МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ ЗАСОБАМИ ФРЕЙМВОРКУ LARAVEL»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.І. Шевченко
(ініціали та прізвище)

Керівник доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шевченку Максиму Ігоровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет магазину засобами фреймворку Laravel

керівник роботи Панасенко Євген Валерійович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проєктування та реалізація інформаційної системи інтернет-магазину.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.01.2025	
2.	Збір вихідних даних.	20.01.2025	
3.	Обробка методичних та теоретичних	17.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	24.03.2025	
5.	Розробка третього розділу.	20.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2025	

Студент _____ М.І. Шевченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Є.В. Панасенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет магазину засобами фреймворку Laravel»: 64 с., 44 рис., 1 табл., 16 джерел.

ВЕБЗАСТОСУНОК, ГОСТЬОВИЙ КОШИК, ЕЛЕКТРОННА КОМЕРЦІЯ, ІНТЕРНЕТ-МАГАЗИН, КАТЕГОРІЇ ТОВАРІВ, ПРОГРАМНА ІНЖЕНЕРІЯ, РОЗГОРТАННЯ, СЕРВЕРНА ЛОГІКА, ТЕСТУВАННЯ, BLADE, FRAMEWORK, LARAVEL, LIVEWIRE, MVC, PHP.

Об’єкт дослідження – процес розробки вебзастосунків для електронної комерції.

Мета роботи: створення функціонального вебзастосунку інтернет-магазину на базі Laravel із підтримкою гостьового кошика, адміністрування, перегляду замовлень та зручного інтерфейсу для користувача.

Метод дослідження – аналіз технічної літератури, моделювання системи за допомогою UML-діаграм, застосування архітектури MVC, реалізація за допомогою Laravel 11, Livewire та Tailwind CSS.

У роботі розглянуто створення вебзастосунку типу інтернет-магазину з використанням Laravel. Обґрунтовано вибір технологій, реалізовано функціонал управління товарами, категоріями, кошиком, замовленнями. Особливу увагу приділено реалізації гостьового кошика та адміністративної панелі. Забезпечено захист від CSRF, SQL-ін’єкцій, XSS. Інтерфейс побудовано з використанням Blade і Tailwind CSS. Результатом є функціональна система, придатна до розгортання та масштабування.

SUMMARY

Bachelor’s qualifying paper “Development of an Online Store Using the Laravel Framework”: 64 pages, 44 figures, 1 table, 16 references.

WEB APPLICATION, GUEST CART, E-COMMERCE, BLADE, ONLINE STORE, PRODUCT CATEGORIES, SOFTWARE ENGINEERING, DEPLOYMENT, SERVER LOGIC, TESTING, FRAMEWORK, LARAVEL, LIVEWIRE, MVC, PHP.

The object of the study is – the process of developing web applications for e-commerce.

The aim of the study is – to create an online store using Laravel that enables product and order management with a user-friendly interface.

The method of research is – framework analysis, system modeling (UML), development using MVC architecture.

The paper presents the development of a web-based e-commerce system using the Laravel framework. The choice of technologies is justified; modules for managing products, categories, cart, and orders are implemented. Special focus is given to guest cart functionality and admin panel tools. Security measures against CSRF, SQL injection, and XSS are ensured. The interface is built with Blade and styled using Tailwind CSS. The resulting system is fully functional, scalable, and ready for deployment.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 8
1 Аналіз та вибір фреймворку для створення вебсайту 10
1.1 Огляд сучасних PHP-фреймворків 10
1.2 Основні переваги та особливості Laravel 11
1.3 Обґрунтування вибору Laravel для реалізації проєкту 12
1.4 Порівняльна характеристика Laravel і альтернатив 13
1.5 Перспективи використання Laravel у подальшій розробці 16
1.6 Актуальність теми дослідження 17
2 Проєктування інформаційної системи інтернет-магазину 18
2.1 Аналіз предметної області 18
2.2 Визначення вимог до системи 20
2.3 Архітектура системи 21
2.4 Вибір технологій та інструментів 23
2.4.1 Серверна частина 24
2.4.2 Клієнтська частина 24
2.4.3 База даних 24
2.4.4 Додаткові інструменти 25
2.4.5 Обґрунтування вибору 25
2.5 Моделювання системи 25
2.5.1 Діаграма прецедентів 26
2.5.2 Діаграма класів 26
2.5.3 Діаграма послідовності 27
2.6 Організація та структура бази даних 28
2.6.1 ER-діаграма бази даних 29
2.6.2 Основні таблиці бази даних 30

2.6.3 Логіка роботи таблиць у системі 31
2.7 Забезпечення безпеки та захисту даних у системі 34
3 Реалізація інформаційної системи інтернет-магазину 36
3.1 Загальний опис реалізації 36
3.2 Реалізація функціоналу користувача 37
3.2.1 Перегляд товарів та категорій 37
3.2.2 Детальна картка товару 39
3.2.3 Додавання товару до кошика 40
3.2.4 Перегляд і редагування кошика 42
3.2.5 Оформлення замовлення 46
3.2.6 Перегляд історії замовлень 48
3.3 Реалізація адміністративного функціоналу інформаційної системи 50
3.3.1 Адміністративна панель: загальний опис 50
3.3.2 Керування товарами (CRUD) 52
3.3.3 Перегляд покинутих кошиків 57
3.4 Особливості реалізації гостьового кошика 59
Висновки 61
Перелік посилань 63

ВСТУП

Сучасний етап розвитку цифрових технологій характеризується стрімким зростанням обсягів електронної комерції, що зумовлює підвищений попит на ефективні та зручні вебзастосунки, здатні забезпечити автоматизацію процесів купівлі-продажу товарів через Інтернет [12]. В умовах глобалізації ринку онлайн-торгівля стала невід'ємною складовою бізнес-стратегій багатьох компаній, а створення інтернет-магазинів – актуальним завданням для фахівців у галузі інформаційних технологій.

Одним із пріоритетних напрямів розробки вебсистем є створення програмних засобів, які забезпечують інтуїтивно зрозумілий інтерфейс, зручну навігацію, адаптивний дизайн та належний рівень захисту персональних даних користувачів. Успішне вирішення цих завдань передбачає використання сучасних фреймворків, які дозволяють розробникам оперативно створювати масштабовані, розширювані та підтримувані рішення.

Особливу увагу в процесі розробки слід приділяти правильному вибору технологій, побудові логіки взаємодії з базою даних, формуванню структурованого інтерфейсу користувача, а також реалізації адміністративного функціоналу. Важливо забезпечити як зручність використання ресурсу для кінцевих споживачів, так і ефективність керування вмістом системи для адміністратора.

Актуальність теми дипломної роботи обумовлена необхідністю створення вебзастосунків, які не лише відповідають технічним вимогам та очікуванням користувачів, а й відповідають сучасним стандартам програмної інженерії. В рамках цієї роботи реалізовано інформаційну систему типу інтернет-магазину з використанням PHP-фреймворку Laravel, що дозволило застосувати гнучкий підхід до побудови архітектури, а також забезпечити масштабованість і надійність системи.

Метою даної роботи є розробка інформаційної системи інтернет-магазину, яка дозволяє здійснювати управління товарами, категоріями,

кошиком, замовленнями та забезпечує зручну взаємодію з користувачем. Для досягнення поставленої мети необхідно було вирішити низку задач, зокрема: провести аналіз предметної області, обрати інструменти реалізації, спроєктувати архітектуру системи, реалізувати функціональні модулі, протестувати працездатність програми та оцінити її ефективність.

1 АНАЛІЗ ТА ВИБІР ФРЕЙМВОРКУ ДЛЯ СТВОРЕННЯ ВЕБСАЙТУ

1.1 Огляд сучасних PHP-фреймворків

У сучасному світі інформаційних технологій створення вебдодатків стало невід'ємною частиною діяльності багатьох організацій, компаній та приватних розробників. Завдяки широкому доступу до мережі Інтернет, вебсайти перетворилися на ключовий елемент комунікації між бізнесом та його цільовою аудиторією. Саме тому важливість вибору надійного, гнучкого і функціонального фреймворку для реалізації вебпроєкту важко переоцінити.

PHP залишається однією з найпопулярніших мов програмування для веброзробки, і на її основі створено безліч фреймворків, кожен з яких має свої переваги, недоліки, особливості та сферу застосування [6]. Серед найвідоміших PHP-фреймворків можна виділити такі: Laravel, Symfony, Yii, CodeIgniter, Zend Framework [1].

Symfony – це потужний фреймворк, який орієнтований переважно на розробників з досвідом. Він надає надзвичайну гнучкість та масштабованість, дозволяючи реалізовувати складні проєкти із великою кількістю взаємопов'язаних компонентів. Проте саме ця гнучкість може бути перешкодою для новачків, яким буде важко налаштувати проєкт із нуля без достатньої підготовки.

Yii відзначається високою продуктивністю та простотою використання. Цей фреймворк розроблений для швидкої побудови вебзастосунків і надає розробнику зручні інструменти для генерації коду. Незважаючи на це, Yii має дещо обмежену екосистему порівняно з деякими конкурентами, що іноді створює труднощі при реалізації нестандартних рішень.

CodeIgniter – це легкий фреймворк, який відзначається надзвичайною простотою. Він ідеально підходить для невеликих проєктів, не потребує значних знань і забезпечує базову функціональність. Однак через свою

мінімалістичність він поступається більш функціональним рішенням у великих або складних проєктах.

Zend Framework – це ще один потужний інструмент, який активно використовується у корпоративному середовищі. Він пропонує надзвичайно високу модульність і масштабованість, але водночас має досить складну структуру та високу криву навчання.

На фоні цих фреймворків Laravel виглядає найбільш збалансованим рішенням. Він поєднує в собі простоту використання, велику кількість вбудованих інструментів, активну спільноту, чудову документацію та багату екосистему додаткових пакетів. Завдяки цьому Laravel завойовує все більше прихильників серед розробників у всьому світі.

1.2 Основні переваги та особливості Laravel

Laravel, як сучасний фреймворк, пропонує цілий спектр можливостей для побудови повнофункціонального вебдодатку. Його філософія базується на простоті, елегантності та зрозумілості коду. Фреймворк надає чітку структуру проєкту, де логіка, відображення та обробка даних розділені між собою відповідно до принципу MVC (Model-View-Controller). Це дозволяє легко підтримувати, розширювати і масштабувати проєкт в майбутньому.

Однією з найсильніших сторін Laravel є Eloquent ORM – інструмент для зручної роботи з базами даних, що дозволяє здійснювати запити у вигляді об’єктно-орієнтованого коду. Завдяки цьому розробник може уникати ручного написання SQL-запитів, що не лише пришвидшує розробку, але й знижує ризик помилок.

Laravel також має вбудовану систему маршрутизації, яка дозволяє легко організовувати навігацію в межах сайту. Blade – власний шаблонізатор Laravel – дозволяє створювати динамічні вебсторінки, зберігаючи при цьому читабельність та простоту шаблонів [13].

Не менш важливим є інструмент Artisan – консольна утиліта, яка дозволяє автоматизувати рутинні завдання: створення моделей, контролерів, запуск міграцій, створення сідерів, тестів тощо. Це значно спрощує та пришвидшує процес розробки, особливо у великих проєктах.

Фреймворк також має високий рівень безпеки. Він підтримує механізми захисту від типових вебзагроз, таких як SQL-ін’єкції, міжсайтові підробки запитів (CSRF), міжсайтові скриптові атаки (XSS) тощо.

Екосистема Laravel надзвичайно розвинена: існує безліч готових пакетів, які можна легко підключити до проєкту для розширення його функціональності – наприклад, для реалізації авторизації, платежів, черг, email-повідомлень тощо. Окрім того, фреймворк підтримує роботу з REST API, що дозволяє створювати бекенд для мобільних додатків чи SPA.

У підсумку, Laravel – це не просто фреймворк, а повноцінне середовище для ефективної та швидкої розробки сучасних вебдодатків, яке надає все необхідне для професійного створення інтернет-магазину.

1.3 Обґрунтування вибору Laravel для реалізації проєкту

Сучасний веброзробник стоїть перед широким вибором інструментів для реалізації власного програмного продукту. У випадку розробки повноцінного інтернет-магазину, до якого висуваються високі вимоги щодо стабільності, швидкодії, масштабованості та безпеки, вибір фреймворку стає одним із найважливіших рішень на етапі початкового проєктування. У цьому контексті доцільно орієнтуватися не лише на технічні характеристики системи, але й на наявність підтримки, темпи розвитку, відповідність сучасним тенденціям і легкість інтеграції з іншими технологіями.

Laravel, як сучасний PHP-фреймворк, пропонує рішення, що одночасно відповідають цим критеріям. Він надає широкий набір інструментів «із коробки», які дозволяють не витрачати час на розробку типових

функціональностей: аутентифікації, маршрутизації, роботи з базами даних, відправлення email-повідомлень тощо. Крім того, Laravel підтримує роботу з RESTful API та GraphQL, що є необхідною умовою для сучасної багатокомпонентної архітектури.

Особливу увагу слід приділити системі шаблонів Blade, яка дає змогу будувати складні, але зручні у супроводі інтерфейси [14]. Інструмент Artisan забезпечує автоматизацію багатьох рутинних дій, а Eloquent ORM спрощує взаємодію з базами даних, перетворюючи SQL-запити у зрозумілий програмісту об’єктно-орієнтований стиль.

Також варто зазначити, що Laravel має дуже активну спільноту та чудово структуровану документацію, що робить його ідеальним як для індивідуальних розробників, так і для командної роботи. Це дозволяє суттєво зекономити час на навчання, а також значно знижує ймовірність виникнення критичних помилок.

З огляду на вищезазначене, вибір Laravel є раціональним та виправданим як з технічної, так і з організаційної точки зору. Це фреймворк, який забезпечує не лише інструменти, а й філософію розробки, яка сприяє створенню ефективних, зручних для користувача та безпечних вебрішень будь-якої складності.

1.4 Порівняльна характеристика Laravel і альтернатив

Вибір технологічного інструменту для реалізації вебдодатку завжди потребує ретельного аналізу можливих альтернатив, особливо коли йдеться про фреймворки, на основі яких будуватиметься архітектура усього проєкту. Саме тому порівняння Laravel з іншими популярними PHP-фреймворками є необхідним етапом дослідження, який дозволяє не лише обґрунтувати вибір, а й усвідомити переваги та потенційні обмеження, пов’язані з кожним варіантом.

Зокрема, Symfony давно зарекомендував себе як надзвичайно гнучкий і потужний фреймворк, який активно використовується в складних та великомасштабних корпоративних проєктах. Проте його основним недоліком вважається досить складна структура та високий поріг входження. Для початківців чи невеликих команд Symfony може бути занадто важким для освоєння. Також варто зазначити, що Symfony вимагає більше часу на первинне налаштування проєкту, а також часто вимагає ручного налаштування багатьох аспектів, що автоматично доступні в Laravel.

Yii, у свою чергу, позиціонується як фреймворк з високою продуктивністю, який ідеально підходить для швидкої розробки. Завдяки генераторам коду та простому API, Yii має низький поріг входження. Проте з огляду на меншу спільноту та обмежену кількість доступних розширень, Yii поступається Laravel в контексті довготривалих, масштабованих рішень, особливо там, де йдеться про складні взаємозв’язки між модулями, активне використання API, чи необхідність побудови систем з багаторівневим доступом.

