МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ОНЛАЙН-МАГАЗИНУ ПОБУТОВОЇ ТЕХНІКИ ІЗ ЗАСТОСУВАННЯМ ФРЕЙМВОРКІВ VUE ТА LARAVEL»**

Виконала: студентка 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
П.І. Іванова
(ініціали та прізвище)

Керівник декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(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 СТУДЕНТЦІ

Івановій Поліні Ігорівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку для онлайн-магазину побутової техніки із застосуванням фреймворків Vue та Laravel

керівник роботи Гоменюк Сергій Іванович, 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 моделювання вебзастосунку онлайн-магазину побутової техніки.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 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.	09.01.2024	
2.	Збір вихідних даних.	02.02.2024	
3.	Обробка методичних та теоретичних	22.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	29.04.2024	
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 _____ П.І. Іванова
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.І. Гоменюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку для онлайн-магазину побутової техніки із застосуванням фреймворків Vue та Laravel»: 57 с., 25 рис., 2 табл., 8 джерел.

БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, КОНТРОЛЕР, КОМПОНЕНТ, МОДЕЛЬ, ОНЛАЙН-МАГАЗИН, ФРЕЙМВОРК.

Об’єкт дослідження – методи розробки та інтеграції компонентів вебзастосунків для електронної комерції.

Предмет дослідження – процес розробки вебзастосунку в контексті його етапів та виявлення потенційних труднощів, що можуть виникнути під час цього процесу.

Мета роботи – розробка вебзастосунку для онлайн-магазину побутової техніки за допомогою фреймворків Vue та Laravel.

Метод дослідження – порівняння, аналіз, описовий, структурний.

Результати та їх новизна: досліджено етапи розробки вебзастосунку для онлайн-магазину побутової техніки, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення.

Взаємозв’язок з іншими роботами: ця робота базується на дослідженні процесу розробки вебзастосунку та виявлених труднощів, що виникають під час цього процесу.

У кваліфікаційній роботі розглянуто вебзастосунки, які використовуються для реалізації електронної комерції. Зростання попиту на онлайн-покупки побутової техніки робить важливим розв’язок задач ефективного створення, розгортання та підтримки онлайн-магазинів. Тому розробка вебзастосунку для онлайн-магазину побутової техніки є актуальною задачею.

SUMMARY

Bachelor's qualifying paper "Development of a Web Application for an Online Store of Household Appliances Using Vue and Laravel Frameworks": 57 p., 25 figures, 2 tables, 8 references.

DATABASE, WEB APPLICATION, CONTROLLER, COMPONENT, MODEL, ONLINE STORE, FRAMEWORK.

Object of the study: methods of development and integration of components for e-commerce web applications.

Subject of the study: the process of developing a web application in the context of its stages and identifying potential difficulties that may arise during this process.

Aim of the study: to develop a web application for an online household appliance store using Vue and Laravel frameworks.

Methods of research: comparison, analysis, descriptive, structural.

Results and novelty: the stages of developing a web application for an online household appliance store were investigated, key problems were identified, and new methods for collecting and analyzing software requirements were applied. These changes are aimed at improving the management of the development process and increasing the efficiency of the software.

Relationship to other works: this work is based on a study of the web application development process and the difficulties encountered during this process.

Significance and conclusions: the study examined web applications used for e-commerce implementation. The growing demand for online purchases of household appliances makes it important to address the challenges of efficient creation, deployment, and maintenance of online stores. Therefore, the development of a web application for an online household appliance store is a relevant task.

ЗМІСТ

ВСТУП

У сучасному світі сфера розробки програмного забезпечення є однією з найдинамічніших та інноваційних галузей. Щороку вимоги до програмних продуктів зростають, а користувачі стають все більш вибагливими. У цьому контексті розробка комерційних продуктів набуває особливого значення, вимагаючи від розробників не тільки високого технічного рівня, а й глибокого розуміння ринкових вимог та споживчих потреб.

Зростаюча конкуренція та швидкий розвиток технологій у сфері електронної комерції вимагають від розробників не лише технічної майстерності, але й здатності ефективно поєднувати різні інструменти для досягнення високої якості продукту.

Метою цієї кваліфікаційної роботи є аналіз етапів розробки комерційного продукту. Як приклад обрано процес створення вебзастосунку для онлайн-магазину побутової техніки, оскільки цей вид продукту широко представлений в аутсорсингових компаніях та є актуальним об’єктом дослідження. Основні етапи розробки включають:

– аналіз стеку технологій;
– проєктування моделей проєкту;
– розробка готового проєкту;
– тестування готового проєкту;

Об’єктом дослідження є процес розробки вебзастосунку для онлайн-магазину побутової техніки. Предметом дослідження є визначення ключових етапів та вирішення можливих труднощів у цьому процесі.

У першому розділі роботи розглядається технічне завдання від замовника та обґрунтовується вибір технологічного стеку для реалізації проєкту.

Другий розділ присвячений глибокому аналізу предметної області, включаючи створення схеми даних на основі виділених сутностей та аналіз взаємозв’язків між ними.

