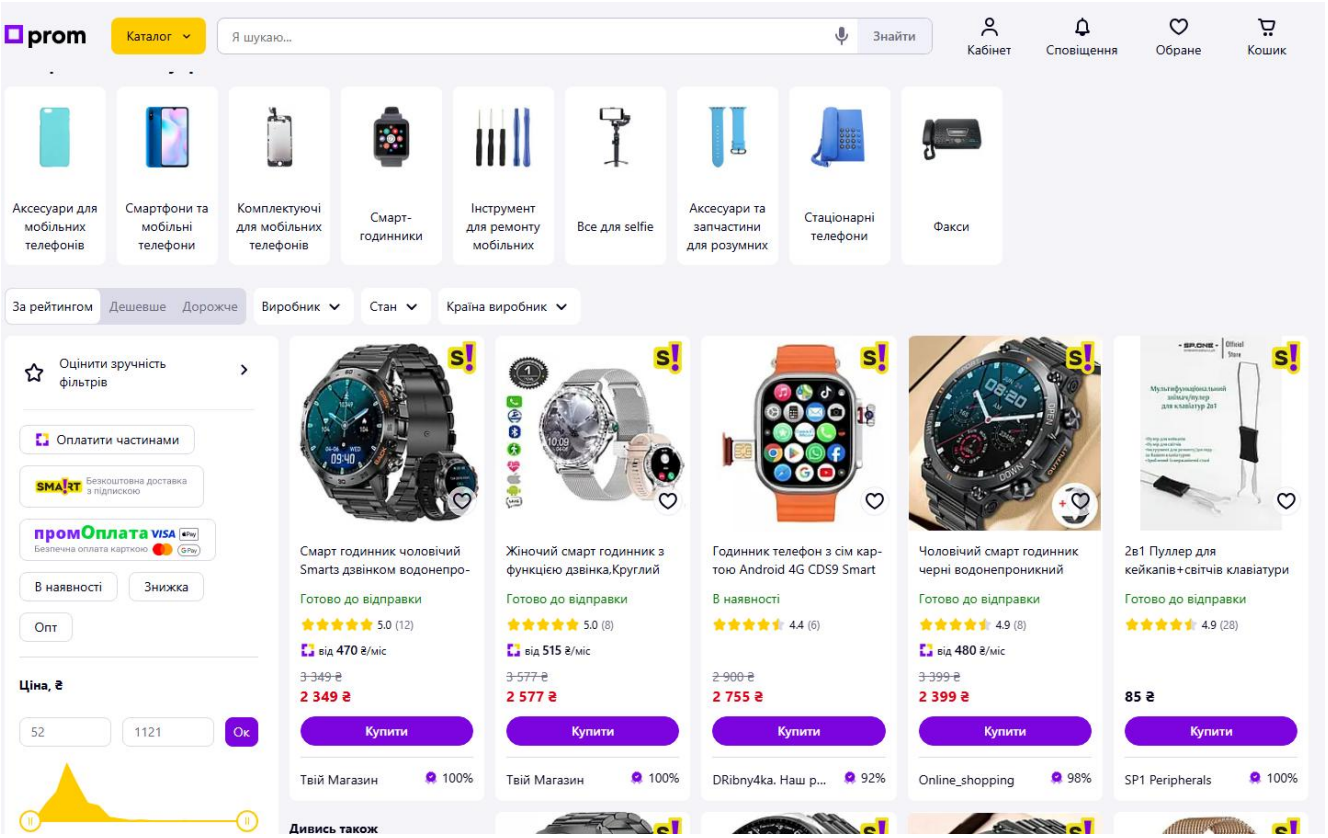


Рисунок 1.3 – Інтерфейс prom.ua

Підсумовуючи результати аналізу, можна зробити висновок, що на ринку існує потреба в платформі, яка б поєднувала простоту використання для приватних осіб із швидкодією та інтерактивністю сучасних вебтехнологій. Більшість існуючих рішень страждають від застарілих архітектурних підходів до побудови клієнтської частини, що призводить до повільної роботи інтерфейсу та незручної комунікації.

Саме ці недоліки створюють передумови для розробки платформи «MarketFlow». Використання реактивних технологій дозволить забезпечити миттєве оновлення даних на клієнті, а впровадження сучасного чату вирішить проблему повільної комунікації, наблизивши досвід використання вебзастосунку до стандартів сучасних мобільних месенджерів.

1.4 Постановка завдання на розробку програмного забезпечення

На основі проведеного аналізу предметної області та виявлених недоліків існуючих систем електронної комерції, виникає обґрунтована необхідність створення нової, оптимізованої платформи онлайн-оголошень.

Головним завданням роботи є проєктування та програмна реалізація веборієнтованої платформи «MarketFlow». Ця система повинна функціонувати за принципом односторінкового застосунку, що дозволить усунути проблему повільного перезавантаження сторінок та забезпечить користувачам досвід, наближений до використання нативних мобільних або десктопних програм.

Для досягнення поставленої мети необхідно спроєктувати та реалізувати наступний функціонал програмного забезпечення:

- реалізація надійних механізмів реєстрації, авторизації та аутентифікації користувачів для забезпечення безпеки системи;
- створення особистого кабінету користувача з можливістю управління та редагування персональних даних;
- інтеграція динамічного модуля додавання платіжних реквізитів з автоматичним розпізнаванням типу банківської картки;
- розробка модуля управління контентом, що включає створення, редагування та видалення користувацьких оголошень;
- впровадження інструментів повнотекстового пошуку за ключовими словами в назвах та описах товарів;
- розробка алгоритмів гнучкої фільтрації пошукових результатів за вибраними категоріями та сортування;
- створення модуля миттєвого обміну повідомленнями між покупцями та продавцями для обговорення угод без використання сторонніх месенджерів;
- впровадження лічильника переглядів для відстеження популярності та формування статистики кожного окремого оголошення;

- реалізація механізму додавання товарів до списку «Обраного» для зручної організації відкладених покупок;
- розробка прозорої системи рейтингів та текстових відгуків для формування рівня довіри до продавців на платформі.

З технічної точки зору, завдання полягає у побудові надійної бекенд-архітектури, здатної обробляти паралельні запити та забезпечувати цілісність даних у реляційній базі даних. Клієнтська частина повинна бути розроблена з використанням компонентного підходу, що дозволить легко масштабувати інтерфейс та повторно використовувати елементи коду. Усі мережеві запити між клієнтом та сервером мають бути захищені від поширених вебвразливостей.

Результатом виконання поставленого завдання має стати готовий до розгортання програмний продукт, який вирішує проблеми повільної комунікації та складного пошуку в сегменті C2C класифайдів, пропонуючи користувачам швидкий, безпечний та інтуїтивно зрозумілий інструмент електронної комерції.

1.5 Обґрунтування плану виконання завдання

Розробка сучасних веборієнтованих інформаційних систем, зокрема платформ електронної комерції рівня маркетплейсу, є складним інженерним процесом, що вимагає чіткої регламентації та застосування стандартизованих методологій. Для забезпечення високої якості кінцевого програмного продукту та мінімізації ризиків на етапах проєктування і програмування, виконання завдання базувалося на ітеративній моделі життєвого циклу програмного забезпечення з інтеграцією сучасних практик веброзробки.

Вибір ітеративного підходу обґрунтовується необхідністю поступового нарощування функціоналу: від базової реєстрації користувачів та розробки логіки бази даних до впровадження складних взаємопов'язаних модулів, таких як географічна фільтрація та чат у реальному часі. Весь процес виконання кваліфікаційного завдання було логічно поділено на чотири основні етапи:

– Аналітичний етап та збір вимог включав детальне дослідження предметної області, аналіз конкурентів та формування специфікації вимог до програмного забезпечення. На цьому етапі було визначено межі проєкту, ролі користувачів та сформовано технічне завдання на розробку.

– Етап концептуального та логічного проєктування передбачав розробку архітектури системи до написання вихідного коду. За допомогою уніфікованої мови моделювання було створено моделі бізнес-процесів. Паралельно проєктувалася логічна та фізична структура реляційної бази даних для СУБД MS SQL Server.

– Етап програмної реалізації – найбільш ресурсомісткий етап, який включав розробку серверної частини на базі фреймворку Laravel з використанням патерну MVC та створення клієнтської частини у вигляді Single Page Application за допомогою бібліотеки React та технології Inertia.js.

– Етап тестування та верифікації. Проведення перевірки працездатності створених компонентів. Особлива увага приділялася функціональному тестуванню системи повідомлень, валідації форм та перевірці безпеки маршрутів для захисту персональних даних користувачів.

Такий послідовний план гарантує, що жоден з критичних аспектів розробки системи електронної комерції не буде пропущений. Кожен етап має чітко визначені критерії завершення, що дозволило ефективно контролювати процес розробки та вчасно оптимізувати архітектуру платформи.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було проведено комплексний системний аналіз предметної області електронної комерції, зокрема сегменту платформ онлайн-оголошень. Дослідження існуючих на ринку рішень, таких як OLX та eBay, дозволило виявити їхні архітектурні та функціональні недоліки, серед яких: перевантаженість інтерфейсів, застарілі підходи до рендерингу

сторінок та відсутність оптимізованих систем миттєвої комунікації між користувачами.

На основі виявлених проблем було сформульовано мету та завдання розробки нової, оптимізованої веборієнтованої платформи «MarketFlow». Визначено, що система має функціонувати за принципом Single Page Application, забезпечувати інтерактивний пошук із географічною фільтрацією, підтримувати обмін повідомленнями в реальному часі та містити прозору систему рейтингів продавців.

Для досягнення поставлених цілей було розроблено та обґрунтовано поетапний план виконання завдання, що базується на ітеративній моделі життєвого циклу програмного забезпечення. Обрана методологія гарантує послідовний перехід від збору вимог до програмної реалізації, що створює надійне підґрунтя для переходу до етапу специфікації вимог та UML-моделювання системи в наступному розділі роботи.

2 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АНАЛІЗ НАУКОВИХ ДЖЕРЕЛ

2.1 Обґрунтування вибору технологій розробки

Процес моделювання об'єкту та предмета розробки вимагає аналізу сучасних наукових досліджень у сфері проєктування вебзастосунків. В умовах стрімкого розвитку інформаційних технологій вибір інструментарію має базуватися не лише на практичному досвіді, а й на академічних дослідженнях ефективності, безпеки та продуктивності програмних рішень. Для реалізації платформи ключовими технологіями було обрано фреймворк Laravel (PHP) та концепцію SPA.

З боку клієнтської частини значна увага в наукових джерелах приділяється порівняльному аналізу одно та багатосторінкових застосунків. Згідно з результатами досліджень [4], технологія SPA забезпечує значну перевагу у продуктивності та оптимізації взаємодії з користувачем порівняно з традиційними багатосторінковими архітектурами. Зазначається, що SPA підхід мінімізує кількість запитів до сервера та суттєво зменшує обсяг переданих даних, оскільки після початкового завантаження сторінки динамічно оновлюється лише необхідний контент, а не вся сторінка цілком [4]. Це критично важливо для маркетплейсів, де необхідні швидкий пошук, миттєве оновлення списків товарів та робота системи повідомлень у реальному часі без перезавантаження інтерфейсу.

Аналіз наукових публікацій також підтверджує високу ефективність використання сучасних PHP фреймворків для побудови бекенду. У дослідженні, присвяченому еволюції та застосуванню PHP у веброзробці [6], зазначається, що використання фреймворків дозволяє суттєво оптимізувати процес розробки серверної частини. Увага акцентується на тому, що сучасні PHP рішення автоматизують безліч рутинних завдань, надають готові інструменти для роботи з базами даних, забезпечують гнучку маршрутизацію та вбудовані механізми безпеки [6]. Це дозволяє розробникам зосередитися на реалізації бізнес-логіки та

забезпеченні надійності системи. Забезпечення такого рівня безпеки та масштабованості є критично важливим фактором для систем електронної комерції, які безперервно обробляють конфіденційні дані тисяч користувачів.

Таким чином, результати аналізу актуальних наукових джерел повністю підтверджують доцільність обраного інструментарію. Технологічний стек на базі Laravel та концепції SPA відповідає сучасним академічним та інженерним стандартам проєктування високонавантажених вебплатформ, забезпечуючи надійну, масштабовану та захищену основу для розробки системи [4, 6].

2.2 Функціональні та інформаційні моделі

Якщо структурне моделювання описує статичну архітектуру системи, то функціональне та інформаційне моделювання спрямоване на відображення динаміки бізнес-процесів та трансформації даних у часі. Для опису цих процесів у платформі «MarketFlow» розроблено алгоритмічні моделі, які наочно демонструють маршрутизацію інформації від моменту її введення користувачем до збереження у реляційній базі даних.

Головним інформаційним процесом системи є алгоритм пошуку та фільтрації товарів. Цей процес ініціюється на стороні клієнтського застосунку, коли користувач вводить пошуковий запит або обирає специфічні фільтри, такі як категорія чи географічна локація. Клієнтська частина формує асинхронний HTTP-запит, який містить параметри фільтрації у форматі JSON, і відправляє його на сервер. Бекенд-система перехоплює цей запит, проводить санітизацію вхідних даних для запобігання ін'єкціям та формує складний багатотабличний запит до бази даних MS SQL Server за допомогою засобів об'єктно-реляційного відображення. Після отримання результуючого набору даних сервер здійснює розбиття на сторінки та повертає інформацію клієнту. Важливим побічним інформаційним процесом на цьому етапі є автоматичне оновлення лічильника переглядів: при кожному відкритті детальної картки товару унікальним

користувачем серверний алгоритм інкрементує відповідне поле в базі даних, формуючи таким чином статистику популярності оголошення.

