МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ МАРКЕТПЛЕЙСУ ДЛЯ РОЗМІЩЕННЯ ТА ПРОДАЖУ ТОВАРІВ КОРИСТУВАЧАМИ»

Виконав: здобувач освіти 4 курсу, групи К121-22
Спеціальність 121 Інженерія
програмного забезпечення
(шифр і спеціальність)
Олег ІЩЕНКО
(ім'я та ПРІЗВИЩЕ)
Керівник Юлія БОРИСОВСЬКА
(ім'я та ПРІЗВИЩЕ)
Рецензент доцент каф. програмної інженерії ЗНУ, к.т.н., доцент Юлія ЛИМАРЕНКО
(посада, вчене звання, науковий ступінь, ім'я та ПРІЗВИЩЕ)

Запоріжжя
2026

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо–кваліфікаційний ступінь фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Т.М. Смолянкова
(підпис)

“ 05 ” червня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ІЩЕНКО Олегу Юрійовичу
(ПРІЗВИЩЕ, ім’я та по– батькові)

1. Тема роботи «Проєктування та розробка вебзастосунку маркетплейсу для розміщення та продажу товарів користувачами»

Керівник роботи БОРИСОВСЬКА Юлія Олександрівна
(науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

Затверджені наказом ВСП ЕПФК ЗНУ від «01» грудня 2025 р. № 1728-с

2. Строк подання студентом роботи 05.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково– пояснювальної записки (перелік питань, які потрібно розробити)
1.Сучасні інформаційні системи
2. Розробка проєкту інформаційної системи
3. Програмна реалізація інформаційної системи

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація до захисту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01.12.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	постановка задачі	Грудень 2025	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Січень 2026	виконано
3.	Розробка першого розділу:		
	огляд інформаційних систем та технологій для їхнього створення	Лютий 2026	виконано
4.	Розробка другого розділу:		
	розробка проєкту інформаційної системи та вибір технологій	Квітень 2026	виконано
5.	Розробка третього розділу:		
	розробка застосунку	Травень 2026	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2026	виконано
7.	Захист кваліфікаційної роботи	16.06.2026	виконано

Здобувач освіти _____ Олег ІЩЕНКО
(підпис) (ім’я ПРІЗВИЩЕ)

Керівник роботи _____ Юлія БОРИСОВСЬКА
(підпис) (ім’я ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ Юлія БОРИСОВСЬКА
(підпис) (ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 49 сторінок, 16 рисунків, 1 таблиця, 12 джерел.

Об'єкт дослідження: процес проєктування та розробки вебзастосунків для підтримки електронної комерційної діяльності.

Предмет дослідження: методи та інструменти проєктування веборієнтованих систем на основі архітектурного шаблону MVC та сучасних PHP-фреймворків для розробки платформ електронної комерції типу «маркетплейс»..

Мета роботи: проєктування та реалізація вебзастосунку маркетплейсу, що надає користувачам можливість розміщувати власні товари, здійснювати їх пошук та оформлювати замовлення в єдиному цифровому середовищі.

Метод дослідження: системний та порівняльний аналіз ринкових аналогів, методи об'єктно-орієнтованого проєктування, моделювання реляційних баз даних, а також експериментальне тестування програмного забезпечення..

У кваліфікаційній роботі проведено детальний аналіз предметної області маркетплейсів, вивчено існуючі аналоги – OLX, Prom.ua та Rozetka, виявлено їхні переваги та недоліки. На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до розроблюваної системи.

У процесі роботи спроєктовано архітектуру застосунку за принципами патерну MVC, розроблено структуру реляційної бази даних MySQL, що охоплює сутності користувачів, категорій, товарів, замовлень та їхніх позицій. Обґрунтовано вибір технологічного стеку: мова програмування PHP 8.2, фреймворк Laravel 10, шаблонізатор Blade, система управління базами даних MySQL 8.0.

Розроблено та реалізовано повнофункціональний вебзастосунок маркетплейсу з модулями реєстрації та автентифікації користувачів, управління товарами, пошуку та фільтрації, оформлення і відстеження

замовлень, а також особистого кабінету. Проведено функціональне тестування розробленого застосунку, виявлені недоліки усунено.

Результатом виконаної роботи є готовий до розгортання вебзастосунок маркетплейсу, що демонструє практичне застосування сучасних вебтехнологій і може бути використаний як самостійна торговельна платформа або слугувати основою для подальшого масштабування.

PHP, LARAVEL, BLADE, MVC, MYSQL, МАРКЕТПЛЕЙС, ЕЛЕКТРОННА КОМЕРЦІЯ, ELOQUENT ORM, BOOTSTRAP

SUMMARY

Qualification Work: 49 pages, 16 figures, 1 table, 12 references.

Object of Study: The process of designing and developing web applications to support electronic commerce activities.

Subject of Study: A marketplace web application for listing and selling goods by users.

Purpose of the Work: To design and implement a marketplace web application enabling users to list their own products, search for goods, and place orders within a unified digital environment.

Research Methods: Analytical, structural, comparative analysis, experimental.

The qualification work presents a detailed analysis of the marketplace subject area, examining existing analogues – OLX, Prom.ua, and Rozetka – with identification of their advantages and shortcomings. Based on this analysis, functional and non-functional requirements for the system under development have been formulated.

The application architecture was designed according to the MVC pattern, and the structure of a MySQL relational database was developed, covering the entities of users, categories, products, orders, and order items. The choice of technology stack is justified: PHP 8.2 programming language, Laravel 10 framework, Blade templating engine, MySQL 8.0 database management system.

A fully functional marketplace web application was developed and implemented, comprising modules for user registration and authentication, product management, search and filtering, order placement and tracking, and a personal account area. Functional testing of the developed application was carried out and identified deficiencies were corrected.

The result of the completed work is a deployment-ready marketplace web application demonstrating the practical application of modern web technologies. It

can be used as a standalone commercial platform or serve as a foundation for further scaling.

PHP, LARAVEL, BLADE, MVC, MYSQL, MARKETPLACE, E-COMMERCE, ELOQUENT ORM, BOOTSTRAP

ЗМІСТ

Завдання на кваліфікаційну роботу студенту 2
Реферат 4
Summary 6
Вступ 9
1 Огляд предметної області 11
1.1 Поняття та сутність маркетплейсів 11
1.2 Аналіз існуючих аналогів 12
1.3 Функціональні та нефункціональні вимоги до системи 15
1.4 Висновок до розділу 1 17
2 Проєктування інформаційної системи 18
2.1 Постановка задачі та загальна архітектура системи 18
2.2 Загальна архітектура системи 18
2.3 Вибір технологічного стеку 20
2.4 Моделювання системи 21
2.5 Проєктування бази даних 24
2.6 Опис API та взаємодії компонентів 25
3 Програмна реалізація 28
3.1 Реалізація серверної частини 28
3.2 Реалізація клієнтської частини 29
3.3 Реалізація бази даних 33
3.4 Реалізація основного функціоналу 35
3.5 Тестування системи 42
3.6 Розгортання вебзастосунку 44
Висновки 46
Перелік використаних джерел 48

ВСТУП

Стрімкий розвиток інформаційних технологій та глобальне поширення мережі Інтернет кардинально трансформували підходи до ведення комерційної діяльності. Електронна комерція перетворилась на невід'ємну складову сучасної економіки, надаючи мільйонам людей можливість здійснювати покупки й продажі у зручний час та з будь-якої точки світу. Серед різноманіття інструментів е-комерції особливе місце займають маркетплейси – онлайн-платформи, що об'єднують продавців і покупців на єдиному майданчику, забезпечуючи зручний пошук товарів, безпечну взаємодію сторін та ефективне управління угодами.

За даними аналітичних досліджень, глобальний ринок маркетплейсів демонструє стабільне зростання. В Україні активно функціонують великі торговельні платформи, зокрема OLX, Prom.ua та Rozetka, якими щомісяця користуються мільйони відвідувачів. Водночас потреба у спеціалізованих, гнучких і масштабованих рішеннях залишається надзвичайно актуальною – зокрема для нішевих ринків і регіональних платформ, де великі гравці не завжди здатні задовольнити специфічні запити аудиторії.

Актуальність теми даної кваліфікаційної роботи обумовлена потребою у розробці сучасного, масштабованого та зручного у використанні вебзастосунку маркетплейсу, що надасть користувачам повноцінний інструментарій для розміщення, пошуку та придбання товарів. Вибір технологічного стеку на основі мови програмування PHP та фреймворку Laravel зумовлений їх широкими функціональними можливостями, зрілістю екосистеми, а також великою спільнотою розробників і багатством відкритих бібліотек.

Метою даної кваліфікаційної роботи є проєктування та розробка вебзастосунку маркетплейсу типу C2C для розміщення та продажу товарів користувачами з використанням фреймворку Laravel, мови програмування PHP та шаблонізатора Blade.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати предметну область та дослідити існуючі рішення у сфері онлайн-маркетплейсів;

– визначити функціональні та нефункціональні вимоги до системи;

– спроєктувати загальну архітектуру системи та структуру реляційної бази даних;

– обрати та обґрунтувати технологічний стек для реалізації проєкту;

– розробити та реалізувати вебзастосунок маркетплейсу;

– провести функціональне тестування розробленого застосунку та усунути виявлені недоліки.

Об'єктом дослідження є процес проєктування та розробки вебзастосунків для підтримки електронної комерційної діяльності.

Предметом дослідження є вебзастосунок маркетплейсу для розміщення та продажу товарів.

Методами дослідження є аналітичний метод – для вивчення предметної області та існуючих рішень; структурний метод – при проєктуванні архітектури системи та схеми бази даних; порівняльний аналіз – при виборі технологічного стеку; а також експериментальний метод – при тестуванні розробленого застосунку.

Практичне значення результатів полягає у розробці повнофункціонального вебзастосунку, готового до розгортання на реальній серверній інфраструктурі. Застосунок може використовуватися як самостійна торговельна платформа або слугувати технічною основою для подальшого розвитку і масштабування.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та переліку використаних джерел. Перший розділ присвячено огляду предметної області маркетплейсів та аналізу існуючих рішень. У другому розділі розглядаються питання проєктування системи, вибору технологій та моделювання бази даних. Третій розділ описує практичну реалізацію застосунку, а також результати тестування та процес розгортання.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та сутність маркетплейсів

Маркетплейс – це цифрова платформа, що забезпечує взаємодію між продавцями та покупцями, надаючи їм єдине середовище для розміщення, пошуку та придбання товарів або послуг. На відміну від традиційного інтернет-магазину, який реалізує товари власного асортименту, маркетплейс виступає посередником: він надає технічну та організаційну інфраструктуру, тоді як самі товари пропонують незалежні продавці [1].

