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»

СТВОРЕННЯ УНІВЕРСАЛЬНОГО ФРЕЙМВОРКА ДЛЯ МАРКЕТПЛЕЙСІВ НА LARAVEL

CREATING A UNIVERSAL FRAMEWORK FOR MARKETPLACES ON LARAVEL

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

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

Виконав: здобувач вищої освіти
Групи ІПЗ-42
Остапович Андрій Миколайович
_____
(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

Остаповичу Андрію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *«Створення універсального фреймворка для маркетплейсів на 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: *15 рисунків.*

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		4,5 %	
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.*	*викон.*
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	*викон.*
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	*викон.*
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	*викон.*
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	*викон.*
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	*викон.*
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	*викон.*

Здобувач вищої освіти _____ (підпис) *Остапович А. М* (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: Остапович Андрій Миколайович

Тема: «Створення універсального фреймворка для маркетплейсів на Laravel»

Коротка характеристика кваліфікаційної роботи. Кваліфікаційна робота зосереджена на аналізі сучасних фреймворків для маркетплейсів та розробці універсального рішення для швидкої, гнучкої та безпечної розробки маркетплейсів. Робота містить теоретичний аналіз, проектування структури бази даних, реалізацію основних функцій та тестування. Розроблений фреймворк сприяє ефективній розробці маркетплейсів, підтримуючи розвиток електронної комерції та задовольняючи сучасні потреби користувачів.
Самостійні розробки і пропозиції автора. У цій розробці є можливість створення потужного інструменту, який зробить процес розробки маркетплейсів більш ефективним та зручним для розробників, що дозволить їм швидше відкривати та масштабувати свої проекти.
Практичне значення роботи. полягає у його доступності для всіх розробників, оскільки він розміщений на сайті GitHub. Це означає, що будь-хто може легко отримати доступ до його вихідного коду, вивчити його структуру та функціональність, а також внести свої внески через систему контролю версій.

Недоліки. Одним з недоліків роботи можна вважати фронтенд частину, так як вона є статичною і не забезпечує гнучкості дизайну.

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

Рецензент: к.пед.н., доцент, завідувач каф. ЦОТ Бабак В.В.
(прізвище, ім'я, по-батькові, посада)

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 *Остапович Андрій Миколайович*
група *ІПЗ-42*

Тема кваліфікаційної роботи: *Створення універсального фреймворка для маркетплейсів на Laravel*

Керівник: *Ліщина Наталія Миколаївна*

Актуальність теми: сучасні маркетплейси є складними системами, які вимагають інтеграції різних компонентів та інструментів для забезпечення їх функціональності та надійності. Існуючі фреймворки не завжди відповідають потребам сучасного електронного бізнесу через їхню недостатню гнучкість, складність у використанні або обмежену підтримку нових технологій.

Об'єкт дослідження – є програмне забезпечення для створення та управління маркетплейсами, що включає в себе різноманітні технології, інструменти та методи для забезпечення функціональності, безпеки та продуктивності.

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи: робота базується на глибокому аналізі сучасних фреймворків для маркетплейсів, включаючи розробку унікальної концепції універсального фреймворку з інтеграцією важливих сервісів, таких як API ПриватБанку та платіжна система Fondy. Це забезпечує високу гнучкість, масштабованість, безпеку та надійність системи. Розроблений фреймворк сприяє швидкому створенню маркетплейсів, що відповідають сучасним вимогам, тим самим підтримуючи розвиток електронної комерції та задовольняючи потреби користувачів.

Зауваження та недоліки: істотних недоліків не виявлено.

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

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

«7» 06 2024 р.

АНОТАЦІЯ

Остапович А. М. Створення універсального фреймворка для маркетплейсів на Laravel. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

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

У першому розділі здійснено аналіз сучасного стану проблеми розробки комп’ютерних ігор і сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено сам фреймворк. У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості використання фреймворку для створення інтернет магазинів

Ключові слова: фреймворк, маркетплейс, laravel, back-end.

ABSTRACT

Ostapovich A. M. Creating a Universal framework for marketplaces on Laravel. Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions and proposals, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of developing computer games was carried out and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, the framework itself was developed and tested. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of using the framework to create online stores.

Keywords: framework, marketplace, laravel, back-end.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні маркетплейси є складними системами, які вимагають інтеграції різних компонентів та інструментів для забезпечення їх функціональності та надійності. Існуючі фреймворки не завжди відповідають потребам сучасного електронного бізнесу через їхню недостатню гнучкість, складність у використанні або обмежену підтримку нових технологій. Серед практично розв’язаних задач виділяються аутентифікація користувачів, управління товарами та послугами, а також інтеграція платіжних систем. Проте, існує прогалина у створенні універсального рішення, яке б поєднувало всі ці аспекти в одному фреймворку.

На світовій арені розробки веб-додатків спостерігається тенденція до створення більш модульних та розширюваних фреймворків. Популярні інструменти, такі як Laravel, пропонують значну гнучкість і багатофункціональність, що дозволяє розробникам швидко адаптуватися до нових вимог ринку [1]. Важливою тенденцією є також акцент на безпеку та продуктивність систем, а також підтримка інтеграції з різними зовнішніми сервісами через API [2].

Метою цієї роботи є створення універсального фреймворку для маркетплейсів, який забезпечить зручність розробки, гнучкість, безпеку та продуктивність. Цей фреймворк призначений для використання в будь-яких проектах, що передбачають розробку маркетплейсів, незалежно від їхнього масштабу та галузі. Розробка даного фреймворку базується на кращих практиках та досвіді використання існуючих рішень, таких як Laravel. Вона інтегрується з популярними інструментами та бібліотеками для забезпечення максимальної функціональності та зручності. Використання сучасних підходів та технологій, таких як RESTful API та інтеграція з платіжними системами, дозволяє забезпечити взаємодію з іншими системами та сервісами, розширюючи можливості розробників [3].

Об’єкт дослідження – є програмне забезпечення для створення та управління маркетплейсами, що включає в себе різноманітні технології, інструменти та методи для забезпечення функціональності, безпеки та продуктивності.

Предметом дослідження – є процес розробки універсального фреймворку для маркетплейсів, що включає архітектурні рішення, використання сучасних фреймворків та бібліотек, інтеграцію зовнішніх сервісів, а також тестування та налагодження системи.

Виходячи із мети сформулюємо завдання:

- дослідження сучасного стану ринку фреймворків;
- визначення основних переваг та недоліків наявних рішень;
- визначення вимог до універсального фреймворку;
- розробка концепції майбутнього фреймворку;
- створення структури бази даних;
- реалізація основних функціональних можливостей;
- розробка системи аутентифікації користувачів;
- використання фабрик та сідерів для тестових даних;
- проведення тестування та дебагінг;
- інтеграція публічного API ПриватБанку для курсу валют;
- інтеграція платіжної системи Fondy;
- створення роутів та контролерів для API;
- документування API за допомогою спеціалізованих інструментів;
- створення документації користувача та розробника;
- підготовка системи до впровадження та експлуатації.

РОЗДІЛ 1
АНАЛІЗ УНІВЕРСАЛЬНОГО ФРЕЙМВОРКА ДЛЯ МАРКЕТПЛЕЙСІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Індустрія електронної комерції в останні десятиліття зазнала значних змін, ставши однією з найдинамічніших і найперспективніших галузей у світі. Маркетплейси стали невід'ємною частиною цієї екосистеми, забезпечуючи зручність та ефективність процесів купівлі-продажу товарів і послуг. Платформи, такі як Amazon, eBay, Etsy та інші, пропонують споживачам широкий вибір продукції, одночасно надаючи продавцям можливість доступу до глобального ринку. Однак, разом з розвитком електронної комерції, постають нові виклики та вимоги, які існуючі фреймворки маркетплейсів не завжди здатні задовольнити. У цьому контексті виникає потреба у створенні універсального фреймворку, який би відповідав сучасним потребам ринку та сприяв розвитку індустрії в майбутньому.

Маркетплейси стали ключовими гравцями на ринку електронної комерції завдяки своїй здатності об'єднувати продавців і покупців на одній платформі. Вони пропонують споживачам широкий асортимент товарів та послуг, доступ до конкурентних цін і можливість зручного порівняння продуктів. Для продавців маркетплейси відкривають можливості виходу на нові ринки, зниження витрат на маркетинг та логістику, а також забезпечення більш ефективного управління продажами.

Незважаючи на успіхи маркетплейсів, існуючі фреймворки мають ряд обмежень, які можуть стримувати їх подальший розвиток. Однією з основних проблем є недостатня гнучкість і універсальність існуючих рішень. Багато фреймворків не здатні ефективно адаптуватися до змін у технологіях та потребах ринку, що ускладнює впровадження нових функцій та інновацій.

Крім того, існуючі фреймворки часто є складними у використанні, що створює бар'єри для швидкої та ефективної розробки нових маркетплейсів.

Складні архітектури, недостатня документація та обмежена підтримка з боку розробників можуть призводити до затримок у впровадженні нових функцій та оновлень. Це особливо актуально для малих та середніх підприємств, які не мають значних ресурсів для подолання таких викликів.

Іншим важливим аспектом є обмежені можливості налаштування та масштабування існуючих фреймворків. Багато з них не дозволяють легко розширювати функціональність або адаптувати платформу до специфічних потреб бізнесу. Це може стримувати зростання та розвиток маркетплейсів, особливо у випадках, коли компанії прагнуть швидко адаптуватися до змін у попиті та умовах ринку.

Враховуючи вищезазначені обмеження, стає очевидною необхідність створення універсального фреймворку для маркетплейсів, який би поєднував у собі гнучкість, швидкість розробки та забезпечував належний рівень підтримки для розвитку різноманітних маркетплейсів. Такий фреймворк мав би стати ефективним інструментом для розробників, що дозволило б швидко та ефективно створювати маркетплейси, а також легко розширювати їх функціональність у відповідності з потребами користувачів та ринку.

Створення універсального фреймворку вимагатиме інноваційного підходу та орієнтації на майбутні тенденції в електронній комерції. Важливо, щоб він підтримував інтеграцію з різноманітними технологіями та платформами, забезпечував високу продуктивність та безпеку, а також мав зручний та інтуїтивно зрозумілий інтерфейс для розробників. Такий підхід дозволить створювати маркетплейси, які будуть готові до викликів майбутнього, відповідатимуть вимогам сучасних користувачів і сприятимуть розвитку електронного бізнесу в цілому.

Фреймворк повинен бути гнучким і універсальним, щоб підтримувати різноманітні моделі бізнесу та адаптуватися до змін у технологіях і ринку. Він повинен дозволяти легко інтегрувати нові функції та технології, такі як штучний інтелект, машинне навчання, блокчейн та інші. Також повинен мати зручний та інтуїтивно зрозумілий інтерфейс для розробників, забезпечуючи легкість у

використанні та швидкість розробки. Це включає в себе добре структуровану документацію, підтримку та інструменти для налагодження та тестування. Повинен включати в себе можливість масштабованості, щоб підтримувати зростання бізнесу та збільшення навантаження. Це включає в себе можливість горизонтального масштабування, розподілу навантаження та ефективного управління ресурсами.

Індустрія електронної комерції продовжує стрімко розвиватися, і маркетплейси відіграють у цьому процесі ключову роль. Однак, існуючі фреймворки для маркетплейсів мають ряд обмежень, які можуть стримувати їх подальший розвиток. Враховуючи потреби сучасного ринку та вимоги до гнучкості, простоти використання, масштабованості, безпеки та інтеграції, стає очевидною необхідність створення універсального фреймворку для маркетплейсів.

