МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ ІЗ ЗАСТОСУВАННЯМ ФРЕЙМВОРКІВ LARAVEL ТА VUE.JS»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.М. Данілов
(ініціали та прізвище)
Керівник завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

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

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Данілову Данилу Миколайовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет-магазину із застосуванням фреймворків Laravel та Vue.js

керівник роботи Гребенюк Сергій Миколайович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

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

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

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.01.2024	
2.	Збір вихідних даних.	05.02.2024	
3.	Обробка методичних та теоретичних	26.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2024	

Студент _____ Д.М. Данілов
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.М. Гребенюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет-магазину із застосуванням фреймворків Laravel та Vue.js»: 51 с., 14 рис., 10 джерел.

ФРЕЙМВОРК, API, CSRF, ELOQUENT, MARIADB, MYSQL, MVC, PHP, REST.

Об’єкт дослідження – процес розробки інтернет-магазину з використанням фреймворків Laravel і Vue.js.

Мета роботи: створення високоякісного інтернет-магазину на базі фреймворків Laravel і Vue.js, що забезпечить ефективне управління товарами та замовленнями.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання та проєктування програмного забезпечення.

У кваліфікаційній роботі було проведено аналіз предметної області, огляд існуючих фреймворків для розробки вебдодатків та описано засоби для створення вебдодатків. На основі цих теоретичних відомостей розроблено проєкт інтернет-магазину із застосуванням сучасних фреймворків Laravel та Vue.js. Результатом виконання кваліфікаційної роботи є інтернет-магазин з клієнт-серверною архітектурою, написаний з використанням бази даних MariaDB, бекенд-розробки на Laravel та фронтенд-розробки на Vue.js.

SUMMARY

Bachelor’s qualifying paper “Development of an Online Store Using Laravel and Vue.js Frameworks”: 51 pages, 14 figures, 10 references.

FRAMEWORK, API, CSRF, ELOQUENT, MARIADB, MYSQL, MVC, PHP, REST.

The object of the study is the process of developing an online store using the Laravel and Vue.js frameworks.

The aim of the study is to create a high-quality online store based on the Laravel and Vue.js frameworks, which will ensure efficient product and order management.

The methods of research are methods of collecting and analyzing software requirements, methods of modeling and designing software.

The qualification work analyzed the subject area, reviewed existing frameworks for developing web applications, and described tools for creating web applications. Based on this theoretical information, an online store project was developed using modern Laravel and Vue.js frameworks. The result of the qualification work is an online store with a client-server architecture, written using the MariaDB database, backend development in Laravel and frontend development in Vue.js.

ЗМІСТ

Завдання на кваліфікаційну роботу……..…..2
Реферат 4
Summary 5
Вступ..... 8
1 Аналіз вимог та об’єкту дослідження 10
1.1 Аналіз існуючих Back-end фреймворків..... 10
1.1.1 PHP Фреймворк Symfony 10
1.1.2 PHP Фреймворк CodeIgniter..... 12
1.2 Аналіз існуючих Front-end фреймворків 14
1.2.1 JavaScript фреймворк React..... 14
1.2.2 JavaScript фреймворк Angular..... 16
1.3 Обрані засоби розробки вебдодатка..... 19
1.3.1 Бізнес-логіка додатку за допомогою Laravel 20
1.3.2 Користувацький інтерфейс вебсайту за допомогою Vue.js..... 22
1.3.3 База даних з допомогою MariaDB..... 25
2 Проєктування вебдодатку 27
2.1 Технічне завдання 27
2.1.1 Функціональні вимоги..... 28
2.1.2 Нефункціональні вимоги..... 29
2.1.3 Технологічні вимоги..... 31
2.2 Технічні вимоги..... 32
2.3 Етапи розробки..... 34
2.3 Діаграма прецедентів..... 36
2.4 Проєктування схеми бази даних..... 37
3 Програмна реалізація..... 39
3.1 Розробка серверної частини вебдодатка..... 39
3.1.1 Створення проєкту та його ініціалізація 39

3.1.2 Ініціалізація бази даних 40
3.1.3 Розробка модуля оплати 41
3.2 Реалізація користувацького інтерфейсу вебзастосунку 45
Висновки 49
Перелік посилань 51

ВСТУП

Розробка інтернет-магазинів являється невід'ємною частиною сучасного бізнесу, оскільки онлайн-торгівля продовжує швидко зростати і займати все більшу частку ринку. Інтернет-магазини дозволяють підприємствам пропонувати свої товари та послуги широкій аудиторії, знижуючи витрати на оренду фізичних приміщень і персонал. Крім того, вони забезпечують зручність для клієнтів, дозволяючи здійснювати покупки в будь-який час і з будь-якого місця.

Для створення високоякісного інтернет-магазину необхідно застосовувати сучасні технології та інструменти, які забезпечують надійність, продуктивність і зручність використання системи. Вибір правильних фреймворків для фронтенду і бекенду є ключовим аспектом у процесі розробки. Одними з найпопулярніших фреймворків для цієї мети є Laravel і Vue.js. Laravel, як back-end фреймворк, відомий своєю елегантною синтаксичною структурою та широкими можливостями для розробки вебдодатків, тоді як Vue.js є потужним інструментом для створення інтерфейсів користувача.

Метою даної роботи є створення функціонального та зручного інтернет-магазину з використанням Laravel та Vue.js, що забезпечить ефективне управління товарами та замовленнями. У процесі розробки будуть використані різноманітні методи дослідження, включаючи збір та аналіз вимог, моделювання та проєктування програмного забезпечення.

Аналіз літератури та ресурсів дозволить ознайомитися з теоретичними аспектами та найкращими практиками розробки вебдодатків. Проєктування системи включає створення архітектури додатку та розробку бази даних. Практична розробка охоплює реалізацію функціональності як на стороні сервера, так і на стороні клієнта, а також інтеграцію компонентів. Оптимізація дозволить покращити продуктивність та масштабованість системи, а

порівняльний аналіз допоможе оцінити переваги та недоліки використаних технологій.

Таким чином, дана робота має на меті не лише створити інтернет-магазин, а й дослідити процес розробки, застосовуючи сучасні технології та підходи. Результати цього дослідження можуть бути корисними як для розробників, так і для підприємств, що планують впровадження або вдосконалення своїх онлайн-платформ для торгівлі.

1 АНАЛІЗ ВИМОГ ТА ОБ’ЄКТУ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючих Back-end фреймворків

Laravel є одним із найпопулярніших PHP-фреймворків завдяки своїй елегантній синтаксичній структурі, багатому набору функцій та широким можливостям для розробки вебдодатків. Проте існує кілька інших фреймворків, які пропонують схожі функціональні можливості та архітектуру.

1.1.1 PHP Фреймворк Symfony

Symfony є одним із найпотужніших і найгнучкіших фреймворків у світі PHP. Він широко використовується у розробці масштабних вебдодатків завдяки своїй багатофункціональності, високій продуктивності та модульній архітектурі. Symfony побудований на основі компонентної архітектури, що дозволяє використовувати окремі його компоненти незалежно від основного фреймворку. Ці компоненти можуть бути інтегровані у будь-які інші проєкти, що робить його дуже гнучким. Наприклад, багато популярних проєктів, таких як Drupal та Laravel, використовують компоненти Symfony.

Symfony підтримує архітектуру MVC (Model-View-Controller), що дозволяє чітко розділити бізнес-логіку, представлення та контролери. Це сприяє кращій організації коду та полегшує його підтримку та масштабування. Він також має потужну консоль для командного рядка, яка дозволяє виконувати різноманітні задачі, такі як генерація коду, управління базами даних та тестування. Консольні команди спрощують рутинні завдання та автоматизують багато процесів розробки.

Окрім цього, Symfony використовує шаблонізатор Twig, який дозволяє створювати гнучкі та потужні шаблони з високою продуктивністю. Twig

забезпечує безпечне та ефективне відображення даних у вигляді HTML. Крім того, Symfony інтегрований з Doctrine ORM, що забезпечує потужні можливості для роботи з базами даних. Doctrine ORM дозволяє автоматично генерувати SQL-запити на основі визначених моделей, що значно може спростити роботу з базами даних.

Сам по собі Symfony спроєктований таким чином, щоб забезпечити високу продуктивність та масштабованість. Це досягається завдяки оптимізації коду та ефективному використанню ресурсів. Однією з головних переваг Symfony є його гнучкість. Компонентна архітектура дозволяє розробникам використовувати лише ті частини фреймворку, які необхідні для конкретного проєкту, що робить його дуже добрим для великих та складних систем.

Symfony має велику та активну спільноту розробників, що забезпечує постійний розвиток фреймворку та наявність великої кількості ресурсів для навчання. Документація Symfony є однією з найкращих серед PHP-фреймворків, що робить його доступним для розробників з різним рівнем досвіду. У нього наявна довготривала підтримка (LTS) для своїх версій, що забезпечує надійність та безпеку у довгостроковій перспективі. Це особливо важливо для корпоративних проєктів, які потребують стабільності та підтримки на тривалий період.

Також Symfony має вбудовані механізми для забезпечення безпеки додатків, такі як захист від SQL-ін’єкцій, XSS та CSRF атак, що дозволяє розробникам зосередитися на логіці додатку, не турбуючись про основні аспекти безпеки. Symfony добре інтегрується з іншими популярними бібліотеками та інструментами, такими як PHPUnit для тестування, Monolog для ведення логів, і SwiftMailer для відправки електронної пошти, що забезпечує розробникам широкий спектр можливостей для розробки комплексних додатків.

Через свою гнучкість та багатофункціональність, Symfony може здаватися складним для новачків. Для освоєння цього фреймворку може

знадобитися більше часу, ніж для простіших альтернатив, він вимагає більше налаштувань у порівнянні з іншими фреймворками, такими як Laravel, що збільшує час на початкову настройку проєкту та вимагає більш глибокого розуміння внутрішньої роботи фреймворку. Висока продуктивність Symfony може бути корисною для великих проєктів, але для маленьких проєктів може стати перешкодою через складність та вимоги до ресурсів.