Серед основних функцій маркетплейсу можна виділити [1]:

– реєстрація та автентифікація користувачів у різних ролях (продавець, покупець, адміністратор);

– надання продавцям інструментів для публікації та управління товарними оголошеннями;

– забезпечення покупцям можливостей для пошуку, фільтрації та перегляду товарів;

– підтримка процесу оформлення і відстеження замовлень; управління транзакціями та комунікацією між сторонами.

Залежно від моделі взаємодії учасників маркетплейси класифікують за такими типами [1]:

– B2C (Business-to-Consumer) – взаємодія між бізнесом і кінцевим споживачем. Продавцями виступають юридичні особи або підприємці, які реалізують товари фізичним особам. Типовим представником є Rozetka.

– C2C (Consumer-to-Consumer) – взаємодія між фізичними особами. Будь-який зареєстрований користувач може одночасно виступати як продавцем, так і покупцем. Найбільш відомий приклад – OLX.

– B2B (Business-to-Business) – платформи, орієнтовані на взаємодію між підприємствами. Використовуються для оптових закупівель, реалізації виробничих товарів або надання корпоративних послуг.

За способом розповсюдження товарів маркетплейси поділяють на горизонтальні (продають товари різних категорій, охоплюючи широкий ринок) та вертикальні (спеціалізовані платформи, що зосереджені на конкретній ніші, наприклад, одяг, електроніка або будматеріали) [1].

Для ефективного функціонування маркетплейс повинен вирішувати кілька ключових задач [1]:

– забезпечити довіру між незнайомими сторонами (через відгуки, рейтинги, гарантії повернення);

– надати зручний і швидкий пошук серед великої кількості товарів;

– автоматизувати процес обробки замовлень та управління логістикою; гарантувати безпеку персональних та фінансових даних користувачів.

1.2 Аналіз існуючих аналогів

Для формування вимог до розроблюваного застосунку було проведено порівняльний аналіз трьох провідних українських маркетплейсів: OLX, Prom.ua та Rozetka. Дослідження охоплювало функціональні можливості платформ, моделі монетизації, підходи до управління каталогом та рівень зручності користувацького інтерфейсу.

OLX є найбільшою платформою безкоштовних оголошень в Україні, що функціонує за моделлю C2C (рис. 1.1). Будь-який зареєстрований користувач може безкоштовно розмістити оголошення про продаж товарів або послуг [2]. Платформа пропонує розгалужену категоризацію, вбудований чат між продавцем і покупцем, а також функцію «OLX Доставка», що дозволяє здійснювати безпечні угоди з оплатою та доставкою через партнерські служби.

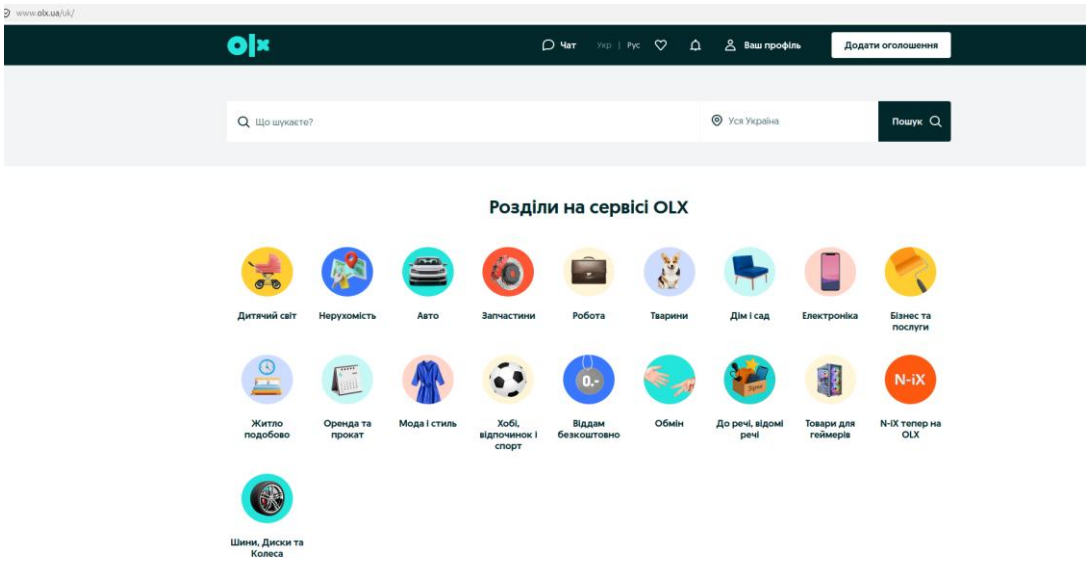


Рисунок 1.1 – Головна сторінка маркетплейсу OLX

Prom.ua є торговельною платформою, орієнтованою переважно на модель B2C, де продавцями виступають зареєстровані підприємці та юридичні особи (рис. 1.2). Платформа надає інструменти для створення власного інтернет-магазину в рамках маркетплейсу, управління каталогом товарів, налаштування умов доставки та обробки замовлень [3]. Монетизація здійснюється через платні тарифні плани для продавців.

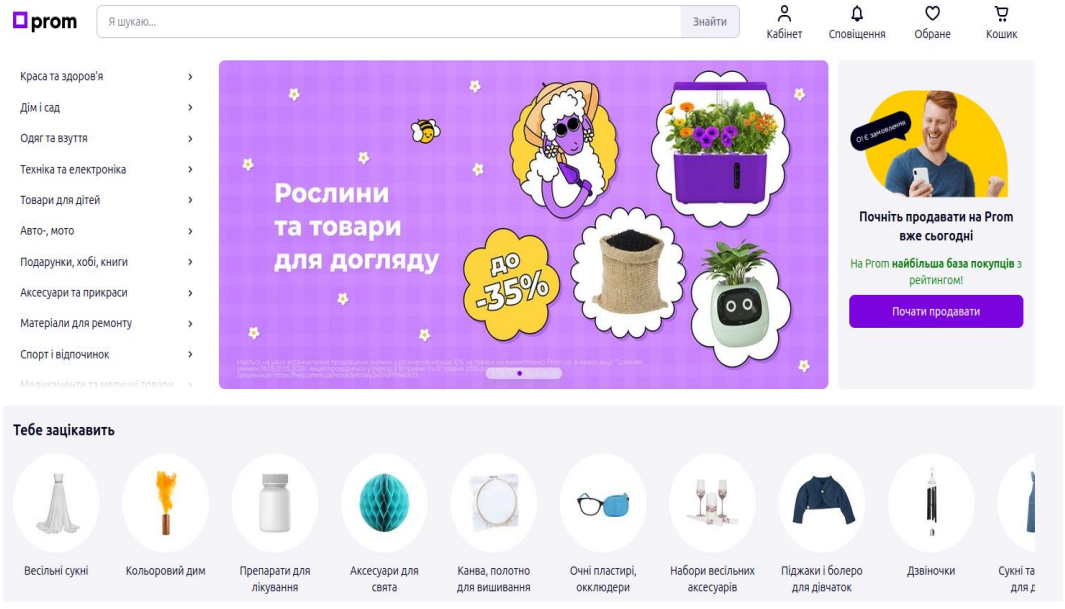


Рисунок 1.2 – Головна сторінка маркетплейсу Prom.ua

Rozetka є найбільшим українським інтернет-ритейлером, що поєднує власний асортимент товарів з маркетплейс-функціоналом (рис. 1.3). Платформа орієнтована на B2C-модель і пропонує широкий вибір товарів переважно у категоріях електроніки, побутової техніки та інших виробів масового попиту [4]. Інтеграція маркетплейсу дозволяє зовнішнім продавцям розміщувати товари поряд із власним асортиментом Rozetka.

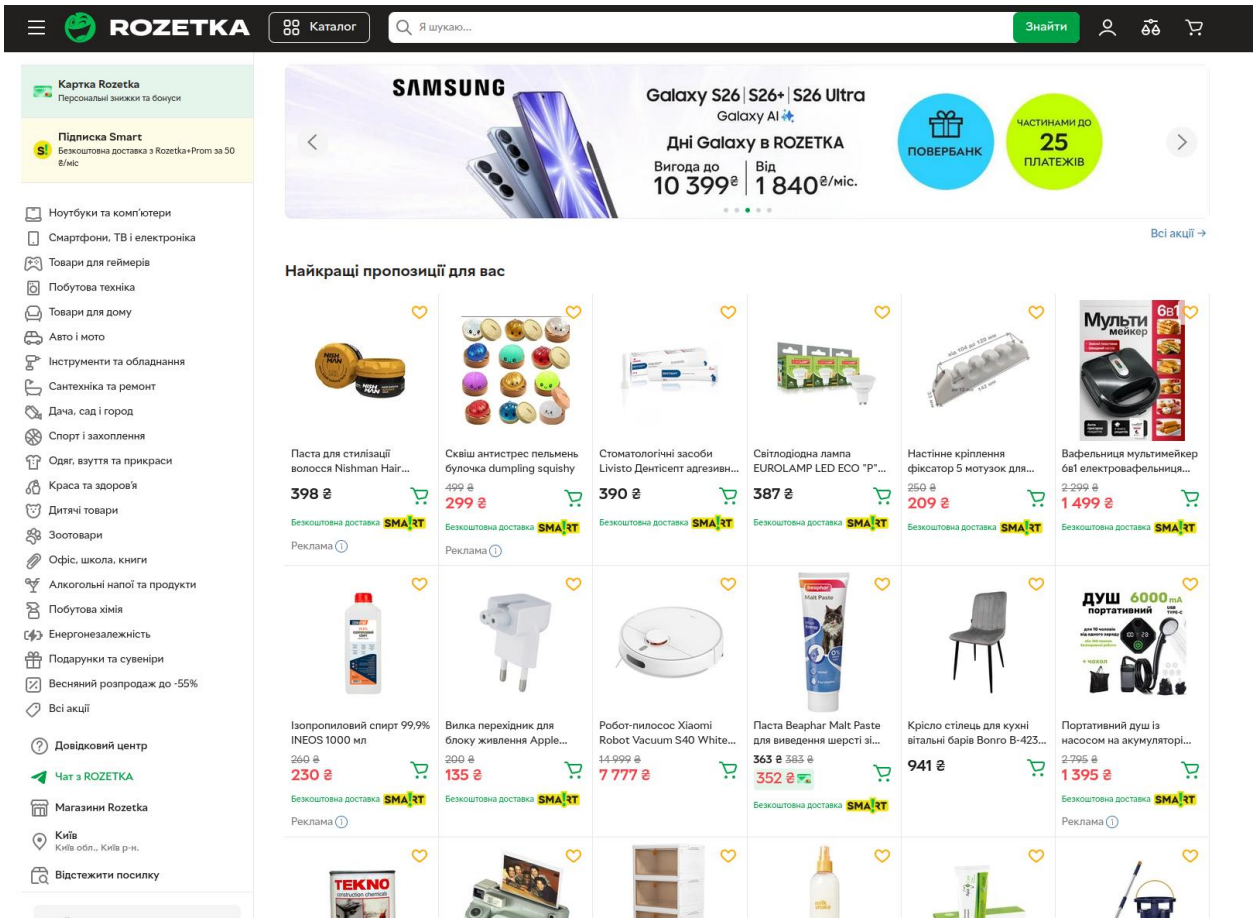


Рисунок 1.3 – Головна сторінка маркетплейсу Rozetka

На основі проведеного аналізу було виявлено ключові недоліки, що визначили вектор розробки власного застосунку: відсутність у жодному з аналогів одночасно простого інтерфейсу, підтримки змішаної C2C/B2C – моделі та відкритого вихідного коду, придатного для кастомізації. Розроблюваний в межах кваліфікаційної роботи вебсервіс маркетплейсу спрямований на забезпечення гнучкого, зручного й масштабованого рішення, що поєднує переваги аналізованих платформ.