Такий фреймворк мав би стати ефективним інструментом для розробників, дозволяючи швидко та ефективно створювати маркетплейси, які будуть готові до викликів майбутнього. Він забезпечить можливість впровадження новітніх технологій, адаптації до змін у ринку та надання найкращого користувацького досвіду. Це сприятиме розвитку електронного бізнесу в цілому та забезпечить конкурентоспроможність маркетплейсів у довгостроковій перспективі.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

З метою розробки універсального фреймворку для маркетплейсів та глибшого осмислення сучасних тенденцій у веб-розробці, було розбито проект на наступні завдання:

- дослідження сучасного стану ринку фреймворків;
- визначення основних переваг та недоліків наявних рішень;
- визначення вимог до універсального фреймворку;
- розробка концепції майбутнього фреймворку;
- створення структури бази даних;

- реалізація основних функціональних можливостей;
- розробка системи аутентифікації користувачів;
- використання фабрик та сідерів для тестових даних;
- проведення тестування та дебагінг;
- інтеграція публічного API ПриватБанку для курсу валют;
- інтеграція платіжної системи Fondy;
- створення роутів та контролерів для API;
- документування API за допомогою спеціалізованих інструментів;
- створення документації користувача та розробника;
- підготовка системи до впровадження та експлуатації.

Висновки до розділу 1

Результати аналізу сучасного стану проблеми в області створення фреймворків для маркетплейсів демонструють важливість розвитку нового, універсального рішення. На сьогоднішній день існуючі фреймворки не завжди відповідають вимогам сучасного електронного бізнесу: вони можуть бути недостатньо гнучкими, складними у використанні або недостатньо підтримувати розвиток маркетплейсів з урахуванням сучасних тенденцій.

Розробка універсального фреймворку для маркетплейсів є надзвичайно важливою для забезпечення ефективного та швидкого створення платформ, які б відповідали поточним і майбутнім потребам ринку.

Універсальний фреймворк має включати в себе можливості для швидкої та простої розробки, добре структуровану документацію, підтримку та інструменти для налагодження та тестування. Це дозволить розробникам швидко створювати та розширювати маркетплейси, забезпечуючи високий рівень продуктивності та безпеки.

Розглядаючи та аналізуючи існуючі фреймворки, ми визначили їхні основні переваги та недоліки, що дозволило нам сформувати концепцію нового

універсального фреймворку, який включає в себе найкращі практики та уникає типових помилок. Створення архітектурного огляду та прототипу дозволило нам краще зрозуміти основні складові та принципи роботи майбутнього фреймворку.

Реалізація базових функціональних можливостей, таких як аутентифікація користувачів, управління товарами та послугами, а також оплата і доставка, підтвердила ефективність обраного підходу. Проведення тестування на валідацію розробленого фреймворку забезпечило його надійність та масштабованість.

У підсумку, розробка універсального фреймворку для маркетплейсів сприятиме розвитку електронного бізнесу, забезпечуючи конкурентоспроможність платформ у довгостроковій перспективі. Він стане ефективним інструментом для розробників, дозволяючи швидко та ефективно створювати маркетплейси, які будуть готові до викликів майбутнього.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

Початково, під час обрання платформи для проектування бази даних, було розглянуто два головні варіанти: Draw.io та Microsoft Access. Визначивши переваги та недоліки кожного з них, було вирішено віддати перевагу Microsoft Access. Переваги цієї платформи полягають у її вбудованій функціональності для роботи з базами даних, а також у зручному інтерфейсі, що спрощує процес розробки.

Почавши проектування, перша мета полягала в розробці структури бази даних, яка б задовольнила всі вимоги проекту. Почавши з сутності користувачів, було визначено необхідні атрибути, такі як: ім’я, електронна пошта, пароль та інші. Подальшим кроком було проектування сутності продавців, де було враховано додаткові атрибути, такі як назва компанії та контактні дані.

Далі було проектування сутностей продуктів та категорій, визначивши їхні взаємозв’язки та атрибути. Це включало в себе не лише опис продукту, але і цінову інформацію, фотографії, категорії товарів тощо.

Наступним етапом було проектування атрибутів, які доповнюють характеристики продуктів, такі як: розмір, колір, вага тощо. Тут важливим було правильно визначити типи даних та зв’язки між атрибутами та продуктами.

Після цього було розглянуто проектування функціональності для відгуків, корзини, замовлень та статичних сторінок, забезпечивши їх відповідними атрибутами та зв’язками.

Цей процес дозволив створити докладну структуру бази даних, яка відповідає всім вимогам проекту та забезпечує ефективну та надійну роботу системи.

Під час проектування бази даних на MySQL, виникали певні труднощі, особливо з визначенням зв’язків між сутностями. Оскільки маркетплейс мав

широкий функціонал, визначення правильних зв’язків та їх імплементація виявилися складним завданням. Наприклад, створення зв’язку «багато до багатьох» між сутностями «категорія» та «продукт» потребувало уважного аналізу та додаткових таблиць-посередників для забезпечення нормалізації бази даних (рис. 2.1) [4].

Щодо можливостей користувачів, база даних маркетплейсу має передбачати різні ролі та права доступу. Наприклад, адміністратор системи має повний доступ до всіх функцій та можливостей, включаючи управління користувачами, продуктами та замовленнями. Продавці мають доступ до своїх власних облікових записів та можуть керувати своїми продуктами та замовленнями. Клієнти, звичайно, мають обмежений доступ та можуть переглядати продукти, робити замовлення та залишати відгуки. Крім того, система повинна передбачати механізми аутентифікації та авторизації, щоб забезпечити безпеку та конфіденційність даних користувачів.

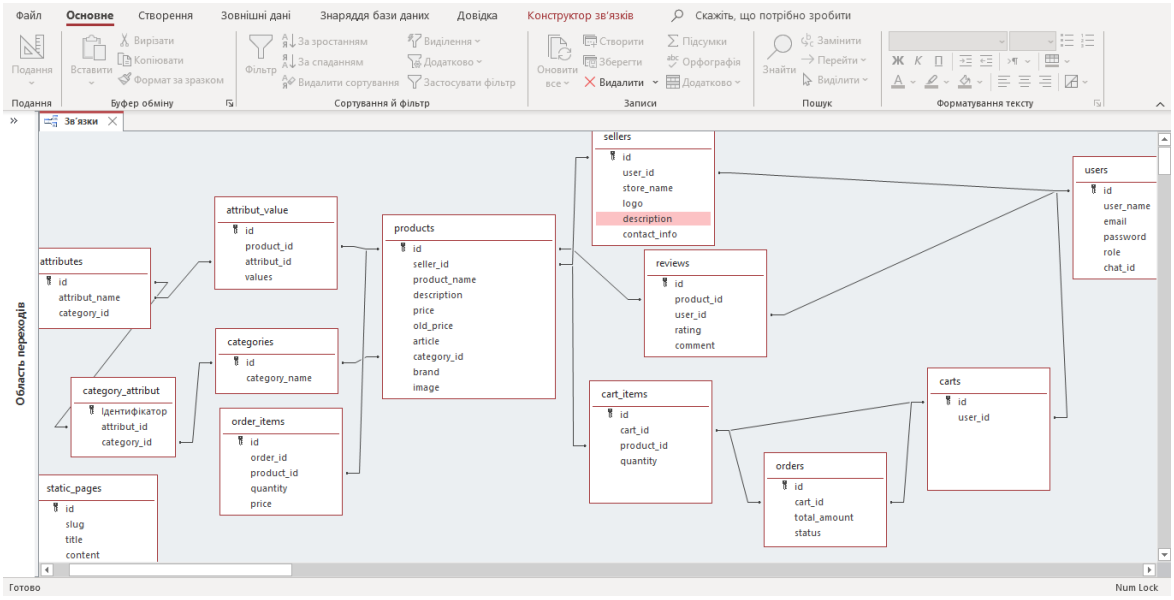

Рисунок 2.1 – Спроектована БД

Також, було розроблено невеликий список функціональних та не функціональних вимог які повинні бути в проєкті, функціональні вимоги включають:

– реєстрацію користувачів – система повинна забезпечити можливість реєстрації нових користувачів;

– управління користувачами – адміністратор системи повинен мати можливість управляти обліковими записами користувачів, включаючи блокування та видалення;

– управління товарами та послугами – система має забезпечувати можливість додавання, редагування та видалення товарів та послуг з маркетплейсу;

– оплату та доставку – користувачі повинні мати можливість здійснювати оплату за товари та послуги через систему, а також вказувати адресу доставки;

– пошук та фільтрацію – система має забезпечувати зручний пошук товарів та послуг з можливістю фільтрації за різними критеріями.

Нефункціональні вимоги включають:

– безпеку – система повинна забезпечувати високий рівень безпеки для даних користувачів та транзакцій;

– продуктивність – система повинна працювати ефективно навіть при великому обсязі даних та транзакцій;

– масштабованість – система має бути легко масштабованою для впровадження нового функціоналу та підвищення навантаження;

– надійність – система повинна бути надійною та стабільною, мінімізуючи можливість виникнення помилок та збоїв.

На основі цього списку було обрано відповідні типи даних (рис. 2.2), структури даних та зв’язки між сутностями, щоб забезпечити ефективну та надійну роботу системи.

Пізніше було додано статі та зв’язано їх з сутністю відгуків, так як часто в маркетплейсів, є якась проста стрічка новин для популяризації сайту.

Загалом проаналізувавши проектування, можна сказати що це досить великий проєкт, і можливе додавання ще багатьох фітч. Та тут грає велику роль гнучкість, та універсальність. Тож увагу акцентовано саме на цьому.

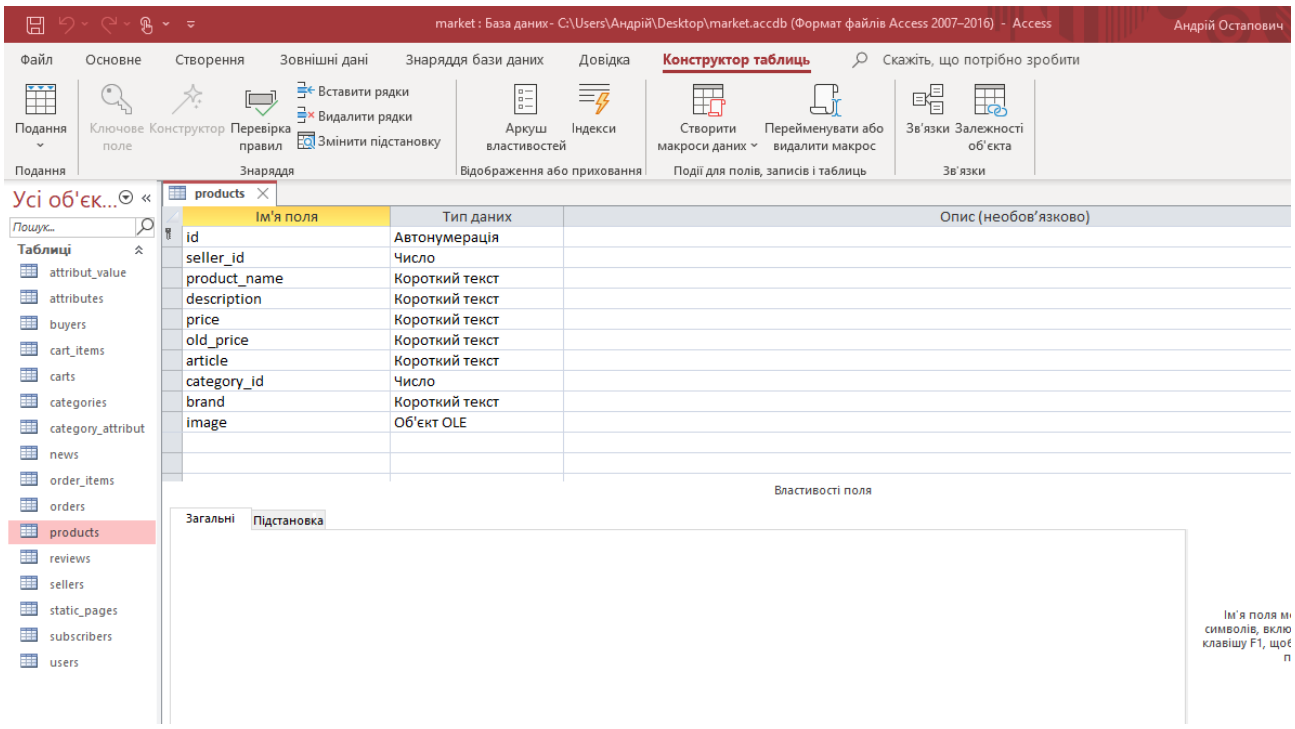

Рисунок 2.2 – Типи даних таблиці products

Основними задачами універсального фремворку стали:

– проектування архітектури:
    1) визначення модульної структури фреймворку;
    2) забезпечення підтримки для різних типів баз даних;
    3) інтеграція з популярними бібліотеками та сервісами;
– розробка основних компонентів:
    1) реалізація системи аутентифікації та авторизації;
    2) створення модулів для управління товарами, категоріями, замовленнями та користувачами;
    3) інтеграція платіжних систем для забезпечення онлайн-оплат;
    4) розробка механізмів для обробки доставок;
– написання документація:
    1) створення детальної документації для розробників;
    2) підготовка прикладів та шаблонів для різних сценаріїв використання;
    3) тестування та налагодження;

4) інтеграція з популярними інструментами тестування;

5) забезпечення можливості автоматизованого тестування та налагодження;

6) розробка інструментів для моніторингу продуктивності та безпеки.

Так як було обрано фреймворк Laravel, то тестування спроектованої БД буде вже пізніше, після додавання міграцій, та фабрик.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Вибір фреймворку для розробки маркетплейсу – це критичний крок у процесі розробки програмного забезпечення. Laravel – є відмінний вибір для створення маркетплейсу через свою гнучкість, широкий функціонал та велику спільноту розробників.

Для початку роботи потрібно обрати IDE, Систему та середовище розробки, також, програми які допоможуть в розробці, налаштування локального середовища, налаштування IDE, MySql, систему управління БД для зручного тестування, та користування по типу PhpMyAdmin, Adminer, систему контролю версій по типу git.

Тож було обрано інструменти, які забезпечать комфортне та ефективне середовище розробки. Нижче описані вибрані технології, та їх переваги.

IDE: Visual Studio Code (VS Code) – це потужний та популярний редактор коду, який обирають багато розробників для роботи над різноманітними проектами, включаючи веб-розробку. Його переваги включають:

– VS Code є легким редактором коду, який швидко завантажується і не вимагає значних ресурсів;

– редактор підтримує широкий вибір розширень для різних мов програмування та фреймворків, що дозволяє налаштувати середовище під конкретні потреби проекту;

– VS Code має вбудовані інструменти для роботи з системою контролю версій Git, що полегшує відстеження змін та спільну роботу над кодом;

– можливість використовувати вбудований термінал для виконання команд прямо з редактора;

– легко налаштовувати середовище розробки завдяки численним параметрам конфігурації та підтримці налаштувань для різних проєктів.

Windows обрано як операційну систему для розробки з наступних причин:

– широка підтримка програмного забезпечення, підтримує більшість інструментів та програм для розробки, що робить її універсальною платформою;

– інтуїтивно зрозумілий та зручним інтерфейс;

– сумісність з різним обладнанням, підтримує широкий спектр апаратного забезпечення, що забезпечує стабільну роботу на різних пристроях;

– Windows пропонує численні можливості для налаштування середовища розробки під конкретні потреби.

Open Server – це потужне локальне середовище розробки, яке обрано з наступних причин:

– легко встановлюється та налаштовується для роботи з різними веб-серверами та базами даних;

– дозволяє використовувати різні версії PHP, що є корисним для тестування сумісності коду;

– підтримує роботу з різними базами даних, включаючи MySQL, що забезпечує гнучкість у виборі бази даних для проекту;

– дозволяє легко керувати різними сервісами, такими як Apache, Nginx, MySQL тощо.

PhpMyAdmin обрано як систему управління базами даних через наступні переваги:

– пропонує інтуїтивно зрозумілий веб-інтерфейс для управління базами даних, що полегшує роботу з ними;

– підтримує більшість функцій для роботи з MySQL, включаючи створення, редагування та видалення баз даних, таблиць та записів;

– дозволяє виконувати складні SQL-запити безпосередньо з інтерфейсу;

– надає різні рівні доступу та аутентифікації для захисту даних.

Git обрано як систему контролю версій через його численні переваги:

– розподілена система версій Git дозволяє кожному розробнику мати повну копію історії проекту, що забезпечує надійність та безпеку;

– легко відстежувати зміни в коді, вносити правки та повертатися до попередніх версій;

– дозволяє створювати та керувати гілками для ізольованої роботи над різними функціями проекту;

– підтримка інтеграції з різними інструментами для розробки, такими як GitHub, GitLab, Bitbucket тощо, що полегшує спільну роботу над проектами.

Також було зроблено добірку пакетів, які спрощують реалізацію, та гнучкість, для вирішення конкретних задач проєкту:

– cloudipsp/php-sdk-v2 – цей пакет дозволяє взаємодіяти з платіжною системою інтернет-магазинів підприємства «ПриватБанк» (CloudIPSP);

– diglactic/laravel-breadcrumbs – використовується для генерації хлібних крихт у Laravel, що дозволяє легко «ходити» по сторінках сайту;

– fomvasss/laravel-lte3 – цей пакет забезпечує інтеграцію з шаблоном AdminLTE для створення адміністративного інтерфейсу;

– fomvasss/laravel-medialibrary-extension – використовується для роботи з медіафайлами, дозволяє зручно зберігати і керувати зображеннями, відео тощо;

– fomvasss/laravel-notification-channel-turbo-sms – інтегрує Laravel з сервісом SMS-розсилки TurboSMS для надсилання повідомлень користувачам;

– fomvasss/laravel-seo – допомагає здійснювати оптимізацію для пошукових систем (SEO) шляхом налаштування мета-тегів та іншого контенту;

– fomvasss/laravel-variables – використовується для зручного керування змінними та налаштуваннями в Laravel;

– guzzlehttp/guzzle – Guzzle дозволяє здійснювати HTTP-запити до зовнішніх сервісів або API;

– kalnoy/nestedset – цей пакет дозволяє працювати зі структурою «вкладених множин» (nested sets) для організації деревоподібних даних, наприклад, категорій;

– laravel/framework – основний фреймворк Laravel, який забезпечує базовий функціонал та структуру додатку;

– laravel/sanctum – дозволяє забезпечити аутентифікацію через API за допомогою токенів;

– laravel/tinker – інтерактивна оболонка Laravel для вивчення та відлагодження коду;

– laravel/ui – надає інструменти для веб-інтерфейсу Laravel, такі як аутентифікація та інші стандартні компоненти;

– lorisleiva/laravel-actions – допомагає в структуруванні та організації дій (actions) в Laravel додатках;

– maatwebsite/excel – дозволяє працювати з Excel-файлами у Laravel;

– orchestra/parser – використовується для обробки та аналізу структурованих даних, таких як XML або JSON;

– rap2hpoutre/laravel-log-viewer – надає інтерфейс для перегляду журналів логів Laravel;

– spatie/laravel-medialibrary – дозволяє зручно керувати медіафайлами в Laravel додатках;

– spatie/laravel-permission – для реалізації системи контролю доступу на основі ролей та дозволів.

Ці пакети використовуються для різних аспектів розробки маркетплейсу і надають різноманітні можливості для реалізації функціональності та оптимізації роботи додатку. Використання таких інструментів допоможе побудувати ефективну та функціональну інформаційну систему.

У розробці будь-якого сучасного веб-додатку важливу роль відіграють принципи CRUD (Create, Read, Update, Delete) та MVC (Model-View-Controller). Використання цих принципів забезпечує структурованість і логічну послідовність в управлінні даними та представленням інтерфейсу.

Платіжна система Fondy була обрана для інтеграції в маркетплейс через її гнучкість та можливість обробки транзакцій з різних джерел [5]. Використання CRUD для операцій з платежами дозволяє ефективно створювати, читати, оновлювати та видаляти платіжні записи. З переваг цієї платіжної системи можна відзначити:

– широкий набір платіжних методів;
– простота інтеграції через API;
– високий рівень безпеки платежів.

Для обробки фінансових операцій важливо мати актуальні курси валют. API валют ПриватБанку надає точні та актуальні дані про курси валют, що дозволяє проводити валютні операції з високою точністю [6]. З переваг API ПриватБанку:

– надійність і точність даних;
– простота інтеграції;
– швидкий доступ до інформації про курси валют.

Для реалізації системи сповіщень використовувалась платформа Турбо SMS, яка дозволяє надсилати SMS повідомлення користувачам про статус замовлень, нові пропозиції та інші важливі події, з переваг можна підкреслити:

– високу швидкість доставки повідомлень;
– надійність і доступність послуги;
– простий API для інтеграції.

Для тестування та відправки електронних листів були обрані Mailtrap і Google SMTP. Mailtrap використовується для тестування електронних листів у безпечному середовищі, тоді як Google SMTP забезпечує надійну відправку реальних листів, про переваги Mailtrap:

– безпечне середовище для тестування листів;

– простий інтерфейс для перегляду та аналізу листів;

– легкість інтеграції з Laravel.

Переваги Google SMTP:

– надійність і стабільність роботи;

– висока швидкість доставки листів;

– простота налаштування та інтеграції.

Laravel використовує шаблонізатор Blade, який забезпечує чистий та зрозумілий спосіб написання HTML з можливістю використання шаблонів та компонентів він: легкий у використанні та інтеграції, має підтримку секцій та макетів для структуризації шаблонів.

Для створення користувацького інтерфейсу був обраний Bootstrap, а для адміністрування – AdminLTE3. Bootstrap забезпечує адаптивний дизайн і простоту у створенні сучасних інтерфейсів, тоді як AdminLTE3 надає потужний інструментарій для створення адмін-панелей [7].

Для вирішення прикладних задач у розроблюваній системі маркетплейсу можуть бути використані різні алгоритми та математичні моделі. Ось деякі можливі аспекти, які можуть бути враховані:

– алгоритми обробки замовлень – розробка алгоритмів для обробки та оптимізації замовлень, включаючи їхнє розподілення, валідацію, підтвердження та відміну;

– математичні моделі для аналізу статистики – розробка математичних моделей для аналізу даних статистики, таких як звіти про продажі, прибуток, активність користувачів тощо;

– алгоритми рекомендацій – використання алгоритмів машинного навчання для рекомендаційних систем, які аналізують поведінку користувачів та пропонують їм відповідні товари або послуги;

– алгоритми пошуку та фільтрації – розробка алгоритмів для забезпечення ефективного пошуку та фільтрації товарів за різними критеріями, такими як ціна, категорія, рейтинг тощо;

– моделі управління користувачами та підписниками – розробка моделей для управління користувачами та їхніми підписками, включаючи аналіз їхньої активності та інтересів;

– моделі аналізу відгуків та рейтингів – розробка моделей для аналізу та обробки відгуків користувачів та їхніх рейтингів, що дозволяє покращити якість обслуговування та товарів;

– алгоритми розміщення товарів – розробка алгоритмів для оптимального розміщення товарів на сторінках магазину з метою підвищення їхньої видимості та продаж.