CodeIgniter можна назвати «старожилом» серед PHP-фреймворків. Його основною перевагою є простота. CodeIgniter дозволяє досить швидко почати розробку, не потребуючи складної конфігурації. Але така простота є водночас і його слабкою стороною. У ньому відсутні деякі сучасні інструменти, які вважаються стандартом у Laravel – наприклад, ORM, шаблонізація з підтримкою компонентів, вбудовані засоби тестування тощо. Тому для великих проєктів, особливо пов’язаних з інтернет-торгівлею, CodeIgniter часто виявляється недостатньо функціональним.

Zend Framework – ще один потужний інструмент, що орієнтований на корпоративний сектор. Він вирізняється модульністю, але водночас має складну структуру, яка вимагає високої кваліфікації розробників і значних ресурсів. Також Zend має менше безкоштовних публічних матеріалів, гірше структуровану документацію порівняно з Laravel, що ускладнює його використання без комерційної підтримки.

Laravel вигідно вирізняється на фоні вищезгаданих фреймворків. Він поєднує в собі простоту і гнучкість, підтримує сучасні підходи до розробки, має потужну екосистему, численні пакети, регулярні оновлення та активну спільноту. Його структура дозволяє як швидко створити MVP, так і масштабувати додаток до рівня великої системи. Крім того, Laravel ідеально підходить для інтеграції з фронтенд-фреймворками та побудови SPA-додатків, що є додатковою перевагою у сучасному світі веброзробки.

Для візуалізації порівняння було побудовано діаграму (рис. 1.1), на якій зображено основні характеристики кожного з фреймворків за критеріями популярності, активності спільноти та простоти використання [11].

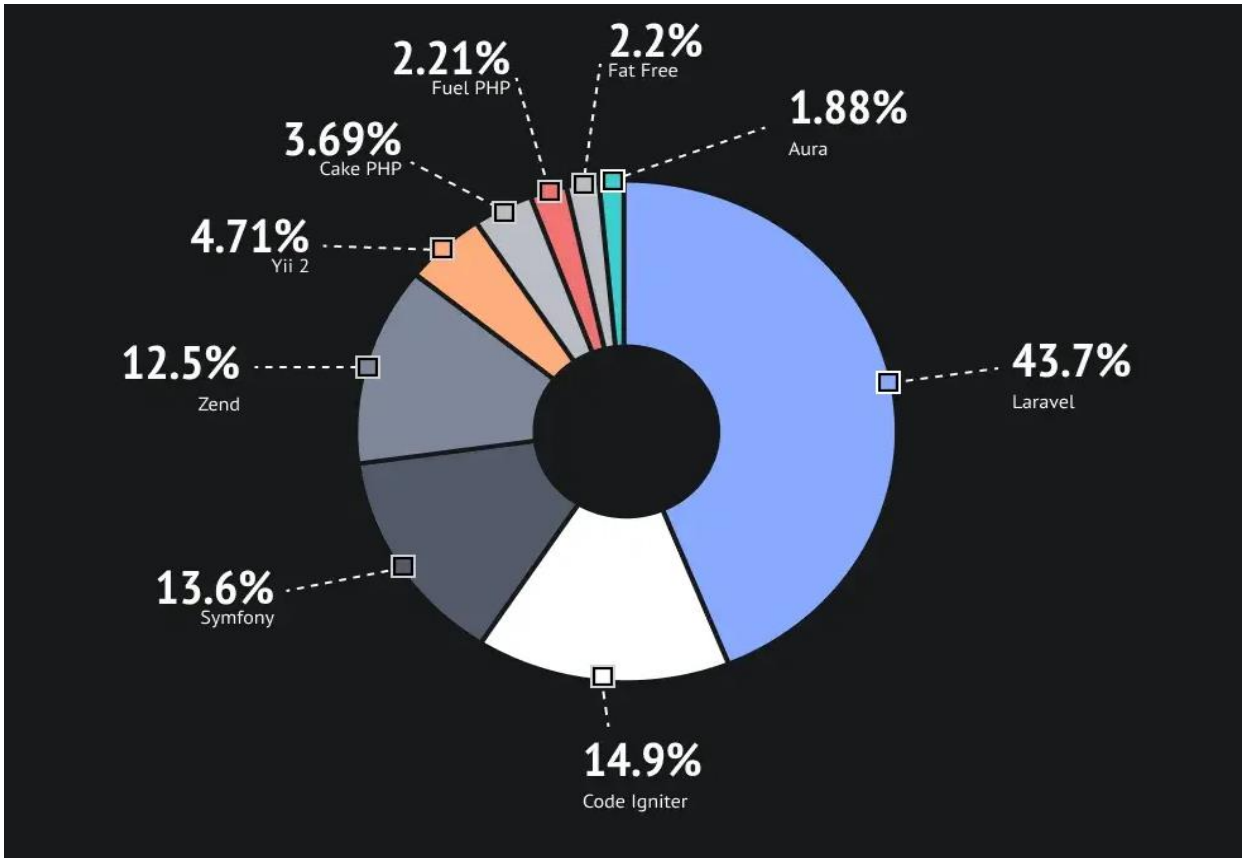


Рисунок 1.1 – Популярність PHP-фреймворків у 2023 році [11]

Як видно з діаграми, Laravel лідирує за всіма трьома показниками, що ще раз підтверджує його доцільність для розробки вебплатформи інтернет-магазину.

1.5 Перспективи використання Laravel у подальшій розробці

У сфері веброзробки важливу роль відіграє не лише початковий функціонал обраного фреймворку, але й його здатність адаптуватися до нових вимог, викликів та технологічних трендів. Саме тому перспективність використання фреймворку у довгостроковому періоді є ключовим фактором при прийнятті рішення щодо його впровадження в конкретному проєкті.

Laravel – один з небагатьох PHP-фреймворків, який постійно оновлюється, розвивається та отримує підтримку з боку авторів, офіційних партнерів і величезної міжнародної спільноти. Кожна нова версія Laravel не лише виправляє помилки чи оптимізує швидкодію, а й привносить інновації, які відповідають сучасним підходам до архітектури, безпеки, фронтенд-взаємодії та тестування.

Особливої уваги заслуговує Laravel Vapor – хмарна платформа для безсерверного розгортання додатків, побудована на основі інфраструктури Amazon Web Services (AWS). Вона дозволяє автоматично масштабувати навантаження, не переймаючись технічними аспектами адміністрування серверів. Це ідеальне рішення для великих проєктів з високими вимогами до продуктивності.

Іншою важливою перспективою є інтеграція з сучасними бібліотеками для створення інтерфейсів: Vue.js [8], React, Alpine.js. Laravel активно підтримує frontend-орієнтовані підходи, що дозволяє створювати SPA-додатки, реактивні компоненти, Livewire-інтерфейси з динамічним оновленням сторінки без перезавантаження.

Слід також відзначити наявність офіційного розширення Laravel Breeze, Jetstream, Nova, що забезпечують швидке створення проєктів з розширеним функціоналом – від автентифікації до адміністративних панелей.

Усі ці можливості роблять Laravel не просто технологією «на зараз», а платформою, яка дозволяє впевнено планувати розвиток проєкту на роки вперед.

1.6 Актуальність теми дослідження

Сучасні економічні умови, посилена цифровізація та глобальна конкуренція змушують бізнес шукати нові ефективні шляхи взаємодії з клієнтами. Інтернет-торгівля стала не просто зручною альтернативою традиційним форматам продажу, а повноцінною та домінуючою моделлю ведення бізнесу. У зв’язку з цим, створення інтернет-магазину перестало бути лише бажаним елементом – це необхідність для будь-якого підприємства, що прагне залишатися конкурентоспроможним.

Однак, попит на вебплатформи супроводжується і підвищеними вимогами до їх якості. Покупець очікує швидкої взаємодії, інтуїтивного інтерфейсу, високого рівня безпеки та адаптивності до мобільних пристроїв. Це означає, що створення інтернет-магазину сьогодні – це не просто розробка сайту, а комплексний технологічний виклик, що вимагає застосування передових підходів.

Вибір фреймворку Laravel як основи розробки такого магазину забезпечує відповідність цим очікуванням: він дозволяє швидко реалізувати сучасний функціонал, забезпечити безпеку та зручність для користувача, а також легко масштабувати рішення в майбутньому.

Таким чином, дослідження технологічних можливостей Laravel у контексті створення вебмагазину є не лише теоретично обґрунтованим, а й практично необхідним у сучасних умовах динамічного розвитку цифрового ринку.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Аналіз предметної області

У сучасних умовах цифрової трансформації бізнес-процесів стрімко зростає потреба у впровадженні ефективних засобів електронної комерції. Онлайн-магазини виступають не лише як зручний канал продажу продукції, а й як повноцінна платформа для побудови взаємодії між продавцем і покупцем, формування бренду, збору аналітичних даних і підвищення лояльності клієнтів [15].

Предметна область, яку охоплює дана дипломна робота – це розробка вебзастосунку для реалізації функціонального інтернет-магазину на базі сучасного технологічного стека. Така система має забезпечувати зручну навігацію по товарах, категоріях, здійснення покупки, керування кошиком, відображення статусу замовлення та інші супутні функції, притаманні типовому e-commerce-рішенню.

Основними учасниками системи є (див. рис. 2.1):

– незареєстрований користувач – має доступ до загального каталогу товарів, фільтрів та перегляду детальної інформації;
– зареєстрований покупець – отримує додаткову функціональність: додавання товарів у кошик, перегляд замовлень, оформлення покупки;
– адміністратор – здійснює управління категоріями, товарами, переглядає покинуті кошики, обробляє замовлення.

У процесі аналізу предметної області слід також враховувати специфіку бізнес-логіки, закладеної в архітектуру системи. На відміну від простих вітрин або лендингів, інтернет-магазин передбачає збереження великої кількості динамічних даних, зокрема:

– інформації про товари (назва, опис, ціна, зображення, категорія);
– відомостей про користувачів і їх замовлення;
– збереження стану кошика, навіть для неавторизованого відвідувача;
– фіксації часу, статусів, кількості тощо.

Для побудови такої системи обрана архітектура, що відповідає шаблону Model-View-Controller (MVC). Такий підхід дозволяє ефективно розділити логіку бізнес-процесів, обробку запитів та відображення інтерфейсу, що позитивно впливає на підтримуваність, масштабованість і зрозумілість коду.



Рисунок 2.1 – Діаграма прецедентів взаємодії користувачів з системою інтернет-магазину

2.2 Визначення вимог до системи

Успішна реалізація будь-якої інформаційної системи безпосередньо залежить від точності формулювання вимог до її функціонування. Саме на цьому етапі визначаються основні можливості, які система повинна реалізовувати, а також параметри, що забезпечують її надійність, продуктивність і відповідність очікуванням кінцевого користувача.

Вимоги до інформаційної системи умовно поділяються на **функціональні** та **нефункціональні**. Перші описують, що система має робити, другі – як вона це повинна реалізовувати з точки зору якості, ефективності, зручності та безпеки.

Функціональні вимоги:

а) *реєстрація та аутентифікація користувача*:
- можливість створення облікового запису;
- вхід до системи з валідацією даних;
- відновлення паролю;

б) *огляд товарів*:
- каталог з можливістю фільтрації за категоріями;
- відображення картки товару з описом, фото, ціною;
- пошук товарів за ключовими словами;

в) *робота з кошиком*:
- додавання/видалення товарів у кошик;
- редагування кількості;
- збереження кошика для гостьового користувача;

г) *оформлення замовлення*:
- форма оформлення із зазначенням контактних даних;
- валідація введених полів;
- відображення підтвердження замовлення;

д) *адміністративна панель*:
- керування товарами (додавання, редагування, видалення);
- керування категоріями;

- перегляд покинутих кошиків;
- перегляд замовлень.

Нефункціональні вимоги:

а) *продуктивність*:
- час завантаження сторінок не повинен перевищувати 2 секунд;
- підтримка одночасної роботи щонайменше 20 користувачів без зниження швидкодії;

б) *безпека*:
- захист від CSRF-атак, SQL-ін’єкцій, XSS;
- шифрування паролів з використанням bcrypt;
- перевірка прав доступу до адмін-панелі;

в) *зручність користування*:
- інтерфейс має бути адаптивним для мобільних пристроїв;
- мінімальна кількість кліків до основних функцій (товари, кошик, замовлення);

г) *масштабованість*:
- можливість розширення функціоналу без радикальної зміни архітектури;
- можливість підключення сторонніх сервісів (платіжних систем, API логістики тощо);

д) *підтримка української мови*:
- вся система інтерфейсно реалізована українською мовою.

2.3 Архітектура системи

Одним із ключових етапів розробки будь-якої інформаційної системи є побудова її архітектури. Вона визначає логіку функціонування проєкту, розподіл обов’язків між компонентами, взаємодію з базою даних, клієнтською частиною та зовнішніми сервісами. Правильно спроєктована архітектура

забезпечує не лише зручність супроводу, а й гнучкість при масштабуванні та розширенні функціоналу (див. рис. 2.2).



Рисунок 2.2 – Архітектура вебсистеми інтернет-магазину

Основою архітектури даного інтернет-магазину є **патерн MVC (Model-View-Controller)**. Він дозволяє чітко розділити логіку представлення, управління та обробки даних у проєкті:

- model (модель) – взаємодіє з базою даних, відповідає за зберігання та обробку інформації. У нашому проєкті це моделі Product, Category, User, Order, CartItem;
- view (представлення) – відповідає за формування інтерфейсу для користувача (всі шаблони реалізовано за допомогою Blade);
- controller (контролер) – приймає запити користувача, обробляє їх та формує відповідь, використовуючи відповідні моделі та шаблони.

Крім стандартної архітектури MVC, проєкт також використовує **Livewire-компоненти** для динамічної взаємодії без повного оновлення сторінки, що підвищує UX (user experience) сайту. Функціонально система складається з наступних взаємопов’язаних блоків:

- головна сторінка (вітрина товарів) – відображає товари з підтримкою фільтрів;
- картка товару – містить опис, зображення, ціну та кнопку додавання до кошика;
- кошик – список обраних товарів, динамічне оновлення кількості, видалення;
- оформлення замовлення – форма введення контактних даних і підтвердження;
- панель адміністратора – управління товарами, категоріями, перегляд замовлень і покинутих кошиків.

Система побудована з урахуванням **REST-підходу**: кожна дія (отримання, створення, редагування, видалення) реалізована у відповідному маршруті та контролері.

Усі маршрути згруповані за функціональністю (гостьові, користувацькі, адмінські).

2.4 Вибір технологій та інструментів

Для створення сучасної, масштабованої та зручно підтримуваної інформаційної системи необхідно підібрати відповідний стек технологій, який задовольняє вимоги продуктивності, безпеки, гнучкості та швидкості розробки.

У межах даного проєкту було обрано набір перевірених засобів розробки з відкритим вихідним кодом, які активно використовуються в галузі вебпрограмування.

2.4.1 Серверна частина

Основою розробки є фреймворк **Laravel 11**, який реалізовано мовою програмування **PHP 8.2**. Laravel був обраний через його відповідність сучасним стандартам розробки, потужну екосистему, наявність великої спільноти, зручну документацію та вбудовані інструменти, такі як Blade, Artisan, Eloquent ORM, Livewire, а також підтримку REST API.