Іншим критично важливим бізнес-процесом є алгоритм публікації нового контенту. Діяльність починається із заповнення користувачем мультипартійної форми, яка містить текстові описи, числові значення ціни та масив медіафайлів. Інформаційний потік на сервері розділяється на дві гілки. Перша гілка відповідає за обробку зображень: система перевіряє розширення файлів, їхній розмір, генерує унікальні імена для уникнення колізій та зберігає фізичні файли у визначену директорію сховища. Друга гілка обробляє текстові дані та шляхи до збережених зображень, формуючи транзакційний запит на вставку пов'язаних записів у таблиці оголошень та галереї. Завдяки використанню транзакцій база даних гарантує цілісність інформації: у разі збою при завантаженні хоча б одного фото весь процес скасовується, і некоректні дані не потрапляють до системи.

Функціональна модель комунікації між користувачами реалізована через алгоритм обміну повідомленнями. Коли покупець ініціює чат, інформаційний потік проходить через контролер повідомлень, який перевіряє права доступу та створює новий запис у таблиці діалогів, пов'язуючи ідентифікатори покупця, продавця та конкретного товару. Усі наступні текстові повідомлення зберігаються із прив'язкою до цього діалогу. Для забезпечення інтерактивності клієнтський застосунок періодично звертається до сервера на наявність нових записів, що дозволяє миттєво оновлювати інтерфейс чату в обох учасників без необхідності повного перезавантаження сторінки.

Логічним завершенням інформаційного циклу взаємодії є алгоритм формування репутації. Після угоди покупець ініціює процес оцінювання, передаючи на сервер числове значення від одного до п'яти та текстовий коментар. Програмний алгоритм записує цей відгук у відповідну таблицю бази даних, після чого автоматично запускається механізм перерахунку середнього рейтингу продавця. Система витягує всі існуючі оцінки цього користувача, застосовує

математичну функцію усереднення та оновлює глобальний показник у профілі продавця, який одразу стає видимим для всіх інших учасників платформи.

2.3 Опис методів, технологій та математичного апарату

Успішна програмна реалізація веборієнтованої платформи та виконання всіх специфікованих функціональних і нефункціональних вимог безпосередньо залежить від правильно обраного технологічного стеку. Процес вибору базувався на критеріях продуктивності, безпеки, швидкості розробки та наявності розвиненої екосистеми підтримки. За результатами аналізу для створення серверної частини було обрано мову програмування PHP та сучасний фреймворк Laravel (рисунок 2.3), а для клієнтської частини – бібліотеку React у поєднанні з технологією Inertia.js. Зберігання даних реалізовано на базі реляційної системи керування базами даних MS SQL Server.

Вибір фреймворку Laravel обґрунтовується його потужною архітектурою MVC, яка дозволяє чітко розділити бізнес-логіку, доступ до даних та їх відображення. Фреймворк надає вбудовані механізми маршрутизації, управління сесіями та захисту від поширених вебвразливостей, таких як підробка міжсайтових запитів. Критичною перевагою Laravel у контексті даного проєкту є наявність об'єктно-реляційного відображення Eloquent ORM. Цей інструмент дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого синтаксису, автоматично генеруючи безпечні SQL-запити, що виключає ризик SQL-ін'єкцій та суттєво пришвидшує процес розробки моделей користувачів, оголошень та повідомлень.

Для забезпечення високої інтерактивності клієнтської частини було обрано бібліотеку React. Її головна перевага полягає у використанні віртуального дерева елементів, що дозволяє миттєво оновлювати лише ті частини інтерфейсу, які дійсно змінилися, не перемальовуючи всю сторінку. Це є критично важливим для реалізації модуля чату в реальному часі та динамічної фільтрації товарів. Щоб уникнути складнощів, пов'язаних із розробкою окремого REST API та

налаштуванням клієнтського роутингу, архітектуру було доповнено прошарком Inertia.js. Ця технологія виступає зв’язуючую між Laravel та React, дозволяючи серверним контролерам безпосередньо повертати React-компоненти з необхідними даними. Такий підхід поєднує швидкість розробки класичного монолітного застосунку з продуктивністю сучасного Single Page Application.

В якості системи керування базами даних обрано MS SQL Server. Ця СУБД корпоративного рівня забезпечує повну підтримку транзакцій, що є обов'язковою умовою для систем електронної комерції. Завдяки надійним механізмам блокування рядків та підтримці складних зв'язків між таблицями, MS SQL Server гарантує цілісність даних маркетплейсу навіть при високих паралельних навантаженнях.

Окрім інформаційних технологій, функціонування системи базується на застосуванні математичного апарату для обробки статистичних даних користувачів. Одним із ключових показників довіри на платформі є глобальний рейтинг продавця, який формується на основі оцінок покупців. Для нівелювання впливу одиничних екстремальних оцінок та забезпечення об'єктивності показника, система автоматично перераховує рейтинг за формулою середнього арифметичного значення.

2.4 Специфікація вимог до програмного забезпечення

1) ***Призначення та межі проєкту:***

1.1) призначення системи: створення зручного та безпечного інструменту для розміщення оголошень, пошуку товарів і комунікації між користувачами;

1.2) погодження: використання термінів SPA (односторінковий застосунок) та API (прикладний програмний інтерфейс);

1.3) межі проєкту: розробка вебклієнта, бекенду, бази даних та чату. Мобільний додаток і платіжні шлюзи не включені.

2) ***Загальний опис:***

2.1) сфера застосування: електронна комерція (C2C та B2C);

2.2) характеристики користувачів: гість (перегляд), авторизований (подача оголошень, чат), адміністратор (модерація);

2.3) загальна структура: фронтенд, бекенд та база даних;

2.4) загальні обмеження: необхідний стабільний інтернет і сучасний браузер із підтримкою JavaScript.

3) Функція системи:

3.1) функція управління оголошеннями:

3.1.1) опис: створення, редагування та видалення товарних пропозицій;

3.1.2) вхід/вихід: вхід – текст, ціна, категорія, локація, фото. Вихід – картка товару;

3.1.3) вимоги: прив'язка до категорій, завантаження до 10 фотографій.

3.2) функція пошуку та фільтрації:

3.2.1) опис: швидкий пошук товару за критеріями;

3.2.2) вхід/вихід: вхід – лючові слова, ціна, місто. Вихід – відфільтровані оголошення;

3.2.3) вимоги: асинхронний повнотекстовий пошук без перезавантаження сторінки.

3.3) функція взаємодії (чат/рейтинги):

3.3.1) опис: забезпечення комунікації та формування довіри до продавця;

3.3.2) вхід/вихід: вхід – повідомлення, оцінки. Вихід – історія чату, середній рейтинг;

3.3.3) вимоги: миттєвий чат із прив'язкою до конкретного оголошення.

4) Вимоги до інформаційного забезпечення:

4.1) джерела даних: заповнені користувачами вебформи та медіафайли;

4.2) довідники: класифікатори категорій та географічних локацій;

4.3) збереження інформації: дані – у реляційній БД, зображення – у файловій системі сервера.

5) Вимоги до інформаційного забезпечення:

Сервер: CPU від 2.4 ГГц, від 2 ГБ RAM, 20 ГБ SSD. До пристроїв кінцевих користувачів специфічних вимог немає.

6) Вимоги до програмного забезпечення:

6.1) архітектура: клієнт-серверна (MVC бекенд, SPA фронтенд);

6.2) системне ПЗ: ОС Windows 11 Pro;

6.3) мережне ПЗ: вбудований вебсервер Internet Information Services;

6.4) ПЗ бази даних: MS SQL Server;

6.5) технології: PHP 8.4 (Laravel) для бекенду, JavaScript (React + Inertia.js) для фронтенду.

7) Вимоги до зовнішніх інтерфейсів:

7.1) інтерфейс користувача: адаптивний;

7.2) апаратний інтерфейс: стандартні пристрої введення-виведення;

7.3) програмний інтерфейс: внутрішній API;

7.4) комунікаційний протокол: HTTP та HTTPS.

8) Властивості програмного забезпечення:

8.1) доступність: цілодобово (24/7);

8.2) супроводжуваність: модульний код із коментарями;

8.3) переносимість: незалежність вебклієнта від операційної системи користувача;

8.4) продуктивність: час відгуку інтерфейсу до 900 мс;

8.5) надійність: захист від втрати даних через транзакції MS SQL Server;

8.6) безпека: хешування паролів Bcrypt, захист від SQL ін'єкцій, CSRF та XSS атак.

9) Інші вимоги:

Основною мовою інтерфейсу та системних сповіщень є українська мова.

Висновки до розділу 2

У другому розділі було проведено глибоке інженерне проєктування системи «MarketFlow». На основі зібраних аналітичних даних розроблено детальну специфікацію вимог до програмного забезпечення, яка чітко розмежовує права доступу для різних категорій користувачів, визначає ключові функціональні модулі платформи, включаючи систему реального часу для обміну повідомленнями та алгоритми географічної фільтрації.

За допомогою інструментарію уніфікованої мови моделювання виконано концептуальне та логічне моделювання об'єкту дослідження. Побудовано діаграми класів та розгортання, які візуалізують розподілену архітектуру системи та зв'язки між її внутрішніми компонентами. Розроблено інформаційні моделі основних бізнес-процесів, що забезпечують безпечний та транзакційний рух даних від клієнтського інтерфейсу до реляційної бази даних.

Крім того, науково обґрунтовано вибір технологічного стеку для практичної реалізації проєкту. Доведено, що використання серверного фреймворку Laravel у поєднанні з клієнтською бібліотекою React та технологією Inertia.js є оптимальним рішенням для створення масштабованого SPA-застосунку. Застосований математичний апарат розрахунку рейтингів дозволяє сформувати прозоре та довірче середовище для користувачів. Отримані проєктні рішення утворюють повну технічну базу, необхідну для переходу до безпосереднього етапу програмної реалізації та тестування розробленого програмного продукту.

3 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

3.1 Об’єктно-орієнтовне моделювання функцій

Будь-яке проєктування починається з візуалізації вимог. У даному випадку перевести уяву у наочний формат допомогла уніфікована мова програмування UML та загальні принципи ООП. Загалом будова платформи розкривається через три базові діаграми. Мова йде про діаграми варіантів використання, класів та розгортання.

Першою розглянуто діаграму варіантів використання. Вона описує роль користувача, який взаємодіє з системою і які він має права. На платформі виділено три основні ролі. Звичайний гість може гортати каталог, шукати потрібні товари через рядок пошуку та дивитися деталі конкретних оголошень. Якщо користувач хоче більшого, він повинен пройти реєстрацію. Після цього гість стає авторизованим користувачем і з'являється можливість створювати власні пропозиції, ховати їх в архів або додавати чужі товари до списку обраного. Також відкривається доступ до приватних повідомлень і системи відгуків. Третя роль – це адміністратор, адже хтось має стежити за порядком на маркетплейсі, модерувати контент та формувати дерево категорій. Діаграму використання можна побачити у додатку А.

Далі переходимо до внутрішньої логіки проєкту. Її найзручніше відображати через діаграму класів. Оскільки під капотом у нас працює фреймворк Laravel, то структура бази даних ідеально лягає на патерн MVC. Головною сутністю виступає клас User. Він зберігає базові дані про людину і пов'язаний ледь не з усіма іншими таблицями. Наприклад, один користувач здатен мати багато оголошень. Саме оголошення містить ціну, опис та певний статус, а ще обов'язково прив'язане до конкретної категорії. Окремої уваги заслуговує клас Review, це відгуки. Він цікавий тим, що має подвійний зв'язок з користувачами, один зовнішній ключ вказує на покупця, який написав коментар, а інший – на продавця, якого саме оцінювали.

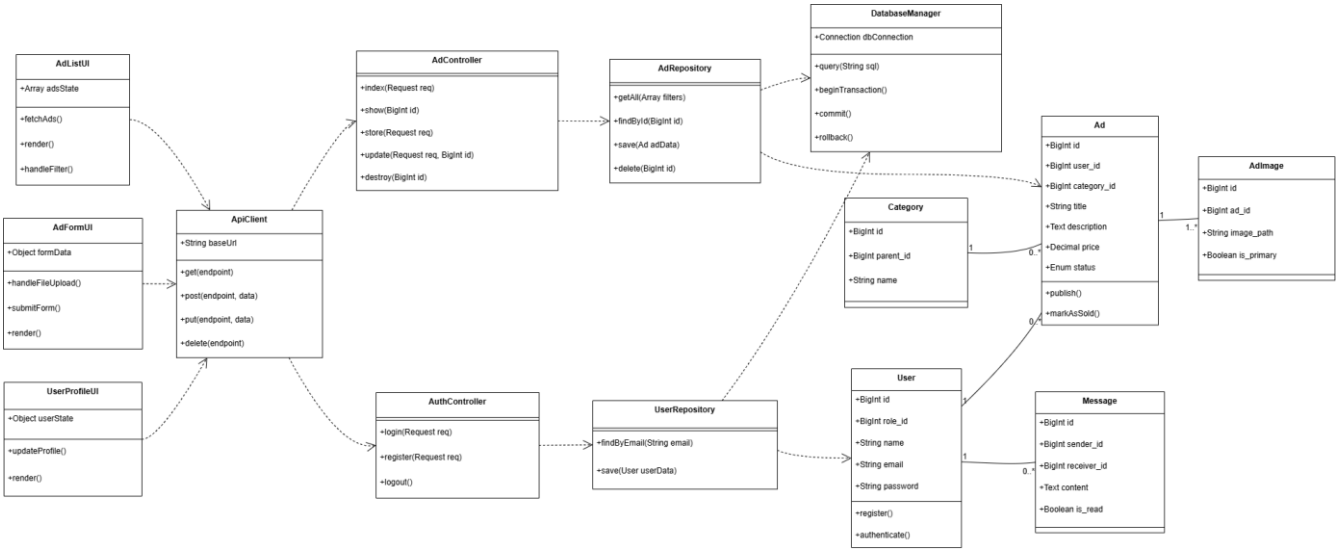


Рисунок 3.1 – Діаграма класів

Окрім об’єктної моделі, ще одну дуже важливу роль відіграє архітектура рівня збереження даних. Оскільки маркетплейси оперують величезними обсягами контенту, виникає потреба у високій швидкості фільтрації та надійності сховища. З цією метою було спроєктовано логічну та фізичну моделі реляційної бази даних. Наочне відображення цих структур через ER-діаграму дає змогу чітко простежити взаємозв’язки між ключовими сутностями платформи та оцінити загальну цілісність системи. Розроблена схема базується на строгих принципах нормалізації, що мінімізує надлишковість інформації та запобігає виникненню аномалій при додаванні чи оновленні записів. Для забезпечення миттєвого відгуку під час складного пошуку та багатопараметричної фільтрації товарів передбачено використання оптимізованих індексів у ключових таблицях каталогу. Такий підхід гарантує необхідну транзакційну надійність під час обробки замовлень і проведення фінансових операцій. У результаті, спроєктований рівень збереження даних утворює стійкий фундамент, повністю готовий до подальшого масштабування та безперебійної роботи за умов пікових навантажень.

