Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ З ПРОДАЖУ ТОВАРІВ ЯПОНСЬКОЇ ТЕМАТИКИ З ВИКОРИСТАННЯМ LARAVEL

DEVELOPMENT OF AN ONLINE STORE FOR THE SALE OF JAPANESE-THEMED GOODS USING LARAVEL

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
Шульга Андрій Олексійович
_____
(підпис)

Керівник:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Н. Ліщина
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Шульзі Андрію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *«Розробка інтернет-магазину з продажу товарів японської тематики з використанням laravel»*

Керівник роботи: *к.т.н., доцент, Ліщина Наталія Миколаївна*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу: *23 рисунків; 6 лістингів; 2 діаграми.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Ліщина Н. М.*		
Специфікації вимог до розробленої системи	*Ліщина Н. М.*		
Розробка об'єкта проєктування	*Ліщина Н. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту		8,3%	
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	

Здобувач вищої освіти _____ (підпис) *Шульга А О.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Ліщина Н. М.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: Шульга Андрій Олексійович

Тема: «Розробка інтернет-магазину з продажу товарів японської тематики з використанням Laravel»

Коротка характеристика кваліфікаційної роботи: Робота бакалавра присвячена розробці інтернет магазину з використанням Laravel, що забезпечує користувачам зручний інтерфейс для пошуку та придбання японських товарів, а також функціональність для управління товарним асортиментом, обробки замовлень та ін.

Самостійні розробки і пропозиції автора: У роботі узагальнено методи розробки інтернет магазину з використанням Laravel підходу, а також доведено ефективність запропонованих технологій та алгоритмів.

Практичне значення роботи: Створення функціонального інтернет-магазину, який не лише сприятиме популяризації японської культури та продукції, але й надасть користувачам зручний та ефективний інструмент для здійснення покупок.

Недоліки: в роботі присутні незначні зауваження щодо наповнення програмного продукту та окремі огріхи у вигляді помилок синтаксису, які суттєво не занижують практичної цінності роботи.

Загальний висновок: Робота виконана на високому науковому рівні, має логічну та послідовну структуру, відповідає встановленим вимогам, може бути представлена до захисту на державній екзаменаційній комісії та заслуговує позитивної оцінки.

Рецензент: к. пед. н., доцент, завідувач кафедри ЧОІ
(прізвище, ім'я, по-батькові, посада)

Кабак В.В.

Рецензент кваліфікаційної роботи _____
(підпис)

«7» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Здобувач вищої освіти *Шульга Андрій Олексійович*
група *ІПЗс-31*

Тема кваліфікаційної роботи: *Розробка інтернет-магазину з продажу товарів японської тематики з використанням Laravel*
Керівник *Ліщина Наталія Миколаївна*

Актуальність теми: Створення інтернет-магазинів з використанням Laravel залишається важливим завдяки його високій продуктивності, безпеці та гнучкості. Цей фреймворк дозволяє розробляти сучасні веб-додатки, які задовольняють потреби як бізнесу, так і користувачів, забезпечуючи швидку та зручну взаємодію, що є ключовим фактором для успішної роботи в умовах зростаючої конкуренції в онлайн-торгівлі.

Об'єкт дослідження – це комплекс заходів з проектування та кінцевий продукт створення інтернет магазину.

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи: у рамках цієї роботи було розроблено ефективний інтернет-магазин на базі фреймворку Laravel, що демонструє високий теоретичний рівень та практичну значущість. Система відзначається зручним інтерфейсом управління контентом та каталогом товарів, що спрощує адміністрування платформи.

Зауваження та недоліки: істотних недоліків не виявлено.

Загальний висновок робота виконана на хорошому рівні та заслуговує оцінки «добре»_за умови успішного захисту.

Керівник _____ (Н. Ліщина)
(підпис) (прізвище, ім'я, по батькові)

«7» 06 2024 р.

АНОТАЦІЯ

Шульга А. О. Розробка інтернет-магазину з продажу товарів японської тематики з використанням Laravel. Рукопис.

Кваліфікаційна робота бакалавра за спеціальністю 121 Інженерія програмного забезпечення. Луцький національний технічний університет. Луцьк, 2024.

Ця кваліфікаційна робота бакалавра присвячена розробці інтернет-магазину з продажу товарів японської тематики з використанням фреймворку Laravel. У роботі проведено аналіз існуючих інтернет-магазинів, що спеціалізуються на японських товарах, виявлено їхні переваги та недоліки.

Застосування фреймворку Laravel дозволить ефективно розробити інтернет-магазин, забезпечивши високу якість коду та зручну адміністративну панель для управління сайтом та товарами.

Цей проект має на меті створити відмінний інтернет-магазин, який задовольнить потреби клієнтів у товарах японської тематики та надасть їм зручний та ефективний інтерфейс для покупок.

Ключові слова: інтернет-магазин, товари японської тематики, розробка, Laravel, електронна комерція, веб-додаток.

ABSTRACT

Shulha A. O. Development of an online store selling Japanese goods using Laravel. Manuscript.

Qualification work of a bachelor in specialty 121 Software Engineering. Lutsk National Technical University. Lutsk, 2024.

This bachelor's thesis is dedicated to the development of an online store for selling Japanese-themed products using the Laravel framework. The work includes an analysis of existing online stores specializing in Japanese products, identifying their advantages and disadvantages.

The use of the Laravel framework will allow to develop an online store efficiently, providing high quality code and a convenient administrative panel for managing the site and products.

This project aims to create an excellent online store that will satisfy the needs of customers for Japanese goods and provide them with a convenient and efficient shopping interface.

Keywords: online store, Japanese goods, development, Laravel, e-commerce, web application.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі електронна комерція займає дедалі вагоміше місце у сфері продажу товарів та послуг. Розвиток інтернет-технологій сприяє виникненню нових можливостей для бізнесу, зокрема у створенні зручних та функціональних онлайн-магазинів. Одним з ефективних інструментів для розробки таких веб-додатків є фреймворк Laravel, що дозволяє створювати високоякісні та надійні веб-ресурси.

Тема даної кваліфікаційної роботи полягає у розробці інтернет-магазину з продажу товарів японської тематики з використанням Laravel. Японська культура, зокрема її традиції, мистецтво, мода та технології, має великий попит у всьому світі, включаючи Україну. Створення спеціалізованого інтернет-магазину, який пропонуватиме різноманітні японські товари, здатне задовольнити інтереси широкої аудиторії та сприяти популяризації японської культури.

Однією з ключових аудиторій, на яку спрямований цей інтернет-магазин, є молодь України. Японська культура давно завоювала серця молодих українців, які цікавляться аніме, мангою, японською модою, кухнею та технологіями.

Дослідження теми розробки інтернет-магазину з продажу товарів японської тематики відображає глибоке розуміння потреб та уподобань цільової аудиторії, а також вимоги до технічної реалізації проекту. Аналіз ринку та вивчення основних категорій товарів дозволяють визначити, які продукти користуються найбільшим попитом серед молоді. Водночас, розгляд технічних аспектів розробки платформи, таких як архітектура системи, розробка бази даних та створення інтерфейсу користувача є критично важливим для забезпечення зручності та безпеки користувачів.

Метою цієї роботи є детальний аналіз процесу розробки інтернет-магазину з використанням фреймворку Laravel, а також реалізація практичних рішень для створення функціонального та зручного у користуванні веб-додатку. У рамках даного проекту буде розглянуто наступні аспекти: проектування архітектури

системи, розробка бази даних, створення інтерфейсу користувача та адміністратора, забезпечення безпеки та оптимізація продуктивності веб-додатку.

Завдання, які потрібно виконати відповідно до поставленої мети кваліфікаційної роботи:

– аналіз та реалізація можливостей створення інтернет-магазину з продажу товарів японської тематики;

– проектування структури бази даних для зберігання інформації про товари, замовлення та користувачів;

– розробка користувацького інтерфейсу, який забезпечить зручність та зрозумілість для користувачів під час покупок;

– реалізація backend частини магазину з використанням фреймворку Laravel для забезпечення безпеки та ефективності роботи системи;

– створення інтерактивних елементів, таких як кошик для покупок та функціонал оформлення замовлення.

Об’єктом кваліфікаційної роботи є інтернет-магазин, спроектований для продажу товарів японської тематики. Його функціональні можливості включають адміністративну панель, перегляд каталогу товарів, додавання товарів до кошика та оформлення. Крім того, магазин має забезпечувати зручний пошук товарів за різними параметрами, відображення детальної інформації про кожен товар, а також можливість зворотного зв'язку з клієнтами через форму зворотного зв’язку.

Предмет даної кваліфікаційної роботи – це розробка інтернет-магазину з продажу товарів японської тематики з використанням фреймворку Laravel. Цей проект спрямований на створення зручного та ефективного майданчика для покупців, які цікавляться японською культурою та товарами. Він також відображає інноваційний підхід до використання сучасних технологій у розвитку електронної комерції.

Наукова новизна даної роботи полягає у використанні фреймворку Laravel для розробки інтернет-магазину з продажу товарів японської тематики. Такий

підхід дозволяє не лише забезпечити високу якість розробленого продукту, але й застосувати сучасні технології у вирішенні практичних завдань електронної комерції. Крім того, унікальність полягає і в самій тематиці магазину, яка спрямована на задоволення попиту на товари японської культури, що може бути актуальним та привабливим для широкого кола покупців.

Практичне значення одержаних результатів дослідження полягає у створенні функціонального та ефективного інтернет-магазину, спеціалізованого на японських товарах. Цей магазин може стати цінним ресурсом для клієнтів, які зацікавлені в придбанні аутентичних японських товарів та продуктів. Такий проект може стимулювати розвиток електронної комерції та культурного обміну між країнами, а також сприяти поширенню та популяризації японської культури серед широкого загалу.

РОЗДІЛ 1
АНАЛІЗ ІНТЕРНЕТ-МАГАЗИНУ ЯПОНСЬКОЇ ТЕМАТИКИ НА LARAVEL І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Онлайн-магазин, або інтернет-магазин, представляє собою електронну торговельну платформу, де товари та послуги продаються та покупаються через Інтернет. Це віртуальний майданчик, який забезпечує зручну та доступну для користувачів можливість здійснювати покупки в будь-який час та з будь-якого місця, де є доступ до Інтернету.

Онлайн-магазини надають ряд переваг як для покупців, так і для продавців. Для покупців вони забезпечують зручність та доступність, оскільки вони можуть швидко та легко знаходити та придбавати потрібні товари, порівнюючи їх характеристики та ціни. Крім того, онлайн-магазини зазвичай пропонують широкий асортимент товарів, який може бути складніше знайти у звичайних кам’яних магазинах.

Для продавців інтернет-магазини відкривають нові можливості для розширення аудиторії та збільшення обсягів продажів. Вони дозволяють ефективно використовувати інтернет-технології для просування своїх товарів та послуг, залучення нових клієнтів та підтримки вже існуючих.

На сьогоднішній день спостерігається значний попит на японські товари як в Україні, так і в інших країнах світу. Японські продукти здобувають популярність завдяки своїй унікальності, якості та культурному значенню.

Серед основних напрямків розвитку цього ринку можна виділити постійне розширення асортименту товарів та покращення їхньої якості. Виробники вкладають зусилля у створення нових інноваційних продуктів, що відповідають потребам споживачів.

За останні роки помітно зросла кількість попиту на японські товари, що призвело до постійного збільшення їхнього асортименту на ринку. Покупці

цінують японські товари за їхню унікальність, якість та естетичний дизайн, що додає привабливості цим товарам.

На сьогоднішній день існують численні приклади успішних інтернет-магазинів, спеціалізованих на японських товарах, які демонструють потенціал цього ринку. Ці інтернет-магазини пропонують широкий вибір товарів, що включає електроніку, одяг, косметику, подарункові товари та інші товари з Японії. Вони відомі своєю надійністю, високою якістю продукції та швидкою доставкою, що приваблює клієнтів з різних куточків світу.

Приклади таких інтернет-магазинів включають «Murasaki», який зображений на рисунку 1.1, «Pulsar», «Fairy Fox» та багато інших. Ці компанії активно розвиваються та залучають нових клієнтів завдяки своєму великому асортименту, конкурентоспроможним цінам та відмінному обслуговуванню клієнтів.

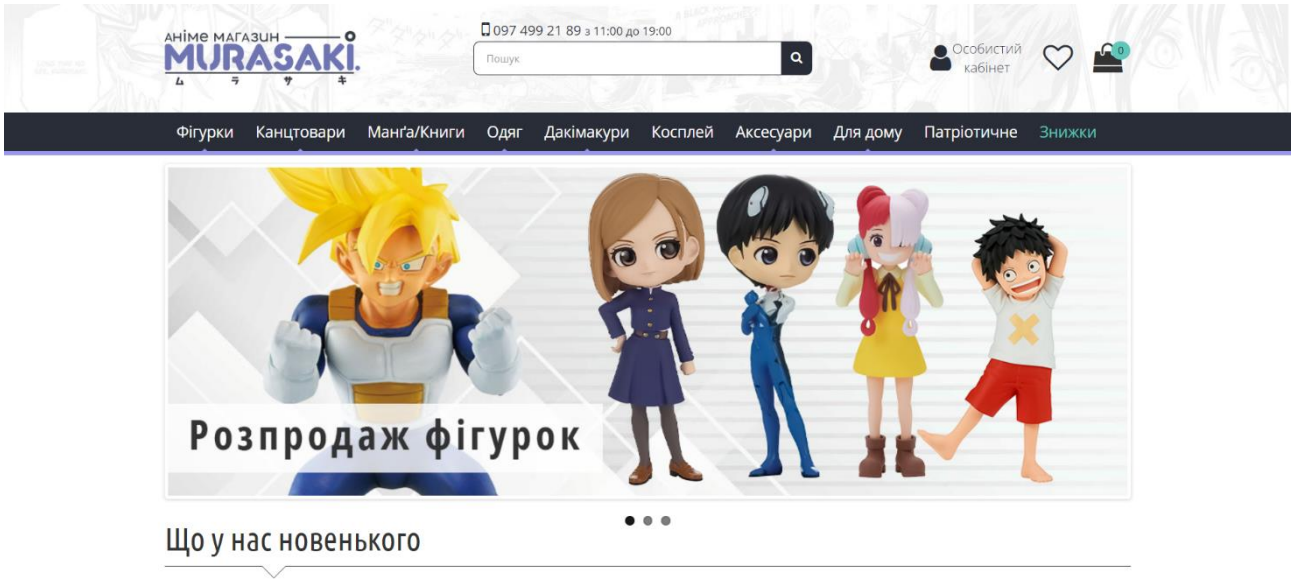

Рисунок 1.1 – Онлайн-магазин «Murasaki» [1]

Успіх цих інтернет-магазинів свідчить про зростаючий інтерес споживачів до японських товарів та підтверджує потенціал розвитку цього сегмента ринку. Це також стимулює нових підприємців до входу на цей ринок та розширення

свого бізнесу, що сприяє подальшому розвитку та розширенню асортименту японських товарів для споживачів у всьому світі.