Висновки до розділу 2

Розробка маркетплейсу є складним і багатогранним процесом, що вимагає ретельного вибору засобів, методів та алгоритмів для досягнення оптимальної продуктивності та функціональності системи. Вибір фреймворку для розробки маркетплейсу є критичним кроком, який значно впливає на ефективність та успіх проекту. Laravel, з його гнучкістю, простотою використання, активною спільнотою та багатим набором можливостей, є відмінним вибором для створення сучасних маркетплейсів.

Основні переваги використання Laravel включають його зрозумілий синтаксис, наявність великої кількості готових компонентів і інструментів, підтримку багатьох маркетплейсових можливостей, активну спільноту розробників, а також вбудовані механізми безпеки. Використання специфічних пакетів, таких як cloudipsp/php-sdk-v2, guzzlehttp/guzzle, spatie/laravel-medialibrary, та інших, дозволяє значно спростити розробку та підвищити гнучкість проекту, забезпечуючи при цьому високу якість та надійність системи.

Для вирішення прикладних задач у розроблюваній системі маркетплейсу можуть бути використані різні алгоритми та математичні моделі, такі як алгоритми обробки замовлень, математичні моделі для аналізу статистики, алгоритми рекомендацій, пошуку та фільтрації, моделі управління користувачами та підписниками, а також алгоритми розміщення товарів. Ці алгоритми та моделі сприяють оптимізації процесів, покращенню користувацького досвіду та підвищенню ефективності маркетплейсу.

Таким чином, вибір Laravel як основного фреймворку для розробки маркетплейсу, разом з ретельно підібраними пакетами та алгоритмами, забезпечить створення високоякісного, безпечного та масштабованого маркетплейсу. Це дозволить не тільки задовольнити сучасні вимоги ринку, але й бути готовим до майбутніх викликів та змін. Використання інноваційних підходів та передових технологій сприятиме успішному розвитку та конкурентоспроможності маркетплейсу, забезпечуючи його довготривалу стабільність та зростання.

РОЗДІЛ 3
РОЗРОБКА УНІВЕРСАЛЬНОГО ФРЕЙМВОРКА ДЛЯ МАРКЕТПЛЕЙСІВ НА LARAVEL

3.1 Практична реалізація об’єкта проєктування

Для початку потрібно створити сам проєкт. Для цього ми запустим наш налаштований Open Server (рис. 3.1).

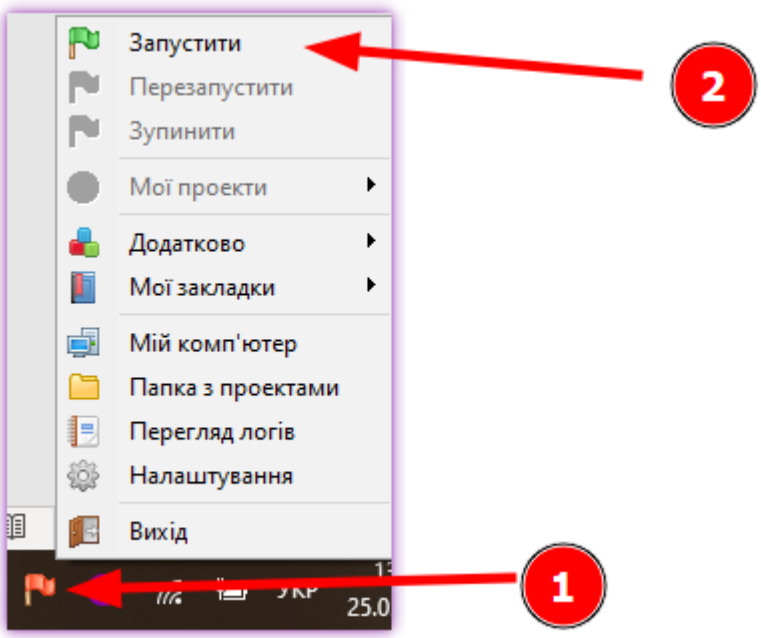

Рисунок 3.1 – Запуск Open Server

Після запуску можна відкрити термінал, було використано термінал Open Server який відразу з налаштованим середовищем PHP (рис. 3.2).

Також за допомогою Open Server можна відкрити мій комп’ютер, папку з проєктами, переглянути логи та налаштування.

В вкладці додатково, можна побачити різні інтерфейси управління базою данних, а також, інформацію про програму, в ній описана версія програми, та інформацію розробників, ну і звичайний калькулятор.

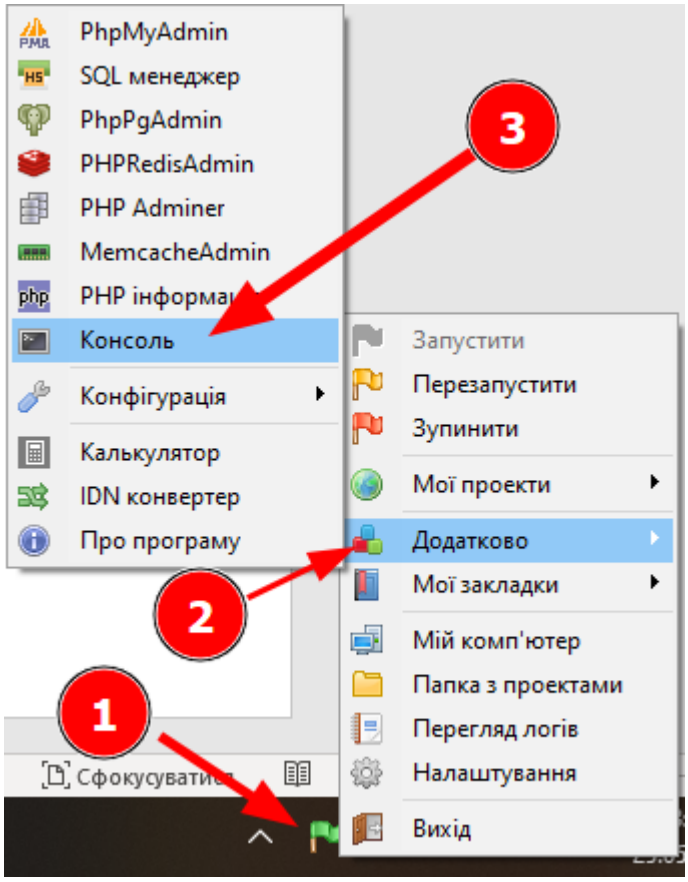

Рисунок 3.2 – Відкриття терміналу

Відкривши термінал, створим новий проєкт Laravel та встановим залежності командами:

– composer create-project laravel/laravel (назва проєкту);

– composer install.

Тоді було налаштування конфігурації файлу .env в неї входить налаштування з’єднання з базою даних, яку створим в phpMyAdmin, після чого вводим назву бази даних, ім’я користувача root а також пароль, за звичай його немає, тоді запустим в терміналі: php artisan migrate, що запустить базові міграції проекту Laravel.

Міграції дозволяють визначати структуру бази даних за допомогою коду, що робить її створення та оновлення контрольованими та відтворюваними. Міграції забезпечували створення таблиць, визначення їхньої структури та відношень між ними. Вони також дозволяють легко оновлювати структуру бази даних у разі змін вимог до проекту.

Тож створимо, наші міграції з моєї спроектованої БД. Так як я маю досвід користування таким фреймворком, я відразу створюю, моделі та міграції, командою php artisan make:model (назва моделі) -m , після чого переніс поля для міграцій (лістинг 3.1). Прописав зв’язки в моделях (лістинг 3.2), так як зв’язків багато, та вони різні то важливо їх правильно описати щоб не виникало конфліктів, та помилок.

Що до моделей то вони представляють собою структуру даних у базі даних та взаємодію з нею. Вони дозволяють маніпулювати даними за допомогою ORM (Object-Relational Mapping). Моделі були створені для всіх основних сутностей. Вони містять атрибути та методи, необхідні для роботи з цими сутностями, такі як відношення з іншими моделями, запити до бази даних, валідація даних тощо.

Лістинг 3.1 – Додавання полів для міграції

```
public function up(): void
 {
     Schema::create('products', function (Blueprint $table) {
         $table->id();
         $table->unsignedBigInteger('seller_id');
         $table->string('name');
         $table->text('description');
         $table->decimal('price', 10, 2);
         $table->decimal('old_price', 10, 2)->nullable();
         $table->string('article');
         $table->unsignedBigInteger('category_id');
         $table->string('brand');
         $table->string('slug');
         $table->timestamps();
     });
 }

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

Кінець лістингу 3.1

Лістинг 3.2 – Описування зв’язків в моделях

```
public function categories()
   {
     return $this->belongsTo(Category::class, 'category_id', 'id');
   }