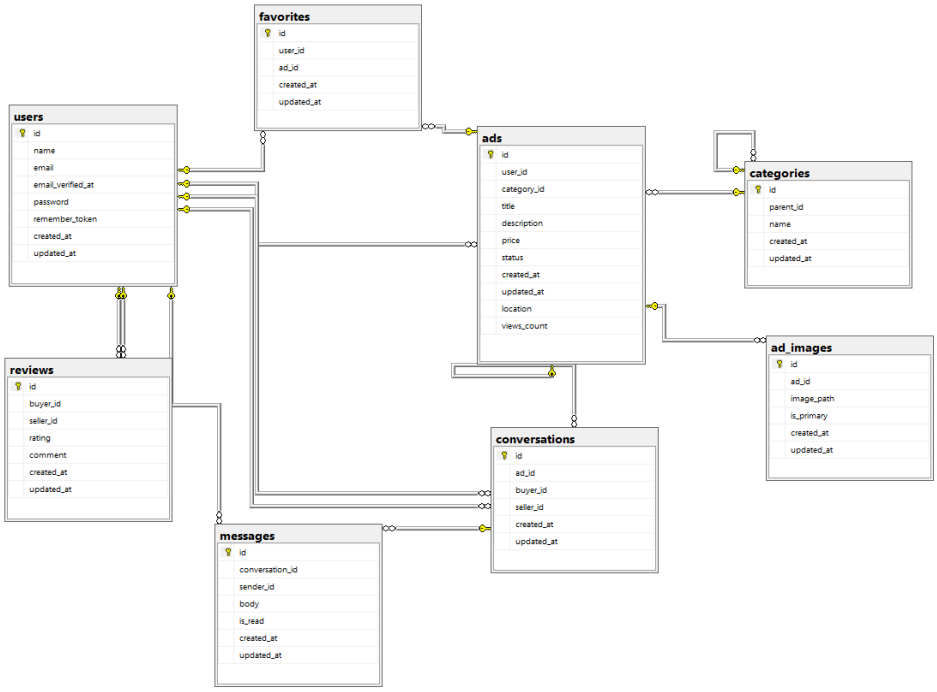


Рисунок 3.2 – ER-діаграма бази даних

Як видно на схемі, архітектура бази даних спроєктована із суворим дотриманням теорії реляційних баз. З шести існуючих нормальних форм, від 1НФ до 6НФ, структуру приведено до третьої нормальної форми. Такий рівень заглиблення є оптимальним балансом: це зроблено навмисно, щоб усунути транзитивні залежності, уникнути дублювання інформації та аномалій при оновленні записів, але при цьому зберегти високу швидкість виконання запитів. Центральною частиною системи, від якої розгалужується більшість зв'язків, виступає таблиця users. Вона зберігає базові облікові дані і безпосередньо пов'язана зовнішніми ключами з усіма іншими процесами – від створення оголошень до написання коментарів.

Головною бізнес-сутністю платформи є таблиця ads, котра використовується для оголошень. Вона містить в собі всю комерційну інформацію: заголовок, текстовий опис, ціну, поточний статус активності та кількість переглядів. Кожне оголошення жорстко прив'язане до свого автора через user_id. Щоб не перевантажувати цю таблицю важкими даними, фотографії товарів винесено в окрему сутність ad_images. Зв'язок тут класичний: одне

оголошення може мати багато картинок. При цьому поле is_primary допомагає системі швидко розуміти, яке саме фото треба показувати на головній сторінці як обкладинку.

Ще одне архітектурне рішення застосовано у таблиці categories. Замість того, щоб створювати окремі таблиці для головних розділів і дрібних підкатегорій, було використано патерн самопосилання. Таблиця має зовнішній ключ parent_id, який посилається на її ж власний ідентифікатор id. Завдяки цьому можна будувати дерево категорій абсолютно будь-якої глибини вкладеності.

Окремо треба подивитись на розділ комунікації між користувачами. Замість примітивного зберігання всіх повідомлень в одній купі, систему чатів розділено на дві взаємопов'язані таблиці: conversations та messages. Таблиця діалогів виступає контейнером, вона фіксує, хто є покупцем, хто продавцем, і якого саме товару стосується ця розмова. А вже всередині таблиці messages зберігаються самі тексти повідомлень із позначкою is_read, що дозволяє легко виводити сповіщення про непрочитані листи.

Завершують структуру бази даних допоміжні таблиці favorites та reviews. Таблиця обраного працює як зв'язок між користувачем та чужим оголошенням. А таблиця відгуків відрізняється тим, що має одразу два зовнішні ключі на таблицю користувачів: buyer_id фіксує автора коментаря, а seller_id вказує на того, чий рейтинг зараз оцінюється. Всі зв'язки у базі налаштовані правилом каскадного видалення. Тобто, якщо адміністратор видаляє користувача з системи, база даних автоматично підчистить усі його оголошення, повідомлення та залишені відгуки, не залишаючи неактуальних даних.

Нарешті, треба чітко розуміти, як це все працює на фізичному рівні. Для цього була побудована діаграма розгортання, вона розбиває проєкт на три логічні вузли. Починається маршрут з клієнтського пристрою. Це може бути звичайний браузер на телефоні чи комп'ютері, де виконується скомпільований код React-застосунку. Всі запити звідти по захищеному каналу відправляються на вебсервер. Там їх приймає Nginx і передає безпосередньо PHP-додатку. Саме на

цьому етапі відбувається обробка бізнес-логіки. А щоб інформація не зникала після перезавантаження, вебсервер постійно спілкується з сервером баз даних. Для цієї ролі було обрано СКБД MySQL. Зв'язок між серверами йде через стандартні мережеві порти, що гарантує надійне збереження профілів, товарів та історій переписок. Діаграму розгортання можна побачити у додатку А.

3.2 Алгоритмічне та математичне забезпечення системи

Такого маленького опису, як простий малюнку класів та зв'язків не вистачить для повноцінного опису. Самій системі ще треба пояснити, як саме вона має працювати крок за кроком. Для цього формується алгоритмічне та математичне забезпечення. Згідно правилам проєктування, процес завжди йде від загального до конкретного. Тобто спочатку ми відповідаємо на питання «Що зробити?», а вже потім детально розбираємо «Як це зробити?».

Оскільки на платформі є система відгуків, тут не обійтися без математики. Функція, яка потребує обчислень – це розрахунок середнього рейтингу продавця. Цей показник оновлюється щоразу, коли хтось залишає нову оцінку. Математично це виглядає як класичне середнє арифметичне. Це значення не просто обчислюється на льоту, а перераховується і кешується у профілі користувача. Такий підхід рятує сервер від зайвих навантажень при кожному відкритті сторінки продавця.

Для детального аналізу функціональної логіки платформи варто розглянути ключовий бізнес-процес будь-якого маркетплейсу – публікацію нового оголошення. Послідовність виконання цього алгоритму наочно змодельовано за допомогою UML-діаграми діяльності.

Процес чітко розділено на дві зони відповідальності: взаємодію з користувачем та внутрішню обробку системою. На першому етапі користувач ініціює створення оголошення, послідовно заповнює атрибути форми товару та завантажує необхідні медіафайли. Після натискання кнопки «Опублікувати» потік управління переходить до програмної частини. Система насамперед виконує

валідацію отриманих даних. У разі виявлення невідповідностей, алгоритм виводить повідомлення про помилки та циклічно повертає користувача на етап редагування форми. Якщо ж дані успішно проходять перевірку, відбувається їх транзакційне збереження в базі даних. На завершальному етапі система генерує фінальне представлення запису та автоматично перенаправляє користувача на сторінку готового оголошення. Діаграму діяльності можна побачити у додатку А.3.

Загалом, описаний високорівневий алгоритм формує надійний базис для програмної реалізації відповідних контролерів бекенду. Чітке розмежування клієнтської та серверної логіки дозволяє ізолювати процеси перевірки та збереження даних, забезпечуючи високий рівень безпеки та стійкості системи до некоректного введення. Подібний модульний підхід до проєктування гарантує гнучкість платформи і легкість її подальшого масштабування.

3.3 Архітектура та вибір компонентів

Щоб усі описані алгоритми працювали швидко і без збоїв, системі потрібен міцний фундамент. Саме тому логічна структура програмного забезпечення базується на чіткому розподілі обов'язків між клієнтською та серверною частинами.

Взаємодія користувача з платформою починається через компоненти інтерфейсу. Це модулі списку оголошень, форми створення товарів та сторінка профілю. Ці фронтенд-компоненти не мають прямого доступу до бази даних. Вони зв’язуються із серверною частиною виключно через спеціалізований програмний клієнт Inertia.js, який виступає надійним мостом між шарами програми.

На стороні сервера запити перехоплюються відповідними контролерами. Зокрема, контролером автентифікації та контролером оголошень. Вони відповідають за валідацію вхідних даних та маршрутизацію бізнес-логіки. А для забезпечення принципу єдиної відповідальності, доступ до даних реалізовано

через патерн «Репозиторій». Замість того, щоб писати складні запити прямо в контролерах, вони змушені звертатися до репозиторіїв користувачів та оголошень. Ті, у свою чергу, через менеджер баз даних взаємодіють з основними сутностями предметної області: об'єктами користувача, категоріями, зображеннями та повідомленнями. Така модульна структура гарантує високу гнучкість системи та легкість внесення змін у майбутньому.

Фізична архітектура мережевої взаємодії також має особливості. Усі мережеві запити від клієнта передаються через захищені протоколи HTTP/HTTPS до вебсервера, який виконує роль зворотного проксі сервера. Його головна мета – балансування навантаження та перенаправлення API-запитів до контейнера з серверним застосунком, де безпосередньо функціонує ядро на базі фреймворку Laravel.

Важливо пояснити, що серверна частина архітектури не є монолітною. Вона розподіляє збереження даних на кілька незалежних вузлів, де для зберігання реляційних даних, таких як профілі користувачів, тексти оголошень та повідомлення чату, використана база даних MySQL. Але звертатися до диска при кожному запиті було б занадто довго, тому для оптимізації швидкодії та управління сесіями користувачів залучено систему кешування Redis, яка працює безпосередньо у швидкій пам'яті сервера.

У цілому, розроблені логічні та фізичні моделі утворюють надійну, масштабовану та захищену архітектуру. Вона повністю здатна витримувати високі навантаження, характерні для сучасних систем електронної комерції та C2C-маркетплейсів.

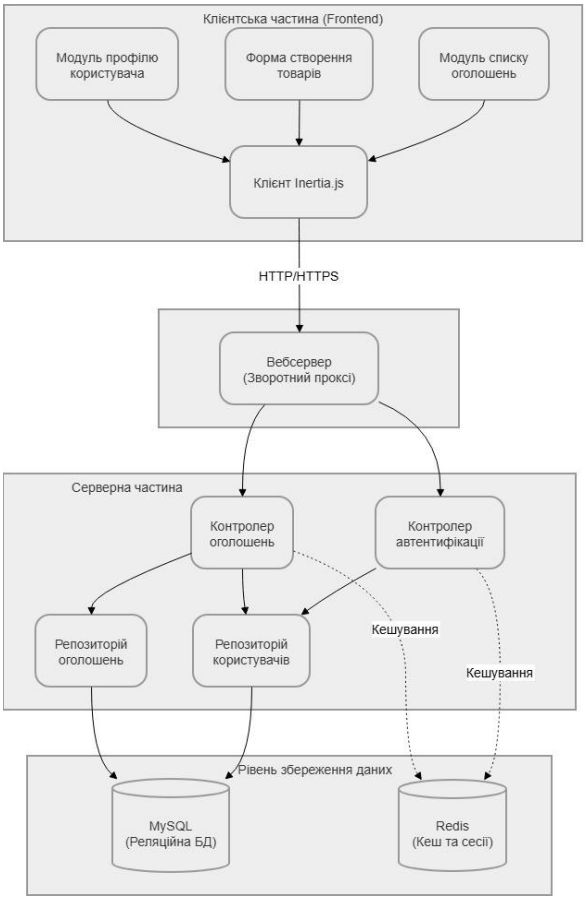


Рисунок 3.4 – Діаграма компонентів

Як видно з показаної вище діаграми, застосована архітектурна модель забезпечує сувору інкапсуляцію процесів. Клієнтська частина ізольована від прямої взаємодії з рівнем даних, що означає мінімізовану вразливості безпеки. У свою чергу, використання патерну «Репозиторій» на бекенді створює єдину точку входу для роботи з MySQL та Redis. Завдяки такій декомпозиції кожен компонент системи може оновлюватися, тестуватися або замінюватися незалежно від інших, що є критично важливим для довгострокової підтримки проєкту.