Для локальної розробки використовувався пакет **XAMPP**, який забезпечує запуск Apache-сервера та сервера баз даних MySQL у локальному середовищі.

2.4.2 Клієнтська частина

Інтерфейс користувача реалізовано із використанням шаблонізатора **Blade**, що є вбудованим у Laravel. Для розробки реактивних компонентів та динамічного оновлення вмісту сторінок застосовано **Livewire** [3] – бібліотеку, яка дозволяє створювати сучасний інтерфейс без написання окремого JavaScript-коду [4].

Стилістика сторінок реалізована за допомогою CSS-фреймворку **Tailwind CSS**, що забезпечує високу гнучкість та адаптивність при верстці [2].

2.4.3 База даних

У якості СКБД обрано MySQL – потужну реляційну базу даних, яка чудово інтегрується з Laravel через ORM Eloquent. Управління схемою бази даних здійснюється через міграції, які дозволяють контролювати зміну структури таблиць версіями та гарантують узгодженість структури на всіх етапах розробки.

2.4.4 Додаткові інструменти

Додатковими інструментами стали:

- Composer – менеджер пакетів PHP для завантаження залежностей проєкту [5];
- NPM + Vite – для збирання фронтенд-ресурсів;
- Git / GitHub – система контролю версій, що використовувалась для зберігання та відстеження змін у коді [7];
- PHPMyAdmin – графічна оболонка для роботи з MySQL, яка використовувалась для перевірки вмісту таблиць та тестування SQL-запитів.

2.4.5 Обґрунтування вибору

Усі обрані технології є відкритими, безкоштовними, мають широку документацію і спільноти. Вони дозволяють швидко реалізовувати функціональні модулі, легко інтегруються між собою та забезпечують масштабовану, безпечну платформу для розробки інтернет-магазину [16].

2.5 Моделювання системи

Етап моделювання є одним з найбільш важливих у процесі проєктування інформаційної системи. Він дозволяє створити уявлення про логіку роботи системи, її структуру, взаємозв’язки між компонентами та ролями користувачів ще до початку безпосередньої розробки. Моделювання слугує своєрідною візуальною документацією, яка дає змогу узагальнено побачити ключові аспекти функціонування системи, виявити потенційні проблеми та оптимізувати структуру ще до її реалізації в коді.

У рамках розробки вебсистеми інтернет-магазину було використано уніфіковану мову моделювання UML (Unified Modeling Language), яка є загальноприйнятим стандартом у сфері проєктування програмного забезпечення. UML дозволяє створювати формалізовані діаграми, що відображають як функціональну поведінку системи (прецеденти, послідовності), так і її статичну структуру (класи, компоненти, об’єкти тощо).

У цьому підрозділі наведено основні типи діаграм, які використовувалися для моделювання логіки інтернет-магазину.

2.5.1 Діаграма прецедентів

Діаграма прецедентів (див. рис. 2.1) дозволяє описати взаємодію зовнішніх акторів (користувачів) із системою. Вона є фундаментальним інструментом для початкового розуміння функціональності системи з точки зору кінцевого користувача. У рамках моделювання було виокремлено три типи акторів: гість, зареєстрований користувач та адміністратор.

Ці ролі реалізують наступні прецеденти: перегляд каталогу товарів, додавання товарів до кошика, оформлення замовлення, керування товарами, аналіз покинутих кошиків тощо. Завдяки цій діаграмі легко визначити, які функції мають бути реалізовані для кожного типу користувача, та забезпечити відповідну диференціацію прав доступу.

2.5.2 Діаграма класів

Для моделювання об’єктної структури системи побудовано діаграму класів (див. рис. 2.3), яка відображає ключові сутності проєкту та зв’язки між ними. Ця діаграма відповідає реалізованим у Laravel Eloquent-моделям: Product, Category, User, Order, CartItem. Зв’язки між класами показують

відношення «один-до-багатьох» або «багато-до-одного», що формують логіку обробки даних.

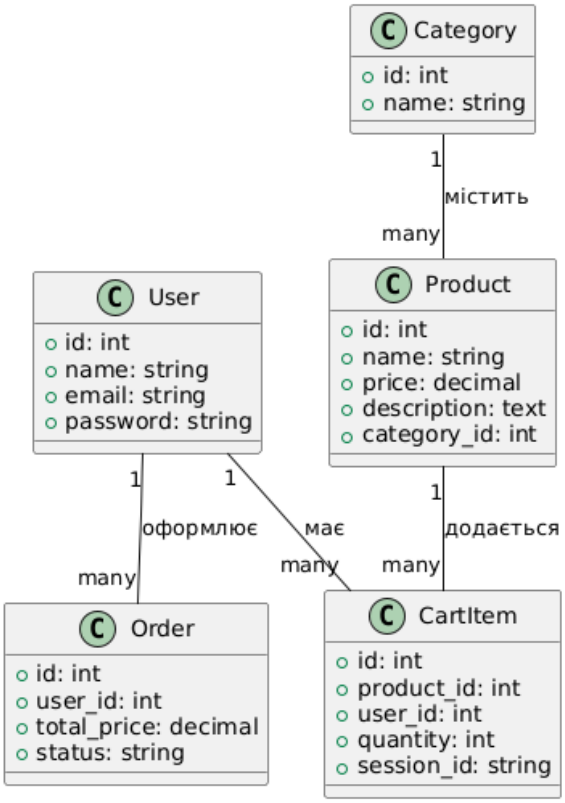


Рисунок 2.3 – Діаграма класів моделей інтернет-магазину

Клас User пов’язаний із замовленнями (Order) та записами в кошику (CartItem), Product належить до певної Category, а CartItem поєднує User Product. Такий підхід дозволяє побудувати гнучку, реляційну модель з можливістю ефективної фільтрації та пошуку інформації. Ця діаграма слугує своєрідною мапою, що дозволяє швидко зорієнтуватися в структурі бази даних і логіці взаємодії моделей.

2.5.3 Діаграма послідовності

Додатково, для візуалізації динамічної поведінки системи, створено діаграму послідовності (див. рис. 2.4). На ній змодельовано один із найважливіших сценаріїв – додавання товару до кошика.

Цей сценарій охоплює взаємодію між інтерфейсом користувача, Livewire-компонентом, контролером, моделлю CartItem та базою даних.

Користувач ініціює дію на фронтенді, після чого запит передається через Livewire до контролера, який здійснює обробку логіки та оновлення відповідної моделі. Далі інформація зберігається у базі даних, а користувач отримує зворотній зв’язок без перезавантаження сторінки.

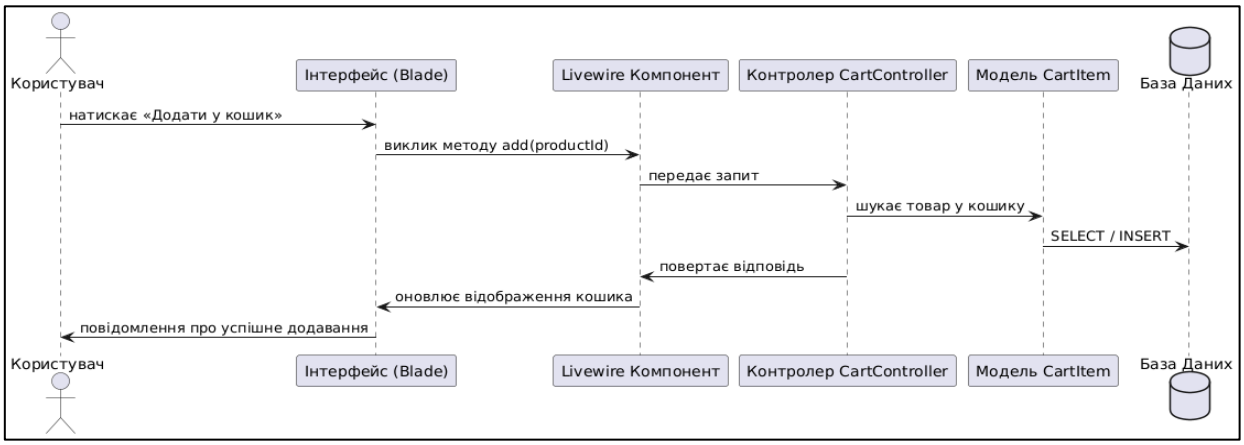


Рисунок 2.4 – Діаграма послідовності «Додати товар у кошик»

Ця діаграма дозволяє глибше зрозуміти, як саме функціонують окремі компоненти системи у процесі взаємодії з користувачем у реальному часі.

Узагальнюючи, зазначимо, що моделювання дозволило сформувати чітке уявлення про логіку проєкту, і стало міцною основою для його технічної реалізації. Структуровані діаграми полегшили як процес розробки, так і подальшу підтримку та тестування системи.

2.6 Організація та структура бази даних

Вебсистема інтернет-магазину, як і більшість сучасних вебдодатків, базується на роботі з даними, які потребують ефективного зберігання, обробки та взаємозв’язку. У цьому контексті розробка правильної структури бази даних є критичним завданням, що забезпечує узгодженість, цілісність та продуктивність функціонування всієї системи.

Для реалізації реляційної бази даних у межах цього проєкту було використано **MySQL**, яка повністю інтегрується з фреймворком Laravel за допомогою ORM Eloquent та системи міграцій. Архітектура бази даних реалізована відповідно до вимог предметної області та логіки взаємодії компонентів вебзастосунку.

2.6.1 ER-діаграма бази даних

Для наочного уявлення про структуру бази даних розроблено **ER-діаграму** (див. рис. 2.5), що відображає всі основні сутності, їх атрибути та зв’язки. Це дозволяє узагальнити структуру даних і підтвердити правильність спроєктованої логіки.

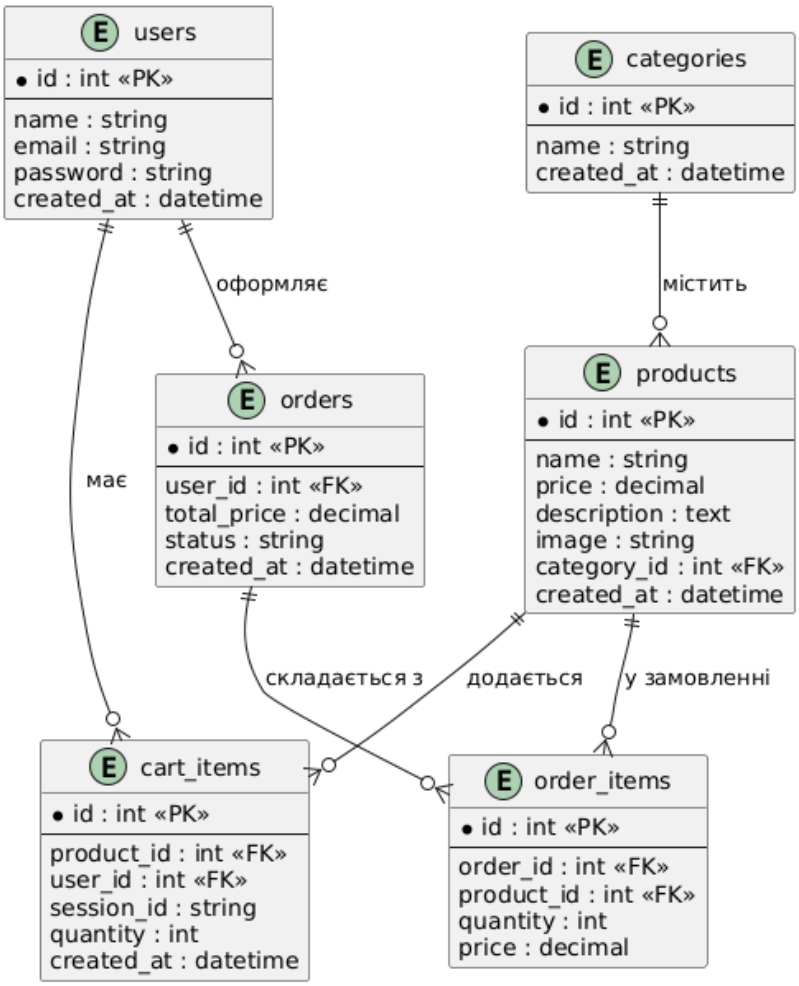


Рисунок 2.5 – ER-діаграма бази даних інтернет-магазину

2.6.2 Основні таблиці бази даних

Ці таблиці пов’язані між собою через **зовнішні ключі** відповідно до логіки «один-до-багатьох» та «багато-до-одного» (див. табл. 2.1, рис. 2.6 – 2.11). Наприклад:

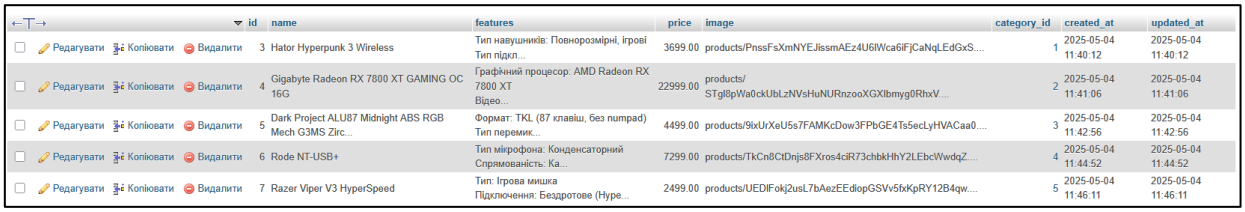
– кожен товар (product) належить до однієї категорії;

– один користувач може мати багато замовлень;

– один кошик (cart_items) пов’язаний із певним товаром та користувачем (або session_id).