Symfony використовується у багатьох великих корпоративних проєктах, що демонструє його потужність та можливості. Наприклад, популярний контент-менеджмент-система Drupal 8 побудована на основі компонентів Symfony. Це дозволяє Drupal використовувати всю міць та гнучкість Symfony для створення складних та масштабованих систем управління контентом. Інший приклад – Laravel, який також використовує деякі компоненти Symfony для своєї базової архітектури, що забезпечує його високу продуктивність та гнучкість.

Symfony також використовується у таких великих проєктах, як BBC, TED, BlaBlaCar та Vogue. Ці компанії обрали Symfony через його стабільність, безпеку та масштабованість, що робить його ідеальним для розробки великих вебдодатків з високими вимогами до продуктивності та надійності.

1.1.2 PHP Фреймворк CodeIgniter

CodeIgniter є ще одним популярним PHP-фреймворком, який вирізняється своєю легкістю, швидкістю та простотою використання. Він широко використовується для розробки вебдодатків малого та середнього розміру завдяки своїй мінімалістичній структурі та високій продуктивності.

Однією з головних особливостей CodeIgniter є його невеликий розмір та швидкість. Основний пакет фреймворку займає всього кілька мегабайтів, що робить його дуже швидким у завантаженні та використанні. Це дозволяє розробникам швидко почати роботу над проєктом без необхідності

встановлення та налаштування складних залежностей. Також CodeIgniter підтримує архітектуру MVC (Model-View-Controller), яка дозволяє розділяти бізнес-логіку, представлення та контролери, що сприяє кращій організації коду та полегшує його підтримку та масштабування.

Важливою перевагою CodeIgniter є його продуктивність. Завдяки мінімалістичній структурі фреймворк працює дуже швидко, що робить його ідеальним вибором для проєктів, де швидкість виконання є критично важливою. Вбудовані бібліотеки для роботи з базами даних, управління сесіями, хешуванням, валідації форм та іншими функціями спрощують розробку та дозволяють зосередитися на логіці додатку.

CodeIgniter має велику та активну спільноту, що забезпечує наявність безлічі ресурсів для навчання та підтримки. Документація фреймворку є детальною та добре структурованою, що робить її доступною для розробників з різним рівнем досвіду. Спільнота активно розробляє та підтримує розширення та бібліотеки, що дозволяє легко інтегрувати додаткові функції у проєкт.

Ще однією перевагою CodeIgniter є його проста настройка та мінімальні вимоги до середовища виконання. Для початку роботи з фреймворком достатньо завантажити архів з офіційного сайту, розпакувати його у кореневу директорію вебсервера та почати розробку. Відсутність жорстких залежностей та необхідність у попередньому налаштуванні роблять CodeIgniter дуже зручним для швидкого старту проєкту.

Проте, незважаючи на всі переваги, CodeIgniter має й певні недоліки. Обмежена функціональність фреймворку може стати перешкодою для розробки великих та складних проєктів. На відміну від більш потужних фреймворків, таких як Symfony або Laravel, CodeIgniter не надає такої ж широкої підтримки для сучасних практик розробки та інтеграції з іншими інструментами. Хоча CodeIgniter підтримує архітектуру MVC, він не нав’язує суворих правил щодо структури проєкту, що може призвести до непослідовності у коді та ускладнити його підтримку у великих командах

розробників. Це може бути недоліком для проєктів, де важлива стандартизація та відповідність конвенціям.

CodeIgniter підходить для розробки невеликих та середніх вебдодатків, де швидкість розробки та продуктивність є пріоритетними. Наприклад, він ідеально підходить для створення блогів, новинних сайтів, невеликих інтернет-магазинів та корпоративних сайтів. Завдяки своїй простоті та легкості у використанні, CodeIgniter дозволяє швидко реалізовувати функціональність та випускати продукт на ринок у короткі терміни.

Популярними прикладами використання CodeIgniter є проєкти ExpressionEngine, Bonfire та PyroCMS. ExpressionEngine – це потужна CMS (система управління контентом), яка використовує CodeIgniter для забезпечення швидкої та надійної роботи. Bonfire є фреймворком на базі CodeIgniter, який надає додаткові інструменти та функції для швидкої розробки вебдодатків. PyroCMS – це ще одна CMS, побудована на CodeIgniter, яка дозволяє створювати гнучкі та масштабовані вебсайти.

1.2 Аналіз існуючих Front-end фреймворків

Vue.js, безсумнівно, займає визначне місце серед front-end фреймворків, пропонуючи простоту використання, гнучкість та широкі можливості для розробки вебдодатків. Проте, на ринку існує кілька інших фреймворків, які також відомі своєю ефективністю та популярністю у веброзробці. Давайте розглянемо деякі з них.

1.2.1 JavaScript фреймворк React

React, розроблений Facebook, є однією з найпопулярніших та впливових JavaScript бібліотек у світі сучасної веброзробки. Він відзначається своєю

декларативною моделлю розробки та компонентним підходом, які дозволяють розробникам створювати складні користувацькі інтерфейси швидко та ефективно.

Однією з ключових особливостей React є Virtual DOM – віртуальне представлення реального DOM. Це дозволяє React відслідковувати зміни в стані компонентів та ефективно оновлювати лише ті частини інтерфейсу, які потребують змін, що забезпечує високу швидкість та продуктивність додатків.

Компонентний підхід React робить код більш організованим та легко зрозумілим. Кожен компонент може містити в собі свій власний стан та логіку, що спрощує розробку та підтримку додатків. Крім того, React пропонує широкий набір інструментів та бібліотек для створення високоякісних інтерфейсів, включаючи Redux для управління станом та React Router для маршрутизації.

Однією з головних переваг React є його широке співтовариство та активна підтримка. Існує безліч відкритих ресурсів, документація є зрозумілою та докладною, що робить навчання та використання React приємним та ефективним процесом.

Хоча React має безліч переваг, він також має свої недоліки та виклики. Один з найбільш поширених викликів – це велика кількість вивчається та концепцій, особливо для початківців. Вивчення та розуміння принципів Virtual DOM, життєвого циклу компонентів та інших ключових концепцій може вимагати певного часу та зусиль.

Ще одним викликом може бути вибір серед широкого спектру додаткових бібліотек та інструментів, таких як управління станом, маршрутизація та стилізація. Вибір правильних інструментів для вашого проєкту може виявитися складним завданням, особливо для новачків.

React використовується у безлічі вебдодатків та сайтів з усього світу. Популярність React зростає з кожним днем, і це завдяки його гнучкості та ефективності. Один з найвідоміших прикладів використання React – це Facebook, який використовує його для побудови інтерфейсу своєї соціальної

мережі. React також використовується у таких відомих компаніях, як Instagram, Airbnb, Netflix та багатьох інших.

Багато проєктів вибирають React через його здатність до компонентної розробки, що спрощує управління складними додатками. Компоненти можна легко повторно використовувати та комбінувати, що дозволяє швидко створювати новий функціонал або редагувати існуючий. Це робить React ідеальним інструментом для командної розробки, де різні члени команди можуть працювати над окремими компонентами паралельно.

У світі мобільних додатків React також зайняв своє місце завдяки фреймворку React Native. React Native дозволяє розробникам створювати мобільні додатки для iOS та Android, використовуючи JavaScript та React. Це дозволяє розробникам повторно використовувати багато коду між веб та мобільними додатками, що зменшує час розробки та спрощує підтримку.

Важливо також відзначити активну спільноту та екосистему навколо React. Існує безліч відкритих бібліотек, компонентів та інструментів, які допомагають розробникам зробити свої додатки кращими та ефективнішими. Також існують різноманітні курси, блоги та конференції, які допомагають розробникам підтримувати свої навички та залишатися в курсі останніх тенденцій у світі React.

У підсумку, React – це потужний інструмент для розробки сучасних вебдодатків та мобільних застосунків. Його декларативний підхід, швидкість та ефективність роблять його вибором №1 для багатьох компаній та розробників з усього світу.

1.2.2 JavaScript фреймворк Angular

Angular, розроблений та підтримуваний Google, є одним із найпотужніших і всеосяжних фреймворків для веброзробки. Його основне

призначення – створення динамічних одно-сторінкових додатків (SPA) з багатим користувацьким інтерфейсом. Angular поєднує в собі широкий набір інструментів та бібліотек, що дозволяють розробникам будувати складні вебдодатки з високою продуктивністю та легкою підтримкою.

Angular базується на архітектурі компонентів, що дозволяє розробникам створювати повторно використовувані та ізольовані частини інтерфейсу. Кожен компонент може мати свій власний шаблон, стиль та логіку, що робить код більш організованим та легким у підтримці. Однією з ключових особливостей Angular є двостороння прив’язка даних (two-way data binding), яка автоматично синхронізує модель та представлення, що значно спрощує роботу з формами та іншими елементами інтерфейсу.

Angular використовує TypeScript, що є надбудовою над JavaScript, надаючи більш строгий синтаксис та типізацію. Це допомагає виявляти помилки на етапі розробки та покращує якість коду. TypeScript також робить код більш читабельним та зрозумілим, що полегшує командну роботу над проєктом.

Одна з ключових переваг Angular – це його модульна структура. Додатки Angular складаються з модулів, які можуть бути легко додані або видалені без порушення цілісності додатка. Це робить додатки Angular дуже масштабованими та зручними для великих проєктів. Angular CLI (Command Line Interface) надає розробникам потужний інструмент для автоматизації рутинних задач, таких як генерація компонентів, сервісів та модулів, а також для запуску тестів та зборки додатка.