Ще одним з прикладів можна навести інтернет-магазин «Yorokobi», що зображений на рисунку 1.2. Цей магазин пропонує широкий вибір японських товарів, зокрема фігурок, аксесуарів, коміксів та подарунків. Однак, на жаль, поруч із його позитивними аспектами можна виокремити кілька суттєвих недоліків.

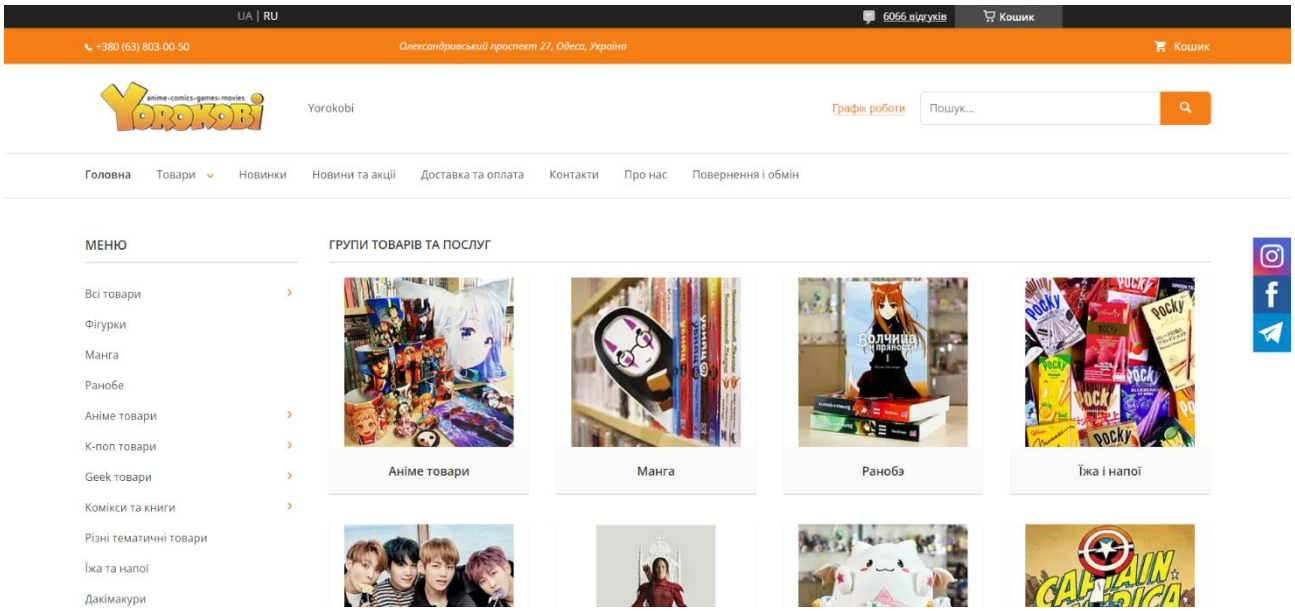

Рисунок 1.2 – Онлайн-магазин «Yorokobi» [2]

Можна також виділити застарілий дизайн, який не відповідає сучасним стандартам веб-дизайну. Крім того, погана адаптація під мобільні пристрої обмежує зручність користування магазином для мобільних користувачів, що може призвести до втрати потенційних клієнтів.

Зайшовши в консоль розробника, можна побачити багато помилок (рис. 1.3), що свідчить про неоптимальну реалізацію коду та може вплинути на швидкість роботи сайту та його загальну продуктивність.

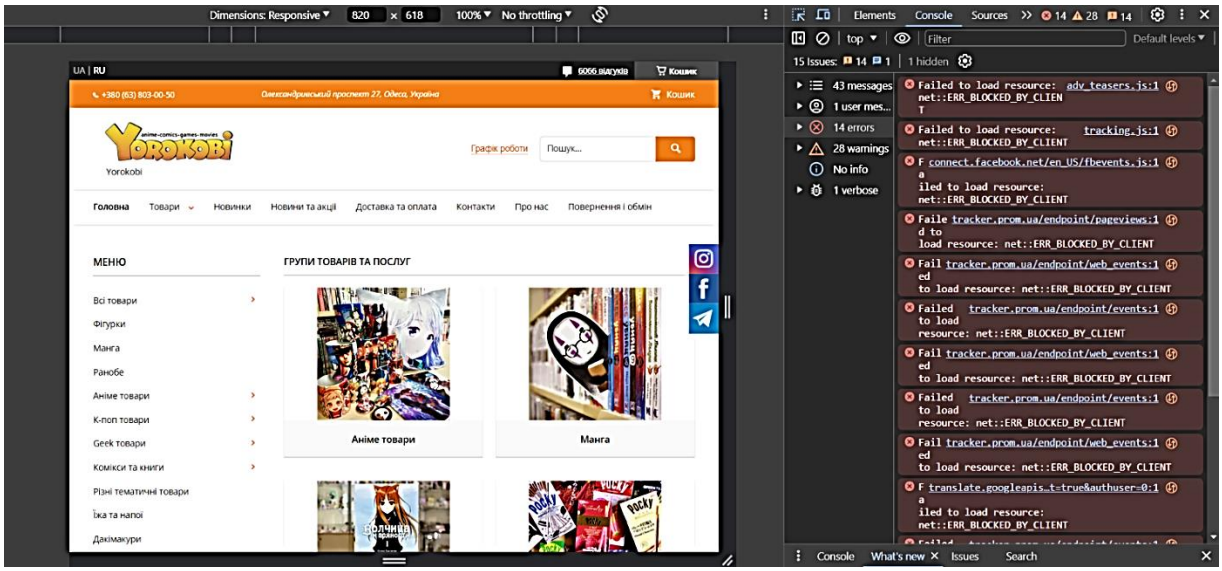

Рисунок 1.3 – Приклад помилок в консолі магазину «Yorokobi»

Також можна побачити продуктивність сайту за допомогою PageSpeed Insights від Google (рис. 1.4). Ефективність цього сайту оцінюється на рівні 41, що свідчить про те, що сайт потребує оптимізації для поліпшення швидкості завантаження сторінок та загальної продуктивності. Низька оцінка ефективності може вплинути на користувацький досвід та знизити конверсію сайту, оскільки швидкість завантаження впливає на час очікування користувачів та їхню відповідь на сайтовий контент.

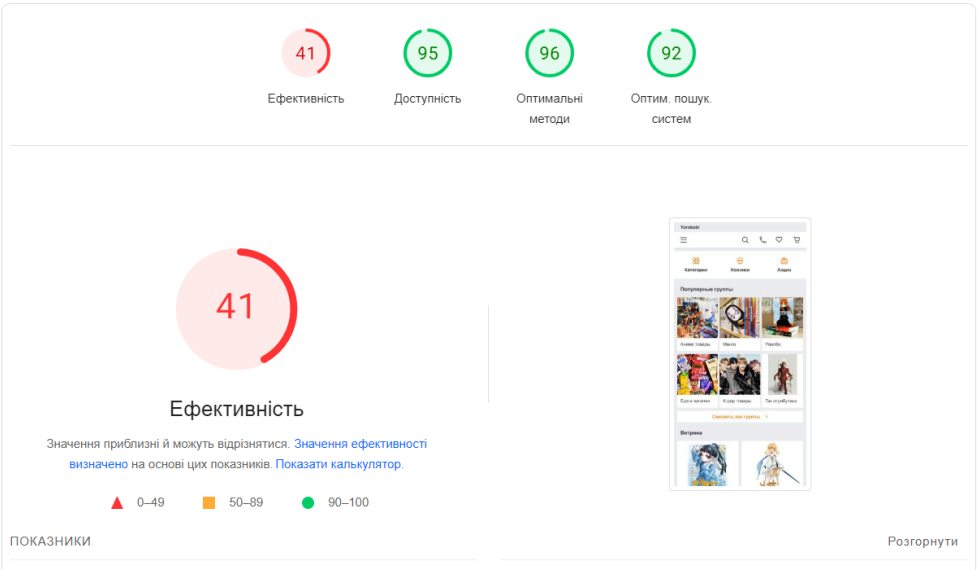

Рисунок 1.4 – Тест швидкодії інтернет-магазину «Yorokobi»

Попри значний попит на японські товари, існуючі інтернет-магазини мають свої переваги та недоліки. До переваг можна віднести широкий асортимент товарів, якість обслуговування та впізнаваність бренду. Однак, серед недоліків можна виокремити застарілий дизайн інтерфейсу, обмежений асортимент у деяких випадках та недостатню акцентуацію на унікальності продукції.

Аналізуючи існуючі інтернет-магазини, можна виявити кілька потенційних ніш та прогалин, які можна ефективно використати для створення нового інтернет-магазину. Наприклад, спеціалізація на ексклюзивних ручно виготовлених товарах з різних регіонів Японії або впровадження програми лояльності для постійних клієнтів може стати ключовими конкурентними перевагами на ринку.

Також варто приділіити особливу увагу сучасному дизайну сайту. Багато існуючих інтернет-магазинів страждають від застарілого дизайну, що негативно позначається на досвіді користувачів. Сучасний дизайн сайту має включати такі елементи:

– чистий та мінімалістичний інтерфейс;
– адаптивний дизайн;
– висока швидкість завантаження;
– якісні зображення та відео;
– інтерактивні елементи;
– зручна навігація;
– покращена пошукова функція.

Впровадження цих аспектів сучасного дизайну дозволить новому інтернет-магазину виділитися серед конкурентів, підвищити рівень задоволення клієнтів і сприяти зростанню продажів.

Окрім цього, варто звернути увагу на зростаючу популярність манги та ранобе, перекладених українським виробництвом. Цей сегмент швидко розвивається та набирає все більшої популярності серед українських читачів.

Включення до асортименту інтернет-магазину манги та ранобе українською мовою може залучити нову аудиторію та значно розширити клієнтську базу.

Таким чином, поєднання спеціалізації на ексклюзивних японських товарах, програми лояльності для постійних клієнтів та продажу перекладених українським виробництвом манги та ранобе, поряд із сучасним дизайном сайту, дозволить новому інтернет-магазину виділитися серед конкурентів, підвищити рівень задоволення клієнтів і сприяти зростанню продажів.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Власне проект розробляється для того, щоб клієнти могли зручно обрати потрібні японські товари та без проблем замовити їх, а менеджери сайту мали змогу легко керувати всіма аспектами роботи сайту через адміністративну панель. Цей проект спрямований на створення інтернет-магазину, який не лише відповідатиме сучасним стандартам веб-дизайну та юзабіліті, але й враховуватиме специфіку українського ринку, включаючи переклад манги та ранобе українським виробництвом.

Інтерфейс користувача (UI) – це зовнішній вигляд та функціональність сайту, з якими взаємодіє відвідувач [3]. Важливо, щоб інтерфейс був інтуїтивно зрозумілим, естетично привабливим та зручним у використанні. Основні критерії елементів інтерфейсу користувача включають:

– чистий та мінімалістичний дизайн, який спрощує навігацію і робить сайт привабливішим;

– адаптивний дизайн, що забезпечує коректне відображення та зручність користування сайтом на різних пристроях, таких як комп'ютери, планшети та смартфони;

– висока швидкість завантаження, що позитивно впливає на користувацький досвід і знижує відсоток відмов;

– зручна навігація, яка полегшує пошук необхідних товарів та інформації;

– фільтри, які дозволяють користувачам швидко відфільтрувати товари за категоріями, ціною, брендом та іншими параметрами, спрощуючи процес вибору та підвищуючи задоволеність від користування сайтом.

Адміністративна частина сайту (бекенд) дозволяє менеджерам керувати контентом, замовленнями, клієнтами та іншими аспектами роботи інтернет-магазину. Важливо, щоб адміністративна панель була функціональною та зручною у використанні. Основні елементи адміністративної частини включають:

– додавання, редагування та видалення товарів, завантаження зображень та описів;

– перегляд, обробка та зміна статусів замовлень, сповіщення клієнтів.

Впровадження цих елементів забезпечить ефективне управління інтернет-магазином, дозволить швидко реагувати на зміни ринку та потреби клієнтів, а також забезпечить високий рівень задоволеності як клієнтів, так і менеджерів.

Висновки до розділу 1

У цьому розділі було проведено аналіз існуючих інтернет-магазинів, які спеціалізуються на японських товарах. Було виявлено, що більшість з них мають як сильні, так і слабкі сторони. Серед основних недоліків часто зустрічаються застарілий дизайн, низька адаптивність під мобільні пристрої, повільна швидкість завантаження та технічні помилки, які негативно впливають на користувацький досвід. Також було зазначено, що деякі магазини не пропонують достатню кількість унікальних та ексклюзивних товарів, що могло б стати їхньою конкурентною перевагою.