1.3 Функціональні та нефункціональні вимоги до системи

На підставі аналізу предметної області та порівняльного дослідження функціональних можливостей існуючих аналогів було сформовано перелік вимог до розроблюваної системи. Вимоги поділяються на дві групи: функціональні, що описують конкретні можливості застосунку, та нефункціональні, що регламентують якісні характеристики системи [5].

Функціональні вимоги визначають поведінку системи з точки зору виконуваних операцій. Нижче розглянемо основі функціональні вимоги до розроблюваної системи.

Модуль автентифікації та авторизації забезпечує керування обліковими записами користувачів та контроль доступу до функціональних можливостей системи:

– реєстрація нових користувачів із зазначенням імені, електронної пошти та пароля;

– вхід до системи за допомогою облікових даних (e-mail та пароль);

– захищений вихід із системи та інвалідація сесії;

– розмежування ролей користувачів: користувач, адміністратор;

– можливість оновлення персональних даних профілю.

Модуль управління товарами відповідає за створення, редагування та керування інформацією про товари користувачів:

– додавання нового товару з зазначенням назви, опису, ціни, кількості та категорії;

– завантаження зображень товару;

– редагування та видалення власних товарів;

– зміна статусу товару (активний, неактивний, продано);

– перегляд власного переліку товарів у особистому кабінеті.

Модуль каталогу та пошуку забезпечує перегляд товарів та зручний пошук необхідної інформації:

– перегляд каталогу товарів з розбиттям на сторінки;

– фільтрація товарів за категорією, ціновим діапазоном та статусом;

– пошук товарів за ключовим словом;

– сортування результатів за ціною, датою додавання та популярністю;

– перегляд детальної сторінки товару з інформацією про продавця.

Модуль замовлень реалізує функціональність оформлення та перегляду покупок користувача:

– оформлення замовлення на обраний товар із зазначенням адреси доставки;

– перегляд історії замовлень в особистому кабінеті.

Нефункціональні вимоги визначають якісні характеристики та обмеження системи.

Вимоги до безпеки визначають механізми захисту даних користувачів та цілісності системи:

– паролі користувачів зберігаються виключно у хешованому вигляді з використанням алгоритму bcrypt;

– усі вхідні дані від користувачів підлягають валідації та санітизації;

– захист від атак типу CSRF (Cross-Site Request Forgery) та SQL-ін'єкцій забезпечується стандартними механізмами фреймворку Laravel;

– сесії захищені із застосуванням HTTP-only cookie.

Вимоги до масштабованості описують здатність системи ефективно розширюватися при зростанні навантаження та обсягу даних:

– модульна архітектура на основі патерну MVC дозволяє незалежно масштабувати окремі компоненти системи;

– структура бази даних розроблена з урахуванням можливості горизонтального масштабування;

– застосунок підтримує розгортання у контейнеризованому середовищі Docker.

Вимоги до зручності використання (UX/UI) визначають принципи побудови інтерфейсу для забезпечення комфортної взаємодії користувача з системою:

– інтерфейс розробляється відповідно до принципів адаптивного дизайну

– коректно відображається на пристроях з різними роздільними здатностями екрану

1.4 Висновок до розділу 1

У першому розділі кваліфікаційної роботи проведено детальний огляд предметної галузі маркетплейсів. Розкрито поняття маркетплейсу, описано його ключові функції та наведено класифікацію платформ за моделями взаємодії учасників (B2C, C2C, B2B) та типом охопленого ринку (горизонтальні та вертикальні).

Порівняльний аналіз трьох провідних українських платформ – OLX, Prom.ua та Rozetka – дозволив виявити їхні сильні сторони та недоліки. Встановлено, що кожна з платформ має власну нішу та обмеження: OLX зручний для C2C-взаємодії, але не пристосований для бізнес-продавців; Prom.ua функціональний для підприємців, але складний для початківців; Rozetka пропонує найвищий рівень гарантій, але має жорсткі умови для партнерів.

На підставі аналізу потреб цільової аудиторії та виявлених прогалин існуючих рішень сформульовано функціональні та нефункціональні вимоги до розроблюваного маркетплейсу. Функціональні вимоги охоплюють чотири ключові модулі: автентифікацію та авторизацію, управління товарами, каталог і пошук, а також замовлення. Нефункціональні вимоги регламентують безпеку, продуктивність, масштабованість та зручність використання системи. Сформовані вимоги слугуватимуть основою для подальшого проєктування та реалізації застосунку.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Постановка задачі та загальна архітектура системи

У межах даної кваліфікаційної роботи передбачається розробка повнофункціонального вебзастосунку маркетплейсу, що дозволяє зареєстрованим користувачам розміщувати власні товари на продаж, а іншим користувачам – переглядати каталог, здійснювати пошук та оформлювати замовлення. Система повинна підтримувати змішану C2C/B2C-модель взаємодії і надавати адміністратору інструменти для управління контентом платформи.

Конкретними результатами, що планується отримати в ході розробки, є:

– веброботок на основі PHP/Laravel з серверним рендерингом сторінок через Blade-шаблони;

– реляційна база даних MySQL зі схемою, що відображає предметну область маркетплейсу;

– система автентифікації та розмежування прав доступу;

– функціонал управління товарами, включаючи завантаження зображень;

– каталог товарів з пошуком, фільтрацією та пагінацією;

– модуль оформлення замовлень;

– особистий кабінет користувача.

2.2 Загальна архітектура системи

Основним архітектурним принципом розробки обрано патерн MVC (Model-View-Controller), що є нативним підходом фреймворку Laravel і

забезпечує чітке розділення відповідальностей між компонентами системи. Ця архітектура передбачає три рівні [6]:

Model (Модель) – відповідає за роботу з даними та бізнес-логіку. У Laravel ця роль покладена на класи Eloquent ORM, що забезпечують об'єктно-реляційне відображення таблиць бази даних. Кожна сутність предметної галузі (користувач, товар, замовлення тощо) представлена окремим класом моделі, що інкапсулює логіку взаємодії з відповідною таблицею [6].

View (Представлення) – відповідає за формування та відображення інтерфейсу користувача. У реалізованому застосунку ця роль виконується Blade-шаблонами – вбудованим механізмом шаблонізації Laravel, що дозволяє впроваджувати PHP-логіку безпосередньо у HTML-розмітку через зрозумілі директиви. Blade підтримує успадкування шаблонів, часткові представлення (partials) та компоненти, що значно спрощує підтримку єдиного стилю інтерфейсу [6].

Controller (Контролер) – виступає посередником між моделлю та представленням. Контролери Laravel обробляють HTTP-запити, викликають відповідні методи моделей, формують набір даних і передають їх у Blade-шаблони для відображення [6].

Застосунок реалізується за клієнт-серверною архітектурою. Вебсервер (Nginx) приймає HTTP-запити від браузера користувача та передає їх до PHP-процесу, де обробляються маршрути Laravel. Сервер повертає готову HTML-сторінку, сформовану Blade-шаблонами. Такий підхід серверного рендерингу (SSR – Server-Side Rendering) забезпечує швидкий початковий час відображення сторінки, коректну індексацію пошуковими системами та спрощену архітектуру без потреби у окремому фронтенд-фреймворку.

2.3 Вибір технологічного стеку

Вибір технологій для реалізації проєкту здійснювався на основі таких критеріїв: відповідність функціональним вимогам системи, зрілість і стабільність технології, доступність документації та спільноти підтримки, продуктивність рішення та перспективи масштабування. На підставі цих критеріїв сформовано технологічний стек, описаний нижче.

PHP є сучасною серверною мовою програмування загального призначення, яка протягом десятиліть залишається провідним інструментом для веброзробки [7].

Laravel – це відкритий PHP-фреймворк загального призначення, що реалізує патерн MVC і надає розробникам розгалужену екосистему готових компонентів для вирішення типових задач веброзробки [8].

Blade – власний шаблонізатор Laravel, що компілюється у стандартний PHP-код і кешується для досягнення максимальної продуктивності. Blade надає зручний синтаксис директив для керування логікою відображення [8].

MySQL 8.0 є провідною відкритою реляційною системою управління базами даних, яка забезпечує надійне зберігання структурованих даних, підтримку транзакцій (ACID), розгалужений набір типів даних, а також потужні механізми індексування та оптимізації запитів [9].

Composer є стандартним менеджером залежностей для PHP, що дозволяє оголошувати, встановлювати та автоматично завантажувати бібліотеки, необхідні для роботи застосунку.

Docker використовується для контейнеризації застосунку при розгортанні на виробничому сервері. Контейнеризація забезпечує ізольованість та відтворюваність середовища виконання незалежно від конфігурації хостової системи [10].

2.4 Моделювання системи

2.4.1 Діаграма прецедентів

Для формалізації вимог до вебзастосунку та візуалізації логіки його роботи було використано мову графічного опису UML (Unified Modeling Language).

Основним інструментом моделювання функціональних вимог є діаграма прецедентів [11]. Вона відображає ролі користувачів (акторів) та перелік дій, які вони можуть виконувати в межах маркетплейсу. Відповідно до функціональних вимог, визначених у розділі 1.3, діаграма прецедентів (рис. 2.1) охоплює такі групи операцій:

Управління профілем: реєстрація, автентифікація та редагування особистих даних.