Angular забезпечує високу продуктивність додатків завдяки використанню Ahead-of-Time (AOT) компіляції, яка перетворює Angular шаблони та компоненти у високоефективний JavaScript код до того, як він буде виконаний в браузері. Це зменшує час завантаження додатка та підвищує його продуктивність. Крім того, Angular підтримує Lazy Loading, що дозволяє завантажувати лише ті частини додатка, які необхідні у даний момент,

зменшуючи час завантаження та покращуючи користувацький досвід.

Незважаючи на численні переваги, Angular має деякі недоліки. Angular є великим та складним фреймворком, який вимагає значних зусиль для освоєння. Вивчення TypeScript, концепцій двосторонньої прив’язки даних, сервісів, ін’єкції залежностей та інших аспектів Angular може вимагати часу та зусиль, особливо для початківців.

Також важно помітити, що неправильне використання прив’язки даних або надмірне використання спостерігачів може призвести до проблем з продуктивністю, особливо в великих додатках. Що у висновку вимагає від розробників ретельного планування та оптимізації коду.

Завдяки своїй потужності та гнучкості, Angular є ідеальним вибором для великих корпоративних додатків, де важлива масштабованість та висока продуктивність. Компанії, які використовують Angular, відзначають його здатність забезпечити стабільну та надійну роботу додатків навіть при великому навантаженні.

Angular є одним з найпотужніших та найгнучкіших фреймворків для веброзробки. Його компонентна архітектура, підтримка TypeScript, висока продуктивність та масштабованість роблять його ідеальним вибором для створення складних вебдодатків. Незважаючи на те, що вивчення Angular може вимагати певних зусиль, його переваги значно перевищують недоліки, роблячи його одним з найпопулярніших інструментів серед веброзробників. Завдяки активній спільноті та постійним оновленням, Angular продовжує розвиватися та залишатися на передовій лінії сучасної веброзробки.

Angular використовується у багатьох великих компаніях та проєктах по всьому світу. Наприклад, такі компанії, як Google, Microsoft, Upwork та PayPal, використовують Angular для створення своїх вебдодатків. Один з найбільш відомих прикладів використання Angular – це Google AdWords, потужна платформа для онлайн-реклами. Angular також використовується у таких додатках, як Google Analytics та Google Cloud Console.

1.3 Обрані засоби розробки вебдодатка

Для розробки представленого вебдодатку прийнято рішення використовувати комбінацію Laravel, Vue.js та MariaDB. Це рішення ґрунтується на детальному аналізі потреб проєкту та можливостей, які надають ці інструменти. Зосередимося на виборі технологій, які забезпечать баланс між продуктивністю, гнучкістю та зручністю розробки.

Laravel був обраний за його потужні можливості та елегантну структуру. Цей фреймворк має чудову документацію і великий об'єм доступної інформації разом із різноманітним вибором користувацьких пакетів, що робить його надійним вибором для створення бекенду. З Laravel можна легко впроваджувати сучасні функції, мається можливість ручного управління базами даних через Eloquent ORM, та швидкої генерації коду за допомогою Artisan. Це дозволяє зосередитися на розробці бізнес-логіки та швидко втілювати заплановані задачі.

Для фронтенду обрано Vue.js через його простоту та потужність. Vue.js є надзвичайно гнучким і може бути легко інтегрований у проєкти будь-якого масштабу. Особливо цінний його компонентний підхід, який допомагає організовувати код та повторно використовувати його в різних частинах додатку. Крім того, Vue.js забезпечує високу продуктивність інтерфейсу користувача, що є критично важливим для забезпечення якісного досвіду для кінцевих користувачів.

Для зберігання даних обрано MariaDB. Ця система управління базами даних відома своєю надійністю та високою продуктивністю. Оскільки MariaDB є форком MySQL, вона підтримує всі функції, які зазвичай використовуються, але при цьому пропонує кращу продуктивність і нові функції. MariaDB легко інтегрується з Laravel, що спрощує процес розробки та забезпечує стабільне і швидке зберігання даних.

Ця комбінація технологій забезпечує всі необхідні інструменти для створення потужного та гнучкого вебдодатку. Використовуючи Laravel,

можна зосередитися на побудові надійного та масштабованого серверного коду, а за допомогою Vue.js – створювати динамічні та інтерактивні інтерфейси, які роблять додаток зручним та привабливим для користувачів. І наприкінці, завдяки MariaDB можна отримати надійне рішення для зберігання даних, яке гарантує високу продуктивність та безпеку.

Рішення використовувати Laravel, Vue.js та MariaDB дозволяє ефективно управляти всім циклом розробки додатку – від серверної логіки до фронтенду та бази даних. Це сприяє більшій узгодженості коду та зменшує час на розробку та підтримку проєкту. Тобто така комбінація забезпечить найкращий результат, дозволяючи швидко реагувати на зміни та виправлення, та легко масштабувати додаток у майбутньому.

1.3.1 Бізнес-логіка додатку за допомогою Laravel

Laravel – це один із найпопулярніших PHP-фреймворків, який був створений для спрощення розробки вебдодатків та покращення робочого процесу розробників. Він вперше був випущений у 2011 році Тейлором Отвеллом і швидко завоював популярність завдяки своїй простоті, елегантності та функціональності.

Laravel пропонує чистий та зрозумілий синтаксис, що дозволяє швидко орієнтуватися в коді та легко його читати. Архітектура MVC (Model-View-Controller), яка використовується в Laravel, допомагає структурувати код, розділяючи логіку додатку, його представлення та контролери, що спрощує підтримку та розвиток проєкту.

Однією з ключових особливостей Laravel є Virtual DOM – віртуальне представлення реального DOM. Virtual DOM дозволяє відслідковувати зміни в стані компонентів та ефективно оновлювати лише ті частини інтерфейсу, які потребують змін, що загалом забезпечує високу швидкість та продуктивність. Компонентний підхід робить код більш організованим та легко зрозумілим і

кожен компонент може містити в собі свій власний стан та логіку, що спрощує розробку та підтримку додатків.

Laravel пропонує широкий спектр вбудованих інструментів, таких як Artisan (командна консоль для автоматизації рутинних задач), Eloquent ORM (потужний об’єктно-реляційний маппер), Laravel Mix (інструмент для компіляції та мінімізації CSS і JavaScript файлів) та багато інших. Такі інструменти роблять фреймворк дуже зручним для розробки, оскільки багато завдань можна виконати швидко і без зайвих зусиль.

Ще однією важливою характеристикою Laravel є його масштабованість. Фреймворк підходить як для малих, так і для великих проєктів завдяки своїй модульній структурі та можливостям масштабування. Laravel також має вбудовані засоби для забезпечення безпеки додатків, наприклад захист від SQL-ін’єкцій, міжсайтового скриптингу (XSS) та підробки міжсайтових запитів (CSRF). Це дозволяє зосередитися на логіці додатку, не турбуючись про основні аспекти безпеки.

Один із важливих факторів, що робить Laravel популярним, це те що у нього існує безліч відкритих ресурсів, та є зрозуміла та докладна документація. Сама офіційна документація є дуже детальною і допомагає швидко освоїти фреймворк, фокусуючись на тільки важливих аспектах. Окрім того, Laravel активно підтримується і постійно розвивається.

Однак, чому обрано саме Laravel замість інших популярних PHP-фреймворків? Відповідь на це питання базується на кількох ключових причинах.

Простота використання. У порівнянні з іншими фреймворками, Laravel має більш простий та зрозумілий синтаксис, що дозволяє швидше почати з ним роботу. Крім того, Laravel пропонує багато готових рішень для поширених задач, таких як аутентифікація, валідація форм, обробка запитів, управління сесіями та хешування. Це дозволяє зосередитися на унікальних аспектах проєкту, замість того щоб витрачати багато часу на розробку базового функціоналу.

Швидкість розробки. Завдяки вбудованим інструментам та функціям (Eloquent ORM і Blade), Laravel дозволяє швидко створювати та налаштовувати вебдодатки. Крім того, Laravel має чудову підтримку для створення RESTful API завдяки вбудованим інструментам та пакетам, таким як Laravel Sanctum для створення та управління простими токенами автентифікації.

Великий функціонал та багато вбудованих інструментів. Серед них наявні вбудовані можливості для тестування та ведення логів. Також Laravel включає в себе Artisan – інтерфейс командного рядка, який автоматизує рутинні завдання, такі як створення міграцій, моделей, контролерів, черг та фонових задач, їх запуск та управління.

Активна спільнота та багата екосистема. Існує безліч відкритих бібліотек, компонентів та інструментів, які допомагають розробникам зробити свої додатки краще та ефективніше. Також існують різноманітні курси, блоги та конференції, які допомагають підтримувати навички та залишатися в курсі останніх тенденцій.

Архітектура Laravel та його модульна структура. Laravel підходить для проєктів різного масштабу, від малих стартапів до великих корпоративних систем. Laravel розроблений з урахуванням принципів сучасного програмування, таких як інверсія керування (IoC), залежності через DI-контейнер, та ін.

1.3.2 Користувацький інтерфейс вебсайту за допомогою Vue.js

Vue.js є прогресивним JavaScript-фреймворком, який дозволяє створювати інтуїтивно зрозумілі та високопродуктивні користувацькі інтерфейси. Його головною перевагою є гнучкість і простота інтеграції, що робить його ідеальним для використання як у невеликих проєктах, так і у великих одно-сторінкових додатках. Vue.js надає декларативний синтаксис,

який значно спрощує розробку та обслуговування коду.

Vue.js легко інтегрується з існуючими проєктами, що робить його привабливим вибором для тих, хто хоче поступово оновлювати свій старий код або додавати нові функціональні можливості. Двостороння прив’язка даних (two-way data binding) у Vue.js дозволяє значно спростити роботу з формами та складними структурами даних. Це забезпечує більш зручну та ефективну роботу з інтерфейсом користувача, особливо у випадках, коли дані повинні бути синхронізовані між моделлю та виглядом у режимі реального часу [3].