   public function seller()
   {
       return $this->belongsTo(Seller::class);
   }
   public function properties()
   {
       return $this->belongsToMany(Property::class);
   }
   public function orders()
   {
       return $this->hasMany(Order::class, 'product_id', 'id');
   }
   public function review()
   {
       return $this->hasMany(Review::class);
   }
```

Кінець лістингу 3.2

Тоді було запущено міграції знову, так як їх було додано і після всіх цих маніпуляцій лишається, додати можливість редагування, видалення та виведення даних, зазвичай це відбувається в адмін панелі якою буде мінятися наповнення.

Так як контролери в Laravel, відповідають за обробку запитів користувачів та керування бізнес-логікою програми. Вони отримують дані від моделей та передають їх у вигляді, зручному для представлення користувачу створюються вони командою php artisan make:controller (Назва контролеру) Controller. Приставка Controller обов’язкова, було створено певну архітектуру, для зручної подорожі по папках (рис. 3.3) також на ньому можна побачити описаний контролер, з майже стандартними CRUD методами, вони були створені для кожного основного ресурсу: продукти, категорії, користувачі, замовлення, відгуки тощо. Вони забезпечують маршрутизацію та управління даними для відповідних сторінок та функцій.

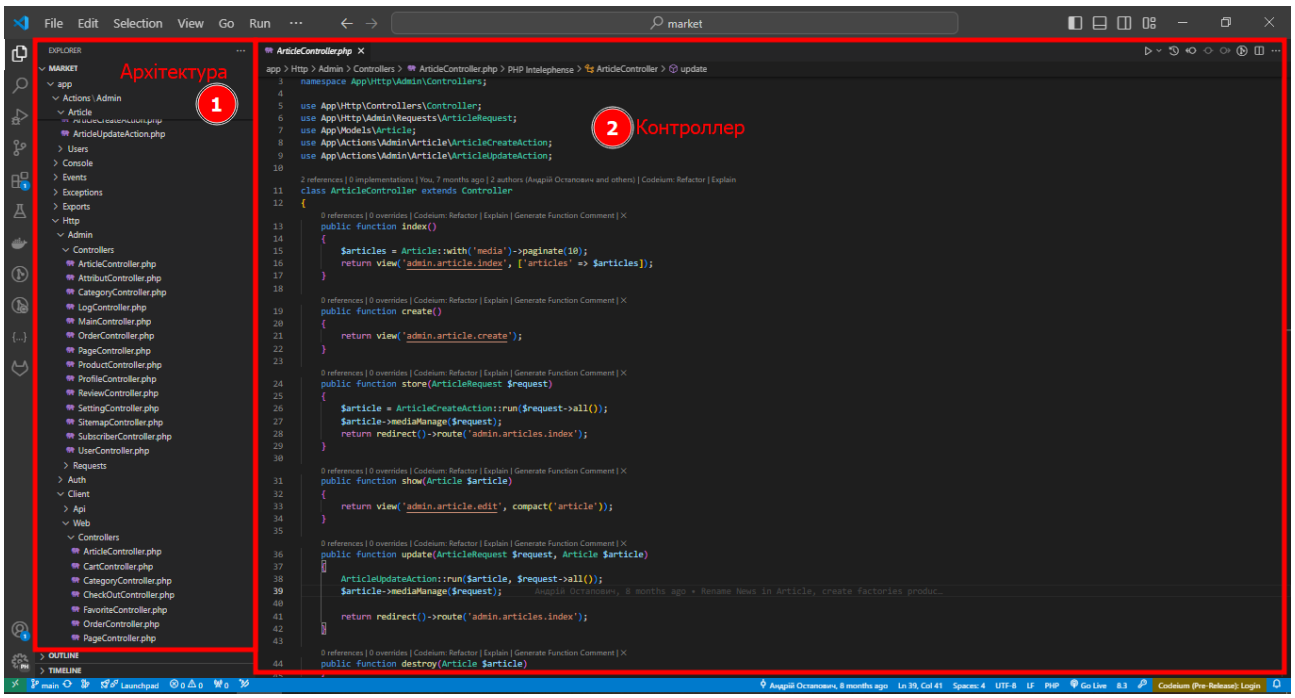

Рисунок 3.3 – Архітектура проєкту та контролеру

Тоді було прописано роути (ендпоінти) для відсліжування url і задав їм методи контролерів додав префікс адмін і перевірку на роль да, через мідвейр та задав них як ресурси, коротко по методу, скажу що він неймовірно зручний для звичайних CRUD контролерів бо створює відразу всі стандартні ендпоінти не прописуючи їхне відслідкування (рис. 3.4).

Роутинг в Laravel дозволяє визначати шляхи, за якими користувачі можуть отримувати доступ до різних частин веб-додатку. Це забезпечує зв’язок між URL та відповідними контролерами.

Однією з переваг можна підкреслити те що він підтримує параметризовані URL-шляхи, шаблони для відповідності динамічним URL та групування маршрутів для організації та забезпечення чистоти коду. Цей підхід дозволяє створювати складні маршрути для різноманітних додатків та забезпечує легкість управління та підтримки.

Також роутинг дозволяє розробникам ефективно керувати тим, як користувачі взаємодіють з веб-додатком, і забезпечує гнучкість та чіткість визначення шляхів доступу до різних функціональних можливостей. Це робить Laravel досконалим вибором для розробки веб-додатків різної складності.

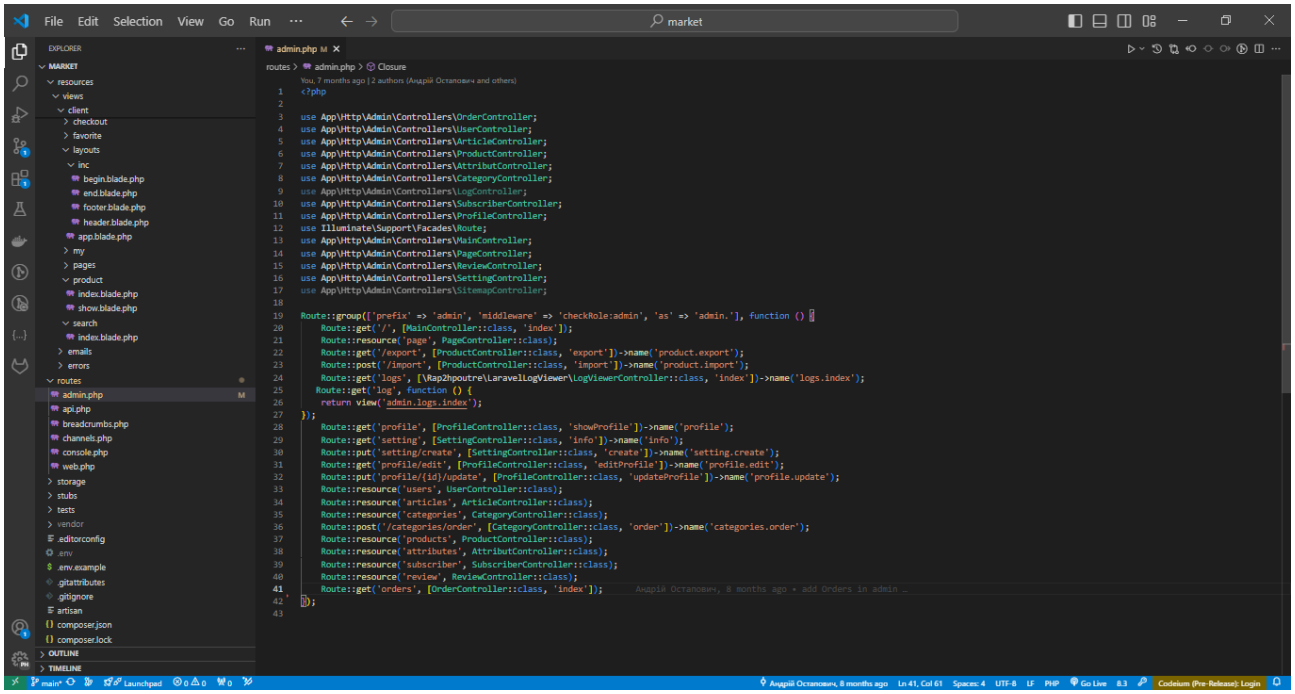

Рисунок 3.4 – Відслідковування ендпоінтів

Пізніше, було розроблено інтерфейс адмін панелі, це відслідковування ендпоінтів, вивід даних з бази, також прокидання їх через контролери, загалом, дуже багато роботи, але так як я це робив на шаблонізаторі Laravel це Blade то це дуже спрощувало задачу, що зменшувало код, та економило час (рис. 3.5).

Blade – це потужний та зручний шаблонізатор, який становить неодмінну частину Laravel. Використання Blade значно полегшує роботу з HTML шаблонами, надаючи можливість легко і ефективно вбудовувати PHP-код безпосередньо в HTML шаблони. Це дозволяє створювати динамічні веб-сторінки та компоненти, забезпечуючи при цьому зручний синтаксис та чистоту коду.

Однією з ключових особливостей Blade є можливість використання секцій, макетів і компонентів. Секції дозволяють визначити блоки контенту, які можуть бути перевизначені в дочірніх шаблонах. Макети дозволяють створювати основний шаблон сторінки з загальними елементами, такими як хедер або футер і використовувати його для всіх сторінок вашого сайту. Компоненти дозволяють

виділити повторювані елементи і створити їх власні шаблони для подальшого повторного використання.

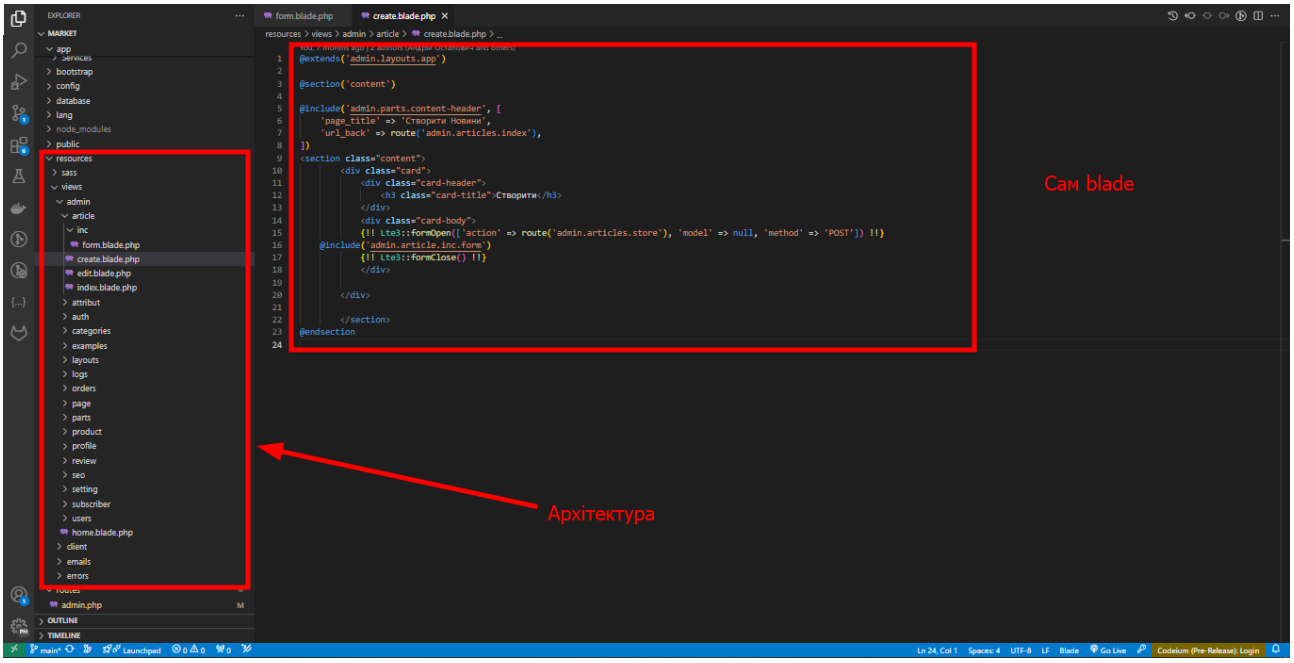

Рисунок 3.5 – Blade та архітектура

Щоб було ще простіше, то було використано пакет, fommvas/adminlte3, який вже з готовими Bootsrap компонентами, які підкреслюють стилі Admin LTE3 (рис. 3.6).

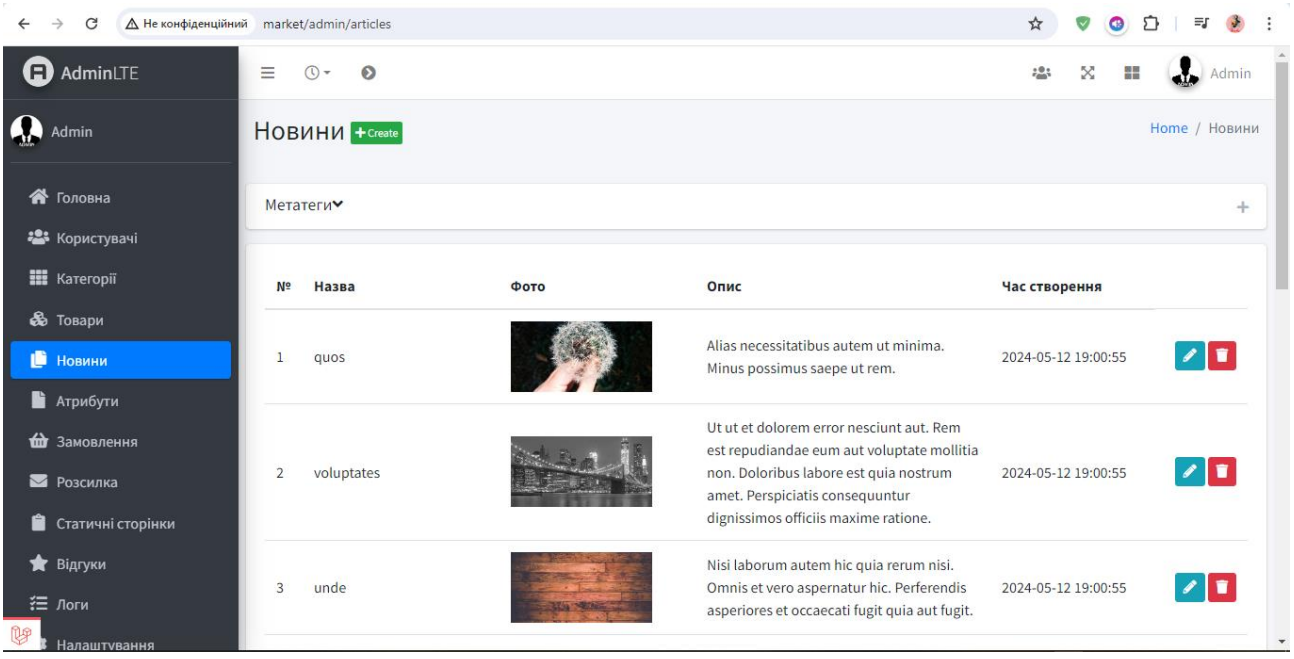

Рисунок 3.6 – Адмін панель

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Для забезпечення належного функціонування та коректності роботи розробленої інформаційної системи, необхідно провести детальне тестування та налагодження. У процесі тестування ми використовуємо кілька підходів, включаючи розробку фабрик та сідерів, а також тестування коду за допомогою інструментів PhpUnitTests та дебагінгу з використанням методів dd() та var_dump().

Фабрики у Laravel дозволяють створювати об’єкти моделей із заповненими даними для тестування. Це особливо корисно для автоматичного генерування великої кількості записів у базі даних, що дозволяє швидко наповнити таблиці необхідними даними для тестування та налагодження. Командою для створення фабрики є php artisan make:factory (назва) Factory. Після чого їх одразу було заповнено (лістинг 3.3).

Фабрики дозволяють легко створювати фейкові дані для різних моделей, що значно спрощує процес тестування, забезпечуючи доступ до необхідних даних без ручного введення.

Лістинг 3.3 – Створення фабрики користувача