Взаємодія з товарами: додавання власних позицій, завантаження зображень (що є обов'язковим прецедентом включення), редагування характеристик та зміна статусів наявності.

Навігація та пошук: перегляд загального каталогу, застосування фільтрів за категоріями та ціною, пошук за ключовими словами.

Процес замовлення: оформлення купівлі обраного товару та перегляд історії транзакцій в особистому кабінеті.

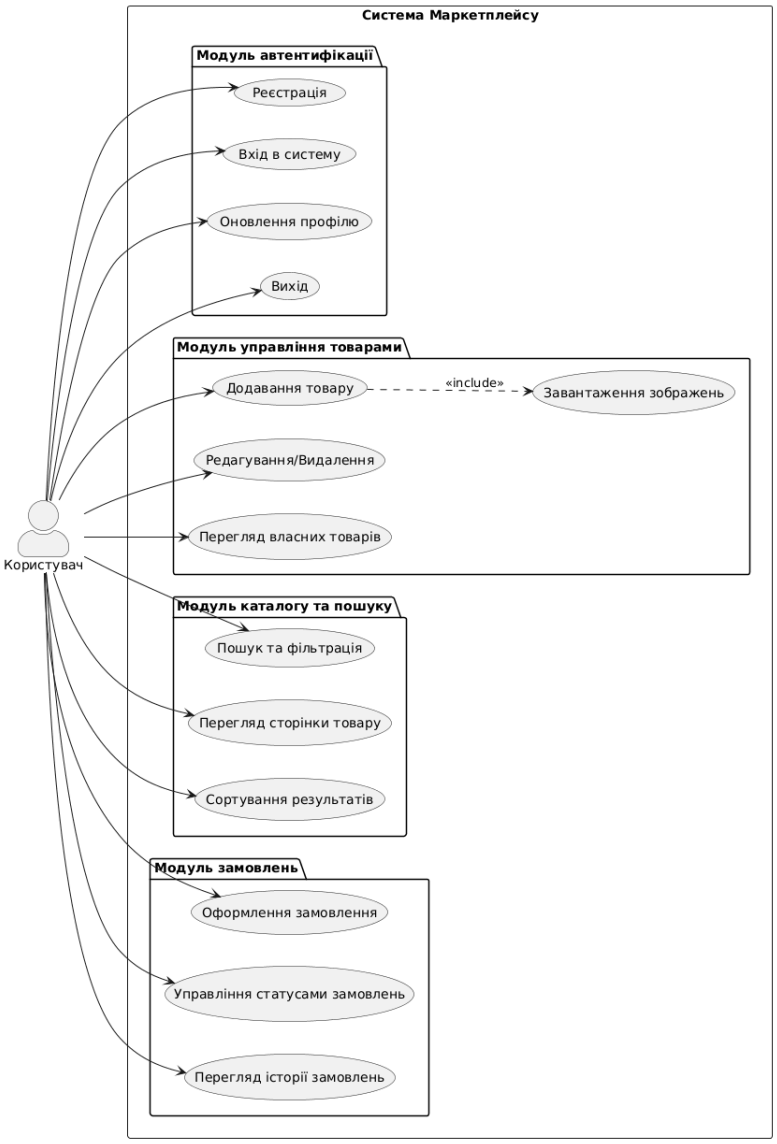


Рисунок 2.1 – Діаграма прецедентів

2.4.2 ER-діаграма бази даних

Модель "сутність-зв'язок" (Entity-Relationship model, ER-модель) є фундаментальним інструментом концептуального проєктування баз даних (БД). Вона оперує трьома ключовими поняттями: "сутність", "атрибут" та "зв'язок" . Основне призначення ER-моделі полягає у формалізованому описі предметної області, для якої створюється база даних. Візуалізація предметної області за допомогою ER-діаграм дозволяє структурувати дані, виявити взаємозв'язки між ними та мінімізувати потенційні помилки на етапі

проєктування, особливо у випадку складних та великомасштабних систем. Важливо зазначити, що ER-модель представляє концептуальний рівень абстракції і не визначає специфічних деталей фізичної реалізації БД [12].

Для розроблюваного маркетплейсу ER-діаграма (рис. 2.2) відображає архітектуру, в центрі якої знаходяться чотири базові сутності:

Сутність «Користувачі» (Users): є головною в системі та пов’язана зі сутностями товарів та замовлень зв’язками типу «один-до-багатьох» (1:N). Це дозволяє одному користувачу виступати одночасно в ролі продавця (розміщувати багато товарів) та покупця (створювати багато замовлень).

Сутність «Категорії» (Categories): виконує роль класифікатора. Зв’язок із товарами типу 1:N гарантує, що кожен товар чітко віднесений до певної категорії для коректної роботи фільтрації.

Сутність «Товари» (Products): містить детальні характеристики лотів. Вона є залежною від категорій та користувачів, що забезпечує ідентифікацію власника товару та його приналежність до товарної групи.

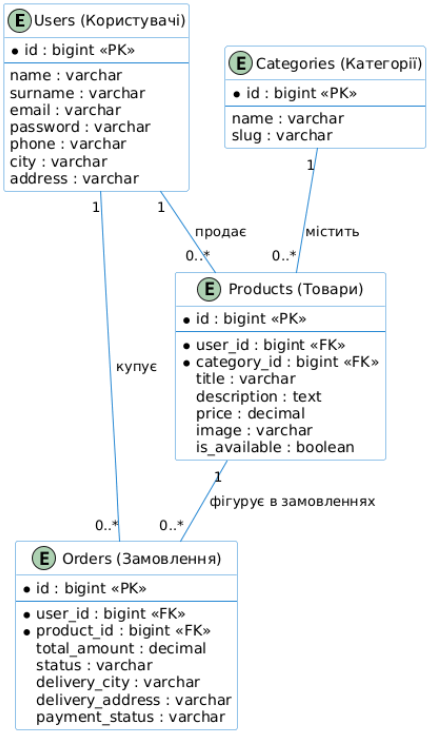


Рисунок 2.2- ER діаграма

2.5 Проєктування бази даних

Для реалізації функціоналу універсального маркетплейсу, що базується на моделі взаємодії C2C (Consumer-to-Consumer), спроєктовано реляційну структуру бази даних. Обрана архітектура забезпечує гнучкість управління контентом, де кожен зареєстрований суб'єкт має можливість одночасно виступати в ролі продавця та покупця без необхідності створення окремих облікових записів.

Таблиця «Users» (Користувачі): є центральною сутністю системи, що містить ідентифікаційні та авторизаційні дані користувачів. Включає унікальний ідентифікатор (Id -первинний ключ, PK), ім'я (Name), електронну пошту (Email) та хешований пароль (Password). Поле Role (за замовчуванням «user») дозволяє розмежувати права доступу між звичайними користувачами та адміністраторами платформи. Поле created_at фіксує дату реєстрації в системі.

Таблиця «Categories» (Категорії) :містить класифікатор товарів за категоріями для зручної навігації. Включає унікальний ідентифікатор (Id -PK), назву категорії (Name) та унікальний людино-зрозумілий ідентифікатор (Slug), який використовується для формування коректних URL-адрес сторінок каталогу.

Таблиця «Products» (Товари/Оголошення): зберігає інформацію про всі оголошення, розміщені на платформі. Містить унікальний ідентифікатор (Id -PK), зовнішній ключ (user_id -FK) для зв'язку з автором оголошення (продавцем) та зовнішній ключ (category_id -FK) для класифікації товару. Таблиця включає назву (Title), опис (Description), ціну (Price), шлях до зображення (image_path) та поточний статус оголошення (Status), наприклад, «активно» або «продано».

Таблиця «Orders» (Замовлення): фіксує інформацію про здійснені угоди. Включає унікальний ідентифікатор (Id -PK) та зовнішній ключ (user_id -FK), що посилається на користувача, який виступає в ролі покупця. Також містить

загальну вартість замовлення (total_price), статус замовлення (Status) та адресу доставки (shipping_address).

Таблиці «Users» та «Products» пов'язані відношенням «один-до-багатьох» через поле user_id (FK) в Products. Це реалізує можливість одного користувача розміщувати необмежену кількість оголошень.

Таблиці «Categories» та «Products» пов'язані відношенням «один-до-багатьох» через поле category_id (FK) в Products, що дозволяє групувати товари за певними категоріями.

Таблиці «Users» та «Orders» пов'язані відношенням «один-до-багатьох» через поле user_id (FK) в Orders. У цьому контексті зв'язок ідентифікує користувача як покупця.

Таблиці «Products» та «Order_Items» пов'язані відношенням «один-до-багатьох» через поле product_id (FK) в Order_Items, що відображає включення конкретного товару до складу угоди

2.6 Опис API та взаємодії компонентів

2.6.1 Основні маршрути системи

У розробленому маркетплейсі замість зовнішнього REST API використовується внутрішня маршрутизація Laravel, яка забезпечує взаємодію між браузером клієнта та серверною частиною засобами HTTP-протоколу. Усі маршрути визначені у файлі routes/web.php та поділені на три групи: публічні, для гостей та для авторизованих користувачів.

Публічна група маршрутів забезпечує доступ до головної сторінки, каталогу товарів та сторінки окремого товару без необхідності авторизації. Маршрут перегляду товару обмежений числовим параметром за допомогою

методу whereNumber(), що запобігає конфлікту зі статичним маршрутом створення оголошення.

Маршрути для гостей - реєстрація та вхід - захищені middleware guest і автоматично перенаправляють авторизованого користувача на головну сторінку. Захищена група маршрутів middleware auth охоплює профіль, управління товарами та оформлення замовлень. Детальний перелік та технічні характеристики всіх реалізованих маршрутів вебзастосунку наведено в таблиці 2.1.

Таблиця 2.1 – Опис маршрутів системи

Тип запиту	Адреса	Призначення
GET	/	Головна сторінка
GET	/catalog	Каталог товарів
GET	/products/{id}	Сторінка товару
GET	/profile	Особистий кабінет
POST	/login	Обробка входу
POST	/logout	Вихід з системи
POST	/products	Створення товару
PATCH	/checkout/{id}	Оновлення статусу замовлення
POST	/profile	Оновлення профілю

2.6.2 Формат запитів та відповідей

Оскільки застосунок реалізований за архітектурою MVC з серверним рендерингом сторінок, обмін даними між клієнтом та сервером здійснюється через HTML-форми з методами GET та POST. Для операцій PUT, PATCH та DELETE використовується прихований елемент форми @method(), що є стандартною практикою у Laravel.