На основі проведеного аналізу, було сформульовано основні завдання для розробки інтернет-магазину, який відповідатиме сучасним стандартам дизайну та юзабіліті. Інтерфейс користувача повинен включати чистий та мінімалістичний дизайн, адаптивний інтерфейс, високу швидкість завантаження, якісні зображення та відео, інтерактивні елементи, зручну навігацію, покращену

пошукову функцію та фільтри для товарів. Адміністративна частина сайту має забезпечити зручне управління товарами, замовленнями, клієнтами, аналітикою та налаштуваннями сайту, а також забезпечити високий рівень безпеки даних.

Впровадження цих елементів дозволить створити інтернет-магазин, який не лише відповідатиме потребам користувачів, але й виділятиметься серед конкурентів, підвищуючи рівень задоволеності клієнтів та забезпечуючи зростання продажів.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

Метою проекту є створення інтернет-магазину з продажу товарів японської тематики, який забезпечуватиме зручність користування, високу продуктивність та надійність. Для досягнення цієї мети необхідно визначити вимоги до програмного забезпечення, які гарантуватимуть задоволення потреб як кінцевих користувачів, так і адміністрації сайту.

Функціональні вимоги до програмного забезпечення включають:

– відображення списку товарів – список товарів повинен містити основну інформацію про кожен товар, включаючи назву, зображення, ціну та короткий опис;

– фільтрація товарів – користувачі повинні мати можливість фільтрувати товари за різними критеріями, такими як категорія, ціна, бренд, наявність на складі тощо. Це допоможе швидко знайти потрібний товар серед широкого асортименту;

– детальна сторінка товару – при натисканні на товар у списку, користувач повинен потрапити на детальну сторінку товару, яка містить розширений опис, технічні характеристики, кілька зображень, відеоогляди (якщо доступні), а також відгуки інших покупців;

– пошук товарів – інтернет-магазин повинен мати функцію пошуку, що дозволяє користувачам швидко знаходити товари за ключовими словами;

– додавання та видалення товарів – користувачі повинні мати можливість додавати товари до кошика одним кліком, а також видаляти їх або змінювати кількість товарів безпосередньо в кошику;

– перегляд вмісту кошика – кошик повинен містити перелік обраних товарів із зазначенням їх кількості, ціни за одиницю та загальної вартості.

Користувачі повинні мати можливість переглядати та редагувати вміст кошика перед оформленням замовлення;

– оформлення замовлення – процес оформлення замовлення повинен бути простим і зрозумілим. Він включає введення контактної інформації, вибір способу доставки та оплати, а також підтвердження замовлення.

В адміністративній частині повинні бути реалізовані наступні функції:

– управління каталогом товарів – адміністратори мають мати можливість додавати нові товари, редагувати дані про існуючі товари та видаляти їх з каталогу. Це включає управління зображеннями, описами, цінами та іншими характеристиками товарів;

– управління замовленнями – адміністратори повинні мати доступ до переліку всіх замовлень з можливістю змінювати їх статус, переглядати деталі кожного замовлення, редагувати інформацію про замовлення та видаляти замовлення за необхідності.

Нефункціональні вимоги визначають якісні характеристики системи, які є критично важливими для її успішного функціонування. Вони не пов'язані безпосередньо з конкретними функціями, але впливають на загальну ефективність, зручність використання та надійність системи.

Нефункціональні вимоги для мого інтернет-магазину включають:

– продуктивність – швидке завантаження сторінок та оптимізація бази даних для ефективного доступу;

– адаптивність – коректне відображення на різних пристроях;

– юзабіліті – простий і зрозумілий інтерфейс, зручна навігація та мінімалістичний дизайн.

Також мною була створена UseCase діаграма (рис. 2.1), яка відображає різні можливості та сценарії взаємодії користувачів з інтернет-магазином. Ця діаграма допомагає краще зрозуміти, які операції доступні різним типам користувачів та як вони взаємодіють з системою.

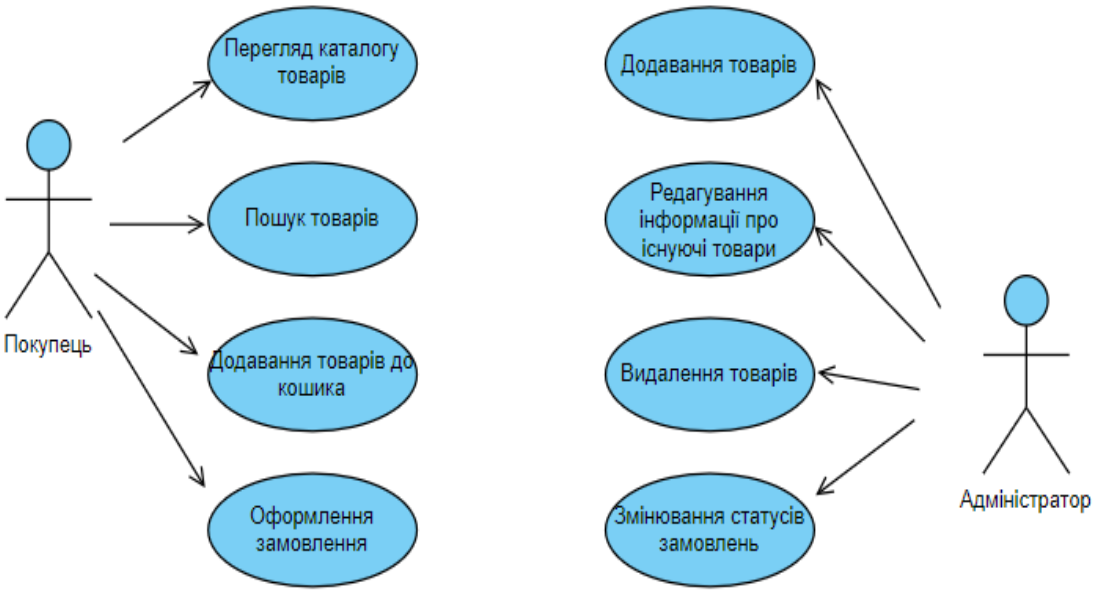

Рисунок 2.1 – UseCase діаграма взаємодії між користувачами системи

На представленій UseCase діаграмі зображено взаємодію між користувачем та адміністратором інтернет-магазину. Ця діаграма відображає основні функціональні можливості, доступні як для звичайних користувачів, так і для адміністраторів системи. Користувачі мають можливість переглядати товари, додавати їх до кошика та здійснювати замовлення. З іншого боку, адміністратори можуть керувати каталогом товарів, переглядати та змінювати статуси замовлень. Ця діаграма є важливим інструментом для розуміння взаємодії між різними типами користувачів та функціоналом системи.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для створення інтернет-магазину, що спеціалізується на продажу товарів японської тематики, я вирішив використати фреймворк Laravel (рис. 2.2). Давайте детальніше розглянемо, що таке Laravel і чому він був обраний для цього проекту.

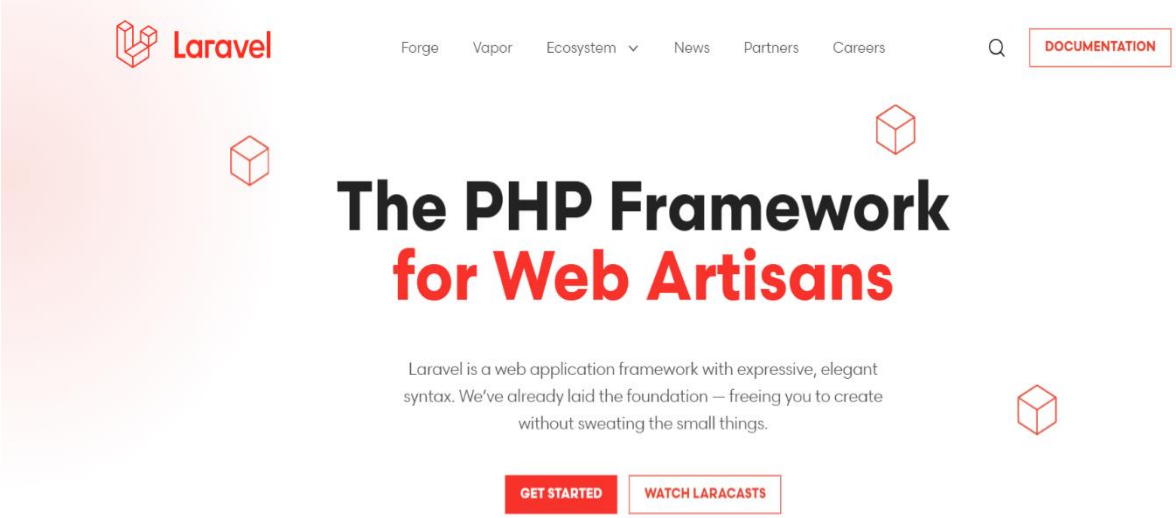

Рисунок 2.2 – Офіційний сайт Laravel [4]

Laravel – це відомий веб-фреймворк для розробки веб-додатків з використанням мови програмування PHP. Він надає зручні інструменти та структуру для розробки швидких і ефективних веб-додатків. Laravel використовується для створення різноманітних веб-додатків, включаючи інтернет-магазини, блоги, форуми та інші онлайн-платформи [5].

Обравши фреймворк Laravel для розробки інтернет-магазину, я розглянув ряд переваг, які він пропонує:

– продуктивність розробки;
– розширюваність;
– зручна документація;
– спільнота та підтримка.

Однак, варто враховувати деякі недоліки, такі як великий розмір і складність. Laravel може бути надмірно складним і важким для невеликих проектів, що може вимагати додаткового часу на його освоєння.

У фронтенд-розробці інтернет-магазину використовувалися HTML, CSS, JavaScript та Bootstrap. Кожна з цих технологій відігравала важливу роль у

створенні зовнішнього вигляду та функціональності веб-сторінок. Давайте детальніше розглянемо, як саме вони застосовувалися у процесі розробки.

HTML є мовою розмітки, що визначає структуру веб-сторінок. Вона використовується для створення різних елементів веб-сторінок, таких як заголовки, параграфи, списки, таблиці та форми. HTML є фундаментальною складовою фронтенд-розробки, оскільки визначає основну структуру та контент веб-сторінок [6].

CSS (Cascading Style Sheets) є основною мовою для стилізації веб-сторінок, але у процесі розробки інтернет-магазину був використаний препроцесор SASS (Syntactically Awesome Stylesheets). SASS дозволяє писати CSS з більшою ефективністю та організованістю, використовуючи функції, змінні, вкладеність та інші корисні функції, що спрощує управління стилями та робить код більш чистим та організованим.

Крім того, увага була приділена адаптивності інтернет-магазину. Адаптивний дизайн дозволяє веб-сторінкам коректно відображатися на різних пристроях та розмірах екранів, таких як комп’ютери, планшети та смартфони. Це досягається шляхом використання медіа-запитів, гнучких макетів та інших технік, які забезпечують оптимальний перегляд контенту незалежно від пристрою користувача.

JavaScript – це мова програмування, яка використовується для надання інтерактивності на веб-сторінках. Вона використовується для створення динамічного контенту, валідації форм, анімацій, взаємодії з користувачем та інших функцій, які покращують користувацький досвід [7].

В розробці також використовувався фреймворк Bootstrap, що зображений рисунку 2.3, став невід’ємною частиною створення інтерактивного та естетичного користувацького інтерфейсу. Bootstrap – це потужний інструмент, який надає широкий спектр готових компонентів та стилів для швидкого створення сучасного та адаптивного дизайну веб-сторінок. Завдяки Bootstrap можна легко і швидко створити такі елементи, як кнопки, форми, навігаційні

панелі, каруселі, модальні вікна та інші, що дозволяє створювати візуально привабливі та функціональні інтерфейси без великих затрат часу та зусиль.

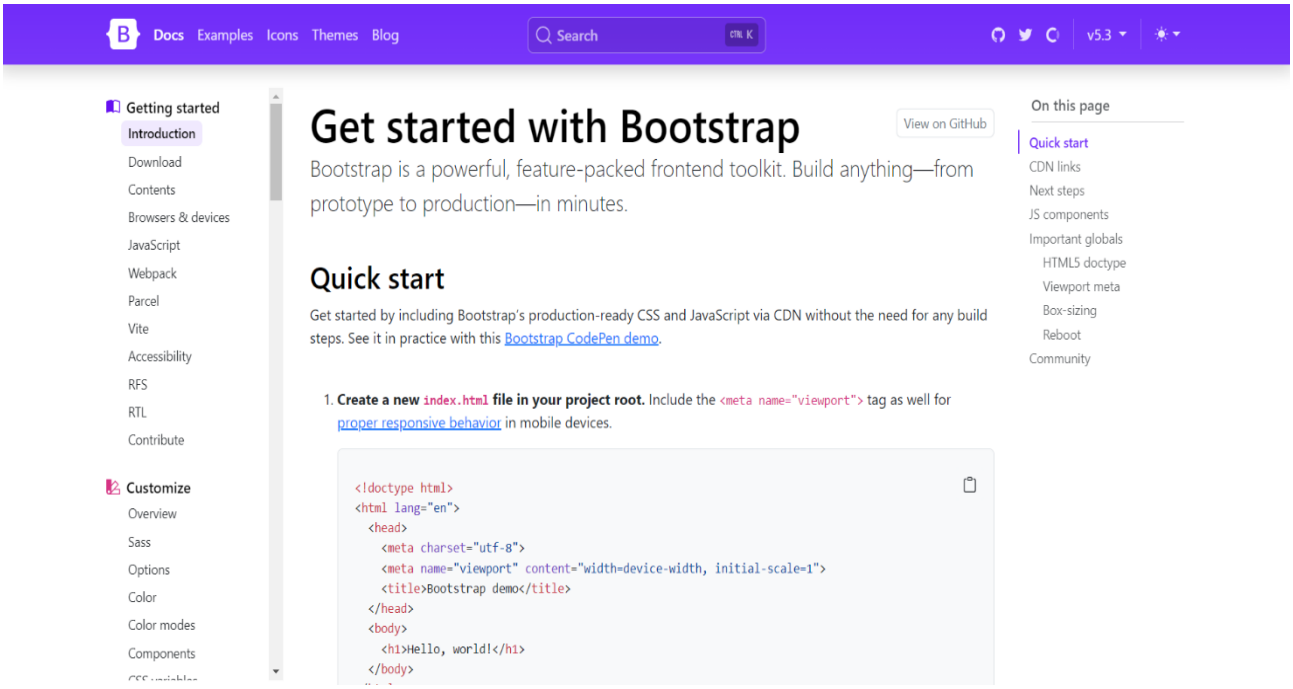

Рисунок 2.3 – Документація Bootstrap 5 [8]

Крім того, Bootstrap має широку підтримку серед розробників та постійно оновлюється, що забезпечує актуальність та сучасність розробленого веб-інтерфейсу. Його модульна структура дозволяє легко налаштовувати та розширювати функціональність, що робить Bootstrap ідеальним вибором для реалізації вимог щодо створення зручного та ефективного інтернет-магазину.