При виборі фреймворку для проєкту, зупинимося на Vue.js замість інших доступних варіантів з кількох причин. По-перше, Vue.js легкий у засвоєнні і має невеликий розмір, що робить його швидким у завантаженні та виконанні. Це особливо важливо для проєктів, де час завантаження і швидкодія мають вирішальне значення. У порівнянні з Angular, який є повноцінним фреймворком і має більш складну архітектуру, Vue.js надає більшу гнучкість та не вимагає використання TypeScript, що може призупинити розробку інтерфейсу вебсайту.

React, з іншого боку, хоч і є потужною бібліотекою для створення інтерфейсів, має деякі недоліки в порівнянні з Vue.js. Одним з таких недоліків є складність в налаштуванні і використанні. React вимагає більшого обсягу коду для реалізації тих самих функцій, які у Vue.js можуть бути реалізовані з меншою кількістю коду. Крім того, Vue.js надає більш структурований підхід до роботи з компонентами і шаблонами, що дозволяє зменшити кількість помилок і полегшує процес налагодження.

Ще однією важливою перевагою Vue.js є його продуктивність. Завдяки оптимізованому рендеру та ефективній роботі з компонентами, Vue.js забезпечує високу швидкість роботи додатків навіть при великій кількості взаємодій з користувачем. Це дозволяє створювати більш швидкі та чутливі додатки, що підвищує загальний досвід користувачів.

Активна спільнота розробників та велика кількість плагінів і розширень

також приваблюють до Vue.js. Велика кількість ресурсів, таких як документація, навчальні матеріали, блоги та форуми, дозволяють швидко знайти відповіді на будь-які питання та вирішити проблеми, що виникають під час розробки. Крім того, активна підтримка та регулярні оновлення забезпечують безпеку та стабільність фреймворку [4].

Вибір Vue.js був обумовлений також його можливістю плавної інтеграції з іншими інструментами та бібліотеками, що допомагає використовувати Vue.js у поєднанні з іншими технологіями, значно розширюючи його можливості та підвищуючи ефективність роботи. Наприклад, Vue.js може легко інтегруватися з Vuex для управління станом додатку або з Vue Router для організації маршрутизації [5].

Таким чином, Vue.js пропонує оптимальний баланс між простотою використання, продуктивністю та гнучкістю, що робить його ідеальним вибором для розробки сучасних вебдодатків. Його легкість у засвоєнні, потужні функції та активна спільнота роблять його привабливим інструментом. У порівнянні з іншими варіантами, наприклад, React та Angular, Vue.js надає більш зручний та ефективний спосіб створення інтерактивних користувацьких інтерфейсів, що дозволяє зосередитися на розробці функціоналу та підвищенні якості кінцевого продукту.

Одним із ключових факторів, що спонукали мене обрати Vue.js, була його орієнтованість на компонентний підхід. Компоненти у Vue.js є самодостатніми модулями, які можуть бути легко повторно використані в різних частинах додатку. Це сприяє кращій організації коду і дозволяє уникнути дублювання логіки, що, у свою чергу, спрощує підтримку та розширення проєкту. Компонентна структура Vue.js надає можливість розробникам зосередитися на створенні ізольованих частин інтерфейсу, що значно покращує масштабованість додатку.

Також важливо відзначити, що Vue.js має прекрасну інтеграцію з сучасними інструментами розробки, такими як Webpack та Babel, що дозволяє легко налаштовувати проєкт відповідно до потреб команди. Це забезпечує

високу продуктивність і дозволяє використовувати новітні можливості JavaScript, не турбуючись про сумісність з різними браузерами. Крім того, Vue.js підтримує серверний рендеринг, що робить його ідеальним для SEO-оптимізованих додатків.

Вибір Vue.js також був обумовлений його зручним та інтуїтивно зрозумілим API. На відміну від деяких інших фреймворків, API Vue.js є простим та легко засвоюваним, що дозволяє швидко розпочати розробку навіть новачкам. Зрозуміла і логічна структура документації також сприяє швидкому навчанні та ефективному використанню фреймворку.

Крім того, Vue.js має відмінну підтримку для створення анімацій та транзицій, що робить додатки більш привабливими та інтерактивними для користувачів.

Загалом, обравши Vue.js для розробки, отримуємо ефективний, продуктивний та гнучкий інструмент, який дозволяє створювати сучасні, інтуїтивно зрозумілі та високо-інтерактивні користувацькі інтерфейси. Його простота, гнучкість, продуктивність та активна підтримка спільноти роблять Vue.js оптимальним вибором для розробників, які прагнуть створювати високоякісні вебдодатки з мінімальними зусиллями та максимальною ефективністю.

1.3.3 База даних з допомогою MariaDB

MariaDB є потужною та надійною системою управління базами даних, обраною для цього проєкту через її численні переваги.

Висока продуктивність MariaDB забезпечується численними оптимізаціями, які дозволяють обробляти великі обсяги даних швидко та ефективно. Такі можливості критичні для проєктів, які потребують оперативного доступу до даних та їх обробки в режимі реального часу.

Надійність та стійкість до збоїв – ще одна ключова перевага MariaDB.

Вона підтримує механізми резервного копіювання та відновлення даних, що знижує ризик втрати інформації. Вбудовані механізми реплікації забезпечують високу доступність та відмово-стійкість системи.

Безпека даних у MariaDB також тримає планку на високому рівні. Система пропонує інструменти для шифрування даних, управління доступом та автентифікацію користувачів, що захищає конфіденційну інформацію.

Загалом, використання MariaDB у цьому проєкті забезпечує високу продуктивність, надійність, безпеку та масштабованість, що є ключовими факторами для успішної реалізації сучасних вебдодатків.

2 ПРОЄКТУВАННЯ ВЕБДОДАТКУ

Для початку проєктування вебдодатку, треба сформулювати його технічне завдання, в залежності від обраної сфери діяльності та підходу до визначених умов, що допоможе мати на меті конкретні задачі та взаємодію між ними.

Опісля цього потрібно вирішити технічні вимоги, наприклад спосіб автентифікації, обробку помилок, взаємодію з базою даних, розробку адміністративної панелі, дотримання безпеки та інше. І останнім пунктом можна буде перейти до самих етапів розробки, що включають до себе наступне:

– планування та аналіз вимог;
– проєктування системи;
– налаштування середовища розробки;
– розробка back-end частини;
– розробка front-end частини;
– тестування.

Окрім цього буде наведена діаграма прецедентів та спроєктована схема бази даних.

2.1 Технічне завдання

Метою проєкту являється розробка інтернет-магазину з продажу товарів із забезпеченням зручного користувацького інтерфейсу та ефективної панелі адміністратора.

2.1.1 Функціональні вимоги

З технічної точки зору, використовується Laravel для бекенду задля забезпечення надійного та масштабованого управління бізнес-логікою та обробки запитів користувачів [1]. На фронтенді – Vue.js, який дозволяє створювати динамічні, інтерактивні інтерфейси, що легко адаптуються до змін і забезпечують швидкий відгук системи [6].

Однією з фундаментальних вимог, являється забезпечення можливості реєстрації та аутентифікації користувачів. Це дозволяє користувачам створювати облікові записи у системі для можливості вебзастосунка взаємодії з ними. Вони можуть здійснювати покупки, переглядати історію замовлень та управління своїм профілем. Для підвищення безпеки будуть наявні підтвердження вхідних даних для уникнення фальшивих реєстрацій.

Центральним елементом інтернет-магазину являється каталог товарів, що дозволяє користувачам переглядати всі доступні продукти. Для зручності навігації, товари будуть призначені відповідними категоріями та підкатегоріями, що будуть реалізовані за допомогою додавання тегів до товарів. Користувачі матимуть змогу застосовувати фільтрацію для сортування товарів за різними параметрами:

– ціна (у обраних користувачем межах);
– бренд (обрання на вибір);
– рейтинг (загальна оцінка продукту користувачами).

Додатково буде реалізовано пошук, що допоможе швидко знаходити потрібні товари за назвою, тегами та ключовими словами.

Для перегляду продукту, буде реалізована картка товару. Вона надає детальну інформацію про товар, його характеристики, опції для вибору інших кольорів та перелік зображень. Це дозволяє користувачу переглядати усе необхідне для вирішення замовлення самого продукту.

Необхідна функція для управління товарами являється кошик. Завдяки ньому користувач може вільно додавати, видаляти, та редагувати кількість

товарів. Також кошик має обчислену суму окремо продуктів у кількості обраних штук, та суму загалом усього замовлення, що дозволяє мати уяву про плановані витрати коштів та обчислення, якщо наявно, обмеженого бюджету. Окрім цього з меню кошика можна зручно перейти до вікна формування замовлення натисканням однієї кнопки, після чого треба підтвердити покупку маючи усі деталі замовлення. Сам кошик виступає випадаючим меню і не заважає клієнту виконувати покупки та переглядати наявні товари.

У адміністрації сайту та товарів нам допоможе адміністративна панель. Для початку, адмін-панель являється відокремленою частиною додатку від основної клієнтської частини, де відбувається управління товарами та слідкування за користувачами. Адміністраторська панель дозволяє виділеним довіреним лицям – адмінам, та головному адміну – супер-адміну, додавати та видаляти товари, оновлювати їх опис, зображення, теги, та ціну. Таким чином забезпечується ефективне управління магазином та підтримка високого рівня обслуговування клієнтів.

Для того щоб користувачі розуміли що виконується з технічної сторони, та у деяких випадках не тільки там, використовується система сповіщень. З її допомогою клієнти розуміють що вони додали товар до кошику, зробили помилку у оформленні замовлення, або отримали сповіщення про нещодавні акції і цікаві новинки. Це допомагає підтримувати зв’язок з користувачами та забезпечувати їх проінформованими про стан замовлень.