```
public function definition(): array
  {
      return [
          'name' => fake()->name(),
          'email' => fake()->unique()->safeEmail(),
          'email_verified_at' => now(),
          'role' => 'seller',
          'password' => bcrypt(Str::random(8)),
          'remember_token' => Str::random(10),
      ];
  }
```

Кінець лістингу 3.3

Сідери ж використовуються для автоматичного заповнення бази даних початковими даними. Це особливо корисно для налаштування системи та забезпечення наявності необхідних даних для її коректного функціонування.

Primary Seeder створює користувача-адміністратора та забезпечує мінімальні дані для запуску проекту (лістинг 3.4).

Лістинг 3.4 – PrimarySeeder

```
        $user = User::create([
                    'name' => 'Admin',
                    'email' => 'admin@app.com',
                    'password' => 'password',
                    'role' => 'admin',
                ]);

      Page::create([
                    'id' => '1',
                    'name' => 'Головна',
                    'content' => 'Вітаємо на сайті',
                    'slug' => 'home',
                    'template' => 'home',
                ]);
```

Кінець лістингу 3.4

Dummy Seeder використовується для наповнення бази даних тестовими даними: товарами, категоріями, атрибутами, новинами, замовленнями, продавцями та звичайними користувачами (лістинг 3.5).

Лістинг 3.5 – DummySeeder

```
        User::factory(10)->create();
              Seller::factory(10)->create()
                  ->each(function (Seller $seller) {
                    $categories = Category::factory(rand(3,10))->create();
                          $attributes = Attribute::factory(rand(2, 4))-
>create();
                     $properties = Property::factory(count($attributes))-
>create();
                                $categories->each(function ($category) use
($attributes) {
                                $category->attributes()->attach($attributes-
>random(rand(1, 2))->pluck('id'));
                      });
                      $products = Product::factory(rand(4, 10))->create([
```

```
                    'seller_id' => $seller->id
                   ]);
                  $properties->each(function ($property) use ($products)
{
                                  $property->products()->attach($products-
>random(rand(1, 2))->pluck('id'));
                   });
                });
```

Кінець лістингу 3.5

Для тестування коду було використано PhpUnitTests, що є стандартним інструментом у світі PHP для написання та виконання тестів. Тести дозволяють перевіряти коректність роботи різних частин системи, забезпечуючи надійність і стабільність додатку.

PhpUnitTests дозволяє створювати та запускати юніт-тести, що перевіряють функціональність окремих модулів та компонентів системи.

Методи dd() (dump and die) та var_dump() використовуються для виведення значень змінних та зупинки виконання коду, що дозволяє детально аналізувати значення та виявляти помилки.

Фабрики та сідери значно спрощують процес тестування та налагодження інформаційної системи, забезпечуючи швидке наповнення бази даних необхідними даними. PhpUnitTests та дебагінг-методи dd() та var_dump() допомагають виявляти та виправляти помилки, забезпечуючи стабільність та надійність роботи системи. Ці інструменти та підходи є невід’ємною частиною процесу розробки та тестування у Laravel, що дозволяє створювати якісні та ефективні веб-додатки.

3.3 Розробка клієнтської частини

Оскільки я є бекенд-розробником, було вирішено завантажити готову фронтенд-частину та інтегрувати її в шаблонізатор Blade (рис. 3.7). Цей процес включав створення нових контролерів, роутів, додавання HTML (рис. 3.8) до блейдів, а також реалізацію функціональності корзини, що охоплює додавання

товарів, видалення, оформлення замовлень для зареєстрованих та незареєстрованих користувачів з оновленням корзини.

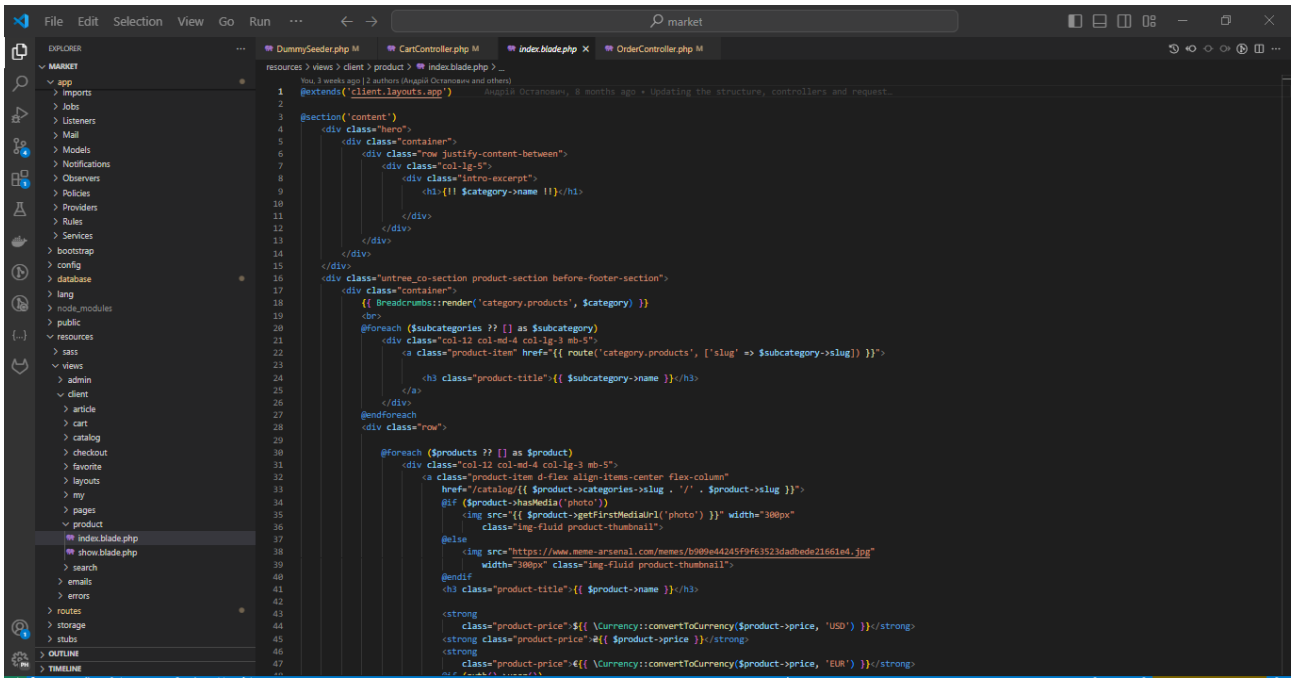

Рисунок 3.7 – Наверстані Blade

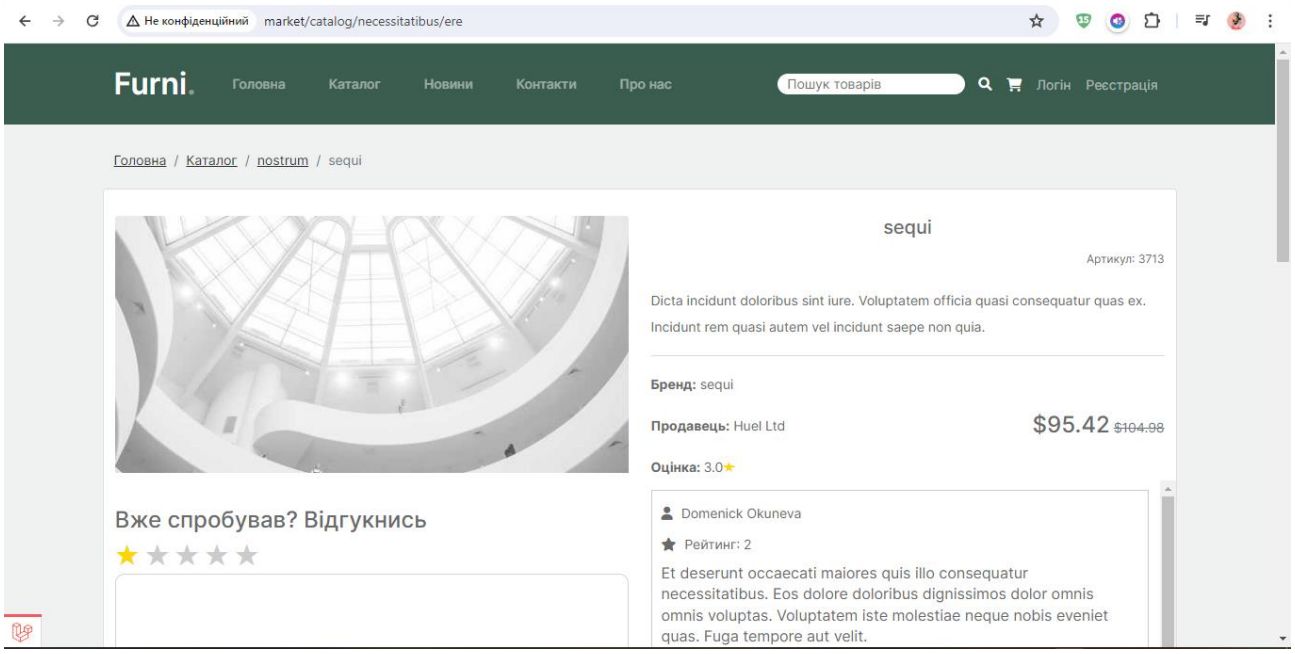

Рисунок 3.8 – Продукт на клієнтській частині

Функціональність корзини включає додавання товарів, їх видалення та оформлення замовлень. Було створено контролери, моделі та роутинг для

забезпечення цієї функціональності. Для зареєстрованих користувачів дані корзини зберігаються в базі даних, а для гостей – у cookie (лістинг 3.6).

Лістинг 3.6 – CartController