Усі запити на зміну даних захищені CSRF-токеном, який автоматично перевіряється middleware VerifyCsrfToken. Токен передається через приховане поле форми, яке генерується директивою @csrf у Blade-шаблоні.

Вхідні дані проходять валідацію через метод validate() Request-об'єкта в контролері. У разі невідповідності правилам валідації Laravel автоматично повертає користувача на попередню сторінку з повідомленнями про помилки, які відображаються у Blade-шаблоні за допомогою директив @error. Відповіді сервера формуються у вигляді HTML-сторінок через метод view() або перенаправлень через redirect().

Завантаження зображень товарів реалізовано через multipart/form-data запит. Файл зберігається у директорії storage/app/public/products/ методом store(), а в базі даних зберігається лише відносний шлях до файлу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Реалізація серверної частини

3.1.1 Структура проєкту

Серверна частина маркетплейсу реалізована на основі фреймворку Laravel 10 та дотримується архітектурного патерну MVC (Model-View-Controller). Ця архітектура забезпечує чіткий розподіл відповідальності між компонентами системи: моделі відповідають за взаємодію з базою даних, контролери - за обробку бізнес-логіки, а представлення - за формування HTML-розмітки.

Структура каталогів проєкту організована відповідно до стандартних угод Laravel. Основні компоненти серверної частини розміщені у директорії app/ та поділені на підкаталоги: Http/Controllers/ містить усі контролери системи, Models/ - Eloquent-моделі, Policies/ - класи авторизації. Директорія database/ включає міграції та сідери, resources/views/ - Blade-шаблони.

Проєкт містить такі контролери: HomeController відповідає за формування головної сторінки; CatalogController - за виведення каталогу товарів із підтримкою фільтрації, пошуку та сортування; ProductController - за повний цикл управління товарами (CRUD); OrderController - за оформлення та обробку замовлень; ProfileController - за управління особистим кабінетом; AuthController - за реєстрацію, авторизацію та вихід із системи.

3.1.2 Реалізація маршрутизації

Маршрутизація є критичною складовою серверної частини. Особливу увагу приділено порядку оголошення маршрутів: статичні маршрути

(наприклад, /products/create) мають бути оголошені до динамічних маршрутів з параметрами ({id}). Для запобігання конфліктам публічний маршрут перегляду товару обмежено методом whereNumber(), що гарантує обробку лише числових ідентифікаторів.

Маршрути згруповано за допомогою middleware: група guest захищає маршрути автентифікації від повторного відвідування авторизованими користувачами, а група auth обмежує доступ до функціоналу управління товарами та замовленнями.

3.1.3 Реалізація автентифікації

Система автентифікації реалізована вручну без використання готових пакетів, що забезпечило повний контроль над логікою. Реєстрація включає валідацію введених даних, хешування пароля за допомогою алгоритму bcrypt через фасад Hash та автоматичний вхід після успішної реєстрації.

Вхід у систему здійснюється через метод Auth::attempt(), який автоматично перевіряє відповідність email та пароля, а також підтримує функцію "Запам'ятати мене" через параметр remember. Після успішного входу відновлюється ідентифікатор сесії для захисту від атак фіксації сесії.

Усі захищені маршрути використовують middleware auth, який при спробі неавторизованого доступу автоматично перенаправляє користувача на сторінку входу. Вихід із системи виконується через POST-запит, що інвалідує поточну сесію та оновлює CSRF-токен.

3.2 Реалізація клієнтської частини

3.2.1 Структура Blade-шаблонів

Клієнтська частина реалізована з використанням шаблонізатора Blade, вбудованого у Laravel. Усі сторінки успадковують базовий макет

layouts/app.blade.php, який містить спільні елементи інтерфейсу: хедер з логотипом, рядком пошуку та навігаційними посиланнями, а також футер і блок для відображення flash-повідомлень.

Директива @yield визначає місця для вставки унікального вмісту кожної сторінки, а @section - наповнює ці місця в дочірніх шаблонах. Для повторюваних елементів інтерфейсу, зокрема картки товару, використовується часткове представлення products/_card.blade.php, яке підключається через директиву @include.

Система налічує такі Blade-шаблони:

– layouts/app.blade.php - базовий макет;

– home/index.blade.php - головна сторінка з героєм та переліком товарів;

– catalog/index.blade.php - каталог з фільтрами;

– products/show.blade.php - деталізована сторінка товару;

– products/create.blade.php та products/edit.blade.php - форми управління оголошеннями;

– orders/checkout.blade.php - форма оформлення замовлення;

– orders/success.blade.php - сторінка підтвердження;

– profile/index.blade.php - особистий кабінет;

– auth/login.blade.php та auth/register.blade.php - форми автентифікації.

3.2.2 Реалізація інтерфейсу користувача

Реалізація інтерфейсу користувача (UI) та користувацького досвіду (UX) є критично важливим етапом розробки маркетплейсу, оскільки візуальна якість та зручність навігації безпосередньо впливають на рівень конверсії та лояльність аудиторії. Інтерфейс розроблюваного застосунку базується на принципах мінімалізму, чистоти візуальних форм та аптивності.

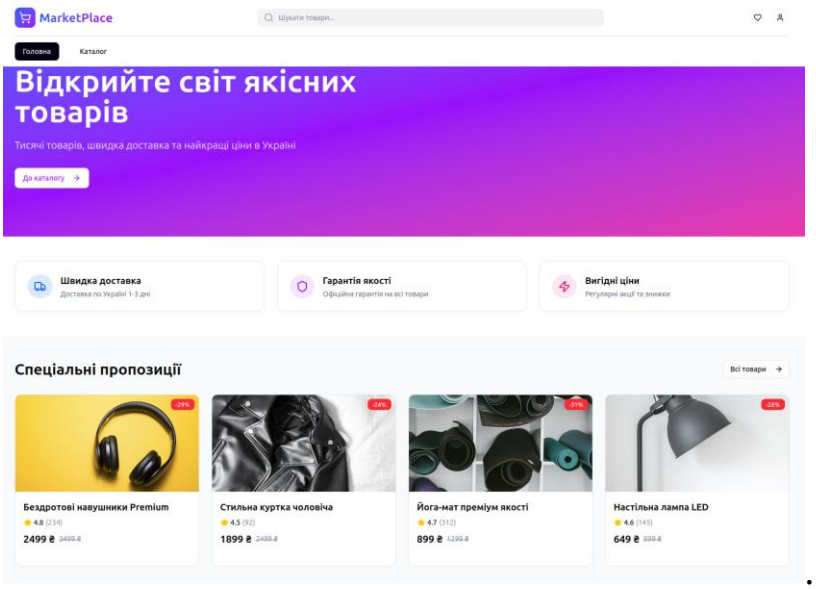


Рисунок 3.1 – Головна сторінка

Головна сторінка (рис. 3.1) виконує роль презентаційного хабу. Центральним елементом є динамічний банер («Відкрийте світ якісних товарів») із закликом до дії (CTA). Нижче розташовано інформаційні блоки з перевагами платформи (доставка, гарантія, ціни) та секція «Спеціальні пропозиції», що демонструє картки товарів із акцентними мітками знижок

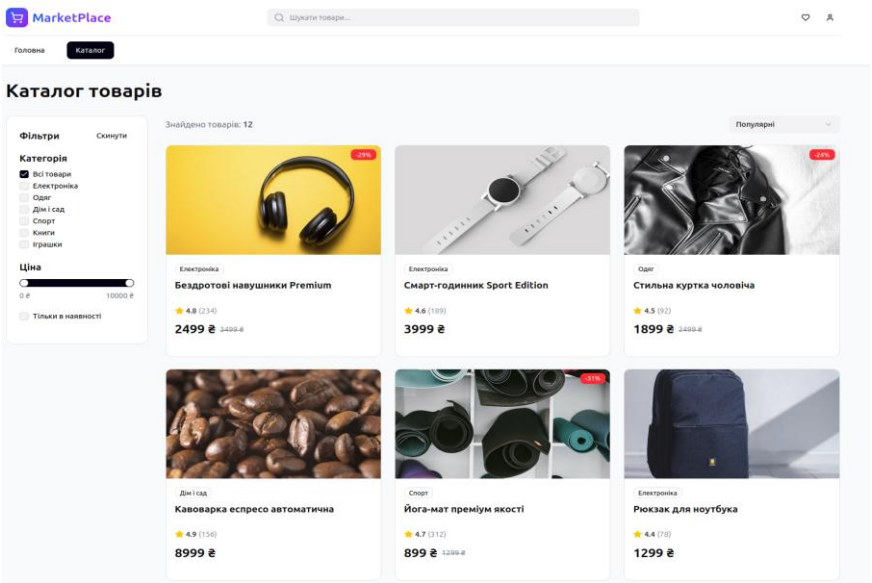


Рисунок 3.2 – Сторінка каталогу

Каталог товарів (рис. 3.2) реалізований за дворівневою структурою. Ліва частина містить функціональну панель фільтрації (за категоріями, ціновим діапазоном та наявністю), що забезпечує швидкий пошук. Права частина представлена сіткою карток товарів, які містять зображення, назву, рейтинг та актуальну ціну.

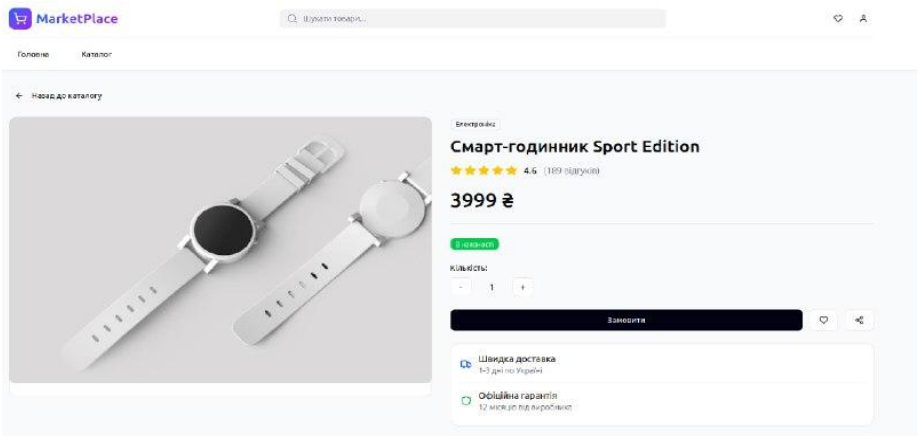


Рисунок 3.3 – Сторінка товару

Сторінка товару (рис. 3.3) спроєктована з фокусом на контент. Ліва частина відведена під великоформатне зображення товару, права – під блок прийняття рішення, що включає назву, рейтинг (відгуки), ціну та інструменти вибору кількості. Акцентна кнопка «Замовити» виділена контрастним кольором для стимуляції цільової дії.