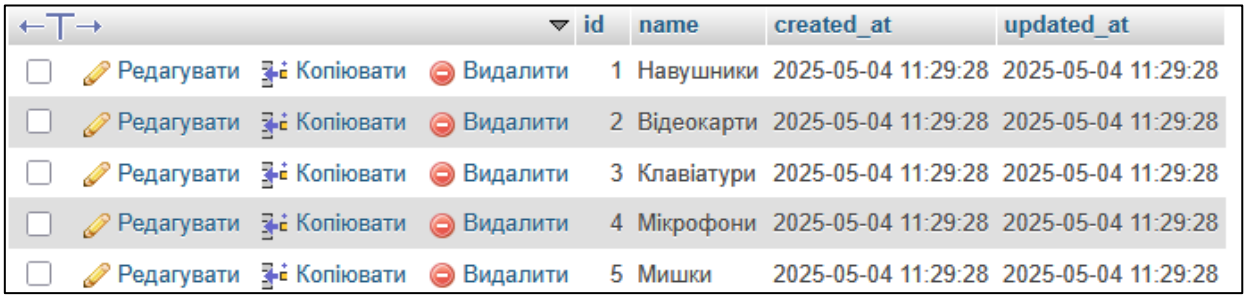
Таблиця 2.1 – Таблиці бази даних інформаційної системи інтернет-магазину

Назва таблиці	Призначення
users	Зберігає дані про зареєстрованих користувачів
products	Містить інформацію про товари (назва, ціна, опис, фото)
categories	Категорії товарів для фільтрації та структурування
cart_items	Містить записи товарів, доданих до кошика користувача
orders	Таблиця оформлених замовлень
order_items	Проміжна таблиця для зв’язку замовлення з товарами



	id	name	features	price	image	category_id	created_at	updated_at
Редагувати Копіювати Видалити	3	Hator Hyperpunk 3 Wireless	Тип навушників: Повнорозмірні, ігрові Тип підкл...	3699.00	products/PnssFsXmNYEJissmAEz4U6lWca6iFjCaNqLEdGxS....	1	2025-05-04 11:40:12	2025-05-04 11:40:12
Редагувати Копіювати Видалити	4	Gigabyte Radeon RX 7800 XT GAMING OC 16G	Графічний процесор: AMD Radeon RX 7800 XT Відео...	22999.00	products/STgl8pWa0ckUbLzNVsHuNURnzooXGXlbmyg0RhxV....	2	2025-05-04 11:41:06	2025-05-04 11:41:06
Редагувати Копіювати Видалити	5	Dark Project ALU87 Midnight ABS RGB Mech G3MS Zirc...	Формат: TKL (87 клавіш, без numpad) Тип перемик...	4499.00	products/9lxUrXeU5s7FAMKcDow3FPbGE4Ts5ecLyHVACaa0....	3	2025-05-04 11:42:56	2025-05-04 11:42:56
Редагувати Копіювати Видалити	6	Rode NT-USB+	Тип мікрофона: Конденсаторний Спрямованість: Ка...	7299.00	products/TkCn8CtDnjs8FXros4ciR73chbkHhY2LEbcWwdqZ....	4	2025-05-04 11:44:52	2025-05-04 11:44:52
Редагувати Копіювати Видалити	7	Razer Viper V3 HyperSpeed	Тип: Ігрова мишка Підключення: Бездротове (Hype...	2499.00	products/UEDlFokj2usL7bAezEEdiopGSVv5fxKpRY12B4qw....	5	2025-05-04 11:46:11	2025-05-04 11:46:11

Рисунок 2.6 – Структура таблиці products у phpMyAdmin



	id	name	created_at	updated_at
Редагувати Копіювати Видалити	1	Навушники	2025-05-04 11:29:28	2025-05-04 11:29:28
Редагувати Копіювати Видалити	2	Відеокарти	2025-05-04 11:29:28	2025-05-04 11:29:28
Редагувати Копіювати Видалити	3	Клавіатури	2025-05-04 11:29:28	2025-05-04 11:29:28
Редагувати Копіювати Видалити	4	Мікрофони	2025-05-04 11:29:28	2025-05-04 11:29:28
Редагувати Копіювати Видалити	5	Мишки	2025-05-04 11:29:28	2025-05-04 11:29:28

Рисунок 2.7 – Структура таблиці categories у phpMyAdmin



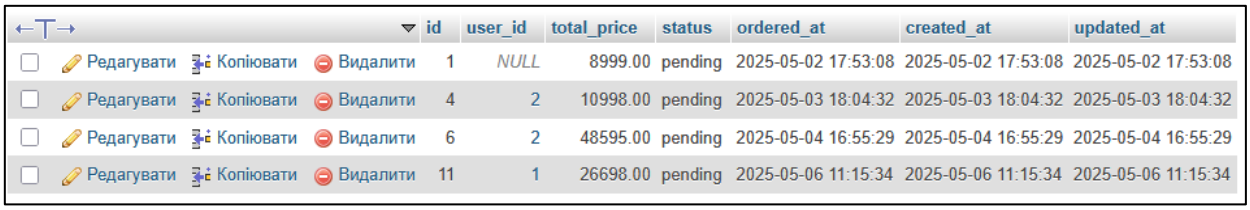
	id	name	email	email_verified_at	password	is_admin	remember_token	created_at	updated_at
Редагувати Копіювати Видалити	1	Test-User	maxsheva04@gmail.com	NULL	$2y$12$nf8DlpXflys9u/euqGlez.dvxNSZcorjzUFK76r9Zu0...	1	li24vs2anJ3urPulpEfxQ6MMuoF5jm7OOVBEuTmoAfFRHATAEq...	2025-05-02 18:06:00	2025-05-02 18:06:00
Редагувати Копіювати Видалити	2	Test-User1	sp1resh0w@gmail.com	NULL	$2y$12$0qdwV3rZEhKZG/PdkrryXOrburLoqpiCk5VyGA.SkQJ...	0	NULL	2025-05-03 17:57:41	2025-05-03 17:57:41

Рисунок 2.8 – Структура таблиці users у phpMyAdmin



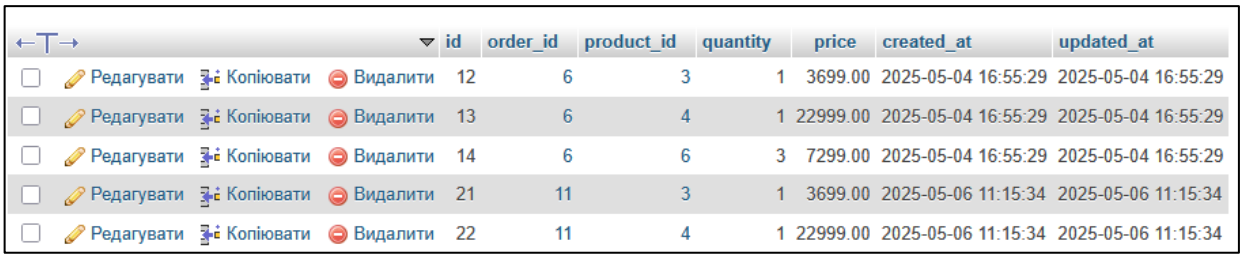
	id	user_id	product_id	quantity	session_id	created_at	updated_at
Редагувати Копіювати Видалити	119	NULL	3	1	ImZn4xtK1xmEdG8un6fev0NnLSJE5D2k10t6XJ78	2025-05-06 11:14:49	2025-05-06 11:14:49
Редагувати Копіювати Видалити	120	NULL	7	1	ImZn4xtK1xmEdG8un6fev0NnLSJE5D2k10t6XJ78	2025-05-06 11:15:00	2025-05-06 11:15:00
Редагувати Копіювати Видалити	121	NULL	4	1	ImZn4xtK1xmEdG8un6fev0NnLSJE5D2k10t6XJ78	2025-05-06 11:15:01	2025-05-06 11:15:01

Рисунок 2.9 – Структура таблиці cart_items у phpMyAdmin



	id	user_id	total_price	status	ordered_at	created_at	updated_at
Редагувати Копіювати Видалити	1	NULL	8999.00	pending	2025-05-02 17:53:08	2025-05-02 17:53:08	2025-05-02 17:53:08
Редагувати Копіювати Видалити	4	2	10998.00	pending	2025-05-03 18:04:32	2025-05-03 18:04:32	2025-05-03 18:04:32
Редагувати Копіювати Видалити	6	2	48595.00	pending	2025-05-04 16:55:29	2025-05-04 16:55:29	2025-05-04 16:55:29
Редагувати Копіювати Видалити	11	1	26698.00	pending	2025-05-06 11:15:34	2025-05-06 11:15:34	2025-05-06 11:15:34

Рисунок 2.10 – Структура таблиці orders у phpMyAdmin



	id	order_id	product_id	quantity	price	created_at	updated_at
Редагувати Копіювати Видалити	12	6	3	1	3699.00	2025-05-04 16:55:29	2025-05-04 16:55:29
Редагувати Копіювати Видалити	13	6	4	1	22999.00	2025-05-04 16:55:29	2025-05-04 16:55:29
Редагувати Копіювати Видалити	14	6	6	3	7299.00	2025-05-04 16:55:29	2025-05-04 16:55:29
Редагувати Копіювати Видалити	21	11	3	1	3699.00	2025-05-06 11:15:34	2025-05-06 11:15:34
Редагувати Копіювати Видалити	22	11	4	1	22999.00	2025-05-06 11:15:34	2025-05-06 11:15:34

Рисунок 2.11 – Структура таблиці order_items у phpMyAdmin

2.6.3 Логіка роботи таблиць у системі

Інформаційна система інтернет-магазину побудована на реляційній структурі бази даних, де кожна таблиця виконує визначену роль у реалізації бізнес-процесів. Всі таблиці поєднані за допомогою зовнішніх ключів у логічно пов’язану схему, що забезпечує повний життєвий цикл користувацької взаємодії – від перегляду товару до оформлення та обробки замовлення.

Формування товарного каталогу. Товари системи зберігаються у таблиці products, яка має зв’язок із таблицею categories через зовнішній ключ category_id. Це дозволяє логічно структурувати товари за категоріями

(наприклад: «навушники», «мікрофони», «відеокарти»), реалізовувати фільтрацію та сортування на фронтенді:

– при завантаженні головної сторінки системи відбувається вибірка всіх товарів з таблиці products;
– якщо користувач обирає певну категорію, запит фільтрується за значенням поля category_id.

Завдяки цьому механізму можлива гнучка категоризація товарів без дублювання даних.

Додавання товарів у кошик. Таблиця cart_items реалізує логіку збереження поточних намірів користувача щодо купівлі товару:

– коли гість або зареєстрований користувач натискає кнопку «Додати до кошика», система виконує перевірку:
  а) якщо такий запис для product_id + user_id (або session_id) вже існує – оновлюється кількість;
  б) якщо ні – створюється новий запис;
– для неавторизованого користувача використовується унікальний session_id, що дозволяє зберігати кошик до моменту входу в акаунт.

Цей механізм дозволяє не втрачати додані товари після авторизації, завдяки подальшому об’єднанню даних за user_id.

Оформлення замовлення. Оформлення замовлення – найважливіший процес у системі. Його реалізація охоплює роботу щонайменше трьох таблиць: orders, order_items, cart_items:

– після заповнення форми користувача система створює новий запис у таблиці orders, прив’язаний до user_id;
– далі за допомогою SELECT-запиту з cart_items вибираються всі позиції кошика;
– для кожного товару створюється запис у таблиці order_items, де фіксується: order_id, product_id, кількість (quantity), ціна на момент замовлення;
– після успішного створення замовлення всі записи з cart_items для цього користувача (або сесії) видаляються.

Завдяки цьому вся інформація про замовлення зберігається в історії, незалежно від подальших змін у товарах чи цінах.

Перегляд та обробка замовлень:

– зареєстрований користувач має доступ до особистого кабінету, де може переглянути історію оформлених ним замовлень (для цього формується вибірка з orders за user_id);
– для кожного замовлення виводяться пов’язані з ним товари через order_items;
– адміністратор має повний доступ до всіх замовлень у системі та може:
    а) змінювати їх статус (pending, confirmed, shipped, canceled);
    б) переглядати дати, суми, склад замовлень.

Керування товарами та категоріями:

– усі дії щодо наповнення каталогу – додавання, редагування, видалення товарів або категорій – реалізуються через взаємодію з таблицями products та categories;
– в рамках адміністративної панелі адміністратор має можливість керувати цими таблицями через інтерфейс, що побудований на основі CRUD-функцій.

Аналітика покинутих кошиків:

– дані в таблиці cart_items, які не перетворились у order_items (тобто товари не були оформлені як замовлення), зберігаються як покинуті кошики;
– система дозволяє адміністратору переглядати ці записи та здійснювати маркетингові дії (наприклад, надіслати email-нагадування, якщо користувач залишив кошик).

Переваги такої логіки побудови бази:

– розділення відповідальностей: кожна таблиця відповідає за окремий аспект функціонування системи;
– гнучкість: легко реалізувати нові функції (знижки, статуси доставок, нові типи користувачів);

– інтегрованість з Laravel: ORM Eloquent автоматично працює з такими зв’язками (hasMany, belongsTo, etc.).

2.7 Забезпечення безпеки та захисту даних у системі

Одним із ключових аспектів проєктування інформаційних систем, що взаємодіють із персональними даними користувачів, є реалізація належного рівня безпеки. Сучасні вебресурси, зокрема ті, що належать до класу електронної комерції, повинні забезпечувати надійний захист даних, стійкість до найпоширеніших загроз та розмежування прав доступу відповідно до ролей користувачів.

У процесі розробки інформаційної системи інтернет-магазину було впроваджено низку заходів щодо забезпечення безпеки. Для цього використано вбудовані механізми фреймворку Laravel, а також додаткові засоби на рівні архітектури системи та бази даних.

Захист облікових даних користувачів. Для автентифікації користувачів застосовано стандартну систему Laravel Auth, яка передбачає збереження паролів у хешованому вигляді (алгоритм bcrypt). Вхід до системи супроводжується серверною валідацією даних, що унеможливлює обхід форм через зміну параметрів на клієнтській стороні.

Захист від CSRF-атак. Усі форми, які використовують метод POST, автоматично включають CSRF-токен. Обробка запитів здійснюється з використанням middleware VerifyCsrfToken, що забезпечує перевірку відповідності токена до поточної сесії користувача.

Захист від SQL-ін’єкцій. Усі запити до бази даних реалізовані за допомогою ORM Eloquent, який автоматично застосовує механізми екранування введених даних. У випадках використання конструктора запитів Laravel або параметризованих SQL-запитів також забезпечується захист через зв’язування змінних, що виключає пряме підставлення небезпечних даних.

Захист від міжсайтових скриптових атак (XSS). Усі змінні, що виводяться у шаблонах Blade, автоматично екрануються. Зокрема, конструкція {{ $variable }} не дозволяє виводити HTML-код у явному вигляді. Таким чином, навіть якщо користувач спробує передати JavaScript-код у формі або через URL-параметр, система не відобразить його у вигляді виконуваного скрипта.

Розмежування прав доступу. Функціонал, доступний лише адміністраторам, захищено за допомогою проміжного програмного рівня (middleware), зокрема auth та спеціалізованого admin, що перевіряє роль поточного користувача. Таким чином, забезпечується поділ між публічним інтерфейсом і адміністративною частиною системи.

Захист сесій та обробка cookies. Сесії зберігаються у захищеному вигляді, використовується механізм шифрування cookies (EncryptCookies). Крім того, конфігураційні параметри дозволяють обмежити доступ до cookies лише з серверного боку (HTTP-only), що унеможливлює їх отримання через JavaScript-код.

Інші заходи безпеки:

- всі сторонні пакети оновлюються через Composer/NPM відповідно до актуальних версій;
- фреймворк за замовчуванням реалізує захищене логування подій;
- адміністративна частина сайту не містить публічного інтерфейсу та недоступна для неавторизованих користувачів;
- система не обробляє банківські або платіжні дані, що виключає додаткову сертифікацію відповідно до стандартів PCI DSS.

Отже, реалізовані в системі механізми захисту відповідають актуальним вимогам щодо безпеки вебзастосунків, а їх застосування дозволяє забезпечити конфіденційність, цілісність та доступність оброблюваних даних відповідно до міжнародних стандартів.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Загальний опис реалізації

Реалізація інформаційної системи інтернет-магазину виконувалась із використанням сучасного технологічного стеку, що включає фреймворк Laravel версії 11 [9], мову програмування PHP 8.2, систему управління базами даних MySQL, а також фронтенд-інструменти Tailwind CSS, Alpine.js та Livewire [10].

Система побудована у відповідності до архітектурного патерну **Model-View-Controller (MVC)**, що дозволяє розділити логіку обробки запитів, взаємодії з даними та формування інтерфейсу. Це сприяє зручності супроводу, масштабованості та забезпечує високу читабельність коду.