```
 public function index(CartService $cartService)
    {
        $cartId = request()->cookie('cart_id');
        $cartItems = $cartService->mergeGuestCart($cartId);

return view('client.cart.index', compact('cartItems'));
    }
    public function addToCart($productSlug, CartService $cartService)
    {
        $cartService->addToCart($productSlug);

        return redirect()->back();
    }
    public function remove($itemId, CartItemService $cartItemService)
    {
        $cartItem = CartItem::find($itemId);
        $cartItemService->removeFromCart($cartItem);

        return redirect()->back();
    }
```

Кінець лістингу 3.6

Таким чином, інтеграція фронтенд-частини, розробка контролерів, моделей, блейдів, роутів та міграцій дозволила створити функціональний маркетплейс з можливістю обробки замовлень як для зареєстрованих, так і для незареєстрованих користувачів.

3.4 Інтеграція платіжної системи та інших API

Платіжна система буде використовуватись Fondy, так як я вже з нею працював, це буде не важко, але важливо врахувати логіку суми оплати, інтеграція населених пунктів України для вибору місця доставки, а також для зручності, та розуміння цін на сайті, інтеграція публічного API приват банку для курсу валют.

Для інтеграції Fondy потрібно зареєструватись отримати конфіги, тестового мерчанта, такі як merchant_id та merchant_secret. Тоді можна відправити запит на їхній url данні, що до оплати такі як сума, валюта, мова, якою буде відображатись їхня сторінка оплати, url для вебхука, який обробляє відповідь платіжної системи, а також час сесії (рис. 3.9).

Коли ми надсилаєм запит на Fondy то він нам повертає певні данні, саме той url на якому людина зможе оплатити, тож ми її відразу редіректим на сторінку оплати (рис. 3.10).

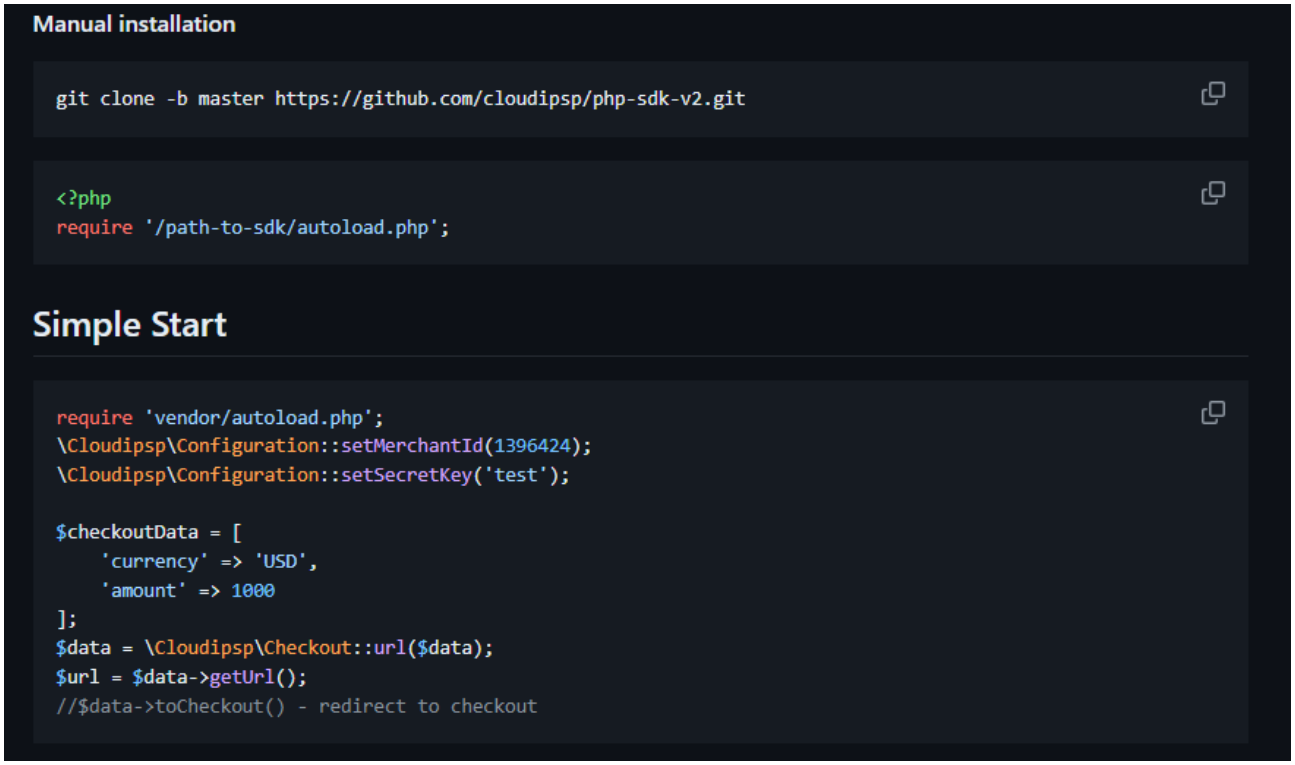

Рисунок 3.9 – Приклад використання Fondy Sdk

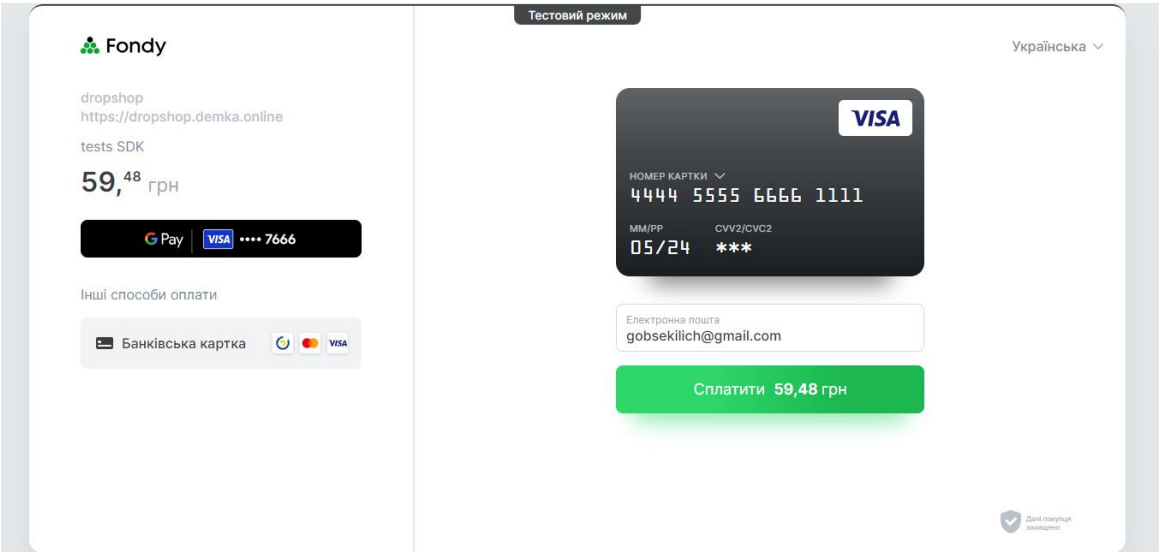

Рисунок 3.10 – Сторінка оплати Fondy

Що до API приват банку то було просто надсилано запит на їхній публічний url та витягання з масива даних що до курсу гривні, долара та євро. Після переводим всі товари і в інші валюти, динамічно, в залежності від курсу.

Також було підключено Smpt, це mail trap та TurboSms вона для того щоб надіслати користувачу який замовив товари, про успішно оформлене замовлення [8].

Тут також потрібно реєструватись, після обрати технологію для .env та скопіювати конфіги, зробивши ці кроки, ми відразу можем, тестувати (рис. 3.11).

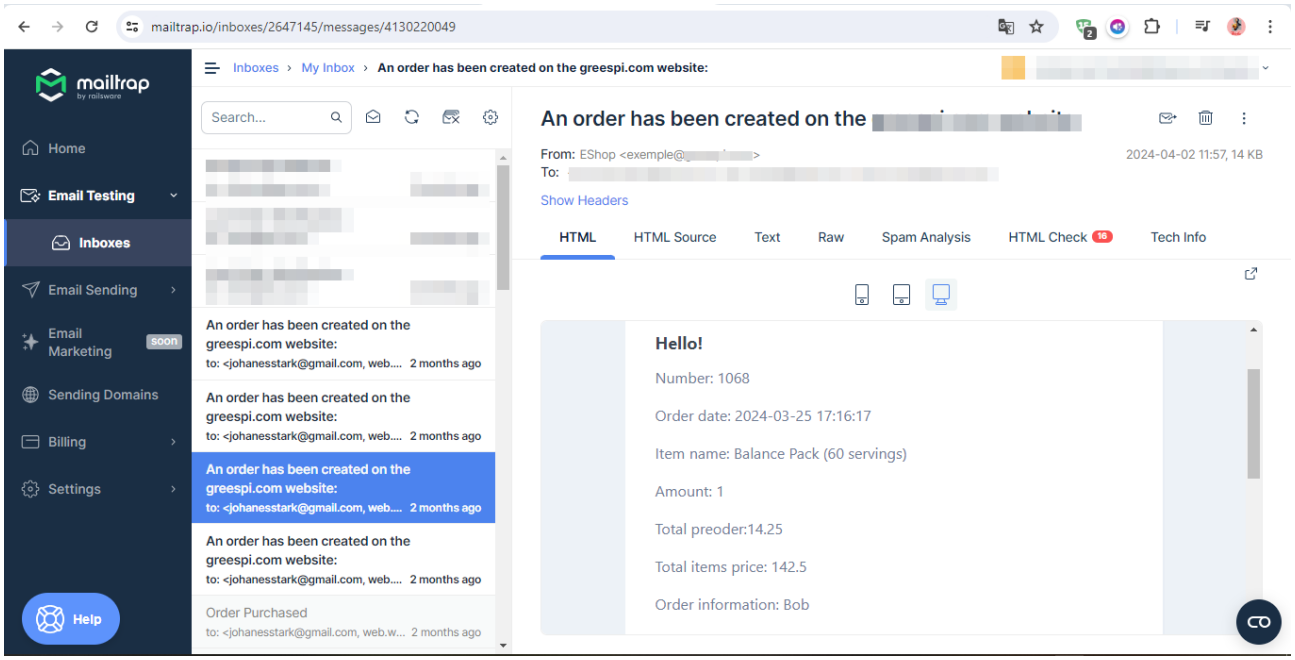

Рисунок 3.11 – MailTrap, тестування

3.5 Розробка RESTful API для майбутніх інтеграцій

Розробка RESTful API є важливою частиною будь-якої сучасної веб-системи, оскільки вона дозволяє іншим додаткам та сервісам взаємодіяти з вашою платформою [9].

REST (Representational State Transfer) – це архітектурний стиль, який використовує HTTP методи для створення веб-сервісів. Основні методи HTTP, що використовуються в REST API, це:

– GET отримання ресурсів;

– POST створення нових ресурсів;

– PUT/PATCH оновлення існуючих ресурсів;

– DELETE видалення ресурсів.

Це загальні підходи, які було використано, для інтеграції (рис. 3.12) [10].

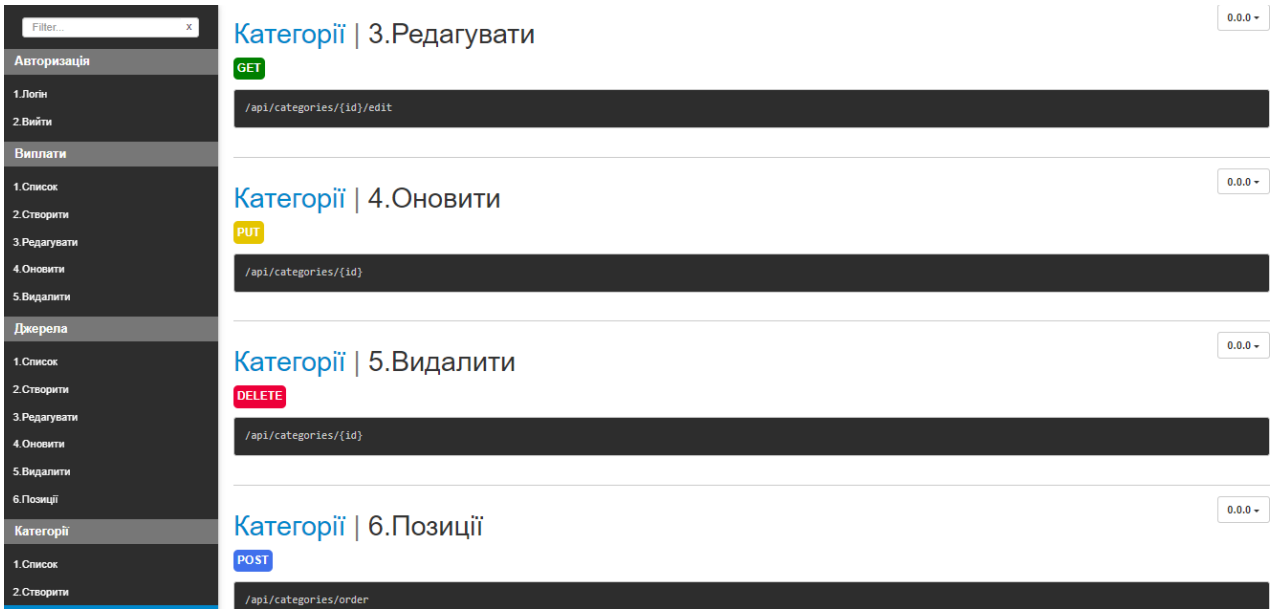

Рисунок 3.12 – Доступні ендпоінти категорії

3.6 Розробка документації

Для створення документації API, використовувався пакет apidoc. Цей пакет автоматично генерує документацію з коментарів у коді, що значно спрощує процес підтримки актуальної документації для всіх доступних ендпоінтів. Встановлюється пакет такою командою: npm install apidoc -g.

У кожному контролері додаються коментарі у спеціальному форматі, який розпізнає apidoc (рис. 3.13).

Він аналізує коментарі, витягує з них інформацію про структуру та параметри API, і автоматично створює документацію у зручному форматі, яка може бути легко переглянута та використана іншими розробниками.

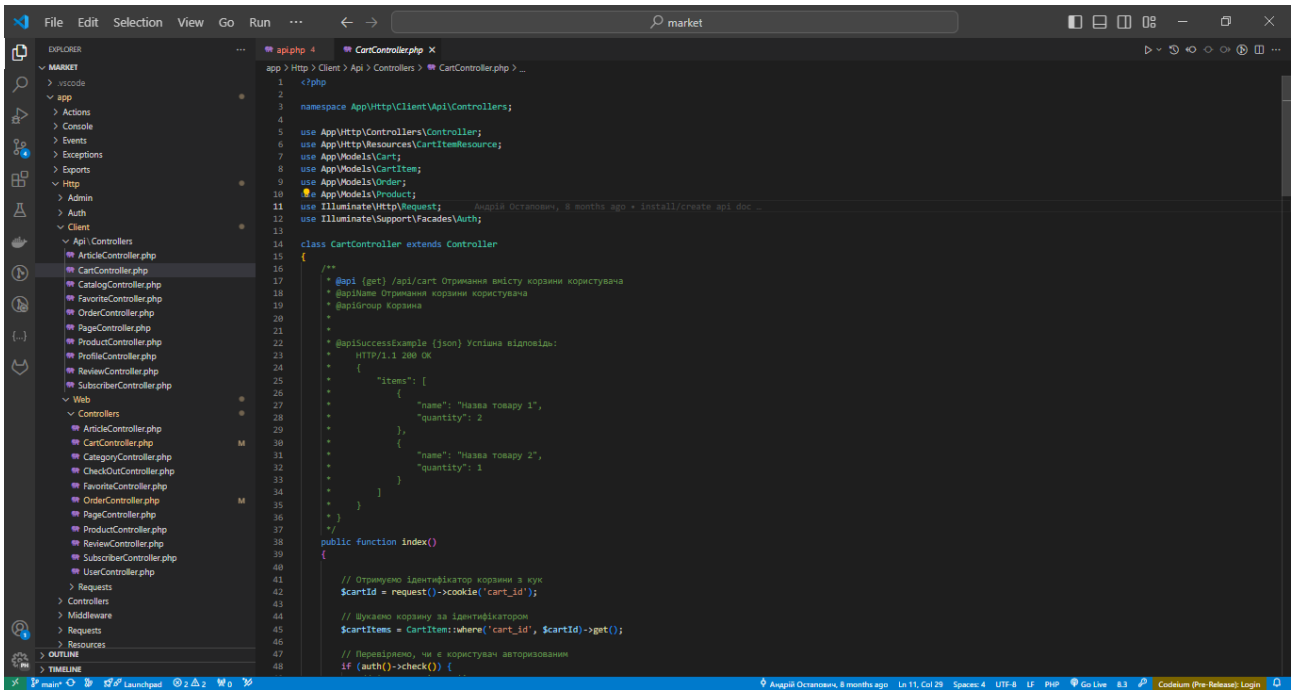

Рисунок 3.13 – Спеціальні коментарі для Api Doc

Цей пакет дозволяє швидко і ефективно створювати документацію, яка буде доступна для перегляду через браузер, що допомагає команді розробників та іншим зацікавленим сторонам краще розуміти функціональність API [11].

Ну і останнім кроком це було написання README.md файлу, в якому було описано, що потрібно для розгортання, як крок за кроком налаштувати проєкт, невелику кількість технологій які були використані. Логін та пароль для першого входу [12].

Висновки до розділу 3

Розробка маркетплейсу в рамках цього проєкту включала декілька важливих етапів, кожен з яких був реалізований з використанням потужних інструментів та методологій.