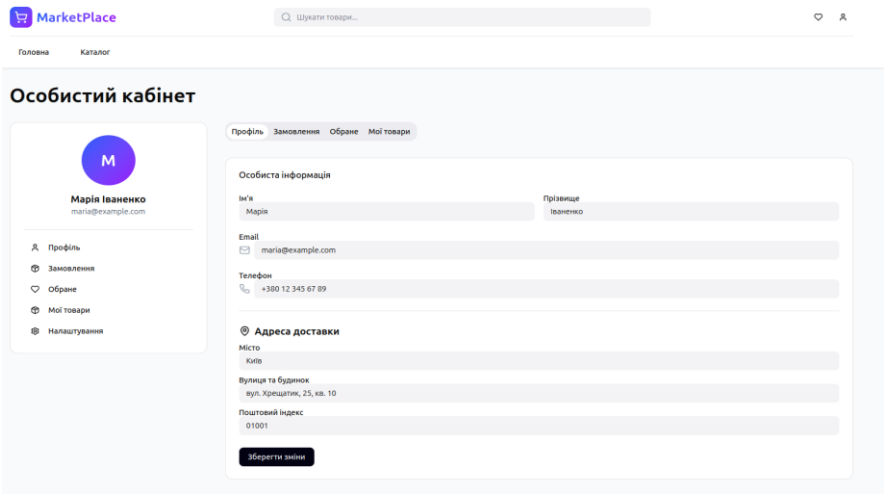


Рисунок 3.4 – Особистий кабінет

Особистий кабінет (рис. 3.4) має модульну структуру з бічним меню навігації. Включає розділи «Профіль» для редагування особистих даних та адреси доставки, «Замовлення» для відстеження історії покупок, а також розділи «Обране» та «Мої товари».

3.3 Реалізація бази даних

3.3.1 Створення таблиць

Схема бази даних реалізована через систему міграцій Laravel, що забезпечує версійний контроль структури БД та можливість відтворення схеми у будь-якому середовищі. Усі міграції виконуються командою php artisan migrate, а початкові дані заповнюються через сідери командою php artisan db:seed. Графічну структуру бази даних маркетплейсу та зв'язки між її сутностями представлено на ER-діаграмі (рис. 2.2).

Таблиця users є центральною в системі та зберігає дані користувачів: ім'я, прізвище, email (унікальний), хешований пароль, телефон та адресу доставки (місто, вулиця, поштовий індекс). Один запис у цій таблиці може одночасно бути і покупцем, і продавцем.

Таблиця categories містить перелік категорій товарів із назвою та slug-ідентифікатором для URL. Початкові дані для шести категорій (Електроніка, Одяг, Дім і сад, Спорт, Книги, Іграшки) додаються через CategorySeeder.

Таблиця products зберігає оголошення продавців. Кожен запис містить зовнішні ключі user_id та category_id, назву, опис, поточну ціну, стару ціну для відображення знижки, шлях до зображення та статус наявності. Зовнішні ключі налаштовані на каскадне видалення, що забезпечує цілісність даних при видаленні користувача або категорії.

Таблиця orders фіксує комерційні транзакції. Для збереження деталей замовлення на момент оформлення всі дані доставки та оплати дублюються безпосередньо у запис замовлення. Це технічне рішення гарантує незмінність даних замовлення навіть у разі зміни профілю користувача після покупки. Статус замовлення реалізований як ENUM з трьома значеннями: pending (очікує відправки), sent (відправлено) та cancelled (скасовано).

3.3.2 CRUD-операції

Усі операції з базою даних реалізовані через Eloquent ORM - об'єктно-реляційний маппер Laravel. Моделі визначають зв'язки між таблицями через методи belongsTo(), hasMany(), що дозволяє отримувати пов'язані дані через механізм eager loading методом with().

Операції читання в каталозі реалізовані через ланцюжок методів запиту: фільтрація за категорією, ціновим діапазоном та статусом наявності, текстовий пошук за назвою через оператор LIKE, сортування та пагінація по 12 товарів на сторінку методом paginate(). Параметри запиту автоматично зберігаються в посиланнях пагінації через метод withQueryString().

Операції запису виконуються через метод create() з масовим заповненням, захищеним властивістю $fillable моделі. Оновлення здійснюється методом update(), видалення - методом delete(). При видаленні товару попередньо виконується видалення пов'язаного файлу зображення зі сховища через Storage::disk('public')->delete().

3.4 Реалізація основного функціоналу

3.4.1 Реєстрація та вхід

Процес реєстрації починається з заповнення форми з полями імені, прізвища, email та пароля з підтвердженням (рис. 3.5). Сервер виконує валідацію: перевіряється заповненість обов'язкових полів, коректність формату email, унікальність електронної адреси в системі та відповідність паролів. Пароль зберігається у БД у хешованому вигляді алгоритмом bcrypt. Після успішної реєстрації користувач автоматично входить у систему та перенаправляється на головну сторінку з повідомленням про успіх.

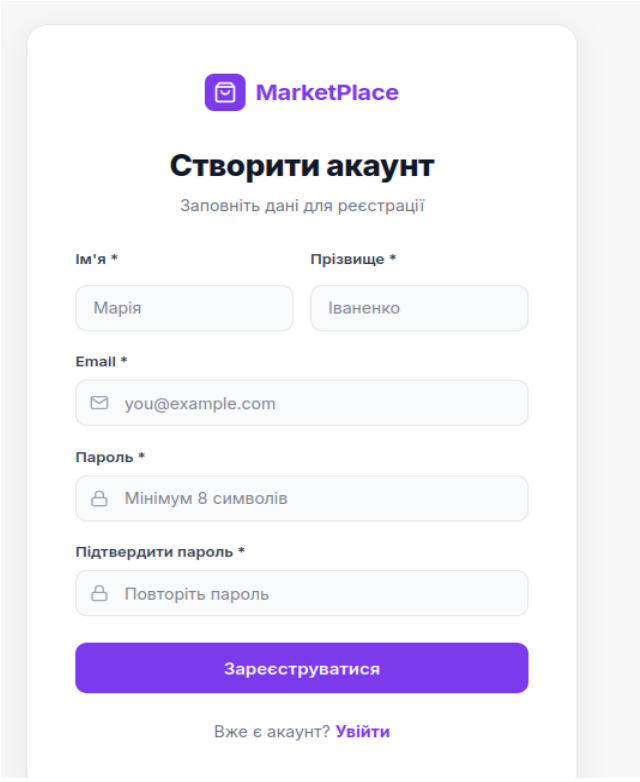


Рисунок 3.5 – Форма реєстрації

Програмну реалізацію методу реєстрації користувача та його валідації на стороні сервера наведено у лістингу

```
  public function register(Request $request)
   {
      $request->validate([
         'first_name' => ['required', 'string', 'max:255'],
         'last_name'  => ['required', 'string', 'max:255'],
         'email'      => ['required', 'email', 'unique:users,email'],
         'password'   => ['required', 'confirmed', Password::min(8)],
      ], [
         'first_name.required' => "Ім'я є обов'язковим",
         'last_name.required'  => "Прізвище є обов'язковим",
         'email.required'      => 'Email є обов\'язковим',
         'email.unique'        => 'Цей email вже зареєстровано',
         'password.required'   => 'Пароль є обов\'язковим',
         'password.confirmed'  => 'Паролі не співпадають',
         'password.min'        => 'Пароль має містити мінімум 8 символів',
      ]);
      $user = User::create([
         'first_name' => $request->first_name,
         'last_name'  => $request->last_name,
         'email'      => $request->email,
         'password'   => Hash::make($request->password),
      ]);
      Auth::login($user);
      return redirect()->route('home')->with('success', 'Реєстрацію успішно завершено!');
   }
```

Форма входу приймає email та пароль (рис 3.6). Метод Auth::attempt() виконує перевірку облікових даних та у разі успіху створює автентифіковану сесію. Функція "Запам'ятати мене" реалізована через передачу булевого параметра до методу attempt(). При невдалій спробі входу відображається повідомлення про помилку.

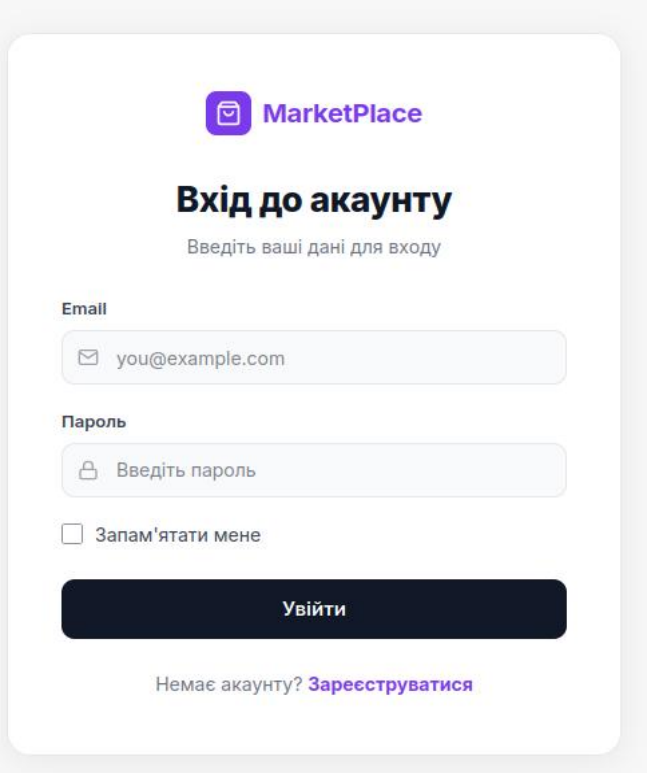


Рисунок 3.6 – Форма входу

3.4.2 Управління товарами

Створення оголошення доступне лише авторизованим користувачам. Форма містить поля назви, опису, категорії, ціни, старої ціни для відображення знижки, статусу наявності та завантаження зображення. Зображення зберігається у директорії storage/app/public/products/ через метод store(), який автоматично генерує унікальне ім'я файлу для запобігання конфліктів (рис 3.7). У базі даних зберігається лише відносний шлях. Програмну реалізацію методу збереження нового товару та завантаження графічного файлу у сховище наведено у лістингу:

```
public function store(Request $request)
    {
        $validated = $request->validate([
            'title'       => ['required', 'string', 'max:255'],
            'description' => ['required', 'string'],
            'price'       => ['required', 'numeric', 'min:0'],
            'old_price'   => ['nullable', 'numeric', 'min:0'],
            'category_id' => ['required', 'exists:categories,id'],
            'status'      => ['required', 'in:available,unavailable'],
            'image'       => ['nullable', 'image', 'max:2048'],
        ]);

        $imagePath = null;
        if ($request->hasFile('image')) {
            $imagePath = $request->file('image')->store('products', 'public');
        }

        Product::create([
            'user_id'     => Auth::id(),
            'title'       => $validated['title'],
            'description' => $validated['description'],
            'price'       => $validated['price'],
            'old_price'   => $validated['old_price'] ?: null,
            'category_id' => $validated['category_id'],
            'status'      => $validated['status'],
            'image'       => $imagePath,
        ]);

        return redirect()->route('profile.index', ['tab' => 'products'])
            ->with('success', 'Оголошення успішно створено!');
    }
```