У процесі розробки, крім використання HTML, CSS, JavaScript та Bootstrap, також активно залучалася бібліотека jQuery (рис. 2.4). jQuery – це швидка, легка та потужна бібліотека JavaScript, яка спрощує взаємодію з документом HTML, обробку подій, анімацію та роботу з AJAX. Вона дозволяє зручно вибирати та маніпулювати елементами веб-сторінки, виконувати анімаційні ефекти, обробляти події користувача та здійснювати взаємодію з сервером без перезавантаження сторінки [9]. jQuery використовувалася для реалізації різноманітних функціональних можливостей, таких як динамічне оновлення вмісту, анімація, валідація форм, взаємодія з API та інше.

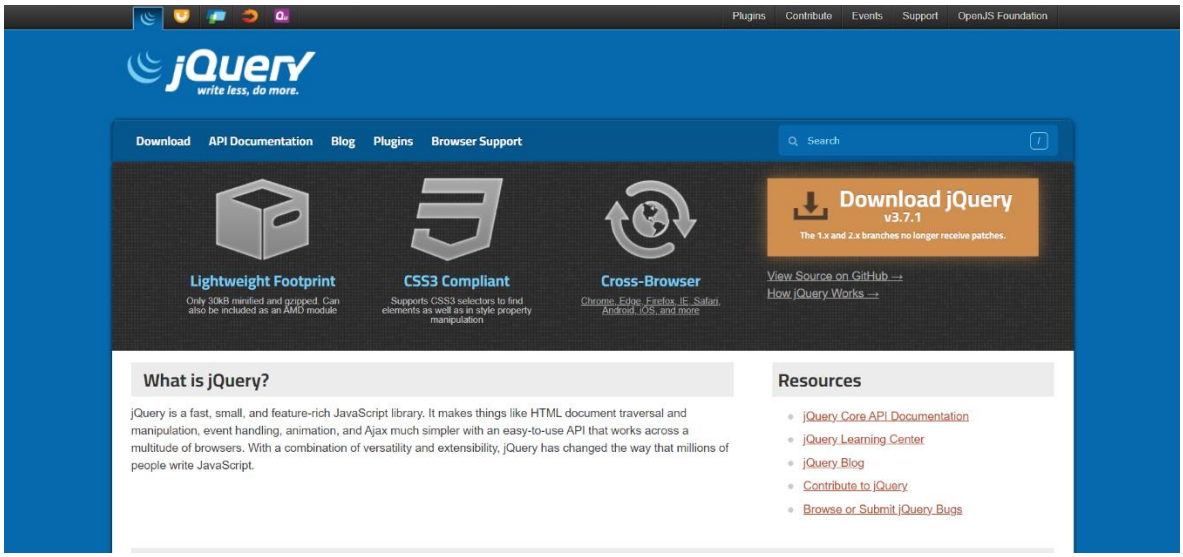

Рисунок 2.4 – Офіційний сайт бібліотеки jQuery [10]

Окрім того, для забезпечення адміністративної частини та більшої зручності управління інтернет-магазином, було використано шаблон адміністративного інтерфейсу AdminLTE (рис. 2.5).

AdminLTE – це безкоштовний та відкритий HTML-шаблон для адміністративних панелей та панелей управління, який має стильний та сучасний дизайн, а також включає в себе готові компоненти та інструменти для швидкого розгортання функціональної адміністративної панелі. Використання AdminLTE спростило процес розробки адміністративного інтерфейсу та дозволило швидко налаштувати потрібні функціональність та елементи керування веб-застосунком.

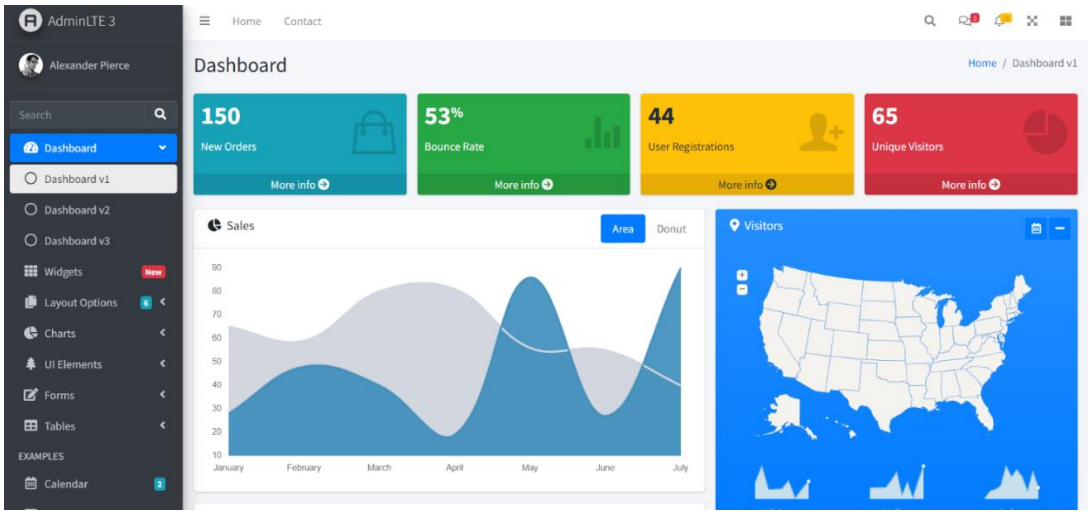

Рисунок 2.5 – Приклад шаблону інтерфейсу AdminLTE 3

Для зручності розробки також були використані Node.js та Vite.

Node.js – це середовище виконання JavaScript на сервері, яке дозволяє виконувати JavaScript-код поза веб-браузером. Він забезпечує можливість створювати серверні застосунки та обробляти запити від клієнтів, а також здійснювати взаємодію з базою даних. Node.js є потужним інструментом для розробки серверних додатків, особливо для тих випадків, коли швидкість та масштабованість мають велике значення [11].

Тепер щодо Vite (рис. 2.6). Vite – це інструмент для розробки фронтенду, який спеціалізується на швидкому збиранні та оптимізації веб-ресурсів. Він надає зручний інтерфейс для розробки веб-додатків та забезпечує ефективне управління ресурсами проекту. Завдяки Vite, є можливість швидко створювати веб-застосунки з високою продуктивністю та ефективністю.

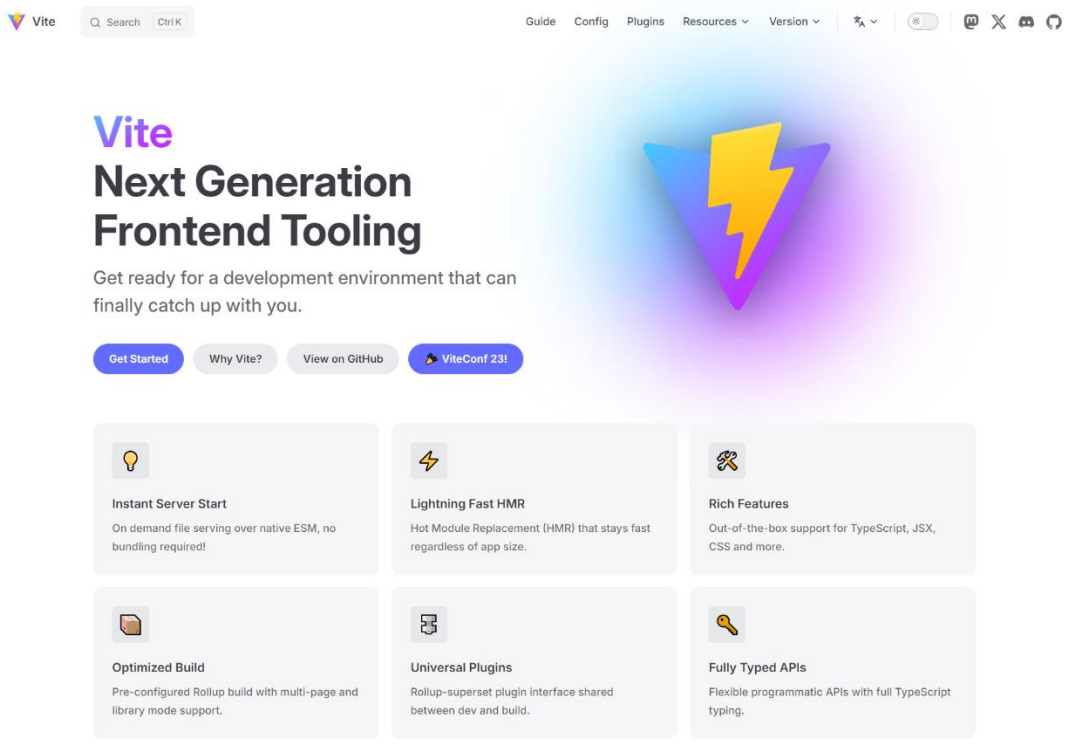

Рисунок 2.6 – Офіційний сайт Vite та його переваги [12]

Node.js було використано для створення та запуску серверної частини додатку, а також для взаємодії з базою даних. З іншого боку, Vite відповідав за швидке компілювання та оптимізацію фронтенду веб-додатка. Використання

цих інструментів дозволило розробляти та розгортати інтернет-магазин з високою продуктивністю та ефективністю.

Висновки до розділу 2

Виходячи з усього описаного в цьому розділі, був проведений ретельний аналіз вимог та проектування програмного забезпечення для інтернет-магазину з японськими товарами. На основі цього аналізу були визначені основні функціональні та нефункціональні вимоги до системи.

Функціональні вимоги охоплюють ключові можливості, такі як управління каталогом товарів, пошук товарів, оформлення замовлень та адміністрування системи. Нефункціональні вимоги визначають критерії продуктивності, безпеки, масштабованості та зручності використання, що необхідно враховувати для забезпечення якісної роботи інтернет-магазину.

Для розробки даного проекту було обрано фреймворк Laravel, враховуючи його переваги, такі як висока продуктивність розробки, розширюваність, зручна документація та активна спільнота. У поєднанні з Laravel були використані Node.js та Vite для забезпечення зручності розробки та оптимізації фронтенду.

Для адміністративної частини застосовувався шаблон AdminLTE, що дозволяє швидко створити функціональний та сучасний адміністративний інтерфейс. Також було створено діаграму класів для ілюстрації взаємодії між користувачами системи (покупцями та адміністраторами). Це допомагає краще зрозуміти організацію та функціональні можливості системи.

РОЗДІЛ 3
РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ З ПРОДАЖУ ТОВАРІВ ЯПОНСЬКОЇ ТЕМАТИКИ З ВИКОРИСТАННЯМ LARAVEL

3.1 Практична реалізація об’єкта проектування

Для початку необхідно встановити Composer. Composer – це пакетний менеджер для PHP, який дозволяє легко встановлювати, оновлювати та управляти залежностями проекту. Composer забезпечує автоматичне завантаження бібліотек і пакетів, які використовуються у проекті, що значно спрощує процес розробки [13].

Для встановлення Composer необхідно виконати наступні кроки. Спершу потрібно перейти на офіційну сторінку завантаження (рис. 3.1). На цій сторінці можна знайти інструкції для кожної операційної системи та завантажити установник, натиснувши на відповідне посилання.

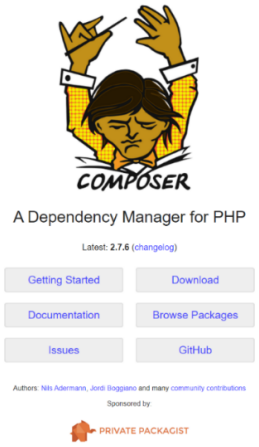

Рисунок 3.1 – Офіційна сторінка Composer [14]

Після завантаження установочного файлу я його запустив. Інсталятор автоматично завантажив та встановив Composer на мій комп'ютер. Під час установки я переконався, що шлях до PHP правильно налаштований, щоб Composer міг знайти і використовувати PHP.

Після успішного завершення установки, я відкрив командний рядок (термінал) та ввів наступну команду: composer.

Коли Composer було встановлено успішно, на екрані з’явився список доступних команд Composer та інформація про встановлену версію (рис. 3.2).

Рисунок 3.2 – Список доступних команд Composer

Для початку роботи створюємо новий проект Laravel (рис. 3.3) за допомогою команди: composer create-project laravel/laravel japanshop.

Ця команда виконується в терміналі і створює проект Laravel з назвою «japanshop», як це вказано в офіційній документації Laravel.

Рисунок 3.3 – Створення нового проекту Laravel

Після створення проекту Laravel генерує спеціальний ключ захисту, відомий як «App key». Цей ключ використовується для шифрування конфіденційної інформації, такої як паролі користувачів або ключі API, перед

зберіганням їх у базі даних або передачею через мережу. Використання App key гарантує, що навіть якщо хтось отримає несанкціонований доступ до бази даних або перехопить мережевий трафік, він не зможе прочитати конфіденційні дані без наявності ключа додатку.