Третій розділ містить детальний опис програмної реалізації на кожному етапі проєкту. Наводяться пояснення деталей та обґрунтування вибору конкретних рішень при імплементації. Розглядаються приклади реалізації, які ілюструють виконання кожного етапу та демонструють ключові аспекти програмного проєкту.

1 ОГЛЯД СТЕКА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ОНЛАЙН-МАГАЗИНУ ПОБУТОВОЇ ТЕХНІКИ ІЗ ЗАСТОСУВАННЯМ ФРЕЙМВОРКІВ VUE ТА LARAVEL

1.1 Загальні поняття стека технологій

Розробка програмного забезпечення є невід'ємною частиною сучасного технологічного прогресу. У цьому контексті стек технологій стає ключовим елементом для досягнення успіху у розробці програмних продуктів. Стек технологій складається з різноманітних компонентів, які використовуються для створення програмного забезпечення та вебдодатків.

Стек в інформатиці та програмуванні це різновид лінійного списку, структура даних, яка працює за принципом «останнім прийшов – першим пішов». Усі операції (наприклад, видалення елемента) в стеку можна проводити тільки з одним елементом, який міститься на верхівці стека та був уведений у стек останнім.

Стек можна уявити як стопку тарілок, з якої можна взяти верхню, і на яку можна покласти верхню тарілку. Інша назва стека – «магазин», за аналогією з принципом роботи магазина в автоматичній зброї.

У сучасному програмуванні стек технологій зазвичай складається з декількох компонентів.

Мова програмування: це основний інструмент для написання коду програми. Різні мови програмування мають свої переваги та обмеження, і вибір мови може вплинути на продуктивність, швидкість та масштабованість проєкту. Наприклад, JavaScript, Python, Java, C++ – лише деякі з популярних мов програмування.

Фреймворки та бібліотеки: фреймворки та бібліотеки надають готові рішення для розв'язання типових задач, таких як робота з вебінтерфейсами, обробка даних або робота з базами даних. Наприклад, у веброзробці популярні

фреймворки, такі як Vue.js, React, Angular для фронтенду та Laravel, Django, Spring для бекенду.

Системи управління базами даних (СУБД): вони відповідають за зберігання та організацію даних в програмі. СУБД можуть бути реляційними (наприклад, MySQL, PostgreSQL) або нереляційними (наприклад, MongoDB, Firebase), залежно від потреб проєкту.

Інструменти розробки та тестування: ці інструменти допомагають розробникам у створенні, тестуванні та налагодженні програмного забезпечення. Вони включають у себе IDE (Integrated Development Environment), системи контролю версій (наприклад, Git), тестувальні фреймворки (наприклад, Jest, PHPUnit) та інші.

Стек технологій може бути підібраний залежно від конкретних потреб та характеристик проєкту, таких як масштаб, складність, швидкість розробки та інші фактори. Ефективне використання стеку технологій дозволяє розробникам ефективно створювати, розгортати та підтримувати програмне забезпечення з високою якістю та продуктивністю.

Сьогоднішній IT-світ надзвичайно різноманітний, і відповідно, існує безліч популярних стеків технологій, які використовуються великими компаніями для вирішення різноманітних завдань у великих проєктах.

Насправді, вибір правильного стеку технологій є ключовим фактором успіху будь-якого IT-проєкту. Він впливає на швидкість розробки, продуктивність, масштабованість та навіть зручність підтримки програмного забезпечення. Тому перед початком будь-якого проєкту розробники зазвичай ретельно вивчають різні варіанти стеків технологій, аналізуючи їх переваги та недоліки.

В цьому контексті пропонуємо розглянути таблицю, яка містить деякі з найпопулярніших стеків технологій, які використовуються у великих IT-проєктах. Безперечно, кожен стек має свої унікальні особливості та застосування, і правильний вибір може значно позитивно вплинути на успішність вашого проєкту.

Давайте розглянемо найпопулярніші стеки технологій та візуалізуємо їх в таблиці 1.1 та побудуємо піраміду стеків за популярність в світі на рисунку 1.1.

Таблиця 1.1 – Найпопулярніші стеки технологій

Мова програмування	**Фреймворки/ Бібліотеки**	**СУБД**	**Інструменти**
JavaScript	React, Vue.js, Angular	MySQL, MongoDB	Git, Jest, Webpack
Python	Django, Flask	PostgreSQL, SQLite	Docker, PyTest
Java	Spring Boot, Hibernate	Oracle, MySQL	Maven, JUnit
PHP	Laravel, Symfony	MySQL, MariaDB	Composer, PHPUnit
Ruby	Ruby on Rails	PostgreSQL, SQLite	RSpec, Capybara
C#	ASP.NET Core, Entity Framework	SQL Server	NuGet, NUnit
Kotlin	Spring Boot, Ktor	PostgreSQL, SQLite	Gradle, JUnit
Swift	SwiftUI, Vapor	CoreData, SQLite	CocoaPods, XCTest
TypeScript	NestJS, Angular, React	PostgreSQL, SQLite	ESLint, Prettier
Rust	Rocket	PostgreSQL	Cargo, Rustfmt
Scala	Play Framework, Akka	PostgreSQL, SQLite	SBT, ScalaTest
Julia	Lapis	PostgreSQL, SQLite	Pkg, Test