Редагування та видалення товару захищені перевіркою права власності: контролер порівнює ідентифікатор поточного користувача з полем user_id товару. При невідповідності повертається відповідь з HTTP-статусом 403. При оновленні фотографії старий файл видаляється зі сховища перед збереженням нового. Програмну реалізацію методу видалення товару наведено у лістингу:

```
public function destroy(Product $product)
 {
    $this->authorize('delete', $product);
    if ($product->image) {
        Storage::disk('public')->delete($product->image);
    }
    $product->delete();
    return back()->with('success', 'Оголошення видалено!');
 }
```

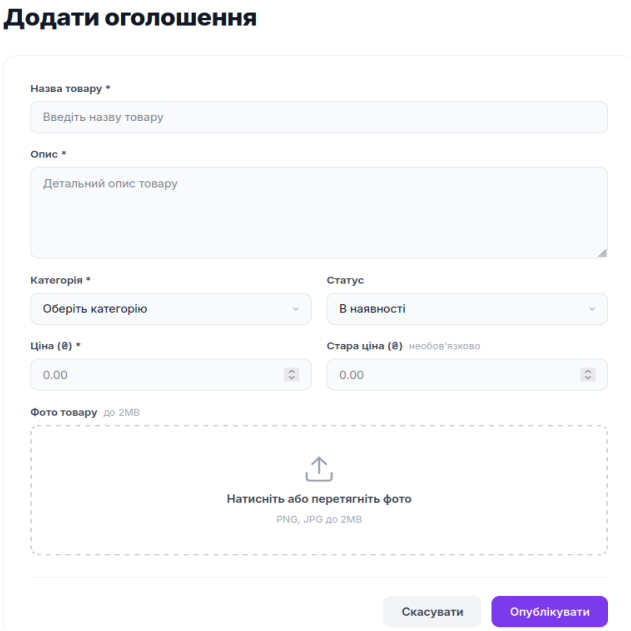


Рисунок 3.7 – Форма входу

3.4.3 Пошук та фільтрація

Пошук і фільтрація реалізовані в CatalogController через динамічне формування запиту до БД. Усі параметри фільтрів передаються через GET-запит, що забезпечує можливість збереження та передачі посилань на відфільтровані результати.

Текстовий пошук виконується за полем title через оператор LIKE з відсотковими символами з обох боків. Фільтрація за категорією реалізована через порівняння поля category_id. Ціновий діапазон задається через

параметри min_price та max_price з відповідними операторами порівняння. Фільтр "Тільки в наявності" відбирає товари зі статусом available. Усі активні фільтри можна застосовувати одночасно, оскільки вони накладаються послідовно на той самий об'єкт запиту

Сортування результатів реалізовано через SELECT-елемент з параметром sort. Підтримуються варіанти: за датою додавання (новинки), за ціною від низької до високої та навпаки, за популярністю (кількість замовлень). При зміні параметра сортування JS-обробник оновлює URL без перезавантаження форми фільтрів (рис 3.8).

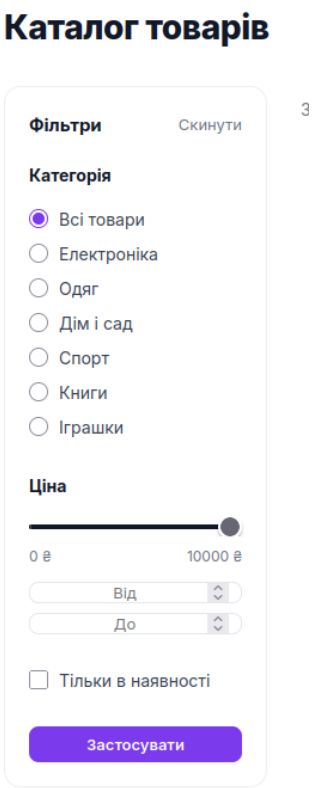


Рисунок 3.8 - Форма входу

3.4.4 Оформлення замовлення

Процес оформлення замовлення починається з натискання кнопки "Замовити" на сторінці товару. Система виконує попередні перевірки: товар

має бути в наявності, а покупець не має бути його власником. При невиконанні умов користувач перенаправляється з відповідним повідомленням.

Форма checkout попередньо заповнюється даними з профілю користувача для зручності. Окрім адреси доставки, форма містить секцію тестової оплати з живим превью банківської картки. Введений 16-значний номер картки форматується JavaScript у групи по чотири цифри для читабельності (рис. 3.9).

Після успішної валідації даних створюється запис у таблиці orders, де зберігаються ідентифікатори покупця та продавця, товару, повні дані доставки та останні чотири цифри картки. Статус товару автоматично змінюється на unavailable, щоб унеможливити повторну покупку. Завершується процес перенаправленням на сторінку підтвердження замовлення.

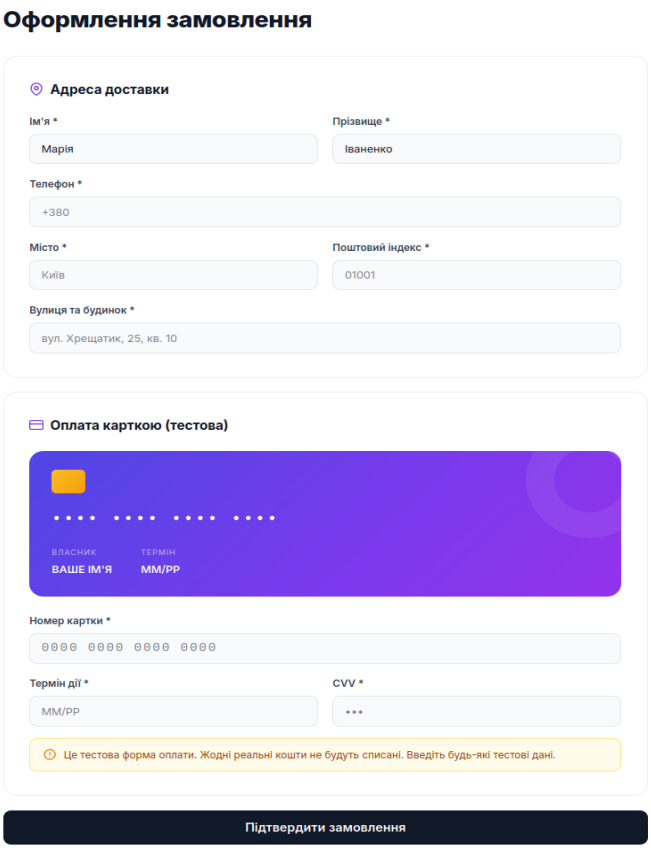


Рисунок 3.9 - Оформлення замовлення

3.5 Тестування системи

3.5.1 Ручне тестування

Тестування розробленого маркетплейсу проводилось методом ручного тестування, який передбачає перевірку функціональності системи шляхом безпосередньої взаємодії з інтерфейсом у браузері. Тестування охоплювало всі ключові сценарії використання застосунку та перевірку обробки граничних і некоректних вхідних даних.

Для проведення тестування було розгорнуто локальне середовище розробки з використанням вебсервера та СУБД MySQL. Тестування проводилось у браузері Google Chrome (рис 3.10), а для перевірки відображення у різних браузерах додатково використовували Mozilla Firefox (рис 3.11).