2.1.2 Нефункціональні вимоги

Продуктивність інтернет-магазину є критично важливим аспектом, оскільки час, який витрачається на завантаження сторінок безпосередньо впливає на користувацький досвід. Сторінки повинні завантажуватися не більше ніж за 2 секунди при стандартному навантаженні, що включає перегляд каталогу товарів, карток товарів, кошика та процесу оформлення замовлення.

Для досягнення цієї мети буде використовуватися оптимізація коду, хешування даних, мінімізація запитів до бази даних та застосування сучасних технологій, таких як CDN (Content Delivery Network) для зменшення затримок у мережі. Ефективно розроблюючи бекенд на Laravel і швидкий фронтенд на Vue.js також сприяє забезпеченню високої продуктивності системи.

Безпека є однією з найважливіших нефункціональних вимог, оскільки інтернет-магазин буде обробляти особисту інформацію користувачів та платіжні дані. Система повинна бути захищена від основних типів атак, таких як SQL-ін’єкції, XSS (міжсайтові скрипти), CSRF (підробка міжсайтових запитів) та інші. Це досягнуто за допомогою використання вбудованих механізмів захисту Laravel, регулярних перевірок безпеки, оновлення системи до актуальних версій та застосування SSL-сертифікатів для шифрування даних, переданих між клієнтами та сервером.

Інтернет-магазин повинен мати можливість розширення функціоналу та підтримувати велику кількість користувачів та товарів. Це включає в себе додавання нових функцій без значних змін у базовій архітектурі системи, а також здатність обробляти високі навантаження під час пікових періодів, таких як святкові розпродажі. Масштабованість забезпечена завдяки використанню модульної архітектури Laravel та Vue.js, а також застосуванню технік горизонтального масштабування бази даних MariaDB та використанню контейнеризації для зручного розгортання та управління вебдодатком.

Інтерфейс користувачів та адміністратора повинен бути інтуїтивно зрозумілим і легким у використанні. Це означає, що користувачі повинні легко знаходити потрібні товари, додавати їх до кошика, оформлювати замовлення та отримувати інформацію про статус своїх покупок. Для адміністраторів інтерфейс забезпечує зручне управління товарами, замовленнями та користувачами. Юзабіліті досягнуто за рахунок застосування принципів UX-дизайну та ручного тестування та аналізу для постійного покращення інтерфейсу.

2.1.3 Технологічні вимоги

Для побудови серверної частини проєкту використовується фреймворк Laravel. Laravel – це потужним PHP-фреймворк, який забезпечує зручну структуру та інструменти для створення сучасних вебдодатків. Завдяки своїй модульній архітектурі, Laravel дозволяє ефективно організовувати бізнес-логіку, забезпечувати безпечну взаємодію з базою даних та легко реалізовувати складні функціональні можливості. Його вбудовані можливості, такі як ORM Eloquent, система маршрутизації, черги, події та інші, сприяють швидкій та якісній розробці надійних вебдодатків [8].

Клієнтська частина проєкту реалізована за допомогою фреймворку Vue.js. Vue.js дозволяє створювати динамічні та інтуїтивно зрозумілі інтерфейси користувача. Завдяки реактивному підходу, Vue.js забезпечує високу продуктивність та зручність у роботі з компонентами, що дозволяє створювати масштабовані одно-сторінкові додатки (SPA). Його гнучкість і простота інтеграції з іншими бібліотеками та існуючими проєктами роблять Vue.js ідеальним вибором для реалізації динамічного та швидкого інтерфейсу користувача.

Для зберігання даних про користувачів, товари, замовлення та інші сутності використовується база даних MariaDB. MariaDB є надійною і продуктивною реляційною базою даних, яка забезпечує високу швидкість виконання запитів та зручність у роботі з великими обсягами даних. Вона сумісна з MySQL, але має додаткові покращення в продуктивності та безпеці. Використання MariaDB дозволяє забезпечувати стабільне та ефективне зберігання і обробку даних, необхідних для функціонування інтернет-магазину [7].

Для управління версіями коду та співпраці між розробниками, використовується система контролю версій Git. Git дозволяє відстежувати всі зміни в коді, повертатися до попередніх версій, створювати різні гілки для паралельної розробки та легко інтегрувати зміни від різних розробників. Це

сприяє організованій та ефективній командній роботі, дозволяє уникати конфліктів у коді та забезпечує високу якість кінцевого продукту. Використання Git також спрощує процес розгортання додатку та подальшої його підтримки.

2.2 Технічні вимоги

У пункті з технічними вимогами будуть описані основні технічні вимоги до проєкту «Розробка інтернет-магазину із застосуванням фреймворків Laravel та Vue.js». Це включає в себе вимоги до архітектури системи, аутентифікації та авторизації користувачів, управління продуктами та замовленнями, адміністративної панелі, front-end функціоналу, оповіщень та сповіщень, оптимізації та продуктивності, безпеки, а також тестування та розгортання.

Архітектура.

Для back-end частини нам знадобиться створення масштабованої, стабільної та безпечної серверної архітектури. І Laravel, як потужний PHP-фреймворк, забезпечує широкий набір інструментів для розробки бекенду, включаючи автентифікацію, маршрутизацію, роботу з базами даних та інші.

Для front-end частини нам потрібно мати можливість створення динамічного та інтуїтивно зрозумілого користувацького інтерфейсу. Vue.js, якраз, як прогресивний JavaScript-фреймворк, надає ефективні інструменти для розробки високоякісних інтерфейсів для створення реактивних компонентів та забезпечення швидкої відповіді на користувацькі дії.

Щодо використання бази даних, наші потреби задовільнить MariaDB. Вона передбачає створення надійного та ефективного механізму зберігання та управління даними, і як відкрита реляційна база даних, забезпечує швидку та надійну роботу, а також підтримку транзакцій та масштабування.

Автентифікація та авторизація.

Користувачі повинні мати можливість вільно створювати, входити та

виходити зі своїх облікових записів.

Повинна бути наявна роздільність звичайних користувачів та специфікованих. Тому буде присутня підтримка ролей для різного рівня доступу до певних функцій.

Для адміністраторів необхідні захищені маршрути та окрема панель керування.

Управління продуктами.

Адміністратори можуть додавати, видаляти, та редагувати продукти.

Адміністратори можуть змінять категорію продуктів та управляти їх тегами.

Адміністратори можуть управляти зображеннями товарів, додавати нові, видаляти зайві, та оновляти старі.

Кошик та замовлення.

Користувачі повинні мати здібність додавання продуктів до кошика.

Замовлення повинні бути мати можливість оформленими користувачами вебдодатка.

Оформлення замовлень повинно бути можливим проходити через онлайн-оплату.

Замовлення повинні мати статуси замовлень (чекає на оплату, оплата успішна, відміна оплати), які динамічно змінюються при оновленні статусу замовлення.

Адміністративна панель.

Доступ до панелі має бути лише у адміністраторів.

Управління адміністраторами повинно здійснюватися лише супер-адміном.

Адміністративна панель має вміщувати у собі можливість переглядати фінансові звіти загалом, та за певні проміжки часу (чи то є конкретна дата, чи то проміжок від-до).

Оптимізація та продуктивність.

Вебзастосунок повинен мати у собі хешування та зменшене

навантаження на сервер задля оптимізації роботи.

Використання механізмів для зменшення часу завантаження сторінок (lazy loading, мінімізація у коді CSS та JavaScript).

Безпека.

Захист від SQL-ін’єкцій, XSS-атак та CSRF-атак.

Шифрування паролів користувачів.

2.3 Етапи розробки

У цьому розділі будуть описані всі основні етапи проєкту від початкового планування та аналізу вимог, через проєктування та розробку системи, до тестування, впровадження та подальшого обслуговування. Кожен етап охоплює конкретні завдання і процеси, які допоможуть досягти поставлених цілей, забезпечити високу якість продукту та задоволення користувачів.

Планування та аналіз вимог:

– визначення цілей проєкту: окреслення основних цілей, функціоналу та характеристик інтернет-магазину;
– аналіз вимог: збір та документування технічних і користувацьких вимог, включаючи функціональні та нефункціональні вимоги;
– оцінка ресурсів: визначення необхідних ресурсів, таких як команда розробників, час, бюджет, програмне забезпечення та обладнання.

Проєктування системи:

– архітектурне проєктування: розробка загальної архітектури системи, включаючи серверну та клієнтську частини;
– проєктування бази даних: створення схеми бази даних MariaDB, визначення основних таблиць, зв’язків та індексів;
– проєктування інтерфейсу: створення макетів та прототипів користувацького інтерфейсу, визначення основних сторінок та

взаємодії між ними.

Налаштування середовища розробки:

– налаштування серверного середовища: встановлення та конфігурація сервера для розробки, включаючи налаштування PHP, вебсервера та MariaDB;
– налаштування клієнтського середовища: встановлення та конфігурація необхідних інструментів для роботи з Vue.js;
– вибір та налаштування системи контролю версій: використання Git для управління версіями коду.

Розробка back-end частини:

– створення структури проєкту: ініціалізація проєкту на Laravel, налаштування основних конфігурацій;
– розробка моделей та міграцій: створення моделей для взаємодії з базою даних, написання міграцій для створення таблиць;
– реалізація бізнес-логіки: створення контролерів та сервісів для обробки запитів, реалізація основних функцій (аутентифікація, управління продуктами, оформлення замовлень);
– розробка API: створення RESTful API для взаємодії з фронтенд-частиною.