3.4 Проєктування інтерфейсу

Зовнішній вигляд застосунку такий же важливий, як і його внутрішня логіка. Навіть найкращий бекенд не врятує ситуацію, якщо користувачу буде незручно ним користуватися. Тому інтерфейс платформи проєктувався з огляду на сучасні тренди UI/UX, де головним правилом є мінімалізм та інтуїтивна зрозумілість.

Для побудови інтерфейсу було обрано Tailwind CSS, що дало змогу замінити об’ємні таблиці стилів гнучким компонуванням елементів. Проєкт реалізовано за принципом Mobile First. Такий принцип означає, що в першу чергу все оптимізувалося для користувачів мобільних пристроїв, а адаптація під ширші монітори була наступним етапом. Завдяки цьому складні блоки, як каталог чи чат, зберігають цілісність на екрані будь-якого гаджета.

На етапі проєктування було створено декілька макетів для ключових екранів системи, серед них був макет головної сторінки. Вона зустрічає користувача великим блоком пошуку та швидкими фільтрами. Нижче розташовується список популярних категорій, а основну площу займає нескінченна стрічка останніх доданих оголошень у вигляді компактних карток із фотографією, ціною та назвою.

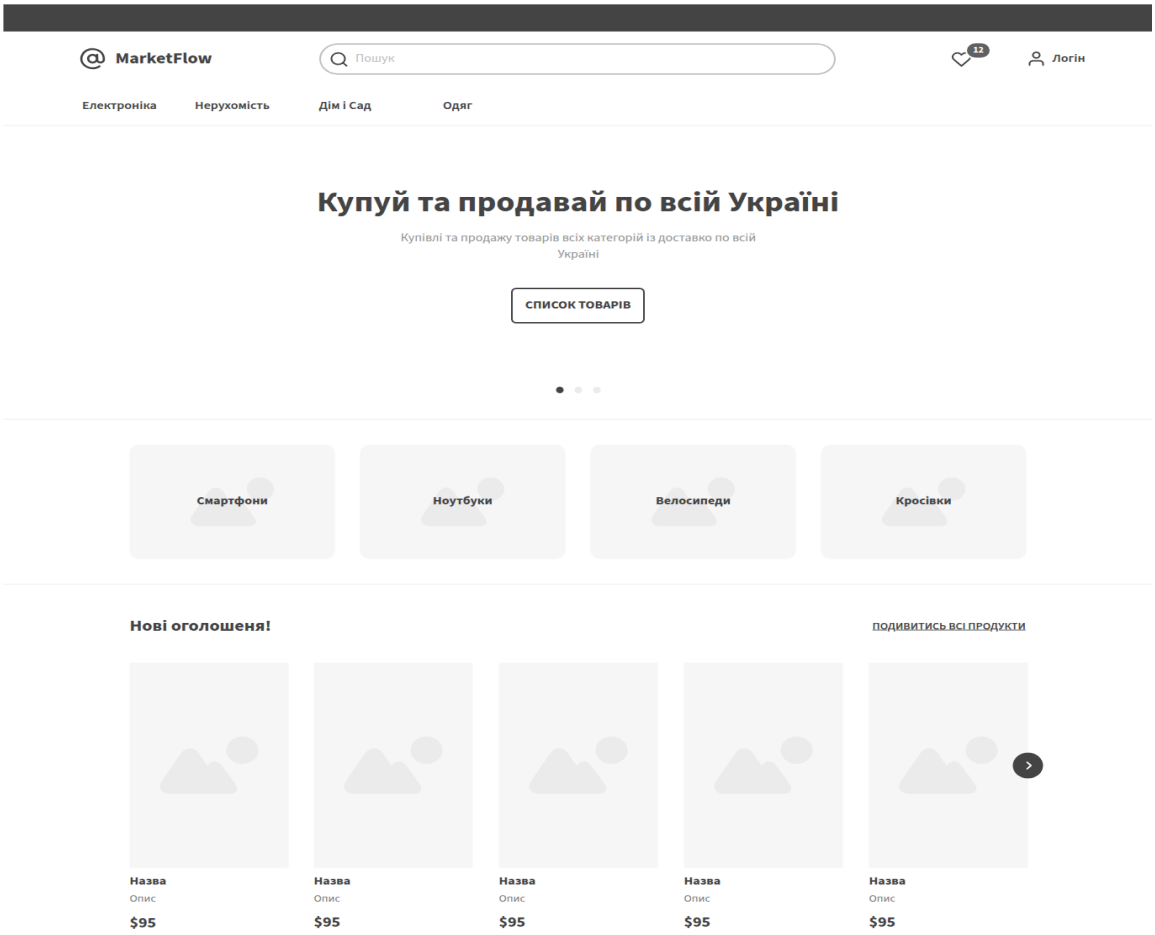


Рисунок 3.5 – Макет головної сторінки

Наступний дуже важливий макет – це детальна картка товару. Вона поділена на дві смислові зони, де ліва частина повністю віддана під візуальний контент, там знаходиться інтерактивна галерея зображень. Права колонка містить усю комерційну інформацію. Там виведено ціну, детальний опис, ім'я продавця та його поточний рейтинг. Одразу під профілем продавця розміщені кнопки цільової дії: «Написати повідомлення» та «Додати в обране». Блок із відгуками інших покупців розташовується трохи нижче основної інформації, щоб не перевантажувати перший екран.

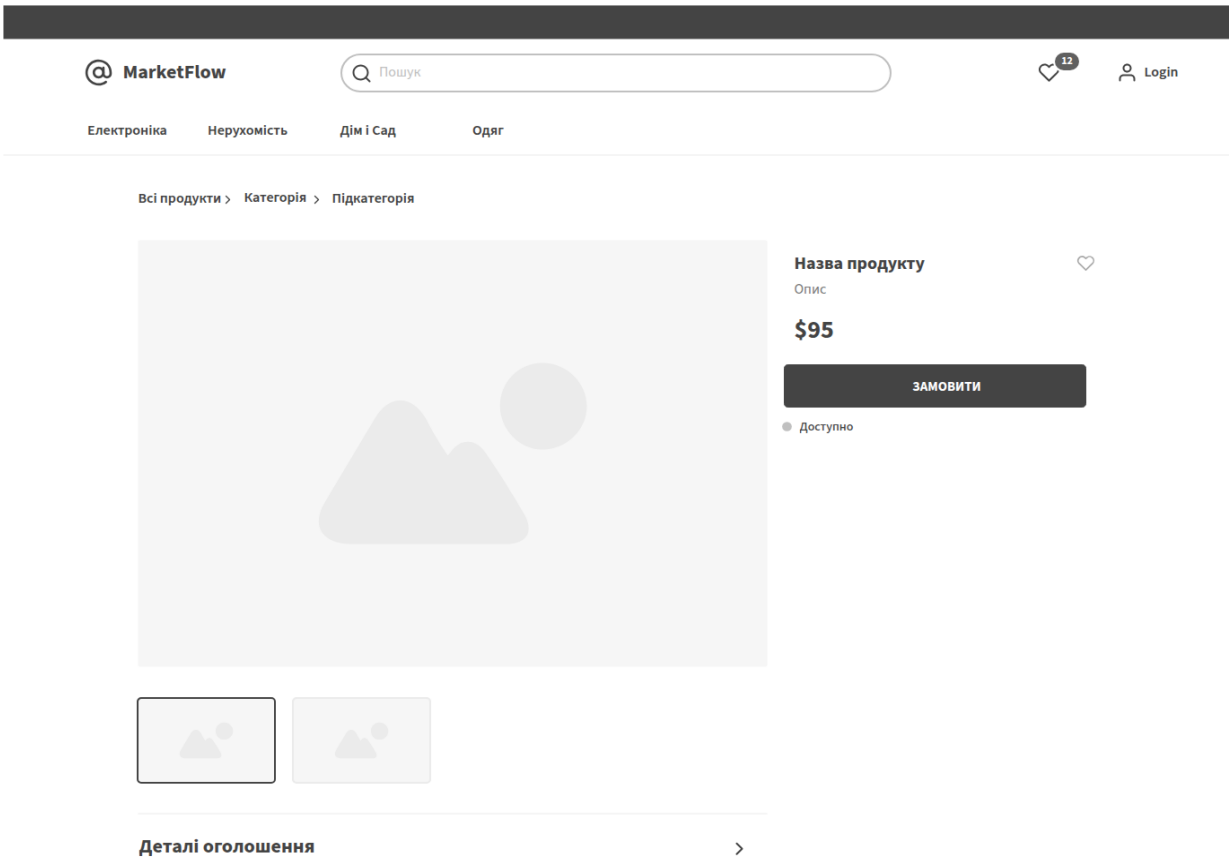


Рисунок 3.6 – Макет картки оголошення

Окремо були спроєктовані інтерфейс особистого кабінету та системи повідомлень. Месенджер виконаний у класичному для сучасних платформ стилі, де можна побачити колонку з активними діалогами, а натиснувши на один з них, відкривається повний діалог з іншим користувачем. Самі повідомлення виглядають дуже мінімалістично та сучасно.

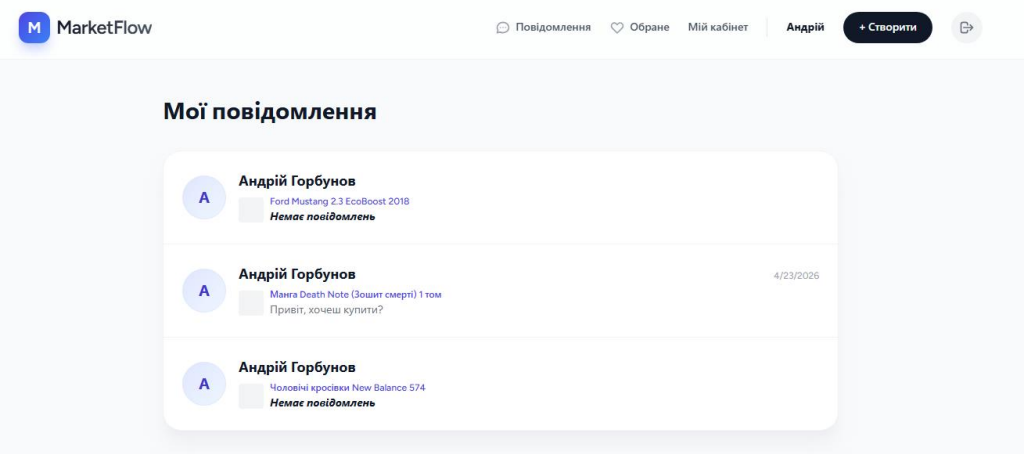


Рисунок 3.7 – Сторінка повідомлень

Таке візуальне компонування елементів управління забезпечує високий рівень ергономіки та інтуїтивно зрозумілу навігацію.

Висновки до розділу 3

Третій розділ став одним із найголовніших, ідеї з голови перейшли до конкретних інженерних рішень. Завдяки застосуванню об'єктно-орієнтованого підходу та стандартів UML вдалося чітко розподілити ролі користувачів, спроєктувати структуру класів та визначити фізичну структуру сервера. Крім того, створена детальна ER-діаграма бази даних та побудовані блок-схеми алгоритмів дали точне розуміння того, як саме система оброблятиме інформацію під капотом. Тепер процес публікації оголошень чи математичний розрахунок рейтингу продавців мають під собою чітку логічну базу.

Багато уваги було приділено архітектурній міцності застосунку. Розподіл обов'язків між фронтендом та бекендом та інтеграція системи кешування Redis довели, що платформа гарно спроєктована, як масштабоване рішення. Обрані технологічні рішення дозволяють системі легко витримувати велике навантаження, що вважається критично важливим для будь-якого сучасного сайту.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

У четвертому розділі наведено результати програмної реалізації розробленої системи електронної комерції. Описано специфікацію програмного забезпечення, визначено структуру даних та подано опис розроблених класів. Крім того, розділ містить результати тестування системи, аналіз її працездатності, а також розроблене керівництво користувача з ілюстраціями реального інтерфейсу на кожному етапі виконання програми.

4.1 Специфікація програмного забезпечення та опис класів

Програмну реалізацію серверної частини платформи виконано з використанням об'єктно-орієнтованої мови програмування PHP та фреймворку Laravel. Відповідно до архітектурного шаблону MVC. Логіку застосунку розділено на моделі даних, контролери та представлення.