Після створення проекту Laravel за допомогою Composer необхідно розгорнути його, щоб встановити всі необхідні залежності. Це можна зробити за допомогою команди: composer install.

Ця команда автоматично встановлює всі необхідні пакети та залежності, які вказані у файлі composer.json вашого проекту.

Після розгортання проекту Laravel необхідно згенерувати App Key, який використовується для шифрування деякої конфіденційної інформації в вашому додатку, такої як сесійні дані та дані про користувачів. Це можна зробити за допомогою команди: php artisan key:generate.

Ця команда створює новий ключ захисту і автоматично оновлює значення APP_KEY в файлі .env вашого проекту, який використовується для зберігання конфіденційних даних. Згенерований ключ допоможе забезпечити безпеку даних додатку, зашифруючи конфіденційну інформацію, перш ніж вона буде збережена у базі даних чи передана по мережі.

Після створення Laravel проекту та встановлення необхідних залежностей, переходимо до налаштування бази даних. Одним з важливих кроків є запуск міграцій, які відповідають за створення та оновлення структури бази даних.

Команда для запуску міграцій в Laravel виглядає наступним чином: php artisan migrate.

Міграції – це інструмент, який дозволяє організувати та керувати структурою бази даних. Вони представляють собою спосіб описувати зміни у схемі бази даних за допомогою коду.

Кожна міграція включає в себе пару методів: up() для виконання змін у базі даних та down() для відкату цих змін.

Міграції у Laravel можна розглядати як контроль версій для схеми бази даних. Вони дозволяють легко керувати структурою бази даних, вносячи зміни

та оновлення через код, що дозволяє підтримувати цю схему в актуальному стані на всіх етапах розвитку проекту [15].

Міграції будуть розглянуті більш детально в наступному пункті розділу «Розробка бази даних».

У Laravel для зручності розробки використовується механізм маршрутизації, який спрощує організацію та керування маршрутами.

Для конфігурації маршрутів для адміністративної частини веб-сайту використовується код, наведений у лістингу 3.1.

Лістинг 3.1 – Конфігурація маршрутів для адміністративної частини

```
<?php


use \Illuminate\Support\Facades\Route;

require __DIR__ . '/auth.php';

Route::group([
    'middleware' => ['auth'],
    'prefix' => 'admin',
    'as' => 'admin.',
], function () {
        Route::get('/',
[App\Http\Admin\Controllers\AdminController::class, '/admin']);
        Route::resource('categories',
\App\Http\Admin\Controllers\CategoryController::class)-
>except(['show']);
        Route::resource('brands',
\App\Http\Admin\Controllers\BrandController::class)-
>except(['show']);
        Route::resource('products',
\App\Http\Admin\Controllers\ProductController::class)-
>except(['show']);
        Route::resource('pages',
\App\Http\Admin\Controllers\PageController::class)-
>except(['show']);
});
```

Кінець лістингу 3.1

У цьому коді визначається група маршрутів, яка містить усі маршрути для адміністративної частини. Ця група має такі характеристики:

– middleware – вказано middleware для автентифікації, щоб перевірити, чи користувач має доступ до адмінських сторінок;

– prefix – встановлено префікс «/admin» для всіх маршрутів адмінки;

– as – вказано префікс «admin» для генерації іменованих маршрутів, що дозволяє зручно використовувати їх у коді.

Далі визначені окремі маршрути для кожного типу ресурсу, такі як категорії товарів, бренди, товари та сторінки, використовуючи функцію resource(). Ці маршрути дозволяють виконувати різноманітні дії з кожним типом ресурсу, такі як створення, читання, оновлення та видалення.

Конфігурація аутентифікації адміністратора означає створення маршрутів, що дозволяють адміністраторам увійти в систему та вийти з неї, а також оброблювати ці запити відповідними методами у контролері. Нижче наведено відповідний код з лістингу 3.2.

Лістинг 3.2 – Налаштування маршрутів для адміністраторів

```
<?php

use App\Http\Auth\Controllers\AuthenticatedSessionController;
use Illuminate\Support\Facades\Route;

Route::middleware(['guest'])->group(function () {

    Route::get('login',
[AuthenticatedSessionController::class, 'create'])
        ->name('login');

    Route::post('login',
[AuthenticatedSessionController::class, 'store']);
});

Route::middleware(['auth'])->group(function () {

    Route::post('logout',
[AuthenticatedSessionController::class, 'destroy'])
        ->name('logout');
});
```

Кінець лістингу 3.2

У цьому фрагменті коду маршрути для автентифікації адміністратора. Якщо адміністратор не автентифікований (гість), йому надається можливість увійти в систему через методи GET і POST для маршруту /login. Після автентифікації адміністратор може використовувати маршрут для виходу, який встановлюється через метод POST для /logout.

Для використання шаблонів AdminLTE3 у нашому проекті, ми спочатку встановлюємо пакет, який відповідає за шаблонізацію форм. Це робиться за допомогою команди: composer require almasaeed2010/adminlte – dev.

Після цього, щоб мати доступ до прикладів шаблонів HTML, нам потрібно скопіювати папку з цими прикладами. Це також виконується за допомогою composer, командою: php artisan migrate.

Після встановлення пакету AdminLTE3 за допомогою Composer та копіювання прикладів шаблонів HTML, ми можемо розгорнути цей пакет у нашому проекті за допомогою команди: php artisan lte3:install.

Ця команда дозволяє автоматично налаштувати необхідні файли та ресурси, щоб забезпечити коректне функціонування шаблонів AdminLTE3 в нашому Laravel-додатку.

Крім того, для поліпшення роботи з файлами та оптимізації зображень, ми встановлюємо додатковий пакет за допомогою Composer: fomvasss/laravel-medialibrary-extension

Цей пакет надає нам більш зручні інструменти для роботи з медіа-файлами, включаючи конвертацію зображень у більш оптимізований формат та розширені функції для їх управління. Це дозволяє нам забезпечити ефективніше та зручніше управління зображеннями та іншими медіа-файлами у нашому проекті.

При створенні Laravel-проекту за допомогою Composer, Vite автоматично стає складовою частиною структури проекту. Після цього можна починати використовувати Vite для розробки фронтенду, завантажуючи необхідні залежності та налаштовуючи проект за допомогою конфігураційних файлів.

На рисунку 3.4 можна побачити уривок з мого файлу package.json, де містяться залежності та скрипти для мого Laravel-проекту.

Рисунок 3.4 – Уривок з файлу package.json

Цей файл package.json використовується для визначення всіх залежностей мого проекту, а також для налаштування різних скриптів, що допомагають у розробці та збірці проекту.

У файлі bootstrap.js я налаштовую jQuery для свого веб-сайту, лістинг 3.3.

Лістинг 3.3 – Налаштування jQuery

```
import jQuery from 'jquery';

import axios from 'axios';

window.$ = jQuery;
window.axios = axios;
```

Кінець лістингу 3.3

Цей файл bootstrap.js використовується для ініціалізації та налаштування різних бібліотек та фреймворків, таких як jQuery та Axios, які використовуються у моєму проекті.

Після завершення налаштування залежностей і основних інструментів для розробки, настав час приступити до верстки інтернет-магазину. Для полегшення

роботи і створення більш організованої та легко змінюваної верстки, я вирішив встановити SCSS (Sassy CSS), що є розширенням мови CSS з багатьма корисними функціями.

Першим кроком було встановлення SCSS за допомогою плагіну. Це було зроблено за допомогою команди: npm i -D sass.

Де npm є менеджером пакетів для JavaScript і sass – пакетом для компіляції SCSS у звичайний CSS. Ця команда дозволяє встановити sass як залежність розробки (-D), щоб використовувати його тільки під час розробки, а не включати його в продакшен-залежності.

Після встановлення плагіну SCSS необхідно підключити його до основного файлу JavaScript проекту для його подальшої компіляції. Це було зроблено в файлі main.js, де я використовував директиву import для включення шляху до файла стилів SCSS: import "../../scss/app.scss".

Ця директива імпортує стилі з файлу app.scss, розташованого у папці scss, в основний файл JavaScript проекту. Таким чином, всі стилі, що знаходяться в файлі app.scss, будуть доступні для використання в проекті і будуть компілюватися у звичайний CSS під час збірки проекту.

Після завершення структурування HTML-коду і розміщення всіх необхідних тегів на сторінці, настав час перейти до надання їм вигляду за допомогою CSS стилів. У багатьох випадках я вирішив скористатися існуючими стилізованими компонентами з бібліотеки Bootstrap, оскільки це дозволяло швидше та з меншими зусиллями розробляти сторінки. Однак, оскільки кожен проект має свої унікальні особливості та дизайнерські вимоги, необхідно було також додати власні кастомні стилі для досягнення бажаного вигляду та відчуття сторінок. Таким чином, поєднання вже готових компонентів Bootstrap і власних CSS стилів дозволило створити гармонійний та естетичний дизайн для мого проекту (рис. 3.5).

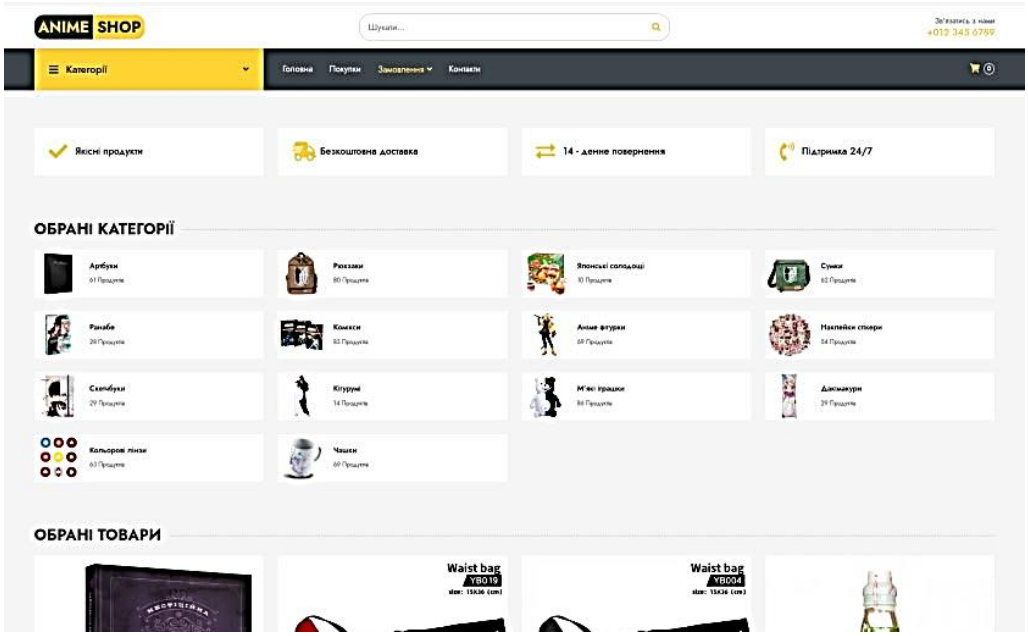

Рисунок 3.5 – Дизайн головної сторінки

Після завершення верстки сторінок інтернет-магазину, настав час для розробки функціоналу додавання товарів до кошика. Для цього було використано JavaScript, і написано код, який активується при натисканні на кнопку «Придбати» для кожного товару (рис. 3.6). У кошику реалізовано можливість збільшувати або зменшувати кількість товару, а також видаляти певний товар із кошика. Також був написаний PHP-код, який зберігає відомості про кошик кожного користувача, включаючи інформацію про додані продукти, їх кількість та ціни. Детальна реалізація цього функціоналу описана у додатку Б.

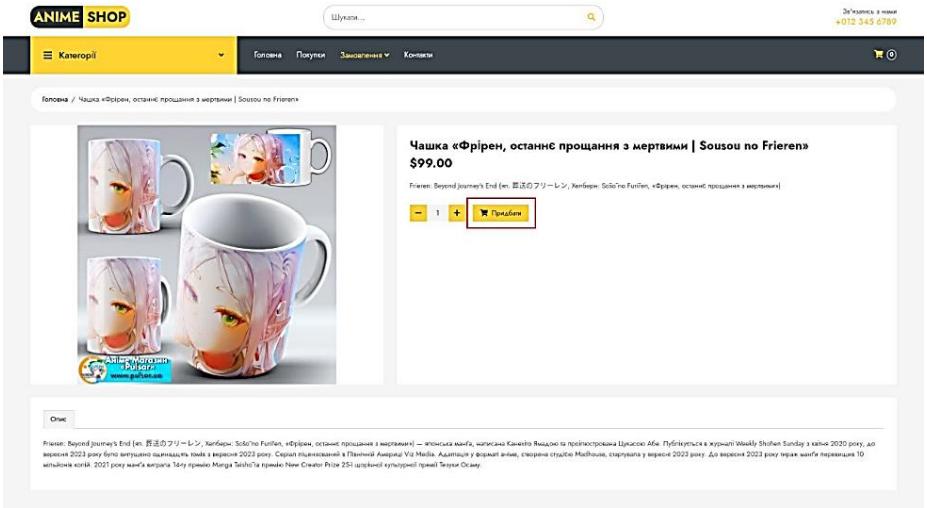

Рисунок 3.6 – Сторінка товару та кнопка «Придбати»

Також окремо був реалізований код для пошуку товарів. Відповідний PHP-код зображений у лістингу 3.4. Цей код дозволяє здійснювати пошук за назвою товару, брендом або категорією.

Лістинг 3.4 – Налаштування пошуку

```
      $builder->when($value = Arr::get($attrs, 'search'), fn ($b) =>
                  $b->where('name', 'like', '%' . $value . '%')
                      ->orWhereHas('brand',   function   ($q)   use
($value) {
                          $q->where('name', 'like', '%' . $value .
'%');
                      })
                      ->orWhereHas('category',  function  ($q)  use
($value) {
                          $q->where('name', 'like', '%' . $value .
'%');
                      })
              );
```

Кінець лістингу 3.4

Наступним кроком є налаштування фільтрації продуктів. Це дозволить користувачам шукати товари за різними критеріями. Для реалізації цієї функціональності я використовував потужний інструмент Laravel Eloquent, який полегшує роботу зі складними запитами до бази даних. Відповідний код наведено у лістингу 3.5.