Розробка фронтенд-частини:

– створення структури проєкту: ініціалізація проєкту на Vue.js, налаштування основних конфігурацій;
– розробка компонентів: створення багаторазових компонентів для інтерфейсу користувача (навігаційні панелі, картки продуктів, форми тощо);
– інтеграція з бекендом: взаємодія з API для отримання та відправки даних, реалізація логіки управління станом додатка;
– реалізація маршрутизації: налаштування Vue Router для організації навігації між сторінками.

2.3 Діаграма прецедентів

Діаграма прецедентів є важливим інструментом у процесі моделювання вимог до програмного забезпечення. На рисунку 2.1 зображено діаграму прецедентів.

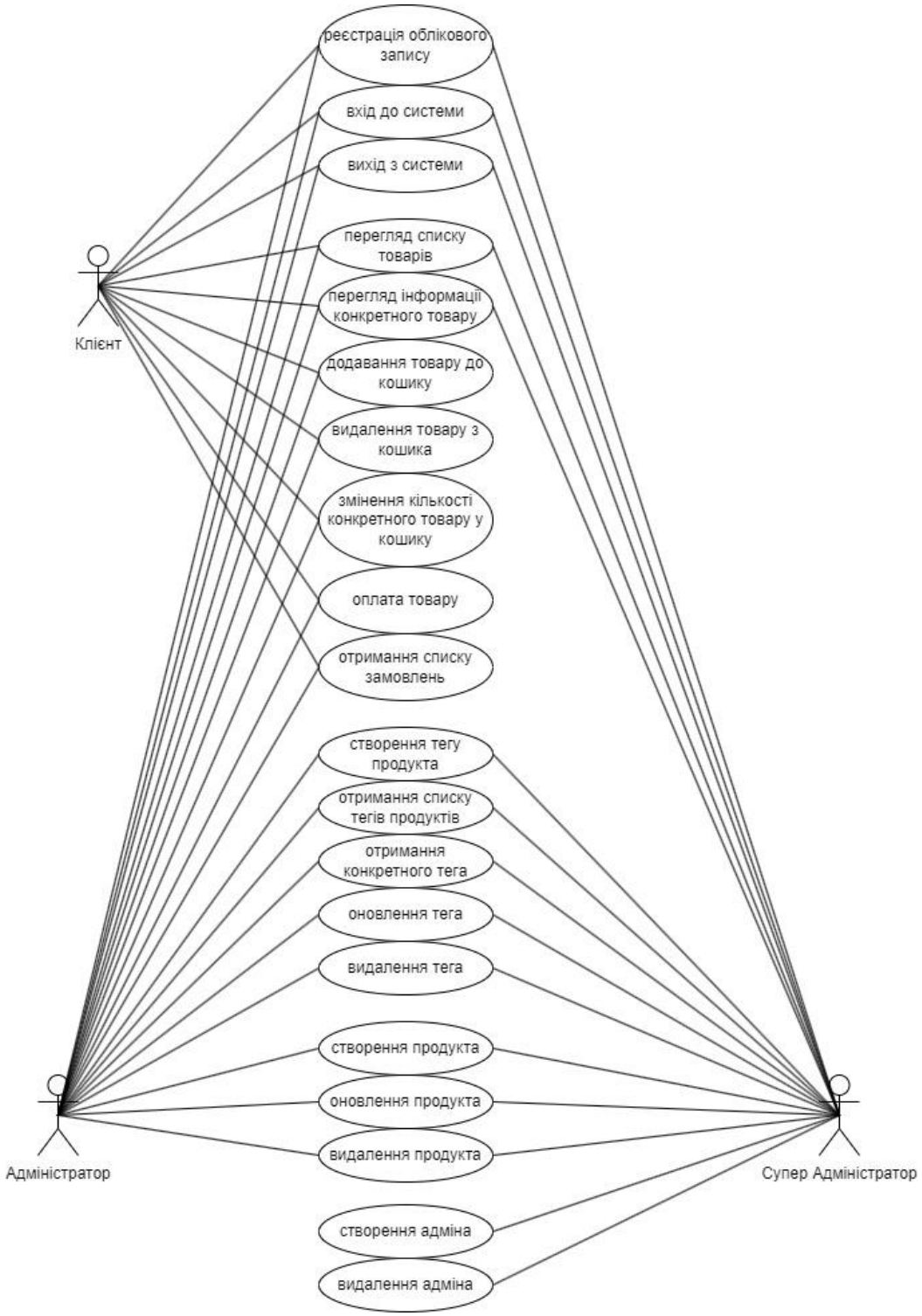


Рисунок 2.1 – Діаграма прецедентів

Вона дозволяє графічно зобразити взаємодію користувачів (акторів) з системою та ілюструє функціональні можливості, які система повинна забезпечувати. Використання діаграми прецедентів допомагає зрозуміти, які саме дії можуть виконувати різні категорії користувачів, що сприяє більш ефективному аналізу та управлінню вимогами до системи.

Діаграма прецедентів складається з двох основних елементів: акторів та прецедентів. Актори – це користувачі або зовнішні системи, які взаємодіють із системою, наприклад, клієнти, адміністратори або платіжні системи. Прецеденти (сценарії використання) описують функціональні можливості системи, тобто дії, які система виконує у відповідь на запити акторів, такі як реєстрація користувача або оформлення замовлення. Ці елементи разом створюють ясну картину взаємодії між користувачами і системою, допомагаючи чітко визначити, які функції повинна виконувати система.

2.4 Проєктування схеми бази даних

На рисунку 2.2 позначена схема бази даних, вона складається з 5 основних сутностей:

- users – сутність яка налічує у собі користувачів системи (вона має унікальні поля номеру телефона та адреси поштової скриньки для кожного користувача, поле збереженого паролю у зашифрованому вигляді та поле ролі, яка розділяє користувачів на 2 частини вебзастосунку: клієнтська та адміністративна);
- products – сутність яка налічує у собі список товарів інтернет-магазину (вона містить поля найменування товару, ціни визначеної магазином, ідентифікатора товарної позиції, опису товару та кількості товарів у наявності);
- tags – сутність зі списком тегів для товарів;
- carts – сутність, яка представляє собою кошик товарів для кожного

користувача (у неї знаходяться поля ідентифікатора користувача, ідентифікатора продукту та кількості товарів);

– checkouts – сутність що являється списком проведених транзакцій (поля налічують у собі ідентифікатор користувача, суму оформленого замовлення та статус оформлення).

Та з 3 суміжних сутностей:

– products_tags – сутність для зв’язку між сутностями products та tags;

– products_images – сутність для зв’язку між сутністю products та зображеннями товарів;

– carts_checkouts – сутність для зв’язку між сутностями carts та checkouts.

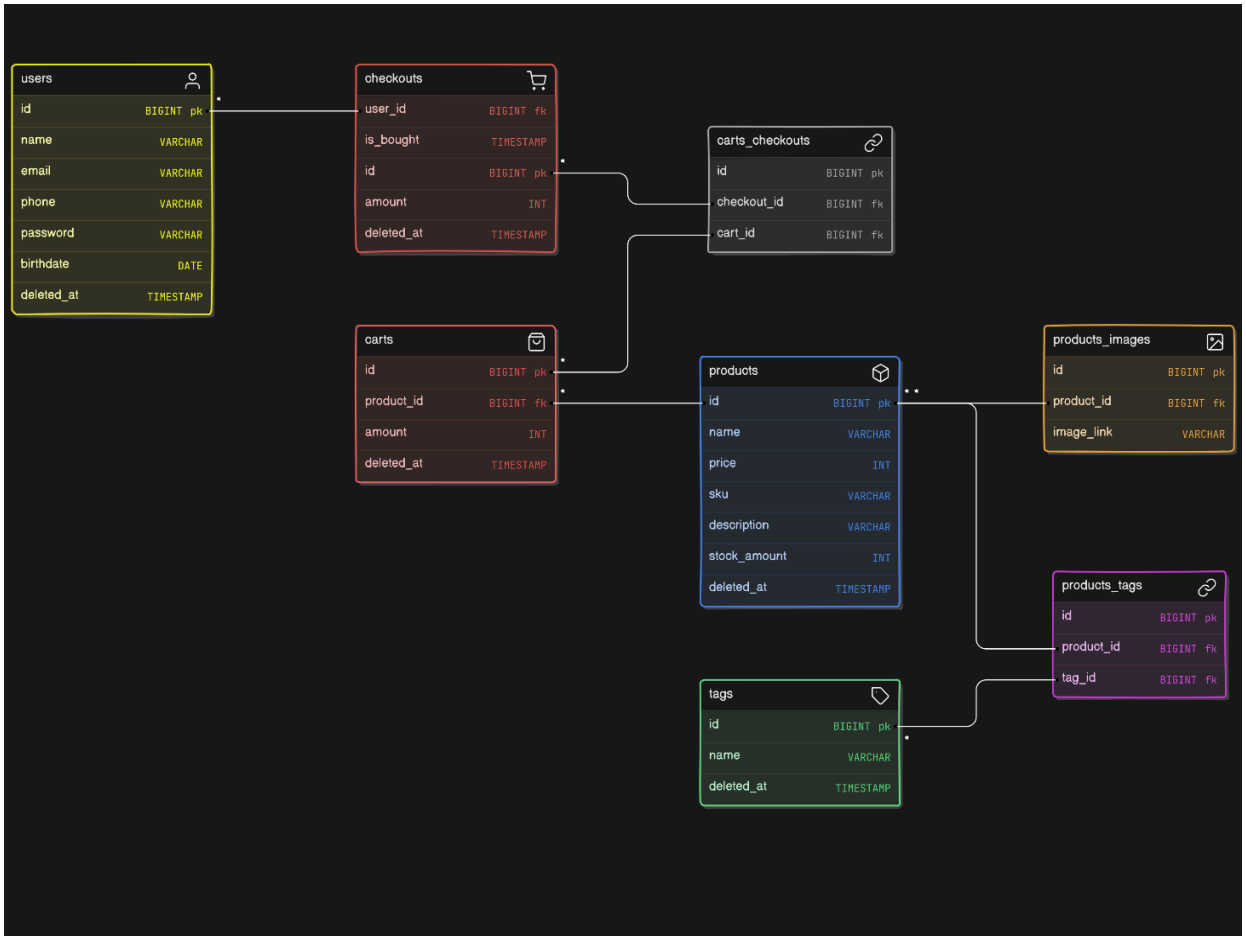


Рисунок 2.2 – ER-діаграма бази даних

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка серверної частини вебдодатка

Перший етап розробки серверної частини буде спрямований на створення API за допомогою платформи Laravel, забезпечуючи високу якість, надійність та ефективність системи.