Модель оголошення реалізовано через клас Ad, основним призначенням якого є взаємодія з таблицею оголошень бази даних та управління об'єктами товарних пропозицій. Серед атрибутів класу визначено масив рядкових даних $fillable, який визначає перелік полів для автоматичного заповнення, такі як title, description, price, location. Для фіксації поточного стану оголошення в системі застосовуються рядкові константи. Специфікація методів включає метод owner(), що реалізує зв'язок «багато до одного» без прийняття зовнішніх змінних та повертає об'єкт класу User, а також метод images(), який реалізує зв'язок «один до багатьох» і повертає колекцію об'єктів AdImage, що складають галерею товару.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Ad extends Model
{
    use HasFactory;
    protected $fillable = ['user_id', 'category_id', 'title', 'description',
'price', 'status', 'location',];
```

```
    public function owner()
    {
        return $this->belongsTo(User::class, 'user_id');
    }

    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    public function images()
    {
        return $this->hasMany(AdImage::class);
    }
    public function favoritedBy()
    {
        return $this->belongsToMany(User::class, 'favorites', 'ad_id', 'user_id')-
>withTimestamps();
    }
}
```

Логіку обробки мережевих запитів, валідацію вхідних даних та збереження відгуків користувачів реалізовано у класі ReviewController. Основною локальною змінною методу є об'єкт $request, який агрегує всі вхідні параметри, передані клієнтом. Крім того, використовується масив $validated для збереження виключно перевірених та безпечних даних після етапу валідації. Ключовим методом контролера є store(), який приймає об'єкт запиту як аргумент та виконує збереження нового запису. У тілі цього методу застосовуються логічні оператори для перевірки відповідності ідентифікаторів, що унеможливлює оцінювання користувачем власного профілю. Результатом його виконання є створення об'єкта Review.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Review;

class ReviewController extends Controller
{
    public function store(Request $request)
{
    $validated = $request->validate([
        'seller_id' => 'required|exists:users,id',
        'rating' => 'required|integer|min:1|max:5',
        'comment' => 'nullable|string|max:500',
```

```
    ]);

    if (auth()->id() == $validated['seller_id']) {
        return back()->with('error', 'Ви не можете оцінити самого себе.');
    }

    Review::create([
        'buyer_id' => auth()->id(),
        'seller_id' => $validated['seller_id'],
        'rating' => $validated['rating'],
        'comment' => $validated['comment'],
    ]);

    return back()->with('message', 'Дякуємо за відгук!');
}
}
```

Управління життєвим циклом оголошень, включаючи маршрутизацію збереження, відображення та пошуку, здійснюється у класі AdController. Його специфікація містить метод show(), який відповідає за формування детальної сторінки товару. Локальною змінною даного методу виступає об'єкт моделі Ad. У тілі методу виконується інкрементація математичної змінної переглядів (views_count), завантаження пов'язаних сутностей (відгуки продавця та галерея), після чого сформований інформаційний масив передається до клієнтської частини за допомогою технології Inertia.js.

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required|string|min:5|max:255',
        'category_id' => 'required|exists:categories,id',
        'price' => 'required|numeric|min:0',
        'description' => 'required|string|min:10',
        'images.*' => 'image|mimes:jpeg,png,jpg,webp|max:5120',
        'location' => 'nullable|string|max:255'
    ]);

    $validated['user_id'] = auth()->id();
    $validated['status'] = 'active';

    $ad = \App\Models\Ad::create($validated);

    if ($request->hasFile('images')) {
        foreach ($request->file('images') as $index => $image) {
            $path = $image->store('ads', 'public');

            \App\Models\AdImage::create([
                'ad_id' => $ad->id,
                'image_path' => $path,
                'is_primary' => $index === 0 ? true : false,
```

```
            ]);
        }
    }

    return redirect()->route('home');
}
```

4.2 Програмна реалізація базових модулей системи

У представленому нижче лістингу реалізовано механізм оновлення інтерфейсу. За допомогою хука useState система відстежує зміни у полях пошуку та сортування. При відправці форми викликається метод router.get з бібліотеки Inertia.js. Параметри preserveState та preserveScroll гарантують, що під час отримання нових даних від сервера сторінка не буде перезавантажуватися повністю, а прокрутка екрану та фокус на полі введення залишаться на своїх місцях. Також у коді передбачено умовний рендеринг, система перевіряє наявність фотографій у масиві товару і, у разі їх відсутності, автоматично підставляє заглушку «Фото очікується», що запобігає виникненню візуальних помилок.

```
import React, { useState } from 'react';
import { Link, router } from '@inertiajs/react';

export default function Index({ ads, categories, filters }) {
    const [values, setValues] = useState({
        search: filters?.search || '',
        category: filters?.category || '',
        sort: filters?.sort || 'newest',
    });

    const handleChange = (e) => {
        setValues(values => ({ ...values, [e.target.name]: e.target.value }));
    };

    const handleSubmit = (e) => {
        e.preventDefault();
        router.get('/', values, { preserveState: true, preserveScroll: true });
    };

    return (
        <main>
            <form onSubmit={handleSubmit}>
                <input name="search" value={values.search} onChange={handleChange}
/>
                <select name="category" value={values.category}
onChange={handleChange}>
```

```
                    <option value="">Всі категорії</option>
                    {categories.map(cat => (
                        <option key={cat.id} value={cat.id}>{cat.name}</option>
                    ))}
                </select>
                <button type="submit">Знайти</button>
            </form>