Лістинг 3.5 – Налаштування фільтрації

```
      $builder->when($value = Arr::get($attrs, 'search'), fn ($b) =>
                  $b->where('name', 'like', '%' . $value . '%')
                      ->orWhereHas('brand',   function   ($q)   use
($value) {
                          $q->where('name', 'like', '%' . $value .
'%');
                      })
                      ->orWhereHas('category',  function  ($q)  use
($value) {
```

```
                        $q->where('name', 'like', '%' . $value .
'%');
                    })
            );
```

Кінець лістингу 3.5

Цей код ілюструє ефективне використання Laravel Eloquent для складання складних запитів до бази даних з урахуванням різних критеріїв фільтрації, що є важливим для створення функціонального і швидкодіючого інтерфейсу для користувачів додатку.

На рисунку 3.7 представлено інтерфейс користувача, який дозволяє користувачам фільтрувати продукти за різними критеріями, такими як бренд, категорія та ціна.

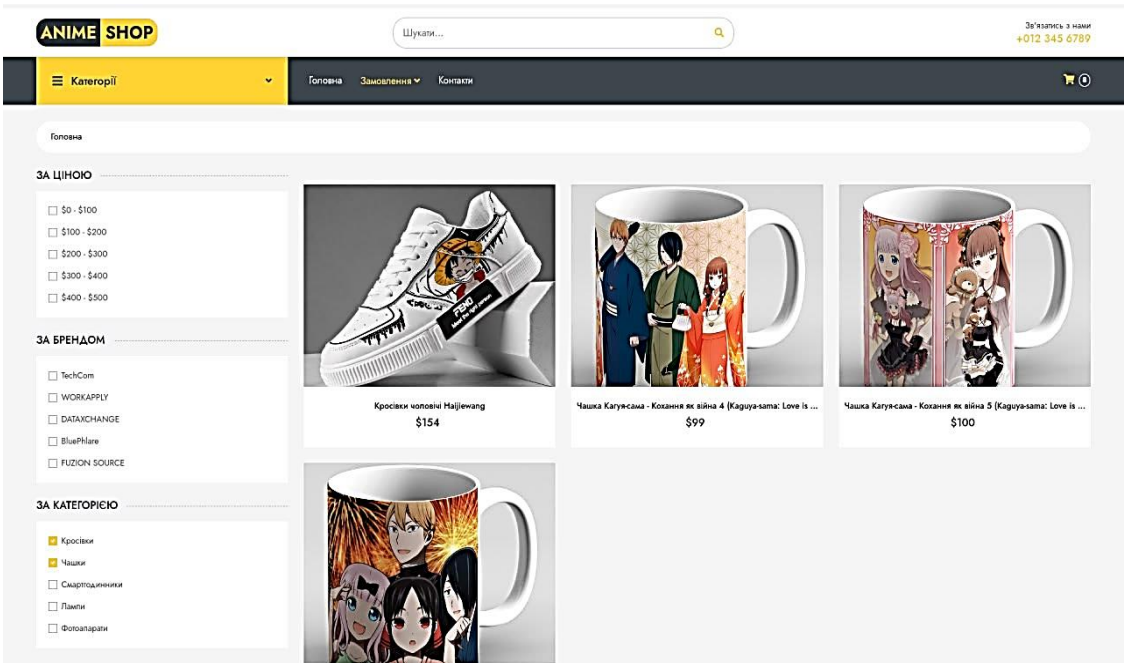

Рисунок 3.7 – Фільтрація товарів

Використання фільтрів спрощує процес пошуку і дозволяє знайти оптимальний варіант товару, відповідно до вимог і потреб користувача, зробивши процес покупки більш приємним та ефективним.

Також не слід забувати про панель адміністратора. Для доступу до неї необхідно до адреси URL додати «/admin». Після цього відкривається сторінка авторизації, де адміністратор має ввести свої облікові дані, такі як email та пароль (рис. 3.8).

Рисунок 3.8 – Сторінка авторизації адміністатора

Цей процес забезпечує безпеку адміністративного доступу, обмежуючи його тільки для авторизованих користувачів. Введення правильної комбінації email та паролю відкриває доступ до панелі управління, де адміністратор може виконувати різні завдання, включаючи управління товарами, категоріями, обробку замовлень та інші адміністративні функції, необхідні для ефективного управління інтернет-магазином.

Беручи до уваги процес редагування та додавання товарів, необхідно спершу перейти до розділу «Товари» у панелі адміністратора. Для цього з лівого боку обираємо відповідний розділ, після чого відкривається сторінка з усіма наявними товарами. На цій сторінці адміністратор може переглядати всі додані товари, редагувати існуючі або додавати нові. Для редагування товару достатньо вибрати потрібний товар зі списку і внести необхідні зміни в його опис, ціну, категорію та інші параметри. На рисунку 3.9 зображено процес редагування

товару, де адміністратор має можливість змінити інформацію про товар, забезпечуючи актуальність та точність даних у каталозі інтернет-магазину.

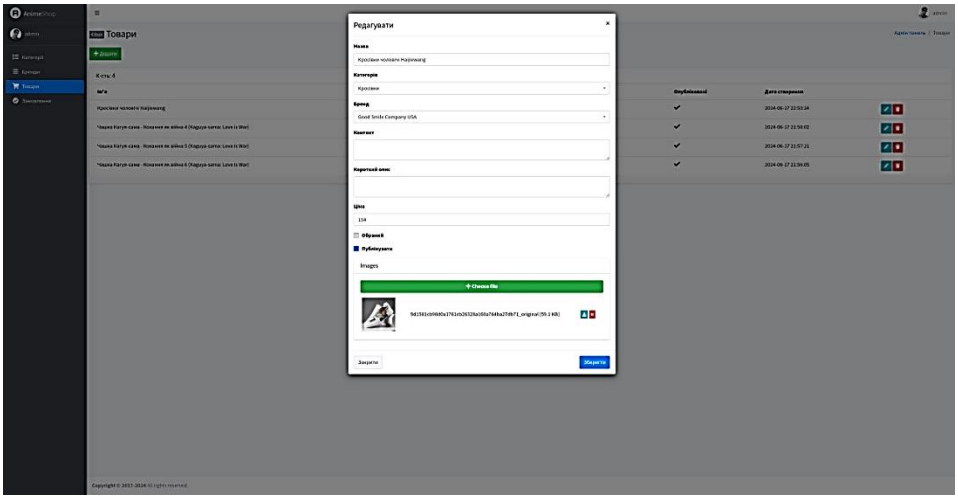

Рисунок 3.9 – Редагування товарів адміністратором

Також було додано можливість адміністратору переглядати та змінювати статуси замовлень. Для цього в панелі адміністратора передбачено розділ «Замовлення». Перейшовши до цього розділу, адміністратор може бачити всі наявні замовлення, аналогічно до розділу з товарами. При натисканні на конкретне замовлення відкривається детальна інформація про нього, включаючи список товарів, дані про клієнта, загальну суму та поточний статус замовлення. Адміністратор має можливість змінювати статус замовлення в залежності від його обробки, наприклад, позначати його як «в обробці», «не оплачено», «оплачено» чи «скасовано».

На рисунку 3.10 показано інтерфейс управління замовленнями, де можна зручно відслідковувати і редагувати статуси замовлень.

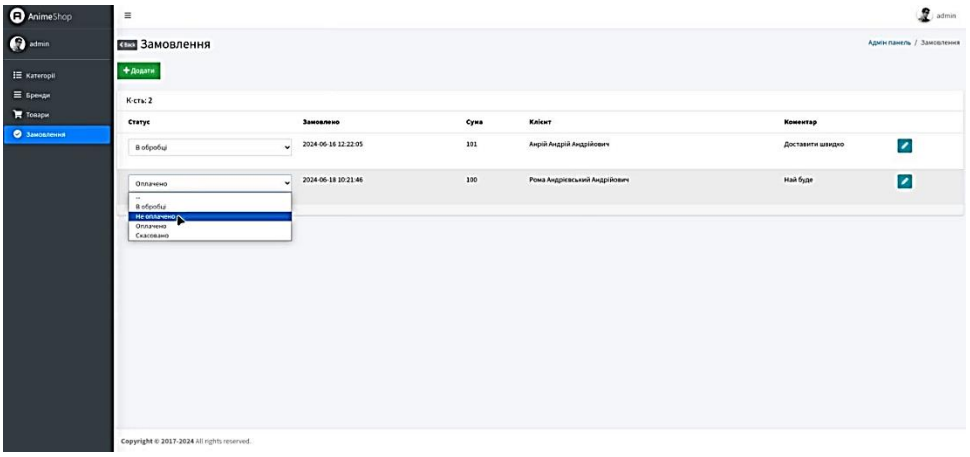

Рисунок 3.10 – Інтерфейс управління замовленнями

Цей інтерфейс дозволяє адміністратору мати повний контроль над усіма замовленнями, що сприяє оперативній обробці замовлень та полегшує управління процесом продажу.

3.2 Розробка бази даних

Бази даних є невід’ємною частиною розробки програмного забезпечення, зокрема інтернет-магазину. Вони є незамінним інструментом для зберігання та організації великих обсягів даних, таких як інформація про товари, користувачів, замовлення та інші аспекти функціонування магазину. Бази даних дозволяють ефективно керувати цими даними, забезпечуючи швидкий доступ до них, а також гарантують надійність та цілісність інформації.

Розробка бази даних для інтернет-магазину передбачає проектування та реалізацію структури даних, необхідної для зберігання детальної інформації про товари, користувачів та їх взаємодію. Ці дані не лише відображаються на веб-сторінках магазину, але й використовуються для обробки замовлень, аналізу продажів, створення звітів та забезпечення інших функціональних можливостей магазину. Таким чином, бази даних виступають у ролі фундаментального елементу, що забезпечує оптимальну роботу інтернет-магазину.

Я вирішив скористатися базою даних MySQL для зберігання і керування інформацією в інтернет-магазині. MySQL відомий своєю надійністю, швидкістю

та розширюваністю, що робить його чудовим вибором для великих та середніх проектів, таких як інтернет-магазини.

Зацікавленою була ідея створення діаграми класів (рис. 3.11), яка наглядно відобразила би організацію бази даних. Це допомагає краще зрозуміти, як дані організовані та взаємодіють між собою, що сприяє ефективній роботі та розвитку інтернет-магазину.

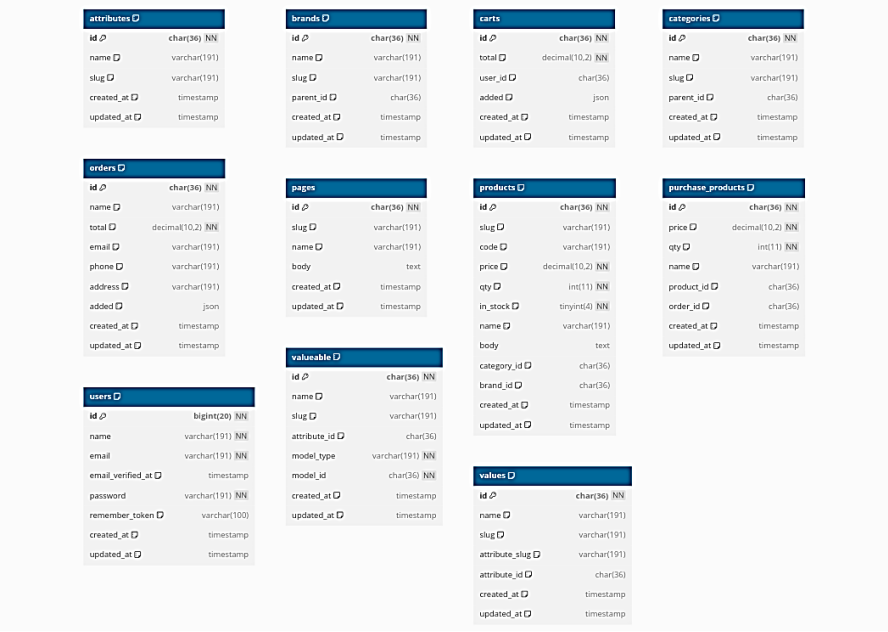

Рисунок 3.11 – Діаграма класів моделей БД

Для побудови схеми бази даних у нашому проекті використані міграції, що є ключовим інструментом у Laravel для управління структурою бази даних. Міграції дозволяють створювати, змінювати та видаляти таблиці та інші об'єкти бази даних за допомогою коду, що полегшує їх версіонування та обслуговування.

Однією з основних таблиць у моєму інтернет-магазині є таблиця products, яка містить інформацію про товари. В лістингу 3.6 зображений код міграції для створення таблиці products.