Метою цього етапу є створення API, що відповідає стандартам REST, інтегрованого з платформою Laravel, та взаємодіє з клієнтським додатком за допомогою формату даних JSON.

3.1.1 Створення проєкту та його ініціалізація

Для початку проєкту Laravel використовувалася команда laravel new, що автоматично створила необхідні файли та структуру каталогів проєкту. Основним результатом цієї команди став файл composer.json, який містить інформацію про проєкт та його залежності, такі як пакети, необхідні для роботи додатка.

Для підтримки автоматичного перезапуску сервера під час розробки був встановлений пакет laravel/laravel-mix, що дозволяє використовувати nodemon. Встановлення зовнішніх пакетів, необхідних для роботи додатка, таких як laravel/ui, laravel/passport, laravel/sanctum та інших, було виконано через Composer.

У файлі config/app.php було налаштовано сервіс-провайдери та аліаси для зручного доступу до різних компонентів Laravel. Також були налаштовані файли середовища .env для різних середовищ (development, staging, production), де визначалися налаштування бази даних, хешування, логування та інші параметри.

Команди Artisan, такі як php artisan serve, були налаштовані для запуску сервера у режимі розробки та продакшену. Був створений та налаштований екземпляр класу застосунку з фреймворку Laravel, а також був оброблений GET-запит за маршрутом за замовчуванням [2].

3.1.2 Ініціалізація бази даних

На основі раніше спроєктованої ER-діаграми, завдяки MariaDB та стандартній функції міграцій у Laravel, було створено схему базу даних.

Базу даних було запущено за допомогою сервісу міграцій Laravel. Laravel надає можливість автоматичного створення бази даних у режимі розробки та тестування. Якщо будь-яке розширення, яке вимагає бази даних, включено у проєкт, і не налаштовано зовнішній datasource, Laravel автоматично запустить відповідну службу міграцій і підключить додаток до цієї служби.

У створеному проєкті була створена директорія migrations у директорії database. Там розміщуються файли міграцій, які містять інструкції для створення та зміни структури бази даних. Кожен файл міграції містить певні кроки, щоб забезпечити стабільність бази даних.

На рисунку 3.1 представлена міграція для створення таблиці у базу даних. Як можна побачити, у міграції наявні два метода, up() та down().

Метод up() виконується при запускі міграції і виконує вказані зміни, коли як метод down() виконується тільки при відкаті міграції. Така функція дуже зручна задля легкого повертання до змін при недогляді помилок, однак треба правильно робити відкат міграції задля уникнення ще більших проблем.

```
<?php

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

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

return new class extends Migration
{
    public function up(): void
    {
        Schema::create("users", static function (Blueprint $table) {
            $table->id();
            $table->string("name");
            $table->string("surname");
            $table->string("phone")->unique();
            $table->integer("role_id");
            $table->string("email")->unique();
            $table->string("password");
            $table->string("birthdate");
            $table->rememberToken();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists("users");
    }
};
```

Рисунок 3.1 – Міграція створення таблиці користувачів

3.1.3 Розробка модуля оплати

Модуль оплати було розроблено для обробки замовлень та взаємодії з платіжною системою. Основний функціонал цього модуля полягає в обробці кошика користувача, створенні запису про замовлення, підготовці даних для платіжної системи та управлінні статусами платежів.

Спочатку відбувається аутентифікації клієнта за допомогою auth("sanctum")->user(), далі отримуються всі товари у кошику, створюється запис оформлення замовлення до бази даних та перехід у метод formLink() сервісу monobankService (див. рис. 3.2, 3.3).

```
public function proceedCheckout(): JsonResponse
 {
    $user = auth("sanctum")->user();
    $carts = $this->cartRepository->findAllByUserWithProducts($user->id);
    if (empty($carts->toArray())) {
       return $this->apiResponse([], 404);
    }
    $sum = 0;
    foreach ($carts as $cart) {
       $sum += $cart->amount * $cart->product->price;
    }
    $checkout = Checkout::query()->create([
       "user_id" => $user->id,
       "sum" => $sum,
       "is_bought" => false
    ]);

    foreach ($carts as $cart) {
       CartsCheckout::query()->create([
          "checkout_id" => $checkout->id,
          "cart_id" => $cart->id
       ]);
       $cart->delete();
    }

    $newTimezone = new DateTimeZone("Europe/Kyiv");
    $datetime = new DateTime("now + " . 5 . "minutes", $newTimezone);
    UpdatePaymentStatus::dispatch($checkout)->delay($datetime);

    return $this->apiResponse($this->monobankService->formLink([
       "amount" => $checkout->sum * 100,
       "ccy" => 980,
       "merchantPaymInfo" => [
          "reference" => (string)$checkout->id,
          "destination" => "ID замовлення " . $checkout->id . ": купівля продукту компанії
JoMama"
       ],
       "redirectUrl" => config("monobank.redirect-url") . $checkout->id,
       "validity" => 300,
       "paymentType" => "debit"
    ]));
 }
```

Рисунок 3.2 – Метод обробки замовлення

```
public function formLink(array $request): array
{
    $response = Http::withHeaders([
        'X-Token' => config("monobank.token")
    ])->post(config("monobank.invoice"), $request);

    Cache::put($request["merchantPaymInfo"]["reference"], $response["invoiceId"], now()->addMinutes(60));

    return ["payment_url" => $response["pageUrl"]];
}
```

Рисунок 3.3 – Сервіс оплати Monobank

Для реалізації модуля оплати використовувавлось відкрите тестове api мобільного банку Monobank. Monobank Api дозволяє створити та відправити запит із заповненою формою платежу до їхньої системи, та отримати посилання для проведення оплати, яке потім отримує користувач [9]. Для цього потрібно мати посилання для створення інвойсу та токен завдяки якому буде проводитись оплата (див. рис. 3.4).

```
MONOBANK_INVOICE=https://api.monobank.ua/api/merchant/invoice/create
MONOBANK_STATUS=https://api.monobank.ua/api/merchant/invoice/status?invoiceId=
MONOBANK_TOKEN=uqa2adzdeGcco9uVhvmU8AC38PNkLu5y9bfDJxoSoqdU
```

Рисунок 3.4 – Config файл з переліком полів для Monobank Api

Окрім оброблення оплати замовлення, була додана реалізація оновлення статусу оформлених замовлень. Для цього спочатку треба мати посилання на отримання статусу оплати з Monobank Api (див. рис. 3.4), вже потім оновляти статус замовлення відповідно до отриманого у api банку [9]. Задля дотримання більшої ефективності, було створено фонову задачу яка запускається кожну хвилину, відправляє запрос до api, і в залежності від результату оновлює або залишає статус замовлення (див. рис. 3.5 – 3.7) [10].

```
class Kernel extends ConsoleKernel
{
    protected function schedule(Schedule $schedule): void
    {
        $schedule->job(UpdatePaymentStatuses::class)->everyMinute();
    }

    protected function commands(): void
    {
        $this->load(__DIR__ . "/Commands");

        require base_path("routes/console.php");
    }
}
```

Рисунок 3.5 – Диспетчер задач

```
class UpdatePaymentStatuses implements ShouldQueue
{
    use Dispatchable;
    use InteractsWithQueue;
    use Queueable;
    use SerializesModels;
    use ApiResponseTrait;

    public function handle(
        CheckoutRepositoryInterface $checkoutRepository,
        CheckoutService             $checkoutService
    ): void
    {
        $checkouts = $checkoutRepository->findAllNotBought();
        foreach ($checkouts as $checkout) {
            $checkoutService->updatePaymentStatus($checkout, false);
        }
    }
}
```

Рисунок 3.6 – Задача на оновлення статусу замовлення

```
class CheckoutService
{
    protected CheckoutRepositoryInterface $checkoutRepository;

    public function __construct(
        CheckoutRepositoryInterface $checkoutRepository
    )
    {
        $this->checkoutRepository = $checkoutRepository;
    }

    public function updatePaymentStatus(Model $checkout, bool $setBadStatus = true): void
    {
        $checkout->refresh();
        $checkoutStatus = $this->getPaymentData($checkout->id);
        if ($checkoutStatus === "success") {
            $checkout->update([
                "is_bought" => true
            ]);
        }
        if ($setBadStatus && !$checkout->is_bought) {
            $checkout->delete();
        }
    }

    public function getPaymentData(int $checkoutId)
    {
        $response = Http::withHeaders([
            "X-Token" => config("monobank.token")
        ])->get(config("monobank.status") . Cache::get((string)$checkoutId));
        return $response["status"];
    }
}
```

Рисунок 3.7 – Сервіс оновлення статусу оплати замовлення

3.2 Реалізація користувацького інтерфейсу вебзастосунку

В цьому розділі розглянемо процес створення інтерфейсу вебзастосунку засобами Vue.js. Інтерфейс взаємодіє з логікою засобами REST API за допомогою JSON та бібліотеки Axios. Всі методи інтеграції зображені на

рисунках 3.8, 3.9 та 3.10. У сховище Vuex зберігається серіалізований результат виконання методів, які асинхронно відправляють об’єкт предмета у back-end [6]. В залежності від відповіді back-end, ці методи або викликають оновлення даних, або відображають повідомлення про помилку. Також, оскільки більшість маршрутів, за якими відправляються ці запити, вимагають авторизованого користувача, з кожним запитом передається JWT-токен із кукі браузера.

```
createTransaction({ commit }, transaction) {
  return axios.post('/transactions', transaction)
    .then(response => {
      commit('ADD_TRANSACTION', response.data);
      return response.data;
    })
    .catch(error => {
      console.error('Error creating transaction:', error);
      throw error;
    });
}
```

Рисунок 3.8 – Метод створення транзакції.

```
getTransactions({ commit }) {
  return axios.get('/transactions')
    .then(response => {
      commit('SET_TRANSACTIONS', response.data);
      return response.data;
    })
    .catch(error => {
      console.error('Error fetching transactions:', error);
      throw error;
    });
}
```

Рисунок 3.9 – Метод отримання транзакцій.

```
updateTransaction({ commit }, transaction) {
  return axios.put(`/transactions/${transaction.id}`, transaction)
    .then(response => {
      commit('UPDATE_TRANSACTION', response.data);
      return response.data;
    })
    .catch(error => {
      console.error('Error updating transaction:', error);
      throw error;
    });
}
```

Рисунок 3.10 – Метод оновлення транзакції.