            <div className="grid-layout">
                {ads.data.map((ad) => (
                    <Link key={ad.id} href={route('ads.show', ad.id)}>
                        {ad.images && ad.images.length > 0 ? (
                            <img src={`/storage/${ad.images[0].image_path}`}
alt={ad.title}/>
                        ) : (
                            <div>Фото очікується</div>
                        )}
                        <h2>{ad.title}</h2>
                        <div>{Number(ad.price).toLocaleString()} ₴</div>
                    </Link>
                ))}
            </div>
        </main>
    );
}
```

4.3 Тестування програмного забезпечення

Будь-яке програмне забезпечення потребує ретельної перевірки перед фінальним релізом. Наявність чистого коду не гарантує його правильної роботи в реальних умовах, тому обов'язковим етапом розробки стало комплексне тестування платформи. Головний акцент робився на функціональному тестуванні за методом чорної скриньки. Логіка перевірялася не зсередини через читання коду, а шляхом імітації дій звичайного користувача через графічний інтерфейс.

Основна увага приділялася критичним шляхам взаємодії: процесу реєстрації, публікації нового контенту та роботі пошукових фільтрів. Також ретельно перевірялася реакція серверної частини на відверто невалідні вхідні дані. Найбільш показові сценарії перевірки зведено у таблицю 4.1.

Таблиця 4.1 – Сценарії перевірки

ID тесту	Опис сценарію	Вхідні дані	Очікуваний результат	Фактичний результат
1	Спроба публікації оголошення з порожнім полем ціни	Назва: «Відеокарта RTX5070» Ціна: [порожньо]	Блокування форми. Виведення помилки: "Поле ціна є обов'язковим"	Успішно. Форма не відправлена, помилку виведено
2	Реєстрація нового акаунта з використанням зайнятої пошти	Email: admintest@mail.com (вже існує в БД)	Повернення на сторінку реєстрації з відповідним попередженням	Успішно. Система заблокувала створення дубля
3	Завантаження файлу некоректного формату замість фото	Файл: document.pdf	Спрацювання валідатора mimes:jpeg,png. Помилка завантаження зображення	Успішно. Файл відхилено до збереження на сервер
4	Перевірка захисту системи відгуків	Натискання кнопки "Оцінити" на власному товарі	Кнопка прихована інтерфейсом.	Успішно. Логіка захисту спрацювала коректно
5	Робота пошуку з використанням часткового збігу слова	Пошуковий запит: "айфон"	Виведення всіх оголошень, де є це слово	Успішно. Inertia.js оновила список без оновлення сторінки

За допомогою аналізу отриманих результатів можна зробити однозначний висновок. Розроблене програмне забезпечення стабільно реагує на дії користувача і не допускає виникнення критичних збоїв. Механізми валідації на стороні бекенду надійно захищають базу даних від потрапляння непотрібної інформації.

Всі функціональні вимоги, які висувалися до платформи на етапі проєктування, виконані в повному обсязі.

4.4 Результати працездатності

Оцінити вдалість проєкту, на жаль, не можна без аналізу реальних результатів. Програмне забезпечення пройшло серію тестів, в яких використовувались різні набори вхідних даних. В основному, увага зверталася на те, як швидко платформа обробляє запити та чи коректно працюють закладені математичні алгоритми.

Щодо алгоритмів, найважливішим було перевірити механізм розрахунку рейтингу продавця. У третьому розділі наводилася відповідна математична модель, на практиці її реалізація працює наступним чином: якщо продавець має базовий рейтинг і система отримує масив нових вхідних даних, запускається перерахунок. Результат обчислення математично усереднюється, заокруглюється до десятих і одразу записується в профіль користувача. Вивід цих розрахунків системою відбувається миттєво, тому дозволяє іншим покупцям одразу бачити актуальний рейтинг продавця та вплинути на їх вибір.

Окремо проведено оцінку якості розробленого ПЗ з точки зору швидкодії. Найбільш ресурсомістким процесом на будь-якому маркетплейсі є фільтрація товарів. Для перевірки якості коду вимірювався час відгуку сервера при роботі з наповненим каталогом. Заміри проводилися для різних типів вхідних даних, а отримані метрики зведено у таблицю 4.2.

Таблиця 4.2 – Метрики тестування фільтрування

Тип вхідних даних	Кількість знайдених результатів	Час обробки	Загальний час генерації сторінки
Пошук за точною назвою	142	18 мс	45 мс
Сортування за ціною	142	24 мс	52 мс
Текстовий пошук	38	41 мс	78 мс
Комбінований запит	12	56 мс	92 мс

Отримані результати показують, що розроблений застосунок є дуже високого рівня та відповідає стандартам продуктивності. Навіть при найскладніших комбінованих запитах із розгалуженою фільтрацією, загальний час генерації відповіді не перевищує 100 мілісекунд. Це набагато менше, ніж загальноприйнятий поріг комфортної роботи користувача у вебзастосунках. Швидкий відгук інтерфейсу напряму впливає на задоволеність клієнтів, знижує відсоток відмов та стимулює їх до подальшої взаємодії з каталогом. Динамічне завантаження зображень також не заторможує систему завдяки роздільній архітектурі. Сервер не витрачає основний час на обробку медіафайлів під час видачі результатів, що дозволяє паралельно обслуговувати велику кількість одночасних сесій. Таким чином, розроблена система здатна безперебійно працювати при великих обсягах вхідної інформації.

4.5 Керівництво користувача

Для забезпечення зручної взаємодії користувача з платформою MarketFlow, було додано детальну інструкцію. Інтерфейс спеціально спроєктований таким чином, щоб користувач міг зрозуміти його інтуїтивно, тому навігація не вимагає спеціальних технічних навичок. Усі ключові етапи роботи супроводжуються візуальними підказками та повідомленнями системи.