Лістинг 3.6 – Міграція таблиці Product

```
<?php

use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint
$table) {
            $table->uuid('id')->primary();
            $table->string('slug')->index()->nullable();
            $table->string('code')->index()->nullable();
            $table->decimal('price', 10, 2)->default(0);
            $table->integer('qty')->default(0);
            $table->tinyInteger('in_stock')->default(0);
            $table->tinyInteger('is_featured')->default(0);
            $table->string('name')->index()->nullable();
            $table->text('body')->nullable();
            $table->uuid('category_id')->index()->nullable();
            $table->uuid('brand_id')->index()->nullable();
            $table->timestamps();
        });
    }

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

Кінець лістингу 3.6

Також у додатку А буде розглянуто міграції для інших таблиць бази даних, таких як users, brands та інші, що забезпечують повну структуру для інтернет-магазину.

Для швидкого наповнення бази даних тестовими даними я використовував фабрики. Фабрики дозволяють автоматично генерувати випадкові дані для наших моделей, що значно спрощує процес тестування та розробки.

Код фабрик наведений у додатку Б. Ці фабрики використовують інструменти Laravel для наповнення бази даних тестовими даними.

Після створення фабрик, можна швидко наповнити базу даних тестовими даними, виконавши команду: php artisan db:seed.

Ця команда запускає фабрики, які генерують та вставляють тестові дані у відповідні таблиці в базі даних MySQL. Це дозволяє швидко створити реалістичне середовище для тестування функціональності інтернет-магазину.

3.3 Тестування та налагодження інтернет магазину

Після завершення основної розробки інтернет-магазину, важливо провести детальне тестування та налагодження для забезпечення коректної роботи всіх функцій. Тестування допомагає виявити та виправити помилки, що можуть виникнути під час роботи з додатком, а також перевірити, чи відповідає проект вимогам специфікації.

Основна мета тестування функціональності полягає в перевірці того, як правильно працюють всі різні частини інтернет-магазину. Моя робота включала такі аспекти:

– перевірка роботи кошика, що охоплює дії, такі як додавання, видалення та оновлення товарів у кошику;

– тестування процесу входу для адміністатора, щоб переконатися, що все працює належним чином;

– тестування адміністративної панелі, включаючи дії з додавання, редагування та видалення товарів, категорій та брендів;

– проведення тестів для перевірки роботи пошуку товарів та фільтрації за різними параметрами.

Для тестування функціональності магазину було створено фабрики та сидери, які автоматично генерують тестові дані в базу даних. Це дозволяє швидко наповнити базу даними та перевірити роботу всіх функцій сайту у реальних умовах.

Одна з важливих перевірок під час тестування та налагодження – це перевірка консолі браузера на наявність помилок JavaScript. Помилки в консолі можуть впливати на функціональність сайту та викликати несподівану поведінку.

На рисунку 3.12 зображено сторінку інтернет-магазину з чистою консоллю браузера, що свідчить про відсутність помилок JavaScript під час завантаження та взаємодії з цією сторінкою.

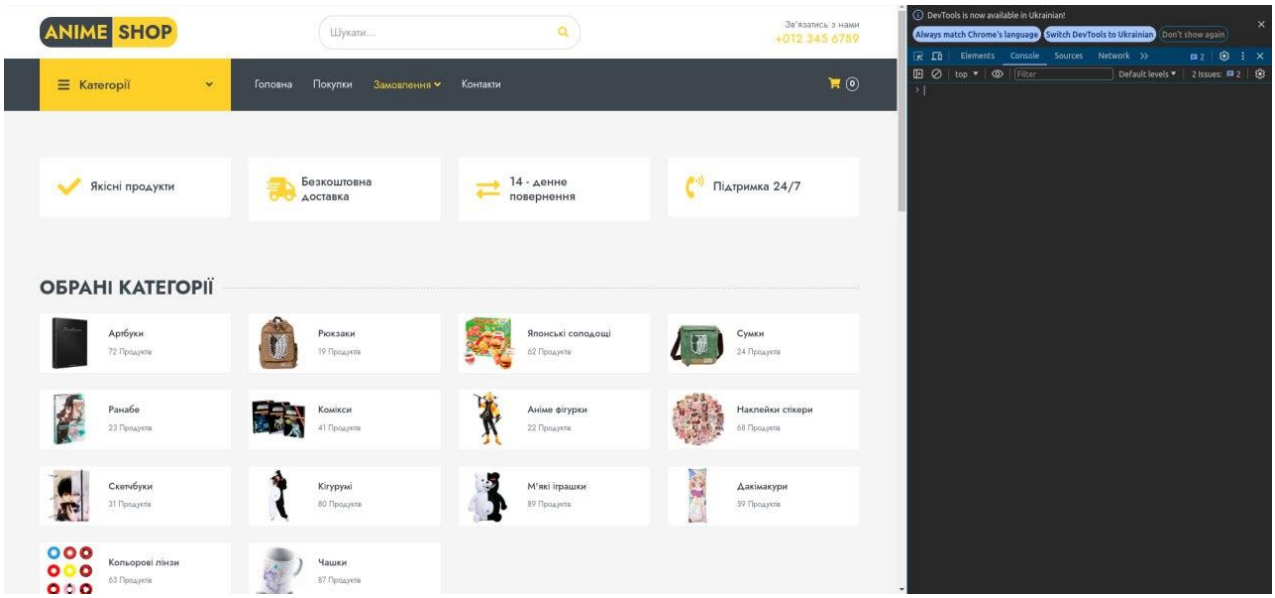

Рисунок 3.12 – Сторінка з чистою консоллю браузера

Пошук товарів є однією з найважливіших функцій веб-сайтів електронної комерції. Він дозволяє користувачам швидко та легко знаходити те, що вони шукають, що призводить до збільшення продажів та покращення досвіду користувачів.

Під час тестування функції пошуку на інтернет-магазині, першим кроком є відкриття інструментів розробника у браузері. Це можна зробити, використовуючи правий клік миші на будь-якій сторінці та обираючи опцію «Інспектувати» або «Перегляд консолі».

Далі необхідно перейти до вкладки «Network». Ця вкладка призначена для моніторингу всіх мережевих запитів, що відправляються та отримуються від сервера під час взаємодії з сайтом.

Після цього вводимо пошуковий запит на сайті магазину, наприклад, назву товару або категорії. Після введення запиту обираємо один з результатів пошуку для переходу на сторінку товару чи категорії.

На вкладці «Network» відображаються всі запити, що стосуються пошуку. Потрібно розглянути ці запити, щоб переконатися, що дані відправляються на сервер з вірними пошуковими параметрами.

Після обробки запиту сервером, отримуємо відповідь, яка також відображається на вкладці «Network». Також потрібно перевірити статус коду відповіді (зазвичай 200, що означає успішний запит), а також дані, які повертаються сервером, наприклад, список товарів або категорій.

І останній етап – це перевірка результатів на самому сайті. Потрібно переконатись, що результати пошуку відповідають нашим очікуванням, тобто товари відображаються коректно, описи є правильними, а фільтри працюють для подальшого вибору (рис. 3.13).

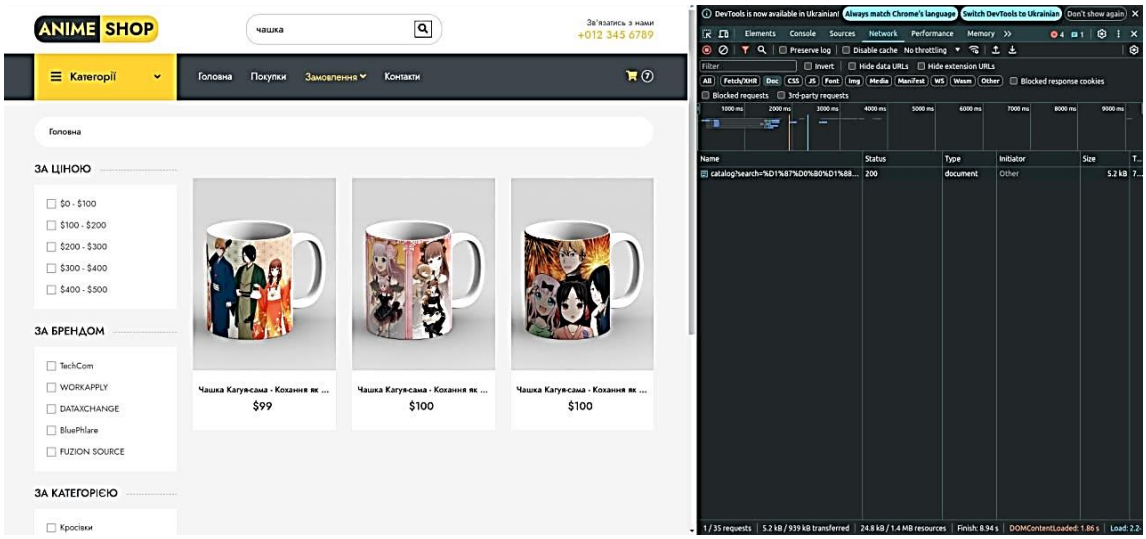

Рисунок 3.13 – Інтерфейс пошуку товарів

Тестування додавання товару до корзини – це важливий крок у забезпеченні правильної роботи інтернет-магазину. Завдяки аналізу запитів та відповідей в Network можна переконатися, що дані про товар передаються та обробляються правильно, а товар успішно додається до корзини користувача.

Тому далі слідує тестування добавлення товару до корзини. Для цього потрібно відкрити інструменти розробника та вибрати Network, після цього натиснути додати до корзини. Очікується, що у відповідь, при успішному додаванні товару до корзини сервер поверне код відповіді 200 (рис. 3.14).

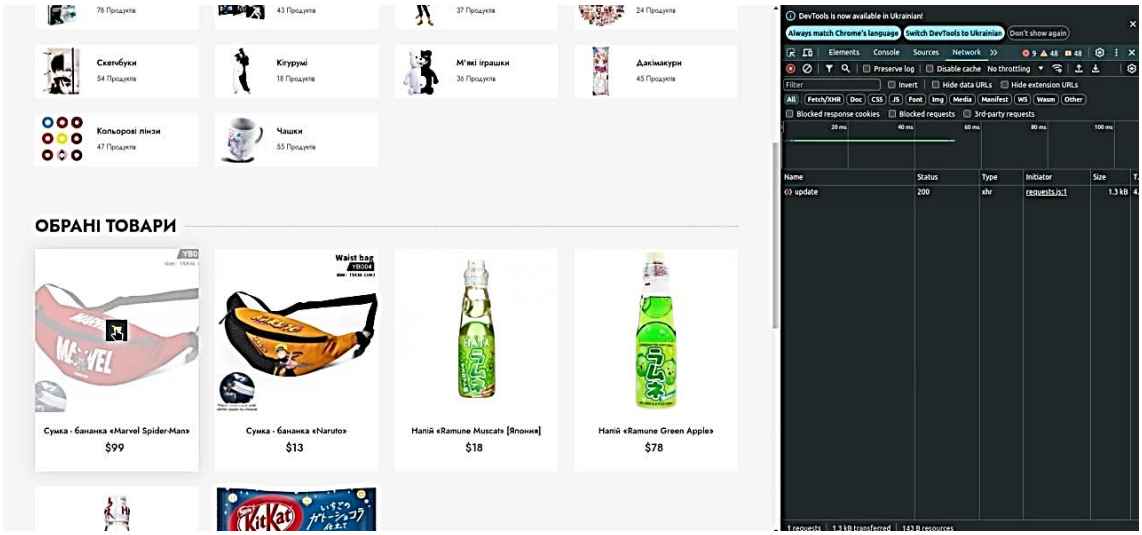

Рисунок 3.14 – Успішне добавлення товару до корзини

Тестування та налагодження є невід'ємною частиною розробки інтернет-магазину. Використання різних інструментів та методів дозволяє виявити та виправити помилки на ранніх етапах розробки, забезпечуючи високу якість кінцевого продукту.

Висновки до розділу 3

У цьому розділі описано різні аспекти створення інтернет-магазину на Laravel для продажу японських товарів. Спочатку були розглянуті основні етапи розробки магазину, включаючи створення проекту Laravel, налаштування

маршрутів, реалізацію адміністративної панелі та інші ключові аспекти. Зазначено використання шаблону AdminLTE3 для створення адміністративного інтерфейсу та SCSS для стилізації веб-сторінок.

Далі був наданий опис міграцій для створення таблиць у базі даних, зокрема таблиці товарів. Також описано процес наповнення бази даних тестовими даними за допомогою фабрик та виконання міграцій.

Також розглянуті ключові аспекти тестування функціональності магазину, такі як робота з кошиком, авторизація адміністратора, адміністративна панель та інші. Узагальнюючи, було детально проаналізовано процес розробки та тестування інтернет-магазину, від початкового створення проекту до функціональної роботи та тестування різних його компонентів.

ВИСНОВКИ

Основною метою даного проекту було успішне виконання усіх поставлених завдань з високою якістю та вчасністю. В процесі розробки був виконаний значний обсяг робіт, спрямованих на досягнення цих цілей.

Початковим етапом було проведено аналіз предметної області, що включав дослідження актуальності обраної теми та веб-технологій. Була детально вивчена архітектура обраних фреймворків, включаючи контролери, компоненти, роути та міграції.

Далі було проведено аналіз сайтів конкурентів (Murasaki, Yorokobi) для виявлення типових проблем, які успішно уникнуто під час розробки інтернет-магазину.

Розробка фронтенд частини інтернет-магазину була проведена з використанням технологій HTML, CSS, JavaScript та Bootstrap, також використовувалася бібліотека jQuery, і для зручності розробки були використані Node.js та Vite. При розробці інтерфейсу користувача (UI) були дотримані такі поставленні критерії як:

– чистий та мінімалістичний дизайн;
– адаптивність;
– висока швидкість завантаження;
– зручна навігація, зручний пошук товарів та інформації;
– фільтри, фільтрація товарів за категоріями, ціною, брендом та іншими параметрами.

Далі було налаштовано середовище розробки, включаючи вибір відповідного хостингу, налаштування версії PHP, увімкнення необхідних розширень для коректного функціонування програмного забезпечення, а також встановлення пакетного менеджера Composer для управління залежностями проекту.

Для розробки адміністративної частини інтернет-магазину був обраний фреймворк Laravel. Цей вибір забезпечив стабільність та ефективність

управління базою даних та логікою системи. При розробці адміністративної частини було досягнуті та реалізовані наступні функції:

– управління каталогом товарів – адміністратори мають можливість додавати нові товари, редагувати дані про існуючі товари та видаляти їх з каталогу. Також реалізовано управління зображеннями, описами, цінами та іншими характеристиками товарів;

– управління замовленнями – адміністратори мають доступ до переліку всіх замовлень з можливістю змінювати їх статус, переглядати деталі кожного замовлення, редагувати інформацію про замовлення та видаляти замовлення за необхідності.

Після завершення розробки адміністративної частини інтернет-магазину було розроблено клієнтську частину сайту. Було реалізовано наступний функціонал:

– відображення списку товарів – список товарів містить основну інформацію про кожен товар, включаючи назву, зображення, ціну та короткий опис;

– фільтрація товарів – користувачі мають можливість фільтрувати товари за різними критеріями, такими як категорія, ціна, бренд, наявність на складі тощо;

– детальна сторінка товару – при натисканні на товар у списку, користувач потрапляє на детальну сторінку товару, яка містить розширений опис, технічні характеристики, кілька зображень;

– пошук товарів;

– додавання та видалення товарів – реалізовано можливість додавати товари до кошика одним кліком, а також видаляти їх або змінювати кількість товарів безпосередньо в кошику;

– перегляд вмісту кошика;

– оформлення замовлення.

Далі була реалізована структура бази даних з використаням MySQL. Також при побудові схем бази даних були використані міграції, що є ключовим інструментом у Laravel для управління структурою бази даних.

Після завершення основної розробки було проведено тестування, що охоплювало перевірку різних аспектів, таких як функціональність, додавання товарів до кошика, оформлення замовлення, а також наповнення сайту контентом.

У результаті мною було успішно створено функціональний інтернет-магазин, який відповідає вимогам та очікуванням користувачів, забезпечуючи зручну та ефективну платформу для придбання японських товарів в онлайн-режимі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Murasaki – інтернет-магазин аніме в Україні. URL: https://murasaki.store/ (дата звернення: 15.03.2024).

2. Аніме магазин Yorokobi. URL: https://yorokobi.com.ua/ (дата звернення: 16.03.2024).

3. Інтерфейс користувача – Вікіпедія. URL: https://en.wikipedia.org/wiki/User_interface (дата звернення: 05.04.2024).

4. Laravel – The PHP Framework For Web Artisans. URL: https://laravel.com/ (дата звернення: 10.04.2024).

5. Переваги Laravel для веб-розробки | Wezom. URL: https://wezom.com.ua/ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii (дата звернення: 10.04.2024).

6. Що таке HTML – HTML українською. URL: https://html.tutorial.in.ua/what-is-html/ (дата звернення: 14.04.2024).

7. Мови програмування для веброзробки та супутні технології | Друкарня. URL: https://drukarnia.com.ua/articles/movi-programuvannya-dlya-vebrozrobki-ta-suputni-tekhnologiyi-wzeAr (дата звернення: 08.05.2024).

8. Introduction – Bootstrap v5.0. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 06.05.2024).

9. Що таке jQuery простими словами: основи роботи з бібліотекою - Heroi.zapisi.cx.ua. URL: https://heroi.zapisi.cx.ua/ukraincyam/shho-take-jquery-prostimi-slovami-osnovi-roboti-z-bibliotekoyu.html (дата звернення: 10.05.2024).

10. jQuery. URL: https://jquery.com/ (дата звернення: 12.05.2024).

11. Що таке Node.js? Основи серверної розробки на JavaScript DevZone. URL: https://devzone.org.ua/post/shcho-take-nodejs-osnovy-servernoyi-rozrobky-na-javascript (дата звернення: 18.05.2024).

12. Vite | Next Generation Frontend Tooling. URL: https://vitejs.dev/ (дата звернення: 20.05.2024).

13. Composer – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Composer (дата звернення: 20.05.2024).

14. Composer. URL: https://getcomposer.org/ (дата звернення: 20.05.2024).

15. Міграція бази даних – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Schema_migration (дата звернення: 01.06.2024).

ДОДАТКИ

Додаток А

Створення міграцій

Міграція users

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void{
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->timestamps();
        });
    }

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

Міграція значень атрибутів

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void{
        Schema::create('values', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('name')->index()->nullable();
            $table->string('slug')->index()->nullable();
            $table->string('attribute_slug')->index()->nullable();
            $table->uuid('attribute_id')->index()->nullable();
            $table->timestamps();
        });
    }
```