Проєкт містить такі ключові компоненти:

– публічна частина (доступна всім відвідувачам сайту) – реалізує перегляд каталогу товарів, фільтрацію за категоріями, пошук, картки товару, додавання до кошика, оформлення замовлення та форму авторизації;
– особистий кабінет користувача – надає змогу переглядати замовлення, керувати кошиком, зберігати обрані товари;
– адміністративна панель (доступна лише авторизованим адміністраторам) – містить інтерфейс для керування товарами, категоріями, замовленнями та перегляду покинутих кошиків.

Система реалізована з використанням стандартних можливостей Laravel, таких як:

– маршрутизація (routes/web.php) – визначає обробку запитів до сторінок сайту;
– контролери – обробляють логіку, взаємодіють з моделями і передають дані до шаблонів;

– blade-шаблонізатор – відповідає за динамічне формування інтерфейсу;
– ORM Eloquent – реалізує об’єктно-орієнтовану роботу з базою даних;
– міграції та сидери – використовуються для створення та наповнення структури БД;
– livewire – забезпечує реалізацію інтерактивного інтерфейсу без використання JavaScript.

Також було застосовано низку кастомних рішень, зокрема реалізація гостьового кошика з подальшим перенесенням у обліковий запис користувача, що підвищує зручність користування ресурсом і наближає його до повноцінних рішень, подібних до Prom.ua.

Реалізацію системи було здійснено поступово – від налаштування середовища до створення окремих компонентів, поступової інтеграції бізнес-логіки та побудови повноцінного інтерфейсу. Після завершення програмної частини було проведено локальне тестування функціоналу з боку користувача й адміністратора.

3.2 Реалізація функціоналу користувача

3.2.1 Перегляд товарів та категорій

На головній сторінці інтернет-магазину реалізовано відображення списку товарів, що представлені у системі. Кожен товар містить короткий опис, назву, ціну та зображення, а також кнопку перегляду або додавання до кошика. Для зручності користувача реалізовано фільтрацію за категоріями, яка дозволяє швидко знайти товари потрібного типу.

Категорії отримуються із таблиці categories та виводяться у вигляді інтерактивного списку з Livewire-компонента (див. рис. 3.1, 3.2). Перемикання категорій не потребує перезавантаження сторінки, що забезпечує зручність використання.

```
{{-- Сітка товарів --}}
  <div class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
    @foreach($products as $product)
      <div class="border rounded p-4 bg-white shadow-sm hover:shadow-md transition flex flex-col justify-between h-[460px]">
        <a href="{{ route('product.view', $product) }}">
          @if ($product->image)
            <img src="{{ asset('storage/' . $product->image) }}" alt="{{ $product->name }}"
              class="w-full h-40 object-contain mb-3 rounded bg-white">
          @else
            <div class="w-full h-40 bg-gray-100 flex items-center justify-center text-gray-400 rounded mb-3">
              Немає зображення
            </div>
          @endif
        </a>
        <a href="{{ route('product.view', $product) }}" class="text-lg font-semibold block hover:text-blue-600">
          {{ $product->name }}
        </a>
        <p class="mt-1 text-sm text-gray-500 font-medium leading-tight">
          {{ number_format($product->price, 2, '.', ' ') }}
          <span class="ml-0.5">₴</span>
        </p>
        <div class="mt-4">
          @livewire('add-to-cart', ['productId' => $product->id], key($product->id))
        </div>
      </div>
    @endforeach
  </div>
```

Рисунок 3.1 – Фрагмент коду Blade-шаблону, що відповідає за вивід товарів на головній сторінці

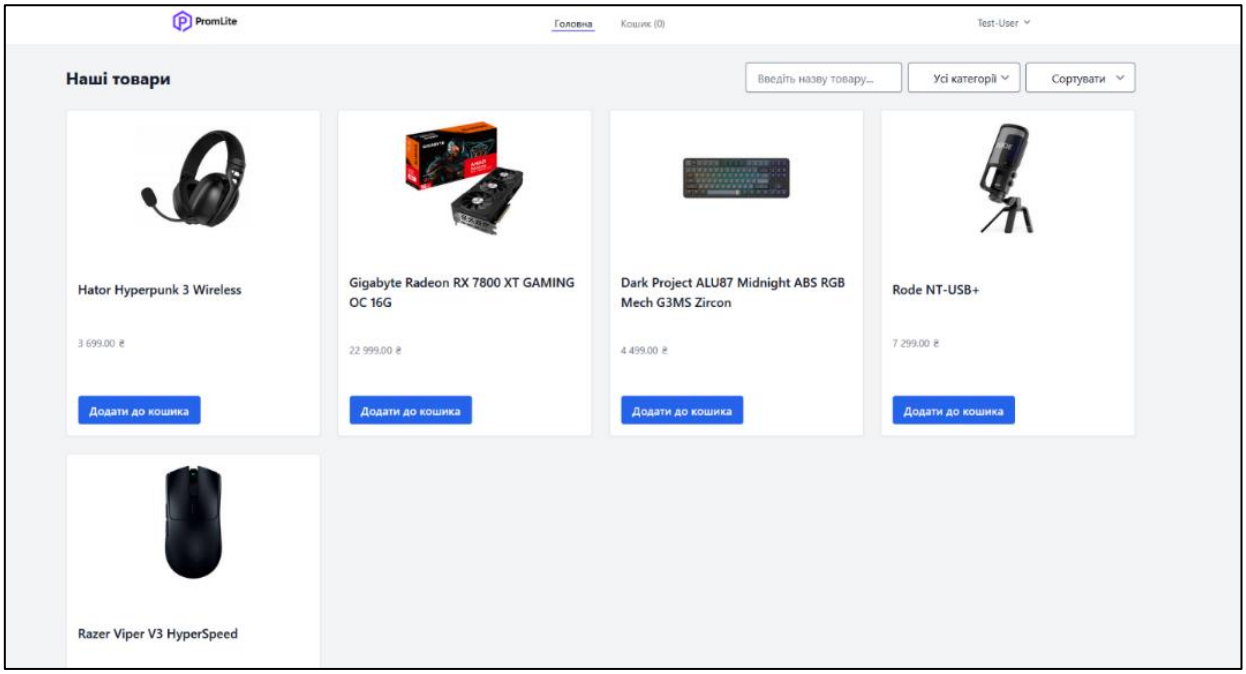


Рисунок 3.2 – Інтерфейс вітрини товарів із фільтром за категоріями

3.2.2 Детальна картка товару

При переході на сторінку окремого товару користувач отримує доступ до розширеної інформації про вибраний продукт. Цей функціонал реалізовано за маршрутом типу /product/{id}, який обробляється відповідним контролером, а відображення вмісту здійснюється через Blade-шаблон (див. рис. 3.3).

```
<div class="bg-gray-100 py-12">
  <div class="max-w-6xl mx-auto px-4 md:px-6 lg:px-8">
    <div class="bg-white rounded-lg shadow-sm p-6 md:p-10 flex flex-col md:flex-row gap-10 items-start">

      {{-- Зображення --}}
      <div class="w-full md:w-1/2 flex justify-center">
        @if($product->image)
          <img src="{{ asset('storage/' . $product->image) }}"
            alt="{{ $product->name }}"
            class="rounded max-h-[400px] object-contain bg-white p-4" />
        @else
          <div class="w-full h-80 bg-gray-100 flex items-center justify-center text-gray-400 rounded">
            Немає зображення
          </div>
        @endif
      </div>

      {{-- Інформація --}}
      <div class="w-full md:w-1/2 space-y-6">
        <h1 class="text-3xl font-semibold text-gray-900">{{ $product->name }}</h1>
        <ul class="text-gray-700 mb-6 space-y-1 list-disc list-inside">
          @foreach(explode("\n", $product->features) as $line)
          @if(trim($line) !== '')
          <li>{{ $line }}</li>
          @endif
        @endforeach
        </ul>

        <p class="text-lg font-medium text-gray-500">
          {{ number_format($product->price, 2, '.', '') }}
          <span class="ml-1 text-sm text-gray-400">₴</span>
        </p>
        {{-- Кнопка "Додати в кошик" --}}
        <div class="pt-4">
          @livewire('add-to-cart', ['productId' => $product->id])
        </div>
      </div>
    </div>
```

Рисунок 3.3 – Картка товару з кнопкою додавання до кошика

На сторінці товару відображаються: зображення продукту (якщо завантажене), назва товару, текстовий опис, ціна у гривнях, кнопка для

додавання у кошик із зазначенням кількості (через Livewire). Інтерфейс є адаптивним і передбачає коректне відображення як на десктопних, так і на мобільних пристроях (див. рис. 3.4).

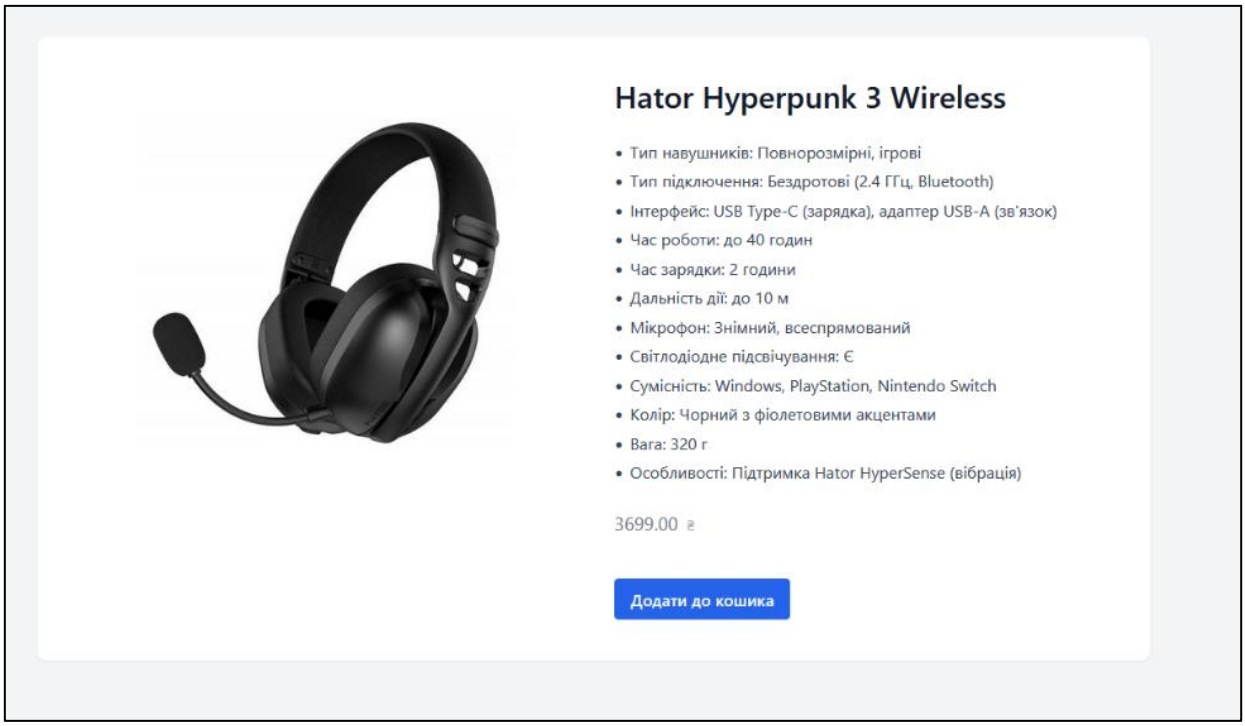


Рисунок 3.4 – Сторінка перегляду товару

3.2.3 Додавання товару до кошика

Функціонал додавання товару до кошика реалізовано з використанням Livewire-компонента AddToCart. Це дозволяє взаємодіяти з кошиком у реальному часі – без перезавантаження сторінки. Компонент працює як для авторизованих користувачів, так і для гостей (через збереження session_id) (див. рис. 3.5).

При натисканні кнопки «Додати до кошика» викликається метод add(), який:

– перевіряє, чи товар уже є у кошику;
– якщо так – оновлює кількість;
– якщо ні – створює новий запис у таблиці cart_items.

```php
<?php

namespace App\Livewire;

use Livewire\Component;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Session;
use App\Models\CartItem;
use App\Models\Product;

class AddToCart extends Component
{
    public $productId;

    public function add()
{
    $userId = Auth::id();
    $sessionId = Session::getId();

    $cartItem = CartItem::query()
        ->where('product_id', $this->productId)
        ->when($userId, fn($q) => $q->where('user_id', $userId))
        ->when(!$userId, fn($q) => $q->where('session_id', $sessionId))
        ->first();

    if ($cartItem) {
        $cartItem->increment('quantity');
    } else {
        CartItem::create([
            'product_id' => $this->productId,
            'quantity' => 1,
            'user_id' => $userId,
            'session_id' => $userId ? null : $sessionId,
        ]);
    }

    $this->dispatch('flash-message', message: 'Товар додано до кошика!');
    $this->dispatch('cartUpdated');
}

    public function render()
    {
        $product = Product::findOrFail($this->productId);
        return view('livewire.add-to-cart', compact('product'));
    }
}
```

Рисунок 3.5 – Livewire-компонент, що реалізує додавання товару до кошика

Система враховує, чи користувач увійшов у систему. Якщо ні – використовує унікальний session_id, що дозволяє реалізувати **гостьовий кошик**, який зберігається між сеансами.

3.2.4 Перегляд і редагування кошика

Вивід товарів, доданих до кошика, реалізовано через окремий Livewire-компонент CartView. Його використання дозволяє динамічно взаємодіяти з вмістом кошика – змінювати кількість товарів, видаляти позиції або переглядати загальну вартість – без необхідності оновлення всієї сторінки (див. рис. 3.6).