Робота з платформою починається з етапу ідентифікації. Неавторизований гість має доступ лише до режиму перегляду каталогу. Щоб отримати можливість купувати або продавати товари, необхідно перейти на сторінку автентифікації. Форма входу та реєстрації містить базові поля: електронну пошту, пароль та ім'я. Система автоматично перевіряє правильність введених даних і, у разі успіху, перенаправляє особу на головну сторінку.

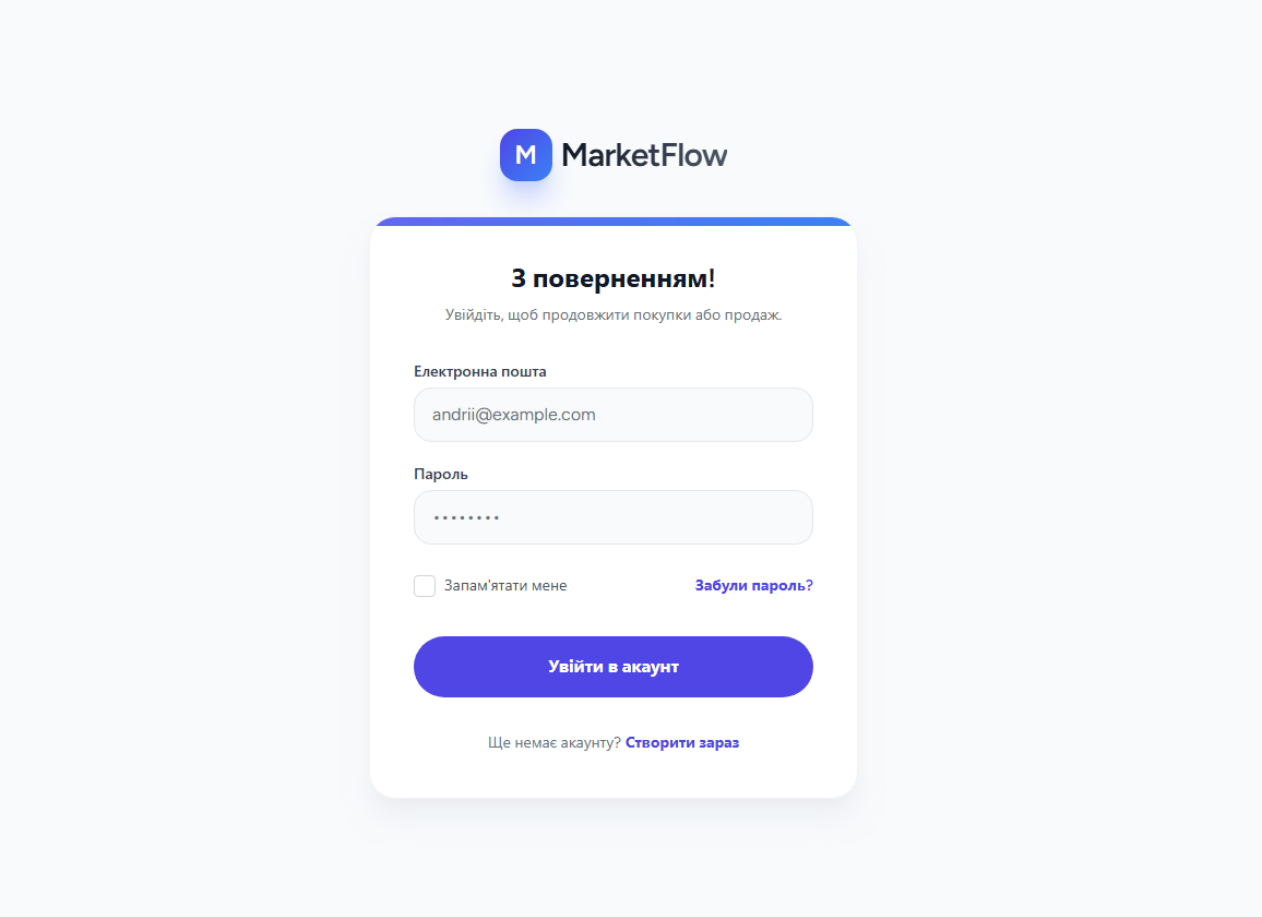


Рисунок 4.1 – Сторінка входу

Головний екран платформи є найголовнішим, бо він є відправною точкою для зацікавлення користувача у придбанні якогось товару. У верхній частині цієї сторінки розташовано навігаційну панель, в якій можна отримати доступ до особистого кабінету, повідомлень та списку улюблених товарів. Центральне місце займає великий блок пошуку, в якому користувач може ввести назву бажаного товару в текстове поле або скористатися випадаючими списками для вибору конкретної категорії та методу сортування. Одразу під блоком пошуку система виводить динамічну сітку з актуальними оголошеннями у вигляді компактних карток із фотографією та ціною. Така структура дозволяє миттєво зорієнтуватися в асортименті та перейти до перегляду деталей лише в один клік. Завдяки продуманому інтерфейсу шлях від першого візиту до вибору потрібної позиції стає максимально швидким та інтуїтивно зрозумілим.

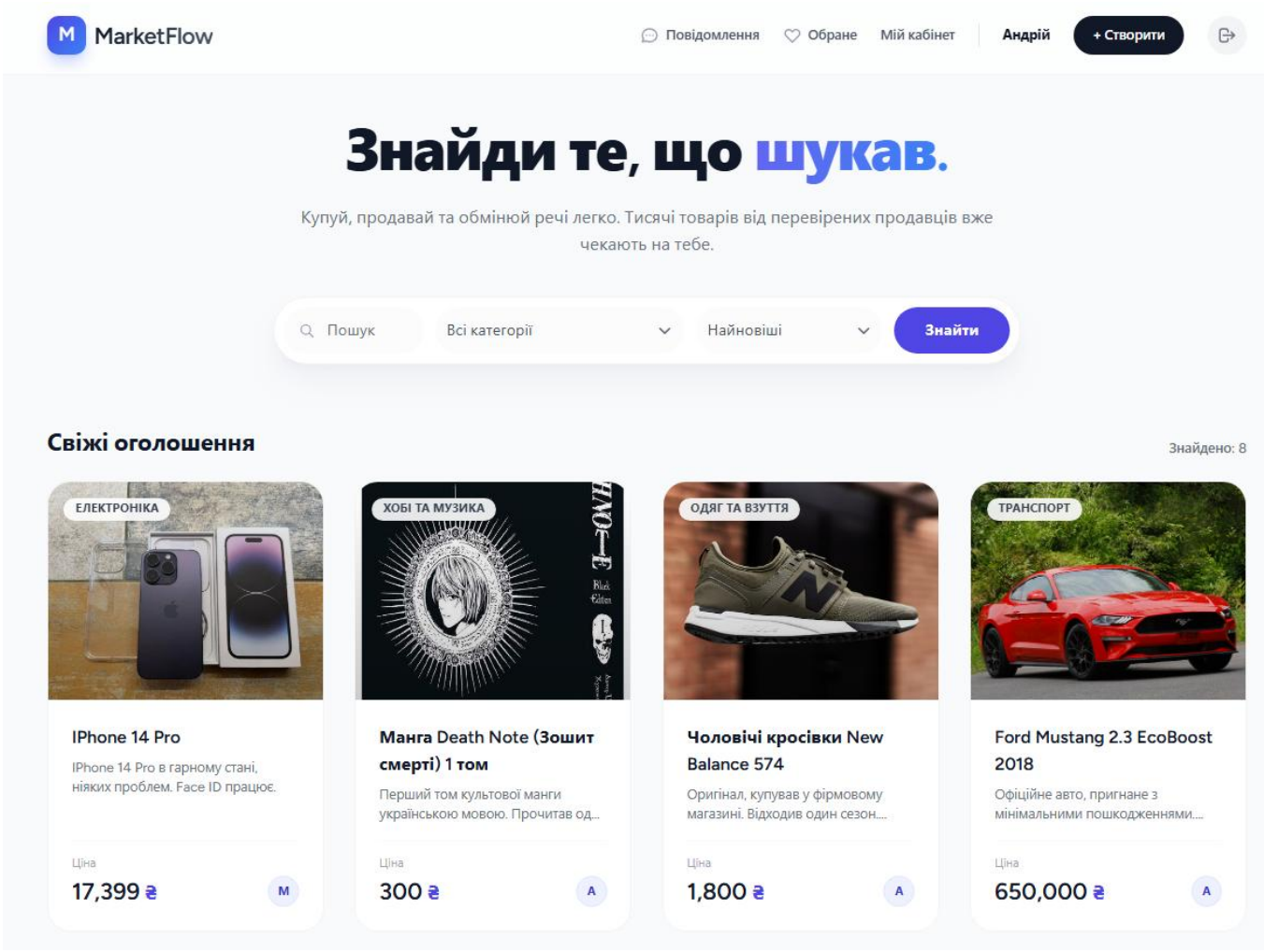


Рисунок 4.2 – Головна сторінка

Для керування власними оголошеннями був розроблений особистий кабінет, у ньому зібрано весь персональний контент. Тут людина може переглянути список своїх активних публікацій, перевірити кількість їх переглядів або перейти до архіву. Через цей самий інтерфейс здійснюється перехід до форми створення нової товарної пропозиції.

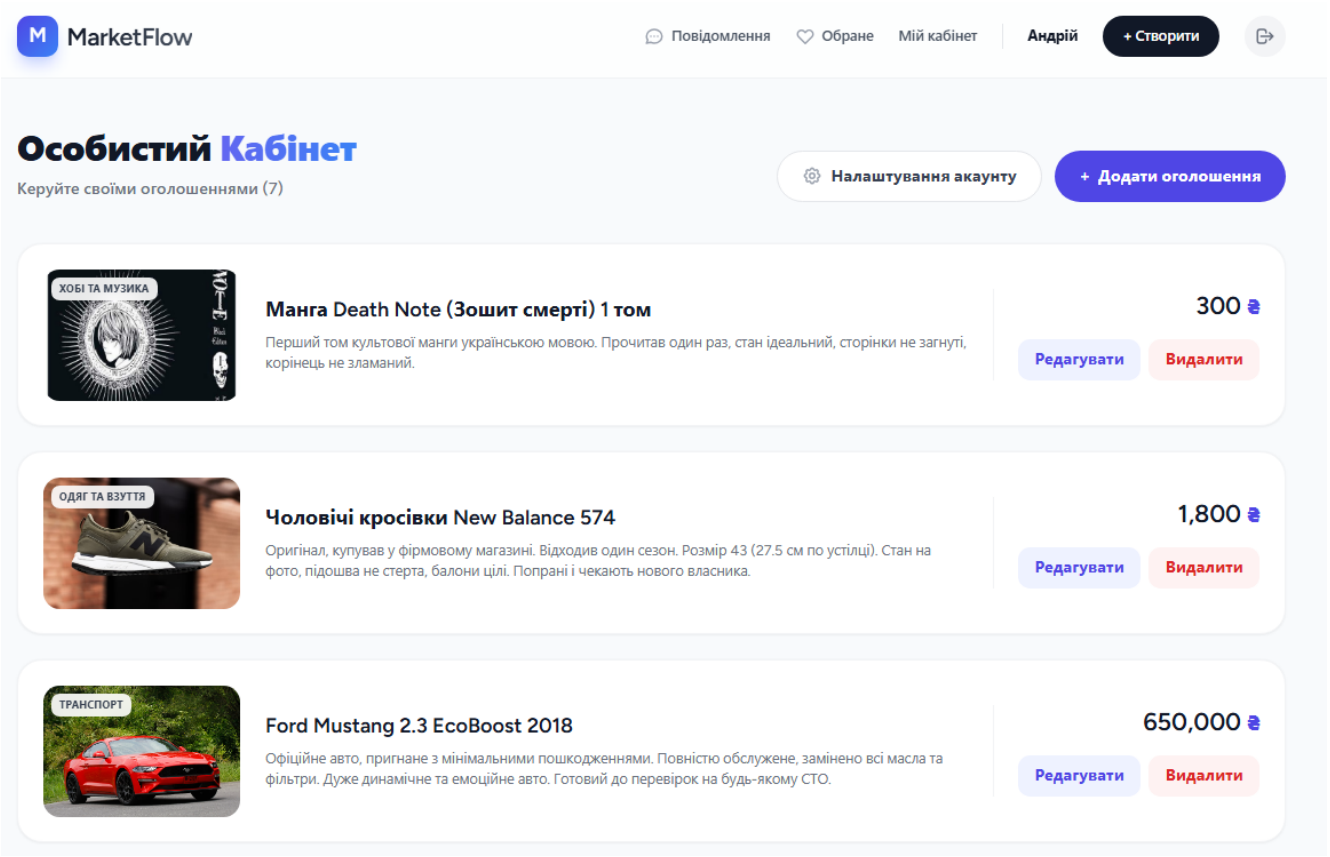


Рисунок 4.3 – Особистий кабінет оголошень

Сам процес створення нового оголошення був максимально спрощений, щоб мінімізувати час на розміщення товару. Користувачу пропонується заповнити форму зі стандартними полями: необхідно вказати заголовок, обрати відповідну категорію з випадаючого списку, встановити ціну та додати детальний опис. Окремим блоком реалізовано зону завантаження медіафайлів, де користувач може вибрати одну або кілька фотографій зі свого пристрою, просто перетягнувши їх у вікно. Система валідації в реальному часі підказує, які кроки залишилися незаповненими, забезпечуючи коректність даних. Тільки після заповнення всіх обов'язкових полів кнопка публікації стає активною, що запобігає появі порожніх або неінформативних карток у загальній стрічці.

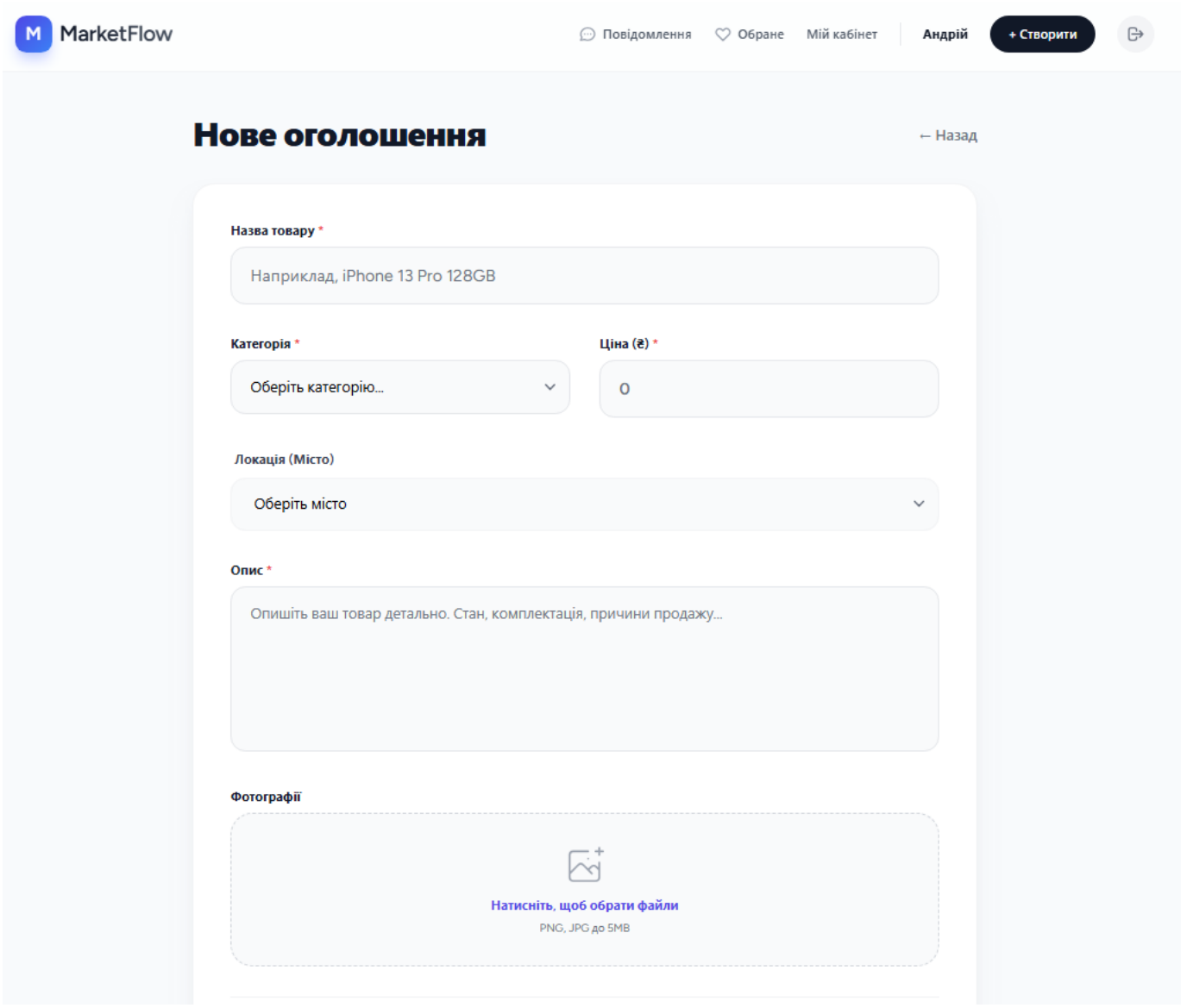


Рисунок 4.4 – Сторінка створення оголошення

Після натискання на картку товару відкривається сторінка з детальним описом. Інтерфейс стає зручним і зрозумілим: зліва – галерея завантажених фотографій, які легко перемикати, а справа – вся важлива текстова інформація: назва, ціна, розширений опис характеристик і блок даних про продавця. Тут розташована головна кнопка для швидкого зв'язку та початку переписки з власником оголошення.

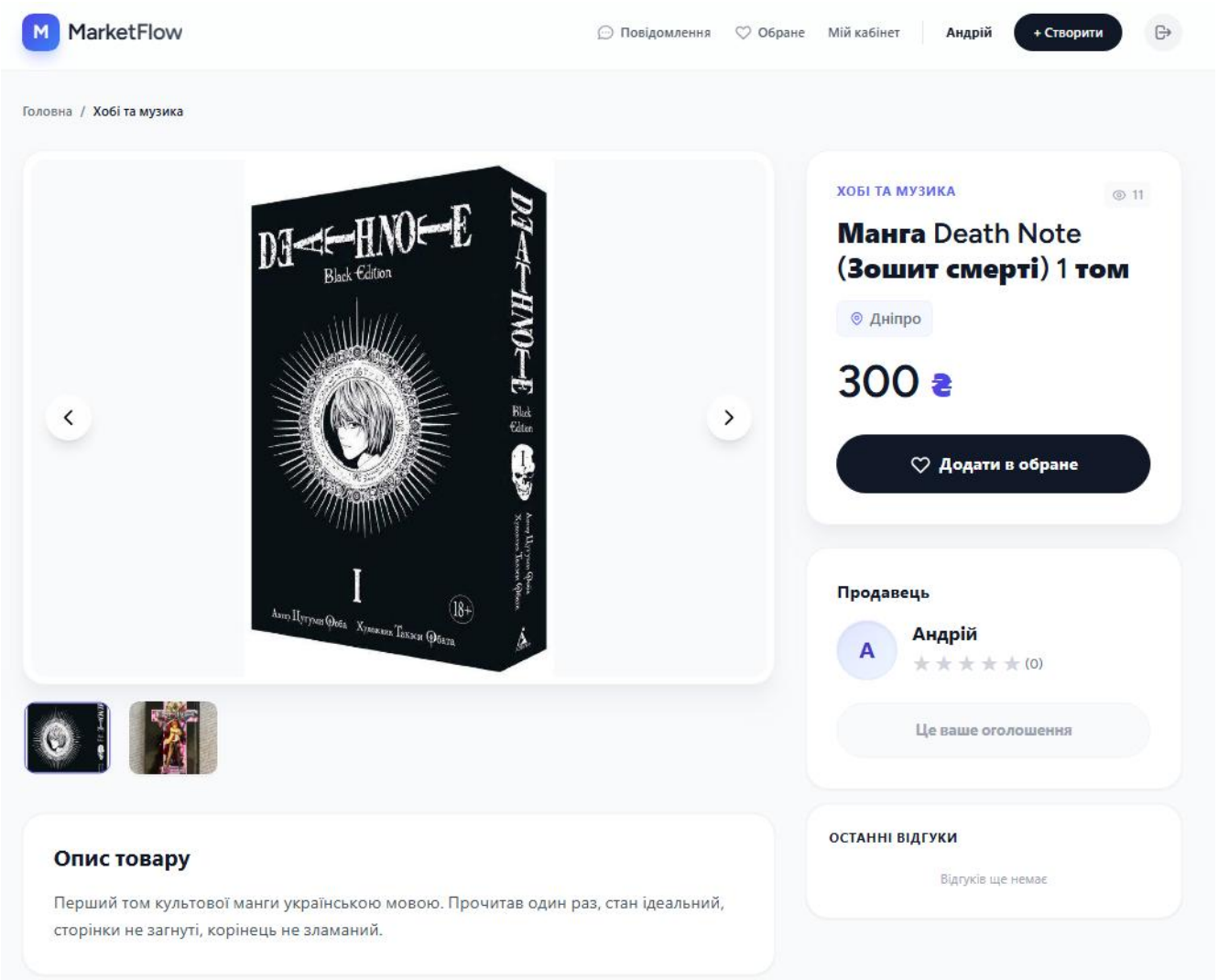


Рисунок 4.5 – Сторінка перегляду окремого оголошення

Окрему увагу треба приділити системі комунікації. Якщо покупець має запитання з приводу товару, він відкриває вбудований месенджер. Інтерфейс чатів побудовано зі звичним для сучасних соціальних мереж дизайном: відправлені повідомлення мають синій колір, а отримані – сірий. Повідомлення відправляються та відображаються у режимі реального часу.

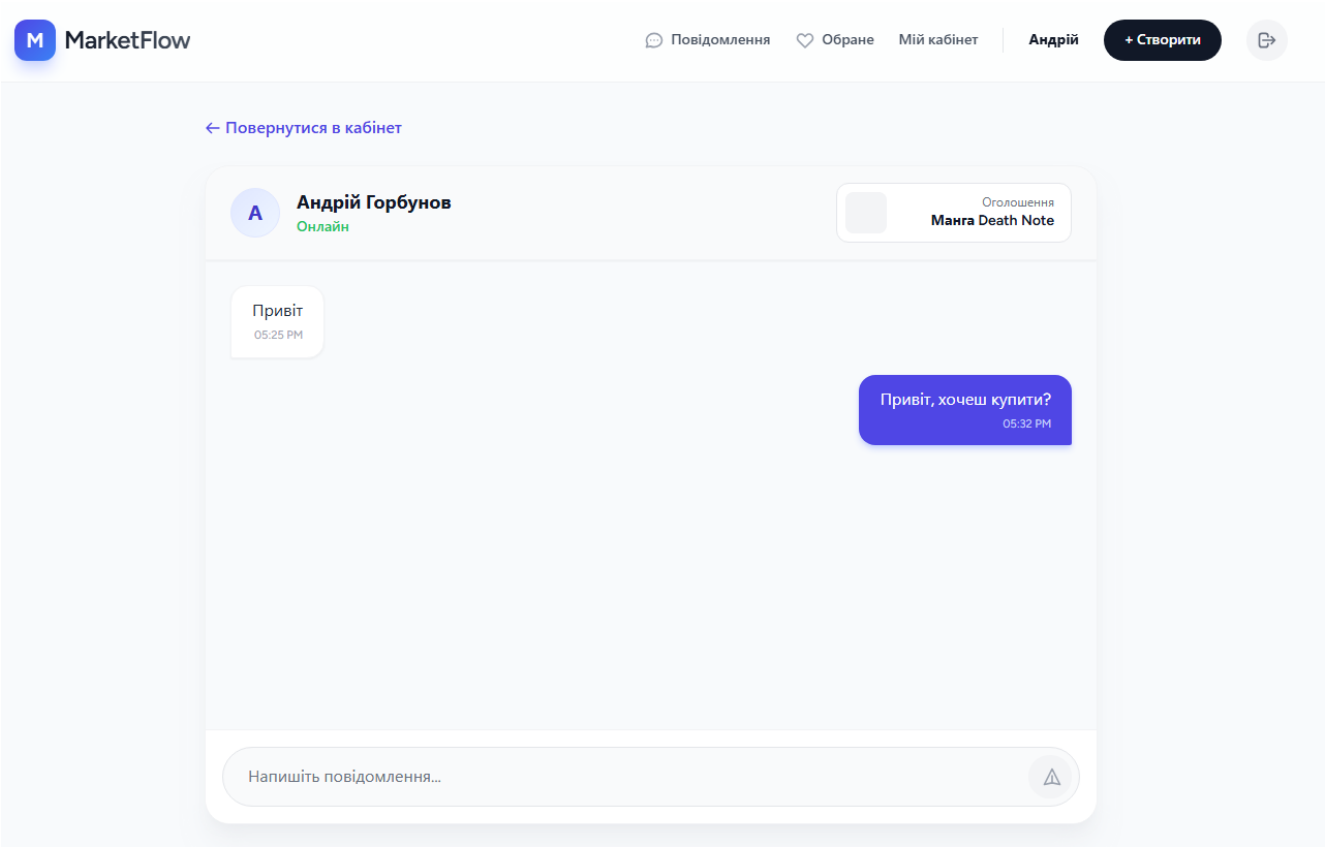


Рисунок 4.6 – Вигляд чату

Останнім етапом успішної угоди є система оцінювання продавця. Перейшовши у профіль, покупець має змогу залишити свій відгук. Для цього використовується спеціальна форма, де потрібно обрати оцінку від 1 до 5 зірочок та, за бажанням, написати текстовий коментар. Для забезпечення об'єктивності показників на рівні програмної логіки впроваджено захист від штучної накрутки, який унеможливлює оцінювання користувачем власних товарів. Після збереження відгуку система автоматично перераховує загальний рейтинг продавця, який одразу стає видимим для всіх інших учасників платформи. Такий прозорий механізм формування репутації підвищує загальний рівень довіри до маркетплейсу та допомагає новим клієнтам робити безпечний вибір.

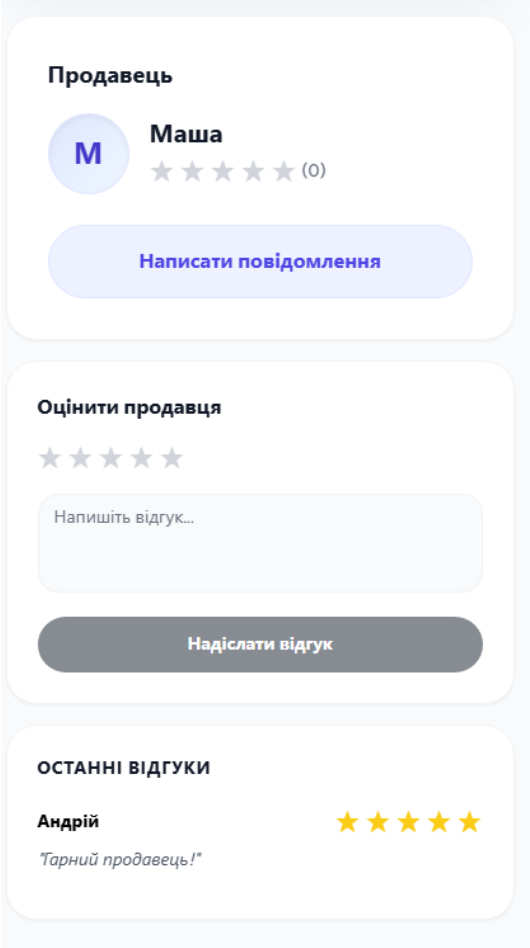


Рисунок 4.7 – Меню оцінювання продавця

Як демонструє наведений рисунок, інтерфейс форми є лаконічним і не перевантажує користувача зайвими елементами. Візуалізація оцінки через систему зірочок забезпечує швидке сприйняття інформації та робить процес залишення відгуку максимально зручним.

Висновки до розділу 4

У четвертому розділі був успішно завершений етап програмної реалізації та практичної верифікації розробленої платформи. Використання сучасного технологічного стеку дозволило перетворити теоретичні архітектурні моделі на стабільний програмний продукт. Опис ключових класів, об'єктів та методів підтвердив дотримання принципів чистого коду та раціональний розподіл обов’язків між клієнтською та серверною частинами застосунку.

Особлива увага була приділена перевірці працездатності системи. Шляхом проведення функціонального тестування за методом чорної скриньки було підтверджено коректність виконання всіх бізнес-сценаріїв: від реєстрації нового користувача до складних механізмів оцінювання та фільтрації оголошень. Проведені заміри швидкодії показали, що час відгуку системи при обробці запитів не перевищує 100 мілісекунд, що свідчить про високу продуктивність та готовність до масштабування.