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

Міграція атрибутів

```
<?php

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

return new class extends Migration{
    /**
     * Run the migrations.
     */
    public function up(): void {
        Schema::create('attributes', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('name')->index()->nullable();
            $table->string('slug')->index()->nullable();
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('attributes');
    }
};
```

Міграція прив’язка значення-атрибут-продукт

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('valueable', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('name')->index()->nullable();
```

```
            $table->string('slug')->index()->nullable();
            $table->uuid('attribute_id')->index()->nullable();
            $table->uuidMorphs('model');
            $table->timestamps();
        });
    }

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

Міграція brands

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('brands', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('name')->index()->nullable();
            $table->string('slug')->index()->nullable();
            $table->uuid('parent_id')->index()->nullable();
            $table->timestamps();
        });
    }

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

Міграція категорій

```
<?php

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

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('name')->index()->nullable();
            $table->tinyInteger('is_main')->default(0);
            $table->string('slug')->index()->nullable();
            $table->uuid('parent_id')->index()->nullable();
            $table->timestamps();
        });
    }

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

Міграція замовлення

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->string('name')->index()->nullable();
            $table->decimal('total', 10, 2)->default(0);
            $table->string('email')->nullable();
            $table->string('phone')->nullable();
            $table->string('address')->nullable();
            $table->json('added')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
```

```
    {
        Schema::dropIfExists('orders');
    }
};
```

Міграція продуктів замовлення

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('purchase_products', function (Blueprint $table) {
            $table->uuid('id')->primary();
            $table->decimal('price', 10, 2)->default(0);
            $table->integer('qty')->default(0);
            $table->string('name')->index()->nullable();
            $table->uuid('product_id')->index()->nullable();
            $table->uuid('order_id')->index()->nullable();
            $table->timestamps();
        });
    }

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

Міграція корзина

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('carts', function (Blueprint $table) {
```

```
            $table->uuid('id')->primary();
            $table->decimal('total', 10, 2)->default(0);
            $table->uuid('user_id')->nullable();
            $table->json('added')->nullable();
            $table->timestamps();
        });
    }

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

Додаток Б

Створення фабрик

Фабрика users

```
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\User>
 */
class UserFactory extends Factory
{
    protected static ?string $password;

    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {

        return [
            'name' => 'admin',
            'email' => 'admin@test.com',
            'email_verified_at' => now(),
            'password' => Hash::make('1234'),
            'remember_token' => Str::random(10),
        ];
    }

    /**
     * Indicate that the model's email address should be unverified.
     */
    public function unverified(): static
    {
        return $this->state(fn (array $attributes) => [
            'email_verified_at' => null,
        ]);
    }
}
```

Фабрика продуктів

```
<?php

namespace Database\Factories;

use App\Models\Brand;
use App\Models\Category;
use App\Models\Product;
```

```
use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Product>
 */
class ProductFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        $priceArray = [
            '540',
            '399',
            '120',
            '310',
            '40',
            '234',
        ];

        $nameArray = [
            'Скетчбук Наруто',
            'Пенал Наруто',
            'Манга Наруто',
            'Кунай Наруто',
            'Кільце Наруто',
            'Чашка Наруто',
            'Скетчбук Наруто1',
            'Пенал Наруто1',
            'Манга Наруто1',
            'Кунай Наруто1',
            'Кільце Наруто1',
            'Чашка Наруто1',
            'Скетчбук Наруто2',
            'Пенал Наруто2',
            'Манга Наруто2',
            'Кунай Наруто2',
            'Кільце Наруто2',
            'Чашка Наруто2',
            'Скетчбук Наруто3',
            'Пенал Наруто3',
            'Манга Наруто3',
            'Кунай Наруто3',
            'Кільце Наруто3',
            'Чашка Наруто3',
        ];

        $code = $this->faker->unique()->numberBetween('111111111', '999999999');
        $qty = $this->faker->numberBetween('1', '35');
        $inStock = random_int(0,1);
        $body = $this->faker->text;
        $category = Category::inRandomOrder()->first()->id;
        $brand = Brand::inRandomOrder()->first()->id;
        $price = $this->faker->randomElement($priceArray);
        $name = $this->faker->unique()->randomElement($nameArray);
```

```
        return [
            'slug' => \Str::slug($name),
            'code' => $code,
            'price' => $price,
            'qty' => $qty,
            'in_stock' => $inStock,
            'name' => $name,
            'body' => $body,
            'category_id' => $category,
            'brand_id' => $brand,
        ];
    }
}
```

Фабрика бренди

```
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Brand>
 */
class BrandFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        $name = $this->faker->unique()->randomElement([
            'Atomic Flare',
            'Kotobukiya Co., Ltd',
            'Bandai Namco',
            'Good Smile Company USA',
            'Crunchyroll (Sony Music Entertainment)',
            'Sentai Filmworks, LLC (AMC Networks)',
            'Viz Media LLC',
            'Kodansha LTD'
        ]);

        return [
            'name' => $name,
            'slug' => \Str::slug($name),
        ];
    }
}
```

Фабрика категорії

```
<?php

namespace Database\Factories;
```

```
use App\Models\Category;
use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Category>
 */
class CategoryFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        $name = $this->faker->unique()->randomElement([
            "Дім та Офіс ",
            "Блокноти і Скетчбуки ",
            "Гобелени ",
            "Гральні карти ",
            "Дакімакури ",
            "Дзеркальця ",
            "Календари и открытки ",
            "Килимки для миші ",
            "Кружки ",
            "Нічники ",
            "Парасолі ",
        ]);

        return [
            'name' => $name,
            'slug' => \Str::slug($name),
        ];
    }
}
```

Фабрика статичні сторінки

```
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Page>
 */
class PageFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
```

```
        $sentence = $this->faker->sentence;

        return [
            'name' => $sentence,
            'slug' => Str::slug($sentence),
            'body' => $this->faker->text,
        ];
    }
}
```

Додаток В

Створення корзини

JavaScript-код

```
import DataQueue from './lib/data-queue';

export const cart$ = new DataQueue();

class Cart {
    products;

    constructor(products) {
        this.products = products || [];
    }

    add(productId, quantity = 1) {
        window.axios.post(/ajax/cart/update, {
            data: [{
                id: productId,
                quantity: quantity,
                is_deleted: false
            }]
        }).then(response => {
            this.products = response.data.products;
            cart$.feed(this);
        })
    }

    remove(productId) {
        window.axios.post(/ajax/cart/update, {
            data: [{
                id: productId,
                quantity: 1,
                is_deleted: true
            }]
        }).then(response => {
            this.products = response.data.products;
            cart$.feed(this);
        })
    }

    count() {
        let count = 0;

        this.products.forEach(product => {
            count += product.quantity;
        });

        return count;
    }
}

document.querySelectorAll('.js-simple-add-to-cart').forEach(item => {
    item.addEventListener('click', event => {
        event.preventDefault();
```

```
        const cart = cart$.current();

        cart.add(item.dataset.productId);
    })
})


document.querySelectorAll('.js-main-add-to-cart').forEach(item => {
    item.addEventListener('click', event => {
        const cart = cart$.current();

        const choosenQuantityItem = item.parentNode.querySelector('.js-cart-
product-quantity');

        cart.add(item.dataset.productId, choosenQuantityItem.value);
    })
})

window.axios.get('/ajax/cart').then(response => {
    const cart = new Cart(response.data.products);

    cart$.feed(cart);
});


cart$.onLatest(cart => {
    document.querySelector('.js-cart-count').innerText = cart.count();
})
```

PHP-код

```
public function view(Request $request)
    {
        / @var Cart $cart */
        $cart = Cart::find($request->session()->get('cartId'));
        $cartProducts = $cart->getAllProducts();
        $subTotal = $cart->total;
        $shippingPrice = config('services.shipping_price');
        $total = $subTotal + $shippingPrice;

        return view('client.cart', compact('cartProducts', 'subTotal', 'shippingPrice',
'total'));
    }

    public function getAllCartProducts(Request $request)
    {
        / @var Cart $cart */
        $cart = Cart::find($request->session()->get('cartId'));

        return response()->json([
            'products' => $cart->getAllProducts(),
        ]);
    }

    public function updateCart(Request $request)
    {
        /** @var Cart $cart */
        $cart = Cart::find($request->session()->get('cartId'));
```

```
        $data = $request->data;

        $cartProducts = $cart->getAllProducts();
        $newCartProducts = [];

        foreach ($data ?? [] as $item) {
            $id = $item['id'];

            $productData = $cart->getProductById($id);

            if ($productData) {
                foreach ($cartProducts as $key => $product) {
                    $productId = $product['id'];

                    // логіка видалення
                    if ($item['is_deleted'] && ($item['id'] === $productId)) {
                        // логіка видалення певної к-сті товарів
                        if ($product['quantity'] > $item['quantity']) {
                            $updQty = $productData['quantity'] - $item['quantity'];
                            $cartProducts[$key]['quantity'] = $updQty;
                            $cartProducts[$key]['total'] = $updQty * $product['price'];
                        }

                        // логіка видалення всього товару
                        if ($product['quantity'] <= $item['quantity']) {
                           unset($cartProducts[$key]);
                        }
                    }

                    // логіка додавання к-сті існуючого товару в корзині
                    if (!$item['is_deleted'] && $productData) {
                        $updQty = $productData['quantity'] + $item['quantity'];
                        $cartProducts[$key]['quantity'] = $updQty;
                        $cartProducts[$key]['total'] = $updQty * $product['price'];
                    }

                }
            }

            // логіка додавання товару
            if (!$item['is_deleted'] && !$productData) {
                $product  =  Product::find($item['id'])->only('id',  'quantity',  'name',
'price');
                $product['quantity'] = $item['quantity'];
                $product['total'] = $item['quantity'] * $product['price'];

                $newCartProducts[] = $product;
            }
        }

        $cartProducts = array_merge($cartProducts ?? [], $newCartProducts);
        $cart->setAttribute('added->products', $cartProducts);
        $cart->setAttribute('total', $cart->getCalcCartTotal());
        $cart->save();

        return response()->json(['message' => 'Продукт успішно доданий!']);

    }
```