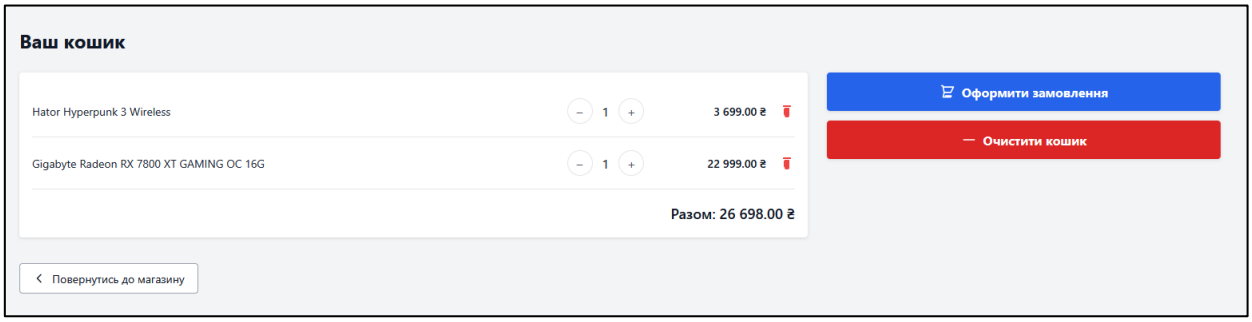


Рисунок 3.6 – Інтерфейс кошика з можливістю редагування кількості товарів

Шаблон компонента cart-view.blade.php відповідає за візуальне представлення списку товарів, зображень, цін, а також кнопок керування (зменшити / збільшити / видалити). Він реалізований із застосуванням Tailwind CSS та директив Livewire. Шаблон Livewire-компонента для перегляду та редагування кошика послідовно наведено на рисунках 3.7 – 3.9 відповідно.

```
<div class="container mx-auto px-4 py-8">
    <h1 class="text-2xl font-bold mb-6">Ваш кошик</h1>

    @if ($items->isEmpty())
        <p class="text-gray-600">Кошик порожній.</p>
    @else
        <div class="flex flex-col md:flex-row gap-6">
            {{-- Ліва колонка: Товари --}}
            <div class="w-full md:w-2/3 bg-white rounded shadow p-4">
                @foreach ($items as $item)
                    <div class="flex items-center justify-between border-b py-4 gap-4">
                        <div class="w-full">
                            <div class="text-sm font-medium text-gray-800">
                                {{ $item->product->name }}
                            </div>
                        </div>
```

Рисунок 3.7

```
                <div class="min-w-[130px] flex items-center gap-1 justify-center">
                    <button
                        wire:click="decreaseQuantity({{ $item->id }})"
                        class="w-8 h-8 rounded-full border text-gray-600 hover:bg-gray-100">−</button>
                    <span class="px-2">{{ $item->quantity }}</span>
                    <button
                        wire:click="increaseQuantity({{ $item->id }})"
                        class="w-8 h-8 rounded-full border text-gray-600 hover:bg-gray-100">+</button>
                </div>

                <div class="min-w-[120px] text-right font-semibold text-sm">
                    {{ number_format($item->quantity * $item->product->price, 2, '.', ' ') }} ₴
                </div>

                <button wire:click="removeItem({{ $item->id }})" class="text-red-500 hover:text-red-700">
                    <svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5" fill="currentColor" viewBox="0 0
20 20">
                        <path fill-rule="evenodd" d="M6 8a1 1 0 012 0v6a1 1 0 11-2 0V8zm4 0a1 1 0 012 0v6a1 1 0
11-2 0V8z" clip-rule="evenodd" />
                        <path fill-rule="evenodd" d="M5 4a1 1 0 011-1h8a1 1 0 011 1v1H5V4zm2 2v10a2 2 0 002
2h2a2 2 0 002-2V6H7z" clip-rule="evenodd" />
                    </svg>
                </button>
            </div>
        @endforeach

        <div class="text-right text-lg font-semibold mt-4">
            Разом: {{ number_format($total, 2, '.', ' ') }} ₴
        </div>
    </div>

    {{-- Права колонка: Дії --}}
    <div class="w-full md:w-1/3 flex flex-col gap-3">
        {{-- Оформити замовлення --}}
        <a href="{{ route('checkout.show') }}"
          class="w-full px-4 py-3 bg-blue-600 text-white rounded font-bold shadow hover:bg-blue-700
transition duration-200 flex items-center justify-center gap-2">
            <svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
                    d="M3 3h2l.4 2M7 13h10l4-8H5.4M7 13L5.4 5M7 13l-1.5 7h11a1 1 0 001-1v-1H9m7 0a2 2 0
11-4 0" />
            </svg>
            Оформити замовлення
        </a>
        {{-- Очистити кошик --}}
        <form wire:submit.prevent="clearCart">
            <button type="submit"
                class="w-full px-4 py-3 bg-red-600 text-white rounded font-bold shadow hover:bg-red-700
transition duration-200 flex items-center justify-center gap-2">
                <svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5" fill="none" viewBox="0 0 24 24"
                    stroke="currentColor">
                    <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
                        d="M20 12H4" />
                </svg>
                Очистити кошик
            </button>
        </form>
    </div>
</div>
```

Рисунок 3.8

```
    {{-- Кнопка "Повернутись до магазину" під усім кошиком --}}
    <div class="mt-8">
      <a href="{{ route('store.front') }}"
        class="inline-flex items-center gap-2 px-4 py-2 text-sm border border-gray-400 rounded bg-white hover:bg-gray-100 transition">
          <svg xmlns="http://www.w3.org/2000/svg" class="h-4 w-4" fill="none" viewBox="0 0 24 24" stroke="currentColor">
            <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 19l-7-7 7-7" />
          </svg>
          Повернутись до магазину
      </a>
    </div>
  @endif
</div>
```

Рисунок 3.9

Логіка компонента CartView реалізована у відповідному PHP-класі. Він містить методи для завантаження вмісту кошика, зміни кількості товарів, підрахунку загальної суми та видалення записів. Джерелом даних виступає таблиця cart_items, пов’язана з товарами через product_id і з користувачами або сесіями через user_id / session_id.

Клас Livewire-компонента, що реалізує логіку редагування кошика, послідовно наведено на рисунках 3.10 та 3.11 відповідно.

```
<?php

namespace App\Livewire;

use Livewire\Component;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Session;
use App\Models\CartItem;

class CartView extends Component
{
  public $items = [];
  public $total = 0;

  public function mount()
  {
    $this->loadCart();
  }

  public function loadCart()
  {
    $query = CartItem::with('product')
```

Рисунок 3.10

```
            ->when(Auth::check(), fn($q) => $q->where('user_id', Auth::id()))
            ->when(!Auth::check(), fn($q) => $q->where('session_id', Session::getId()));

        $this->items = $query->get();
        $this->total = $this->items->sum(fn($item) => $item->product->price * $item->quantity);
    }

    public function increaseQuantity($id)
    {
        $item = CartItem::findOrFail($id);
        $item->quantity++;
        $item->save();
        $this->loadCart();
        $this->dispatch('cartUpdated');
    }

    public function decreaseQuantity($id)
    {
        $item = CartItem::findOrFail($id);
        if ($item->quantity > 1) {
            $item->quantity--;
            $item->save();
        }
        $this->loadCart();
        $this->dispatch('cartUpdated');
    }

    public function removeItem($id)
    {
        CartItem::findOrFail($id)->delete();
        $this->loadCart();
        $this->dispatch('cartUpdated');
    }

    public function render()
    {
        return view('livewire.cart-view');
    }

    public function clearCart()
    {
        $query = CartItem::query();

        if (Auth::check()) {
            $query->where('user_id', Auth::id());
        } else {
            $query->where('session_id', Session::getId());
        }
        $query->delete();

        $this->loadCart(); // обов'язково після очищення
        $this->dispatch('cartUpdated'); // оновити лічильник
    }
}
```

Рисунок 3.11

3.2.5 Оформлення замовлення

Функціонал оформлення замовлення реалізовано через окрему сторінку, яка доступна зареєстрованим користувачам після додавання товарів до кошика. На цій сторінці користувач заповнює контактні дані (ім’я, email тощо), переглядає вміст кошика, загальну суму та підтверджує замовлення (див. рис. 3.12).



Рисунок 3.12 – Форма оформлення замовлення користувачем

Успішне підтвердження замовлення призводить до створення запису в таблиці orders (див. рис. 2.10), а також пов’язаних записів у таблиці order_items (див. рис. 2.11), які зберігають склад кожного замовлення.

Вивід форми та обробка даних реалізовані у Blade-шаблоні з підтримкою серверної валідації та динамічного оновлення суми замовлення. Всі обрахунки виконуються на стороні сервера з оновленням через Livewire. Шаблон форми оформлення замовлення із відображенням вмісту кошика та підтвердженням послідовно наведено на рисунках 3.13 – 3.15 відповідно.

```
@extends('layouts.app')

@section('content')
<div class="container mx-auto px-4">
    <h1 class="text-2xl font-bold mb-4">Оформлення замовлення</h1>
```

Рисунок 3.13

```
@if($errors->any())
    <div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded mb-6">
        <ul class="list-disc pl-5">
            @foreach($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif

@if($items->isEmpty())
    <p class="text-gray-700">Ваш кошик порожній. <a href="{{ route('store.front') }}" class="text-pink-600 underline">Повернутися в магазин</a>.</p>
@else
    <div class="mb-6 overflow-x-auto">
        <table class="w-full table-auto">
            <thead>
                <tr class="bg-gray-200">
                    <th class="px-4 py-2 text-left">Товар</th>
                    <th class="px-4 py-2 text-center">Кількість</th>
                    <th class="px-4 py-2 text-right">Ціна</th>
                    <th class="px-4 py-2 text-right">Сума</th>
                </tr>
            </thead>
            <tbody>
                @foreach($items as $item)
                    <tr class="border-b">
                        <td class="px-4 py-2">{{ $item->product->name }}</td>
                        <td class="px-4 py-2 text-center">{{ $item->quantity }}</td>
                        <td class="px-4 py-2 text-right">{{ number_format($item->product->price, 2) }} грн</td>
                        <td class="px-4 py-2 text-right">
                            {{ number_format($item->quantity * $item->product->price, 2) }} грн
                        </td>
                    </tr>
                @endforeach
            </tbody>
            <tfoot>
                <tr class="bg-gray-100">
                    <td colspan="3" class="px-4 py-2 text-right font-semibold">Всього:</td>
                    <td class="px-4 py-2 text-right font-semibold">
                        {{ number_format($items->sum(fn($i) => $i->quantity * $i->product->price), 2) }} грн
                    </td>
                </tr>
            </tfoot>
        </table>
    </div>

    <form method="POST" action="{{ route('checkout.process') }}" class="max-w-md">
        @csrf

        <div class="mb-4">
            <label for="name" class="block text-sm font-medium text-gray-700 mb-1">Ім’я</label>
            <input type="text" name="name" id="name"
                   value="{{ old('name') }}"
                   class="w-full border rounded px-3 py-2 focus:outline-none focus:ring-2 focus:ring-pink-600" />
        </div>

        <div class="mb-4">
            <label for="email" class="block text-sm font-medium text-gray-700 mb-1">Email</label>
            <input type="email" name="email" id="email"
```

Рисунок 3.14

```
                value="{{ old('email') }}"
                class="w-full border rounded px-3 py-2 focus:outline-none focus:ring-2 focus:ring-pink-600" />
        </div>

        <div class="mb-6">
            <label for="address" class="block text-sm font-medium text-gray-700 mb-1">Адреса доставки</label>
            <textarea name="address" id="address" rows="3"
                    class="w-full border rounded px-3 py-2 focus:outline-none focus:ring-2 focus:ring-pink-600">{{
old('address') }}</textarea>
        </div>

        <button type="submit"
                class="bg-pink-600 text-white px-6 py-2 rounded hover:bg-pink-700">
            Підтвердити замовлення
        </button>
    </form>
  @endif
</div>
@endsection
```

Рисунок 3.15

3.2.6 Перегляд історії замовлень

Інформаційна система інтернет-магазину передбачає можливість перегляду історії замовлень для зареєстрованих користувачів. Реалізація цього функціоналу дозволяє кожному користувачеві бачити власні замовлення, які він оформлював раніше, а також переглядати деталі кожного замовлення – перелік товарів, їх кількість, вартість і статус виконання.

Перегляд списку замовлень. Сторінка з історією замовлень реалізована у вигляді табличного виводу, де кожен рядок містить базову інформацію про замовлення: номер, дата, кількість позицій, сума, а також посилання на детальний перегляд.

Якщо у користувача ще не було оформлених замовлень, виводиться відповідне інформаційне повідомлення (див. рис. 3.16 – 3.17).

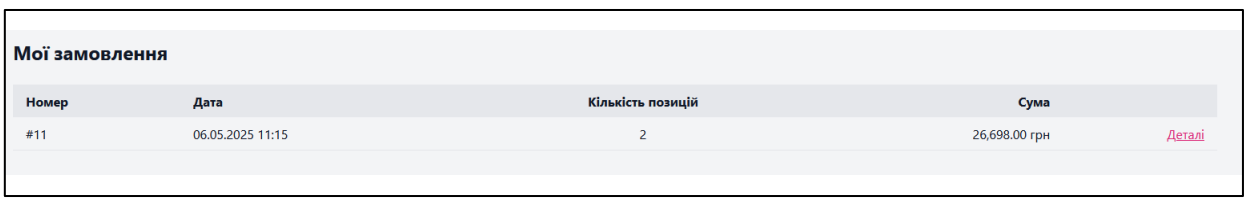


Рисунок 3.16 – Сторінка історії замовлень користувача

```
@extends('layouts.app')

@section('content')
<div class="container mx-auto px-4">
  <h1 class="text-2xl font-bold mb-6">Мої замовлення</h1>

  @if($orders->isEmpty())
    <p>У вас ще не було замовлень.</p>
  @else
    <table class="w-full table-auto mb-6">
      <thead>
        <tr class="bg-gray-200">
          <th class="px-4 py-2 text-left">Номер</th>
          <th class="px-4 py-2 text-left">Дата</th>
          <th class="px-4 py-2 text-center">Кількість позицій</th>
          <th class="px-4 py-2 text-right">Сума</th>
          <th class="px-4 py-2"></th>
        </tr>
      </thead>
      <tbody>
        @foreach($orders as $order)
          <tr class="border-b hover:bg-gray-50">
            <td class="px-4 py-2">#{{ $order->id }}</td>
            <td class="px-4 py-2">
              {{ $order->ordered_at->format('d.m.Y H:i') }}
            </td>
            <td class="px-4 py-2 text-center">{{ $order->items_count }}</td>
            <td class="px-4 py-2 text-right">{{ number_format($order->total_price,2) }} грн</td>
            <td class="px-4 py-2 text-right">
              <a href="{{ route('orders.show', $order) }}"
                 class="text-pink-600 underline hover:text-pink-800">
                 Деталі
              </a>
            </td>
          </tr>
        @endforeach
      </tbody>
    </table>

    {{ $orders->links() }}
  @endif
</div>
@endsection
```

Рисунок 3.17 – Сторінка зі списком замовлень користувача

Перегляд деталей замовлення. Після переходу за посиланням «Деталі» користувач отримує доступ до повної інформації про обране замовлення: дату оформлення, поточний статус, загальну суму, а також список усіх товарів із кількістю, ціною за одиницю та загальною вартістю (див. рис. 3.18 – 3.19).



Рисунок 3.18 – Сторінка деталей замовлення користувача

```
@extends('layouts.app')

@section('content')
<div class="container mx-auto px-4">
  <h1 class="text-2xl font-bold mb-6">Деталі замовлення #{{ $order->id }}</h1>

  <p class="mb-2">
    Дата: {{ $order->ordered_at->format('d.m.Y H:i') }}<br>
    Статус: {{ ucfirst($order->status) }}<br>
    Загальна сума: <strong>{{ number_format($order->total_price,2) }} грн</strong>
  </p>

  <div class="overflow-x-auto mb-6">
    <table class="w-full table-auto">
      <thead>
        <tr class="bg-gray-200">
          <th class="px-4 py-2 text-left">Товар</th>
          <th class="px-4 py-2 text-center">Кількість</th>
          <th class="px-4 py-2 text-right">Ціна за шт.</th>
          <th class="px-4 py-2 text-right">Сума</th>
        </tr>
      </thead>
      <tbody>
        @foreach($order->items as $item)
          <tr class="border-b">
            <td class="px-4 py-2">{{ $item->product->name }}</td>
            <td class="px-4 py-2 text-center">{{ $item->quantity }}</td>
            <td class="px-4 py-2 text-right">{{ number_format($item->price,2) }} грн</td>
            <td class="px-4 py-2 text-right">{{ number_format($item->quantity * $item->price,2) }} грн</td>
          </tr>
        @endforeach
      </tbody>
    </table>
  </div>

  <a href="{{ route('orders.index') }}"
    class="inline-block bg-pink-600 text-white px-6 py-2 rounded hover:bg-pink-700">
    Назад до історії замовлень
  </a>
</div>
@endsection
```

Рисунок 3.19 – Детальна сторінка замовлення із переліком товарів

3.3 Реалізація адміністративного функціоналу інформаційної системи

3.3.1 Адміністративна панель: загальний опис

Адміністративна частина системи є окремим функціональним модулем, призначеним для управління основними сутностями інтернет-магазину: товарами, категоріями, замовленнями та покинутими кошиками. Доступ до цієї частини сайту мають лише користувачі з роллю адміністратора. Перевірка

прав доступу реалізована за допомогою middleware, що обмежує доступ до відповідних маршрутів.