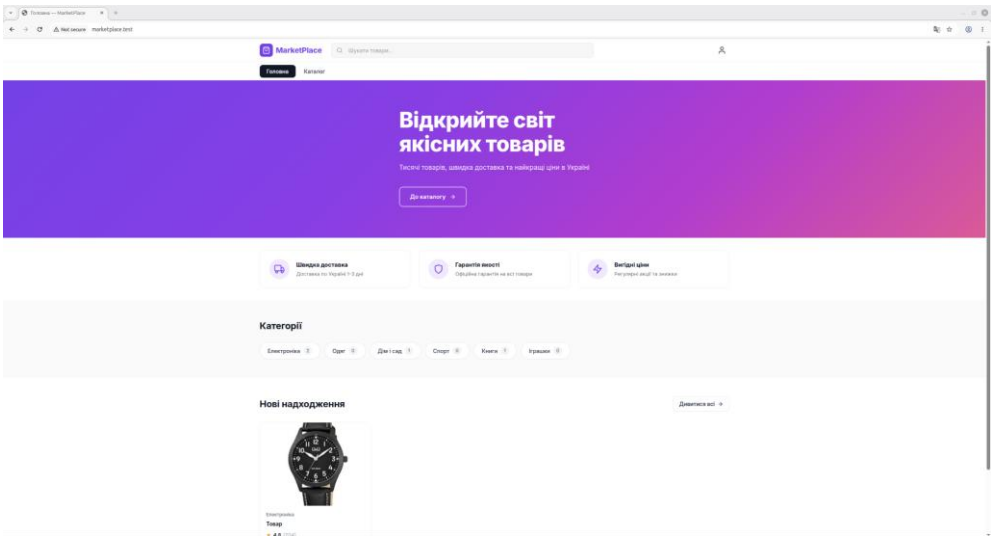


Рисунок 3.10 – Сторінка у Google Chrome

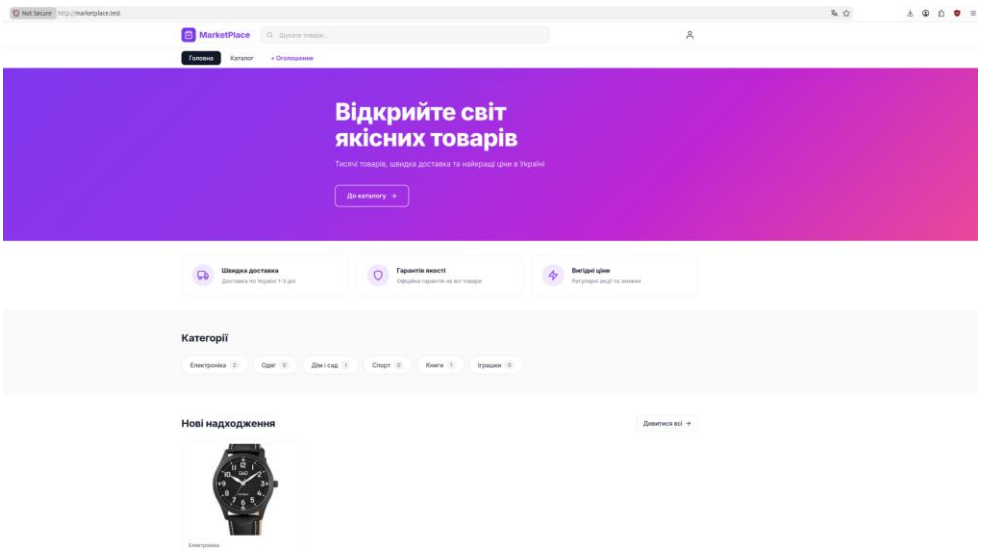


Рисунок 3.10 – Сторінка у Mozilla Firefox

3.5.2 Перевірка основних сценаріїв

Тестовий сценарій реєстрації та авторизації включав: реєстрацію нового користувача з валідними даними - очікуваний результат - успішна реєстрація та вхід у систему; спробу реєстрації з вже існуючим email - результат - повідомлення про помилку; реєстрацію з паролями, що не співпадають - результат - відображення помилки валідації; вхід з коректними обліковими даними - результат - успішна авторизація; вхід з неправильним паролем - результат - повідомлення про помилку без вказання причини.

Тестовий сценарій управління товарами включав: створення оголошення з усіма обов'язковими полями та зображенням - результат - успішне збереження та перенаправлення; спробу створення без назви або ціни - результат - відображення помилок валідації; завантаження файлу нефотографічного формату - результат - помилка валідації типу файлу; редагування власного товару - результат - успішне оновлення; спробу редагування чужого товару через прямий URL - результат - відповідь 403.

Тестовий сценарій оформлення замовлення включав: перехід на checkout з доступним товаром - результат - форма попередньо заповнена

даними профілю; спробу купити власний товар - результат - повідомлення про помилку; оформлення з некоректним номером картки - результат - помилка валідації; успішне оформлення - результат - зміна статусу товару на недоступний, створення запису в БД, відображення сторінки підтвердження.

Тестовий сценарій функціоналу продавця включав: перехід у вкладку "Продажі" - результат - відображення списку отриманих замовлень з даними покупців; натискання кнопки "Відправити" - результат - зміна статусу на sent; відображення оновленого статусу у покупця у вкладці "Замовлення" - результат - "В дорозі".

Всі перевірені сценарії відпрацювали відповідно до очікувань. Валідація форм коректно відображає повідомлення про помилки поруч з відповідними полями. Захист маршрутів через middleware та перевірки в контролерах надійно запобігають несанкціонованому доступу до даних.

3.6 Розгортання вебзастосунку

3.6.1 Налаштування серверного середовища

Для розгортання Laravel-застосунку необхідне серверне середовище, що включає: PHP з розширеннями BCMath, Ctype, Fileinfo, JSON, Mbstring, OpenSSL, PDO, Tokenizer та XML; вебсервер Apache або Nginx; СУБД MySQL версії 5.7 або вище; менеджер пакетів Composer.

У локальному середовищі розробки проєкт розгорнуто з використанням стеку LAMP (Linux, Apache, MySQL, PHP). Для зручності налаштування доменного імені marketplace.test використовується файл /etc/hosts з відповідним записом та конфігурація віртуального хосту Apache, що вказує document root на директорію public/ проєкту.

3.6.2 Процес розгортання

Розгортання застосунку виконується у такій послідовності. По-перше, клонування або копіювання файлів проєкту на сервер та встановлення залежностей командою composer install. По-друге, налаштування файлу оточення: копіювання .env.example у .env та заповнення параметрів підключення до бази даних (DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD) і базового URL застосунку (APP_URL). По-третє, генерація унікального ключа застосунку командою php artisan key:generate. По-четверте, виконання міграцій та наповнення початковими даними командою php artisan migrate --seed. По-п'яте, створення символічного посилання для публічного доступу до завантажених файлів командою php artisan storage:link.

Права доступу до файлової системи налаштовуються так, щоб вебсервер мав право запису у директорії storage/ та bootstrap/cache/. Для коректного відображення завантажених зображень у браузері директорія public/storage має бути доступна через публічний URL.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було повністю досягнуто поставлену мету та розроблено повнофункціональний вебзастосунок маркетплейсу, який забезпечує зручну електронну взаємодію між продавцями та покупцями в єдиному цифровому середовищі.

Відповідно до першого завдання роботи, було проведено аналіз предметної області та існуючих платформ електронної комерції (OLX, Prom.ua, Rozetka), що дозволило визначити основні тенденції ринку та сформувати функціональний набір для навчального маркетплейсу. На основі проведеного аналізу аналогів було успішно виконано друге завдання – визначено та сформульовано функціональні та нефункціональні вимоги до розроблюваної системи.

Для виконання наступних завдань, за допомогою порівняльного аналізу, було обрано та технічно обґрунтовано сучасний технологічний стек. Використання мови програмування PHP версії 8.2 та фреймворку Laravel 10 з шаблонізатором Blade забезпечило високу швидкість розробки, гнучкість налаштування маршрутів та надійний захист даних.

У межах етапу проєктування було реалізовано структурний підхід: побудовано архітектуру вебзастосунку за принципами архітектурного патерну MVC та спроєктовано реляційну базу даних у СУБД MySQL. Створена база даних містить взаємопов'язані таблиці для повного обліку користувачів, категорій, товарів, замовлень та окремих позицій замовлень із забезпеченням умов нормалізації.

Практичні результати роботи полягають у тому, що розроблена система реалізує гібридну модель взаємодії користувачів, за якої кожен зареєстрований користувач може одночасно виступати як продавець та як покупець. У межах вебзастосунку програмно реалізовано систему автентифікації та авторизації з захистом маршрутів, функціонал каталогу товарів з підтримкою текстового

пошуку, фільтрації за категорією та ціновим діапазоном, сортування та пагінації. Також розроблено повний цикл управління оголошеннями продавця, модуль оформлення замовлень із симуляцією платіжної форми, особистий кабінет користувача та захищену адміністративну панель для модерації контенту й керування категоріями. Код структурований відповідно до стандартів Laravel, а інтерфейс є адаптивним для різних екранів.

Згідно з фінальним завданням, за допомогою експериментального методу було проведено комплексне функціональне тестування розробленого застосунку. Усі виявлені під час перевірки інтерфейсу та серверної валідації недоліки було усунено, що підтвердило стабільність роботи платформи.

Серед можливих напрямків подальшого вдосконалення системи варто виділити такі:

1. Інтеграція реальної платіжної системи (LiqPay, Stripe) замість поточної симуляції оплати.

2. Додавання функціоналу відгуків та рейтингів товарів, що підвищить довіру покупців.

3. Впровадження системи сповіщень (email або push) при зміні статусу замовлення.

4. Реалізація функціоналу "Обране" для збереження товарів.

5. Оптимізація запитів Eloquent ORM до бази даних та впровадження механізмів кешування для підвищення продуктивності системи під час зростання навантаження.

Таким чином, виконана кваліфікаційна робота наочно демонструє доцільність використання сучасного стеку технологій – фреймворку Laravel 10, реляційної СУБД MySQL, вбудованої системи міграцій та екосистеми Eloquent ORM, як технологічної основи для розробки вебзастосунків у сфері електронної комерції та демонструє можливість реалізації повноцінного маркетплейсу в рамках навчального проєкту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке маркетплейси і чим вони відрізняються від інтернет-магазинів : MC.today (MC Money and Career). URL: https://mc.today/uk/shho-take-marketplejsi (дата звернення: 25.04.2026).

2. OLX.ua : сервіс оголошень. URL: https://www.olx.ua/uk/ (дата звернення: 25.04.2026).

3. Prom.ua : маркетплейс України. URL: https://prom.ua/ (дата звернення: 25.04.2026).

4. ROZETKA : інтернет-супермаркет. URL: https://rozetka.com.ua/ (дата звернення: 25.04.2026).

5. Функціональні проти нефункціональних вимог: у чому різниця? : Guru99. URL: https://www.guru99.com/uk/functional-vs-non-functional-requirements (дата звернення: 25.04.2026).

6. Що таке паттерн MVC (Model-View-Controller) : Brander. URL: https://brander.ua/technologies/mvc (дата звернення: 25.04.2026).

7. Хто такий PHP розробник і чим він займається : Wezom. URL: https://wezom.com.ua/ua/blog/hto-takiy-php-rozrobnik (дата звернення: 25.04.2026).

8. Documentation : Laravel Framework : official website. URL: https://laravel.com/docs (дата звернення: 25.04.2026).

9. Що таке MySQL, де і як вона використовується : FreeHost. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-mysql/ (дата звернення: 25.04.2026).

10. Що таке Docker? Переваги використання : Westelecom. URL: https://westelecom.ua/blog/287_cto-takoe-docker (дата звернення: 25.04.2026).

11. Use Case (діаграма прецедентів) в UML : Lviv QA Club : блог. URL: https://lvivqaclub.blogspot.com/2008/10/use-case-uml-diagram.html (дата звернення: 25.04.2026).

12. Поняття ER-моделі. Поняття сутності та зв’язку. Атрибути. Типи атрибутів : BestProg.net. URL: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/ (дата звернення: 25.04.2026).

Декларація

академічної доброчесності

здобувача освіти ВСП «Економіко-правничий фаховий коледж ЗНУ»

Я, Іщенко Олег Юрійович , здобувач освіти 4 курсу, спеціальності/освітньо-професійної програми 121 Інженерія програмного забезпечення, групи К121-22, адреса електронної пошти leoi27838m@gmail.com

- підтверджую, що написана мною кваліфікаційна робота на тему «Проєктування та розробка вебзастосунку маркетплейсу для розміщення та продажу товарів користувачами» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений;
- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;
- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 01.06.2026 Підпис _____ Олег ІЩЕНКО

Дата 01.06.2026 Підпис _____ Юлія БОРИСОВСЬКА