Створення контролерів, моделей, Blade-шаблонів, маршрутизація та міграції були основними кроками у створенні функціонального бекенду. Використання Laravel значно спростило ці завдання завдяки своїй інтуїтивно зрозумілій архітектурі та розширеним можливостям.

ВИСНОВКИ

Розробка маркетплейсу є складним та багатогранним процесом, що потребує ретельного планування та вибору технологій. Аналіз сучасного стану фреймворків показав важливість створення нового, універсального рішення, яке б відповідало потребам сучасного електронного бізнесу.

При виконані кваліфікаційної роботи було проведено:

– детальний аналіз існуючих фреймворків, визначені їх основні характеристики, переваги та недоліки. Це дало змогу зрозуміти сучасні тенденції та вимоги ринку;

– проаналізовано різні фреймворки для веб-розробки. Було виділено їхні сильні сторони, а також виявлено недоліки, які слід врахувати при розробці нового фреймворку;

– на основі проведеного аналізу були сформульовані вимоги до нового фреймворку, включаючи його функціональні та не функціональні аспекти, такі як гнучкість, масштабованість, безпека та простота використання.

Також було створено:

– концептуальну модель фреймворку, яка включає основні компоненти та їх взаємодію. Концептуальна модель дозволила визначити архітектурні рішення та ключові функціональні можливості;

– детальну структуру бази даних, яка включає сутності, їх атрибути та взаємозв’язки. Структура бази даних забезпечує фундамент для реалізації всіх необхідних функцій маркетплейсу;

– основні компоненти системи, такі як управління користувачами, товарами, замовленнями та категоріями, ці функції забезпечують базову роботу маркетплейсу;

– механізми реєстрації, входу та управління сесіями користувачів, що гарантує безпеку доступу до системи та захист персональних даних;

– інтеграцію до платіжних сервісів, таких як Fondy, та міст України, платіжна система забезпечує зручність оплати, а дані міст України, логістику для користувачів;

– було запущено сідери які запускають фабрики, для наповнення данних.

– інтеграцію API курсу валют Приват банку що є зручним, при замовленнях, зарубіжними користувачами;

– інструменти для автоматичного створення тестових даних, що спрощує процес розробки та тестування системи;

– проведено тестування компонентів системи з метою виявлення та виправлення помилок це забезпечило надійність та стабільність роботи фреймворку;

– додано функціональність для отримання актуальної інформації про курси валют, що є важливим для розрахунків та транзакцій;

– RESTful API, що забезпечує взаємодію з іншими системами та сервісами. Це робить фреймворк гнучким та розширюваним;

– якісну документації API використано спеціалізовані інструменти, що спрощує розробку та підтримку майбутніх інтеграцій.

Одним з останніх кроків було підготовка детальної документації, яка включає інструкції для користувачів та розробників. Що забезпечує легкість у використанні та налаштуванні системи.

Система була ретельно протестована та підготовлена до впровадження, що гарантує її готовність до реальної експлуатації та подальшого розвитку.

Створення такого проєкту займає багато часу та зусиль, проте він є важливим кроком у розвитку електронного бізнесу. Проєкт є чудовим стартом для новачків, демонструючи основні принципи та підходи до розробки веб-додатків з використанням Laravel та інших сучасних інструментів. Інтеграція різних API та розробка RESTful сервісів дозволяє створювати конкурентоспроможні та масштабовані рішення, готові до викликів майбутнього.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Laravel. URL: http://laravel.com/ (дата звернення: 17.04.2024).

2. Роз’яснення по API. URL: https://dev.ua/news/chto-takoe-api-prostym-yazykom (дата звернення: 20.04.2024).

3. RESTful API. URL: https://foxminded.ua/shcho-take-rest-api/ (дата звернення: 25.04.2024).

4. MySQL. URL: https://www.mysql.com/ (дата звернення: 30.04.2024).

5. Fondy платіжна система. URL: https://fondy.eu/ (дата звернення: 05.05.2024).

6. ПриватБанк API. URL: https://api.privatbank.ua/ (дата звернення: 10.05.2024).

7. AdminLTE3. URL: https://adminlte.io/ (дата звернення: 30.05.2024).

8. TurboSMS. URL: https://turbosms.ua/ (дата звернення: 15.05.2024).

9. Документація API docs. URL: https://docs.phpdoc.org/3.0/guide/references/phpdoc/tags/api.html (дата звернення: 20.05.2024).

10. Документація Guzzle. URL: https://guzzlephp.org/ (дата звернення: 25.05.2024).

11. Роз’яснення по Http запиту. URL: https://qalight.ua/baza-znaniy/http-zapyt-http-request/ (дата звернення: 28.05.2024).

12. Документація cloudipsp/php-sdk-v2. URL: https://github.com/cloudipsp/php-sdk-v2 (дата звернення: 29.05.2024).