Адмін-панель реалізована на основі класичного CRUD-підходу (Create, Read, Update, Delete), що забезпечує зручність у наповненні та оновленні бази товарів, а також надає адміністратору змогу ефективно реагувати на зміни замовлень і поведінку користувачів.

До адміністративної частини входять наступні модулі:

– керування товарами – дозволяє створювати, редагувати, переглядати та видаляти товарні позиції;
– перегляд замовлень – надає повну інформацію про всі оформлені замовлення: дату, вміст, статус, користувача;
– покинуті кошики – окремий модуль для аналітики поведінки користувачів, які не завершили покупку.

Інтерфейс адміністративної панелі витримано у єдиному стилі з основним сайтом, з використанням Blade-шаблонів і Tailwind CSS (див. рис. 3.20).



Рисунок 3.20 – Інтерфейс адміністративної панелі інтернет-магазину

Всі CRUD-операції реалізовано через звичайні маршрути Laravel без використання AJAX або Livewire, що відповідає концепції простоти та надійності адміністрування.

3.3.2 Керування товарами (CRUD)

Операції створення, перегляду, редагування та видалення товарів реалізовані в окремому модулі адміністративної панелі. Функціонал забезпечує адміністраторам можливість керування асортиментом інтернет-магазину у зручному вебінтерфейсі (див. рис. 3.20).

Адміністративний інтерфейс складається з кількох сторінок:

– перегляд усіх товарів у таблиці;
– форма створення нового товару;
– форма редагування існуючого товару;
– спільний шаблон форми (form.blade.php), який використовується як для створення, так і для редагування.

Всі шаблони реалізовані на Blade з використанням Tailwind CSS, а дані обробляються через звичайні маршрути Laravel із CSRF-захистом та валідацією на рівні контролера.

Список товарів. На сторінці index.blade.php адміністратор бачить перелік усіх товарів із можливістю редагувати або видалити кожен з них. Зображення виводиться мініатюрою, ціна та категорія – у відповідних стовпцях. Програмна реалізація адміністративної таблиці товарів із діями послідовно наведено на рисунках 3.21 та 3.22 відповідно.

```
@extends('layouts.app')

@section('title', 'Управління товарами')

@section('content')
<div class="max-w-7xl mx-auto px-4">
   <h1 class="text-2xl font-bold mb-6">Управління товарами</h1>

   @if (session('success'))
      <div class="mb-4 bg-green-100 border border-green-300 text-green-800 px-4 py-2 rounded">
         {{ session('success') }}
      </div>
   @endif
   <div class="flex justify-between items-center mb-4">
      <a href="{{ route('admin.products.create') }}"
        class="px-4 py-2 bg-blue-600 text-white text-sm font-semibold rounded hover:bg-blue-700 transition">
         + Додати товар
      </a>
```

Рисунок 3.21

```
    </div>
    <div class="overflow-x-auto">
        <table class="min-w-full table-auto border border-gray-300 rounded">
            <thead class="bg-gray-100 text-sm">
                <tr>
                    <th class="border px-4 py-2">Зображення</th>
                    <th class="border px-4 py-2">Назва</th>
                    <th class="border px-4 py-2">Ціна</th>
                    <th class="border px-4 py-2">Категорія</th>
                    <th class="border px-4 py-2">Дії</th>
                </tr>
            </thead>
            <tbody class="text-center text-sm">
                @forelse ($products as $product)
                    <tr class="hover:bg-gray-50">
                        <td class="border px-2 py-2">
                            @if ($product->image)
                                <img src="{{ asset('storage/' . $product->image) }}" alt="{{ $product->name }}"
                                    class="w-16 h-16 object-cover rounded mx-auto shadow">
                            @else
                                <div class="w-16 h-16 flex items-center justify-center bg-gray-100 text-gray-400 rounded">
                                    –
                                </div>
                            @endif
                        </td>
                        <td class="border px-4 py-2">{{ $product->name }}</td>
                        <td class="border px-4 py-2">{{ number_format($product->price, 2) }} грн</td>
                        <td class="border px-4 py-2">{{ $product->category->name ?? '–' }}</td>
                        <td class="border px-4 py-2 space-x-2">
                            <a href="{{ route('admin.products.edit', $product) }}"
                               class="text-blue-600 hover:underline">Редагувати</a>

                            <form method="POST" action="{{ route('admin.products.destroy', $product) }}" class="inline-
block">
                                @csrf
                                @method('DELETE')
                                <button class="text-red-600 hover:underline"
                                        onclick="return confirm('Видалити цей товар?')">
                                    Видалити
                                </button>
                            </form>
                        </td>
                    </tr>
                @empty
                    <tr>
                        <td colspan="5" class="border px-4 py-4 text-gray-500 italic">Товари відсутні</td>
                    </tr>
                @endforelse
            </tbody>
        </table>
    </div>
</div>
@endsection
```

Рисунок 3.22

Створення нового товару. Сторінка створення (create.blade.php) містить заголовок та підключення шаблону з формою. Вся бізнес-логіка обробки POST-запиту реалізована в контролері (див. рис. 3.23 – 3.24).

```
@extends('layouts.app')
@section('title', 'Новий товар')

@section('content')
<h1 class="text-2xl font-bold mb-4">Додати товар</h1>

<form method="POST" action="{{ route('admin.products.store') }}" enctype="multipart/form-data">
    @csrf
    @include('admin.products.form')
</form>
@endsection
```

Рисунок 3.23 – Шаблон сторінки створення нового товару

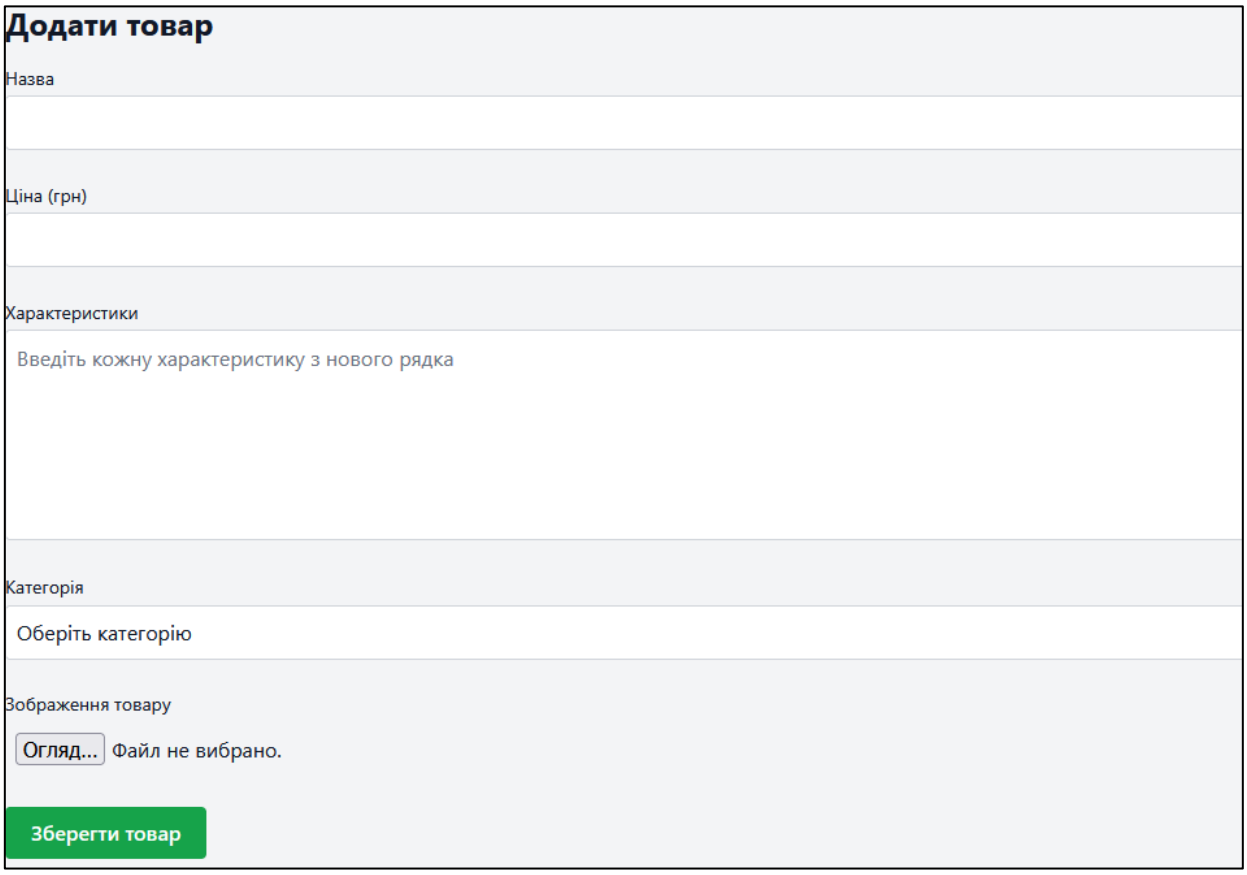


Рисунок 3.24 – Сторінка створення товару

Редагування існуючого товару. Редагування товару реалізовано на окремій сторінці, де заздалегідь заповнюються дані з моделі. У разі виявлення помилок під час валідації вони виводяться над формою (див. рис. 3.25 – 3.27).

```
@extends('layouts.app')

@section('title', 'Редагувати товар')

@section('content')
<div class="max-w-3xl mx-auto px-4">
```

Рисунок 3.25 – Шаблон сторінки редагування товару (1 частина)

```
    <h1 class="text-2xl font-bold mb-6">Редагувати: {{ $product->name }}</h1>

    @if ($errors->any())
        <div class="mb-4 bg-red-100 border border-red-300 text-red-700 px-4 py-2 rounded">
            <ul class="list-disc pl-5 space-y-1">
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

    <form method="POST" action="{{ route('admin.products.update', $product) }}" enctype="multipart/form-data"
class="space-y-6">
        @csrf
        @method('PUT')

        <div>
            <label class="block font-medium text-sm mb-1">Назва</label>
            <input type="text" name="name" value="{{ old('name', $product->name) }}"
                   class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200" required>
        </div>

        <div>
            <label class="block font-medium text-sm mb-1">Ціна (грн)</label>
            <input type="number" step="0.01" name="price" value="{{ old('price', $product->price) }}"
                   class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200" required>
        </div>

        <div>
            <label class="block font-medium text-sm mb-1">Характеристики</label>
            <textarea name="features" rows="6"
              class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200"
              placeholder="Введіть кожну характеристику з нового рядка">{{ old('features', $product->features)
}}</textarea>
          </div>

        <div>
            <label class="block font-medium text-sm mb-1">Категорія</label>
            <select name="category_id"
                    class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200" required>
                <option value="">Оберіть категорію</option>
                @foreach ($categories as $category)
                    <option value="{{ $category->id }}" {{ $product->category_id == $category->id ? 'selected' : '' }}>
                        {{ $category->name }}
                    </option>
                @endforeach
            </select>
        </div>

        <div>
            <label class="block font-medium text-sm mb-1">Поточне зображення</label>
            @if ($product->image)
                <img src="{{ asset('storage/' . $product->image) }}" alt="{{ $product->name }}" class="w-32 h-32
object-cover rounded mb-2">
            @else
                <p class="text-sm text-gray-500 italic">Зображення не додано</p>
            @endif

            <label class="block font-medium text-sm mb-1 mt-2">Замінити зображення</label>
            <input type="file" name="image"
                   class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200">
        </div>

        <div>
            <button type="submit"
                    class="px-5 py-2 bg-blue-600 text-white rounded hover:bg-blue-700 transition font-semibold">
                Оновити товар
            </button>
        </div>
</div>
@endsection
```

Рисунок 3.26 – Шаблон сторінки редагування товару (2 частина)

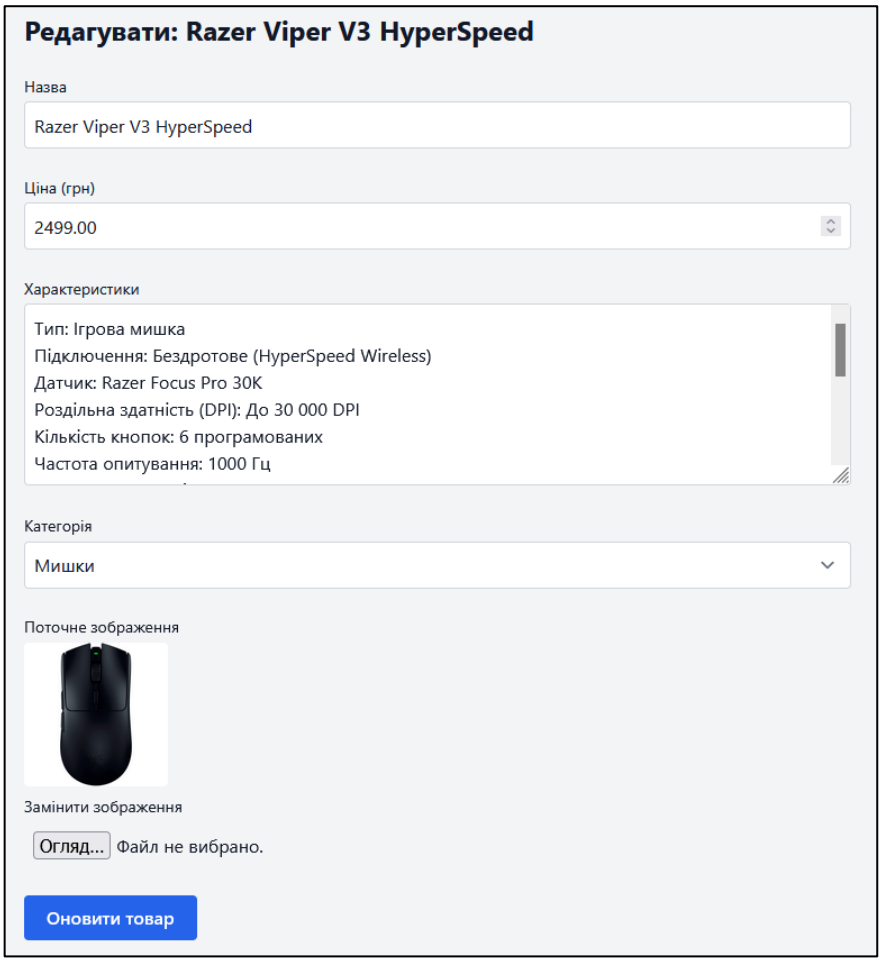


Рисунок 3.27 – Сторінка редагування товару

Форма для створення та редагування (спільна). У шаблоні form.blade.php реалізовано інпути для всіх обов'язкових полів: назва, ціна, характеристики, категорія та зображення. Цей файл підключається як до create, так і до edit (див. рис. 3.28 та 3.29 відповідно).

```
@csrf

@if (isset($product))
    @method('PUT')
@endif

<div class="space-y-6">
    <div>
        <label class="block font-medium text-sm mb-1">Назва</label>
        <input type="text" name="name" value="{{ old('name', $product->name ?? '') }}"
               class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200" required>
    </div>

    <div>
        <label class="block font-medium text-sm mb-1">Ціна (грн)</label>
        <input type="number" step="0.01" name="price" value="{{ old('price', $product->price ?? '') }}"
               class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200" required>
```