На завершальному етапі було підготовлено детальне керівництво користувача. Наявність покрокових інструкцій та скріншотів реального інтерфейсу гарантує легкість освоєння платформи кінцевими споживачами. Таким чином, розроблений програмний продукт є повністю завершеним, технічно перевіреним та придатним для використання в реальних умовах ринку електронної комерції.

ВИСНОВКИ

У ході виконання кваліфікаційної бакалаврської роботи було успішно вирішено науково-практичне завдання з проєктування та розробки платформи електронної комерції MarketFlow. Проведений аналіз предметної області підтвердив актуальність створення C2C-маркетплейсів, які потребують високого рівня захисту даних, швидкодії та зручності користувацького інтерфейсу.

На етапі аналізу та проєктування було розроблено детальну специфікацію вимог і побудовано комплекс UML-моделей. Створення нормалізованої схеми бази даних та опис логічних алгоритмів дозволили забезпечити цілісність інформації та гнучкість системи. Використання сучасної архітектури дало змогу поєднати потужність серверної обробки Laravel із реактивністю інтерфейсу React.js, що є оптимальним рішенням для сучасних вебзастосунків.

Практична реалізація проєкту довела ефективність обраних підходів. Завдяки впровадженню систем кешування та оптимізації SQL-запитів було досягнуто високих показників продуктивності. Тестування системи підтвердило її стійкість до некоректних вхідних даних та надійність механізмів авторизації та валідації. Розроблена система рейтингу та відгуків стала ефективним інструментом забезпечення довіри між користувачами.

У підсумку, мета роботи була повністю досягнута. Створена платформа є повнофункціональним інструментом для здійснення онлайн-торгівлі, що відповідає всім сучасним технічним та ергономічним вимогам. Отримані результати можуть бути використані як база для розгортання реального комерційного сервісу або для подальших досліджень у сфері автоматизації процесів електронної комерції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Faisal M., Nuryana I. K. D. Comparative Analysis Of Laravel, Lumen, Guzzle, Leaf, and Slim Framework Performance On Rest API Using One Way Anova. Journal of Emerging Information Systems and Business Intelligence. 2025. Vol. 6 (2). pp. 77–83. DOI: https://doi.org/10.26740/jeisbi.v6i2.66146 (Accessed: 26.04.2026).

2. He J. Design and implementation of e-commerce system based on the Web. International Conference on Sustainable Computing and Data Communication Systems (ICSCDS). 2023. pp. 138–142. DOI: https://doi.org/10.1109/ICCSN.2011.6014390 (Accessed: 26.04.2026).

3. Inertia.js: The Modern Monolith. URL: https://inertiajs.com/ (Last accessed: 26.04.2026).

4. Jain V. A comparative analysis of single page applications (SPA) and multipage applications (MPA). International Journal of Core Engineering & Management. 2022. Vol. 7 (4). pp. 271–277 DOI: https://doi.org/10.5281/zenodo.14956673 (Accessed: 26.04.2026).

5. Jin Z. Research on Cross border E-commerce Distributed Architecture and Cloud Computing Collaborative Strategy. Distributed Processing System. 2025. Vol. 4 (1). pp. 17–26. DOI: https://doi.org/10.38007/DPS.2025.040103 (Accessed: 26.04.2026).

6. Kuflewski K., Dzienkowski M. A comparative analysis of PHP programming frameworks. Journal of Computer Sciences Institute. 2021. № 21. pp. 367–372DOI: https://doi.org/10.35784/jcsi.2749 (Accessed 26.04.26).

7. Kukreja V., Hemanth P., Dinesh P. Design and Implementation of E-commerce Recommendation System Model Based on Cloud Computing. 7th International Conference on Intelligent Computing and Control System. 2023. pp. 1406–1410. DOI: https://doi.org/10.1109/IPEC51340.2021.9421260 (Accessed: 26.04.2026).

8. Laravel Documentation. The PHP Framework for Web Artisans. URL: https://laravel.com/docs (Accessed: 26.04.2026).

9. Microsoft SQL Server Documentation. URL: https://learn.microsoft.com/en-us/sql/sql-server/ (Accessed: 26.04.2026).

10. NGINX Documentation. High Performance Load Balancer, Web Server, & Reverse Proxy. URL: https://nginx.org/en/docs/ (Accessed: 26.04.2026).

11. Object Management Group (OMG). Unified Modeling Language (UML) Specification. URL: https://www.omg.org/spec/UML/ (Accessed: 26.04.2026).

12. Olta Z., Morina B., Haskuka E. Optimizing Data Retrieval and Update Operations in PHP-MySQL Applications: A Review. Journal of Modern Technology. 2025. Vol. 2 (2). pp. 312–316. DOI: https://doi.org/10.71426/jmt.v2.i2.pp312-316 (Accessed 26.04.2026)[AK1] .

13. React Documentation. A JavaScript library for building user interfaces. URL: https://react.dev/ (Accessed: 26.04.2026).

14. Redis Documentation. The open source, in-memory data store. URL: https://redis.io/docs/ (Accessed: 26.04.2026).

15. Tailwind CSS Documentation. Rapidly build modern websites without ever leaving your HTML. URL: https://tailwindcss.com/docs (Accessed: 26.04.2026).

16. Zulfikri A., Rohman A. Analysis Comparative of Performance Optimization Techniques for PHP Framework Testing: Laravel, CodeIgniter, Symfony. Brilliance: Research of Artificial Intelligence. 2025. Vol. 5 (1). pp. 242–248. DOI: https://doi.org/10.47709/brilliance.v5i1.5989 (Accessed: 26.04.2026).

ДОДАТОК А

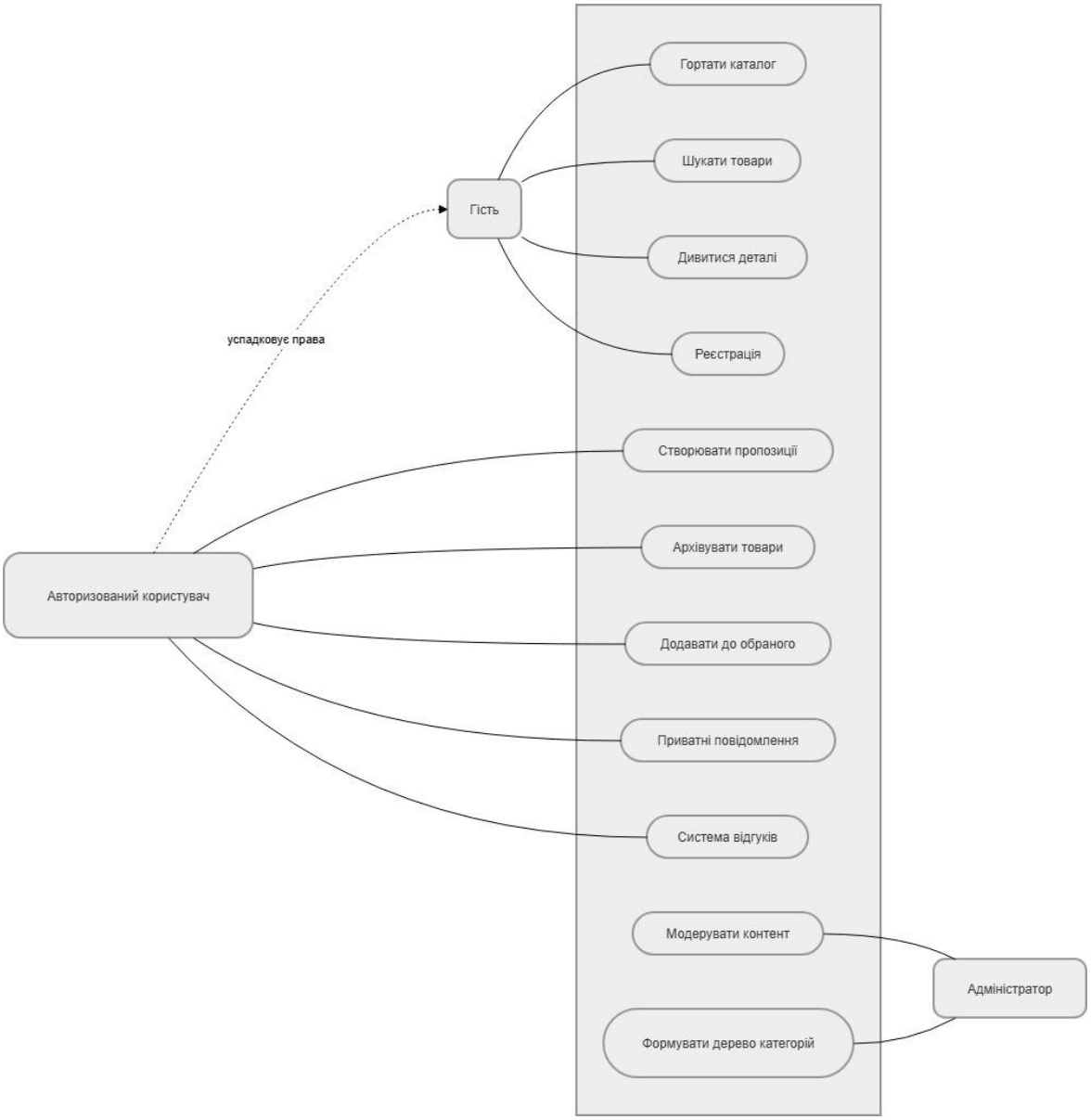


Рисунок А.1 – Діаграма використання

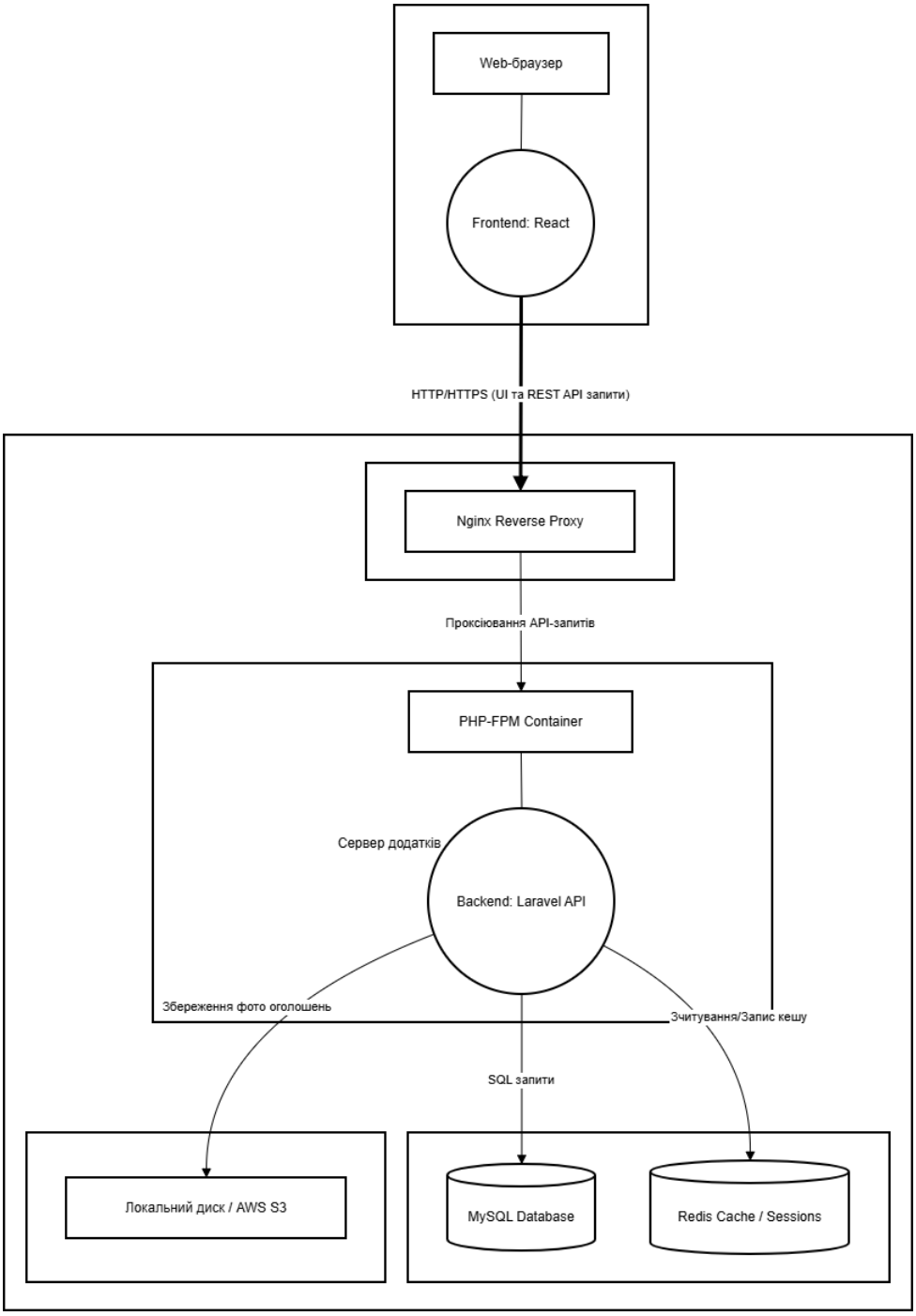
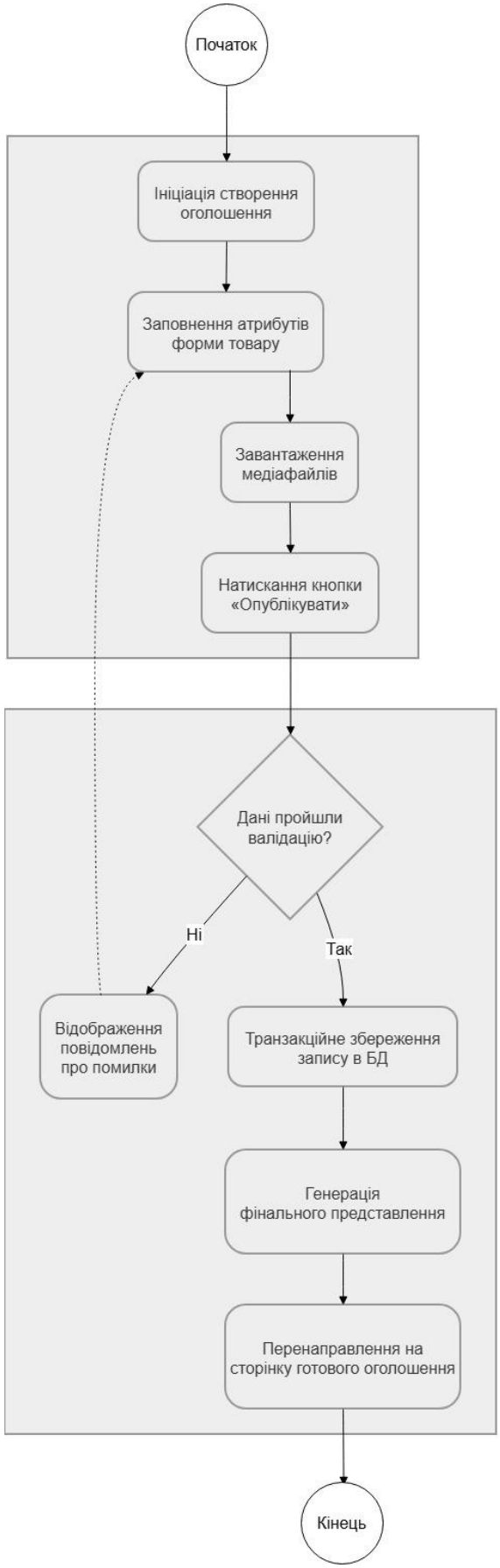


Рисунок А.2 – Діаграма розгортання



Додаток А.3 – Діаграма діяльності