На рисунку 3.11 зображена сторінка входу користувача до системи. Задля того щоб успішно увійти до системи, треба ввести коректні дані облікового запису. Після відправлення асинхронного POST запиту, якщо дані до входу вірні, користувач отримає JWT токен, у противному випадку буде виведена помилка (див. рис. 3.12).

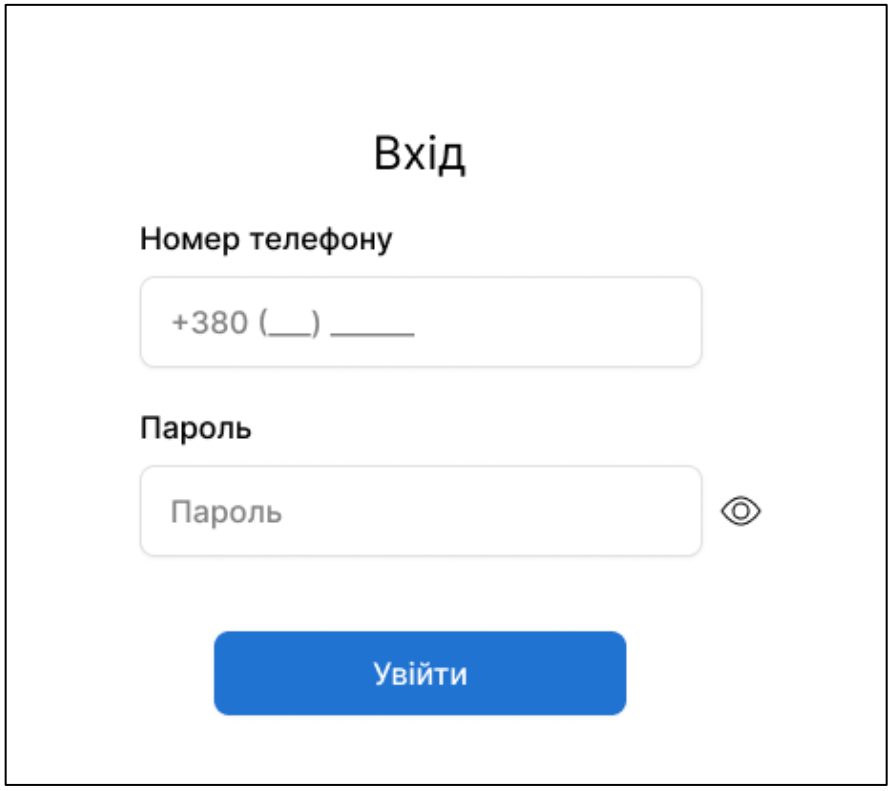


Рисунок 3.11 – Вікно входу до облікового запису

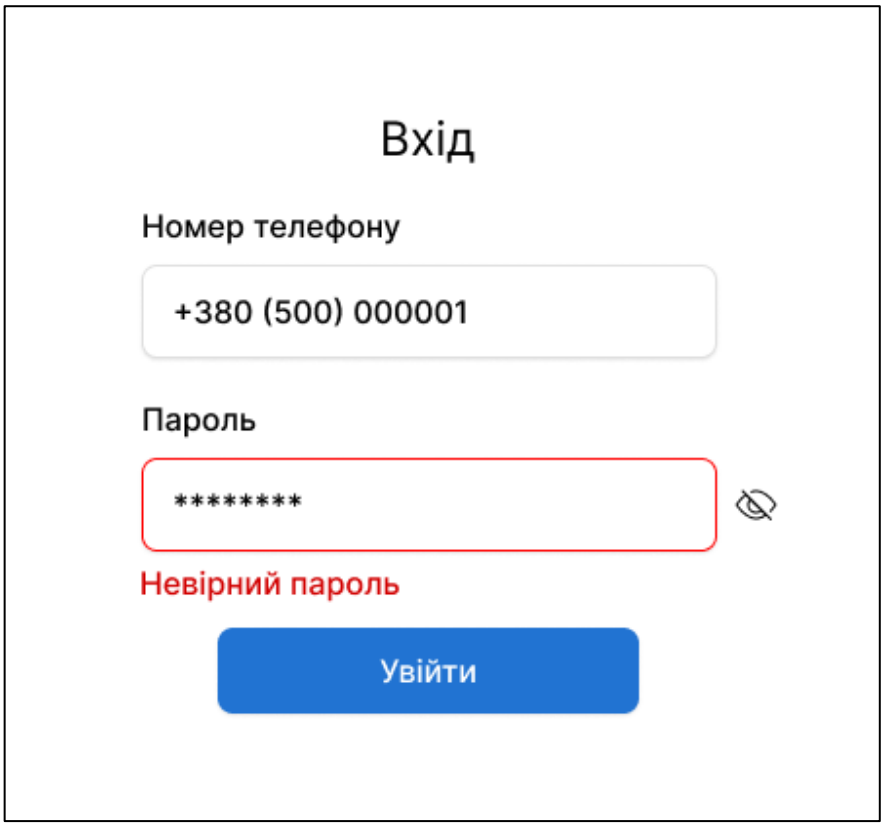


Рисунок 3.12 – Помилка валідації

ВИСНОВКИ

В даній кваліфікаційній роботі було розглянуто принципи роботи з такими фреймворками, як Laravel і Vue.js, для створення вебзастосунку «Розробка інтернет-магазину із застосуванням фреймворків Laravel та Vue.js».

Під час роботи були реалізовані наступні завдання:

– проведено аналіз предметної області, її сутностей та атрибутів; спроєктовано UML-діаграму, а саме діаграму прецедентів;
– реалізовано back-end вебзастосунку за допомогою Laravel;
– впроваджено систему авторизації за допомогою JWT токенів та систему аутентифікації засобами Laravel;
– створено front-end вебзастосунку на основі Vue.js;
– розроблено базу даних з використанням системи управління базами даних MariaDB та проведено її тестування;
– протестований back-end вебзастосунку;
– протестований front-end вебзастосунку.

У процесі виконання теоретичної частини роботи було досліджено обрану предметну область, методи створення та проєктування вебзастосунку засобами Laravel, Vue.js та REST API.

Робота вирішує актуальну проблему замовлення товарів по інтернету, а також їх управління. Крім того, у процесі роботи над проєктом було розглянуто сучасні підходи до безпеки вебзастосунків, включаючи використання методів шифрування даних та захисту від SQL-ін’єкцій. Впроваджено механізми обробки помилок для відслідковування роботи системи та своєчасного реагування на можливі збої.

При виконанні практичної частини вебзастосунку було створено діаграму прецедентів та схему бази даних. Також база даних була реалізована використовуючи інструменти створення міграцій Laravel.

Для зручності користувачів створено інтуїтивно зрозумілий інтерфейс,

який дозволяє швидко та легко орієнтуватися по інтернет-магазину, та зручно керувати списком продуктів для оформлення замовлень. Використано принципи адаптивного дизайну, що гарантує правильне відображення інтерфейсу на різних пристроях.

Проведена робота дозволила створити функціональний і надійний вебзастосунок, який допомагає користувачам ефективно збирати замовлення та зручно їх замовляти. Подальший розвиток проєкту може включати впровадження нових функцій, таких як інтеграція системи бонусів та акцій, нові опції отримання та відстеження замовлень, та розширені можливості налаштування системи під індивідуальні потреби користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. The PHP Framework for Web Artisans. URL: https://laravel.com/ (дата звернення: 21.03.2024).
2. Laravel documentation. URL: https://laravel.com/docs/11.x (дата звернення: 27.03.2024).
3. Macrae C. Vue.js: Up and Running : 1st edition. USA: O'Reilly Media, 2018. 171 p.
4. Mayer D. Vue Handbook: A thorough introduction to Vue.js : 1st edition. Italy : Copes F., 2018. 122 p.
5. Djirdeh H., Murray N., Lerner A. Fullstack Vue: The Complete Guide to Vue.js : 1st edition. USA: Fullstack.io, 2018. 439 p.
6. The Progressive JavaScript Framework. URL: https://vuejs.org/ (дата звернення: 15.04.2024).
7. MariaDB. URL: https://mariadb.org/documentation/ (дата звернення: 17.04.2024).
8. Eloquent ORM. URL: https://laravel.su/docs/10.x/eloquent (дата звернення: 20.04.2024).
9. Monobank open API. URL: https://api.monobank.ua/docs/index.html (дата звернення: 27.04.2024).
10. Fowler M., Beck K., Brant J., Opdyke W., Roberts D. Refactoring: Improving the Design of Existing Code 1st edition. USA : Addison-Wesley, 1999. 431p.