Рисунок 3.28

```
    </div>

    <div>
        <label class="block font-medium text-sm mb-1">Характеристики</label>
        <textarea name="features" rows="6"
         class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200"
         placeholder="Введіть кожну характеристику з нового рядка">{{ old('features', $product->features ?? '')
}}</textarea>
</div>

    <div>
        <label class="block font-medium text-sm mb-1">Категорія</label>
        <select name="category_id"
                class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200" required>
            <option value="">Оберіть категорію</option>
            @foreach ($categories as $category)
                <option value="{{ $category->id }}"
                    {{ (old('category_id', $product->category_id ?? '') == $category->id) ? 'selected' : '' }}>
                    {{ $category->name }}
                </option>
            @endforeach
        </select>
    </div>

    @if (isset($product) && $product->image)
        <div>
            <label class="block font-medium text-sm mb-1">Поточне зображення</label>
            <img src="{{ asset('storage/' . $product->image) }}" alt="{{ $product->name }}" class="w-32 h-32 object-
cover rounded mb-2">
        </div>
    @endif

    <div>
        <label class="block font-medium text-sm mb-1">Зображення товару</label>
        <input type="file" name="image"
               class="border-gray-300 rounded w-full p-2 focus:ring focus:ring-indigo-200">
    </div>

    <div>
        <button type="submit"
                class="px-5 py-2 bg-green-600 text-white rounded hover:bg-green-700 transition font-semibold">
            {{ isset($product) ? 'Оновити товар' : 'Зберегти товар' }}
        </button>
    </div>
</div>
```

Рисунок 3.29

3.3.3 Перегляд покинутих кошиків

Покинуті кошики – це записи про товари, які користувачі додали до кошика, але не оформили як замовлення. Система зберігає такі записи у базі даних, що дозволяє адміністраторам переглядати, аналізувати та здійснювати відповідні управлінські рішення, наприклад, щодо популярності товарів або незавершених сесій (див. рис. 3.30).

Функціонал реалізовано через контролер AbandonedCartController та два Blade-шаблони (див. рис. 3.31 – 3.32).

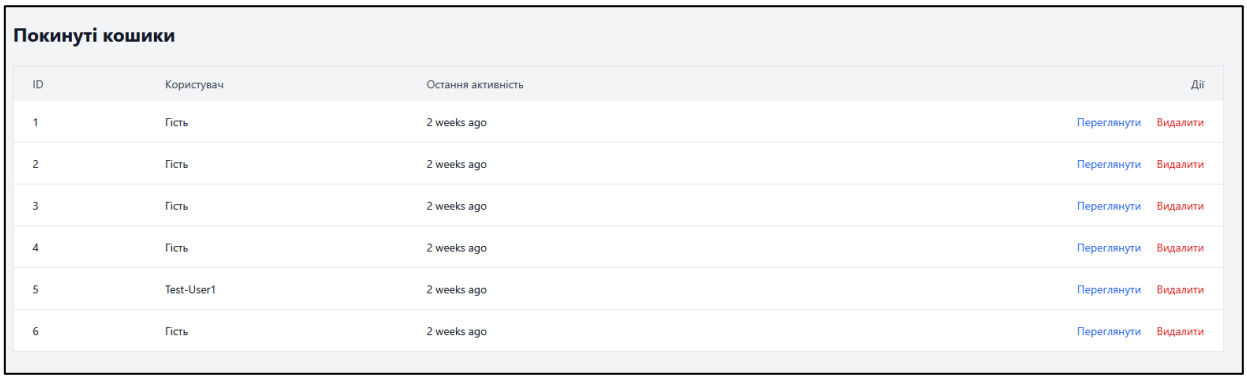


Рисунок 3.30 – Сторінка покинутих кошиків

```
@extends('layouts.app')

@section('content')
  <div class="container mx-auto py-6">
    <h1 class="text-2xl font-bold mb-6">Покинуті кошики</h1>

    @if(session('success'))
      <div class="bg-green-100 text-green-800 px-4 py-2 rounded mb-4">
        {{ session('success') }}
      </div>
    @endif
    <div class="overflow-x-auto">
      <table class="min-w-full bg-white border border-gray-200 rounded">
        <thead class="bg-gray-100 text-left">
          <tr>
            <th class="px-6 py-3 text-sm font-medium text-gray-700">ID</th>
            <th class="px-6 py-3 text-sm font-medium text-gray-700">Користувач</th>
            <th class="px-6 py-3 text-sm font-medium text-gray-700">Остання активність</th>
            <th class="px-6 py-3 text-sm font-medium text-gray-700 text-right">Дії</th>
          </tr>
        </thead>
        <tbody class="divide-y divide-gray-200">
          @foreach($abandonedCarts as $cart)
            <tr>
              <td class="px-6 py-4 text-sm text-gray-900">{{ $cart->id }}</td>
              <td class="px-6 py-4 text-sm text-gray-900">{{ $cart->user->name ?? 'Гість' }}</td>
              <td class="px-6 py-4 text-sm text-gray-900">{{ $cart->updated_at->diffForHumans() }}</td>
              <td class="px-6 py-4 text-sm text-right">
                <a href="{{ route('admin.abandoned.show', $cart) }}" class="text-blue-600 hover:underline
mr-4">Переглянути</a>
                <form action="{{ route('admin.abandoned.destroy', $cart) }}" method="POST" class="inline-
block"
                  onsubmit="return confirm('Ви впевнені, що хочете видалити цей кошик?')">
                  @csrf
                  @method('DELETE')
                  <button type="submit" class="text-red-600 hover:underline">Видалити</button>
                </form>
              </td>
            </tr>
          @endforeach
        </tbody>
      </table>
    </div>
  </div>
@endsection
```

Рисунок 3.31 – Таблиця покинутих кошиків у панелі адміністратора

```
@extends('layouts.app')

@section('title', 'Покинуті кошики')
@section('content')
   <div class="max-w-screen-xl mx-auto px-4 py-8">
      <h1 class="text-2xl font-bold mb-6">Покинутий кошик користувача</h1>

      <div class="bg-white shadow p-6 rounded-lg mb-8">
         <p class="mb-2 text-gray-800">
            Користувач:
         {{ $abandonedCart->user ? $abandonedCart->user->email : 'Гість (сесія ' . $abandonedCart->session_id
. ')' }}
         </p>
         <p class="mb-2 text-sm text-gray-600">
            Остання активність: {{ $abandonedCart->last_activity?->diffForHumans() ?? '–' }}
         </p>
      </div>

      @if(count($carts))
         <div class="overflow-x-auto">
            <table class="w-full text-left border-collapse bg-white shadow rounded-lg">
               <thead class="bg-gray-100">
                  <tr>
                     <th class="px-4 py-2 border-b">ID</th>
                     <th class="px-4 py-2 border-b">Назва</th>
                     <th class="px-4 py-2 border-b">Кількість</th>
                     <th class="px-4 py-2 border-b">Ціна</th>
                  </tr>
               </thead>
               <tbody>
                  @foreach ($carts as $cart)
                     <tr class="hover:bg-gray-50">
                        <td class="px-4 py-2 border-b">{{ $cart['id'] ?? '–' }}</td>
                        <td class="px-4 py-2 border-b">{{ $cart['name'] ?? '–' }}</td>
                        <td class="px-4 py-2 border-b">{{ $cart['quantity'] ?? 1 }}</td>
                        <td class="px-4 py-2 border-b">{{ $cart['price'] ?? '–' }} грн</td>
                     </tr>
                  @endforeach
               </tbody>
            </table>
         </div>
      @else
         <p class="text-gray-500">Немає товарів у цьому покинутому кошику.</p>
      @endif
   </div>
@endsection
```

Рисунок 3.32 – Детальна інформація про покинутий кошик

3.4 Особливості реалізації гостьового кошика

У системі реалізовано повноцінний механізм **гостьового кошика**, який дозволяє користувачам додавати товари до кошика без обов’язкової авторизації. Це є важливою функціональною особливістю, що підвищує

зручність користування сайтом та відповідає сучасним стандартам UX у сфері електронної комерції.

Логіка реалізації. Кожен неавторизований користувач ототожнюється з унікальним ідентифікатором сесії (session_id), що генерується Laravel. Цей ідентифікатор зберігається у cookie браузера та використовується для прив’язки позицій у кошику до конкретної сесії.

Записи з гостьового кошика зберігаються у таблиці cart_items у вигляді:

– session_id – обов’язкове поле, якщо користувач не авторизований;
– user_id – встановлюється лише після входу;
– product_id, quantity – дані про товар та його кількість;

Поведінка після входу. Після авторизації система:

– знаходить кошик користувача за user_id;
– об’єднує або переносить позиції з session_id, якщо такі існують;
– оновлює записи в таблиці cart_items, прив’язуючи їх до user_id.

Це дозволяє користувачеві **не втратити вміст кошика після входу в акаунт**, що значно покращує взаємодію з інтерфейсом і зменшує кількість покинутих покупок.

ВИСНОВКИ

У межах виконання дипломної роботи було здійснено повний цикл розробки інформаційної системи інтернет-магазину, починаючи з аналізу предметної області та вибору технічного стеку, й завершуючи реалізацією функціональних модулів, що охоплюють як користувацьку, так і адміністративну частини.

На першому етапі було проведено **огляд сучасних фреймворків для розробки вебзастосунків**, зокрема PHP-фреймворків, серед яких було обґрунтовано вибір Laravel як найбільш гнучкого, масштабованого та підтримуваного рішення. Проаналізовано основні переваги цієї платформи, її відповідність потребам електронної комерції, можливості щодо швидкої розробки, безпеки, розширення функціоналу та підтримки сучасного підходу до побудови архітектури вебзастосунків.

У другому розділі детально проаналізовано **вимоги до системи**, здійснено моделювання структури бази даних, побудовано UML-діаграми, діаграму прецедентів, класів, ER-діаграму. Проведено формалізацію даних, визначено основні функціональні й нефункціональні вимоги до системи, обґрунтовано вибір інструментів і технологій. Це дало змогу закласти структурну основу для реалізації системи відповідно до сучасних стандартів інженерії програмного забезпечення.

Найбільшу частину роботи присвячено **реалізації функціональних можливостей системи**. Було розроблено модуль користувача, що включає перегляд товарів, фільтрацію за категоріями, детальні картки товару, додавання до кошика, оформлення замовлення та перегляд історії покупок. Окремо реалізовано підтримку гостьового кошика з подальшим збереженням його при вході в систему, що покращує користувацький досвід.

У рамках адміністративної частини реалізовано модулі керування товарами та категоріями, що надають адміністратору змогу створювати, редагувати й видаляти відповідні записи. Також розроблено систему

перегляду **покинутих кошиків**, яка дозволяє аналізувати поведінку користувачів, що не завершили оформлення замовлення.

Усі функціональні частини інтегровані у єдиний інтерфейс, реалізований із використанням Blade-шаблонізатора, CSS-фреймворку Tailwind та Livewire-компонентів для реалізації інтерактивної взаємодії. Особливу увагу приділено **безпеці системи**, включаючи CSRF-захист, серверну валідацію, обмеження доступу до адміністративних модулів та коректну обробку сесій.

Таким чином, у ході дипломної роботи було досягнуто поставлену мету – створено повноцінний вебзастосунок інтернет-магазину, який відповідає сучасним вимогам щодо функціональності, безпеки, зручності та масштабованості. Розроблена система може бути використана як база для подальшого розширення або впровадження в реальний проєкт.

ПЕРЕЛІК ПОСИЛАНЬ

1. Laravel. The PHP Framework for Web Artisans. *Laravel Official Site*. URL: https://laravel.com/ (дата звернення: 10.01.2025).

2. Tailwind CSS – Rapidly build modern websites. *Tailwind CSS*. URL: https://tailwindcss.com/ (дата звернення: 12.01.2025).

3. Livewire – A full-stack framework for Laravel. *Livewire*. URL: https://livewire.laravel.com/ (дата звернення: 19.01.2025).

4. Alpine.js – A rugged, minimal framework for composing JavaScript behavior. *Alpine.js*. URL: https://alpinejs.dev/ (дата звернення: 23.01.2025).

5. Composer – Dependency Manager for PHP. *Getcomposer.org*. URL: https://getcomposer.org/ (дата звернення: 02.02.2025).

6. PHP: Hypertext Preprocessor. *PHP.net*. URL: https://www.php.net/ (дата звернення: 11.02.2025).

7. GitHub – laravel/laravel: Laravel is a web application framework with expressive, elegant syntax. *GitHub*. URL: https://github.com/laravel/laravel (дата звернення: 17.03.2025).

8. R. Sharma. Building Modern Web Applications with Laravel and Vue.js. *Medium*. URL: https://medium.com/@ravi.sharma/building-modern-web-apps-with-laravel-and-vuejs-123456789 (дата звернення: 03.04.2025).

9. TutsMake. Laravel 11 Tutorial for Beginners: Step-by-Step Guide. *TutsMake*. URL: https://www.tutsmake.com/laravel-11-tutorial/ (дата звернення: 19.04.2025).

10. DigitalOcean. How To Deploy a Laravel Application on Ubuntu. *DigitalOcean Docs*. URL: https://www.digitalocean.com/community/tutorials/how-to-deploy-a-laravel-app (дата звернення: 28.04.2025).

11. An Expert Guide to PHP Framework Popularity in 2024-2025. *Devabit*. URL: https://devabit.com/blog/php-framework-popularity-2024/ (дата звернення: 29.04.2025).

12. Дрижов Р. Є. Інформаційна технологія реалізації концепції лаконічного підходу у вебдизайні : робота на здобуття кваліфікаційного ступеня магістра : спец. 122 – комп’ютерні науки / наук. кер. О. В. Власенко. Суми : Сумський державний університет, 2024. 63 с. URL: https://essuir.sumdu.edu.ua/handle/123456789/97630 (дата звернення: 15.05.2025).

13. Singla L. Comparison of Best PHP Frameworks for Website Development in 2025. *Net Solutions*. URL: https://www.netsolutions.com/insights/best-php-frameworks-for-website-development/ (дата звернення: 15.05.2025).

14. Kohl C. 9 Best PHP Frameworks for Web Development. *Medium*. URL: https://medium.com/@cesarkohl/9-best-php-frameworks-for-web-development-7018a503af6a (дата звернення: 15.05.2025).

15. Mallow Technologies. The 7 Most Popular PHP Frameworks for Web Application Development. *Mallow Tech Blog*. URL: https://mallow-tech.com/blog/the-7-most-popular-php-frameworks-for-web-application-development/ (дата звернення: 15.05.2025).

16. Hostinger Tutorials. 11 Best PHP Frameworks For Beginner to Pro Developers. *Hostinger*. URL: https://www.hostinger.com/tutorials/best-php-framework (дата звернення: 15.05.2025).