Продовження таблиці 1.1

Мова програмування	**Фреймворки/ Бібліотеки**	**СУБД**	**Інструменти**
Dart	Flutter, AngularDart	SQLite, Firebase	Pub, Mockito
Lua	Lapis	SQLite, PostgreSQL	LuaRocks, Busted
Clojure	Luminus, Compojure	PostgreSQL, Datomic	Leiningen, Test Check

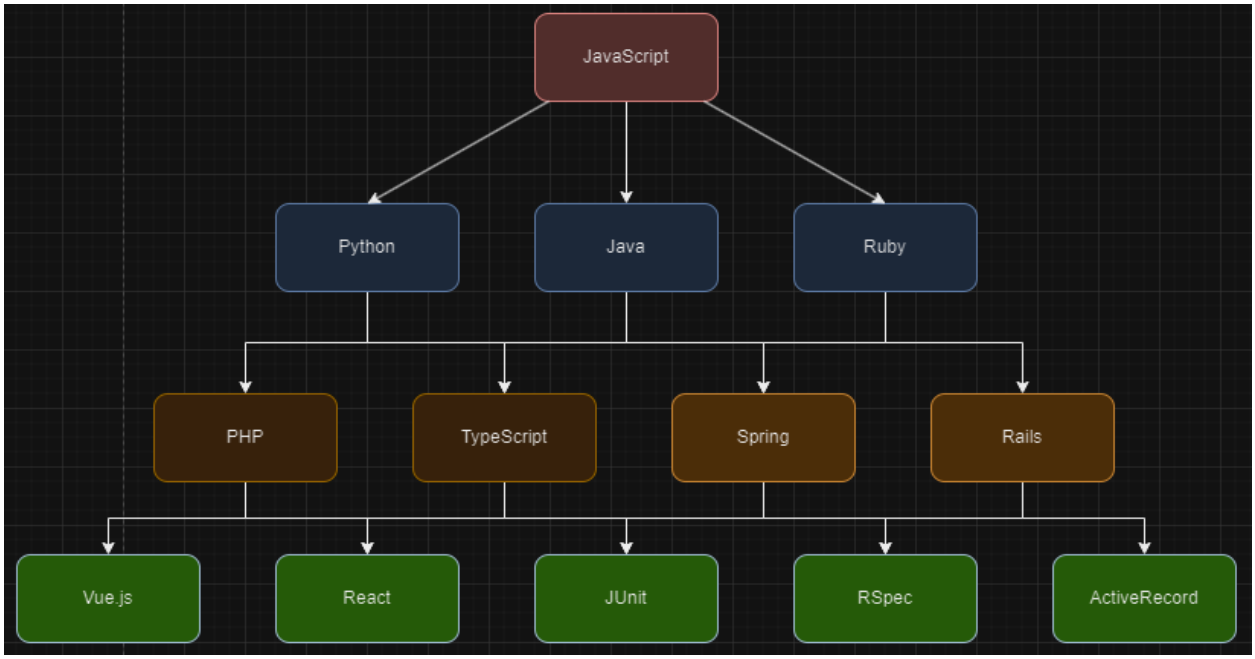


Рисунок 1.1 – Піраміда популярності

Давайте розглянемо декілька душе цікавих, на мою думку, фреймворків у різних мовах програмування, які займають важливе місце в індустрії розробки програмного забезпечення та користуються популярністю серед розробників.

JavaScript:

– **React:** React – це бібліотека JavaScript для створення інтерфейсів користувачів, відомий своєю декларативністю та компонентною структурою, React дозволяє розробникам легко управляти станом

додатку та створювати динамічні UI;

- **Vue.js:** Vue.js – це прогресивний фреймворк JavaScript для створення інтерфейсів користувачів та односторінкових додатків, він простий у використанні, легковаговий та має гнучку архітектуру, що робить його привабливим для початківців та досвідчених розробників;
- **Angular:** Angular – це фреймворк JavaScript, розроблений Google для створення вебдодатків, він має широкий функціонал для створення складних додатків, включаючи роутінг, HTTP-запити та управління станом додатку.

Python:

- **Django:** Django – це високорівневий фреймворк Python для швидкої розробки вебдодатків, він надає вбудовану аутентифікацію, адміністративний інтерфейс та ORM для роботи з базами даних;
- **Flask:** Flask – це легкий фреймворк Python для створення вебдодатків, він має просту структуру та розширювану архітектуру, що робить його ідеальним вибором для маленьких та середніх проєктів.

Java:

- **Spring Boot:** Spring Boot – це фреймворк Java для швидкої розробки вебдодатків, він має вбудовану підтримку для управління конфігурацією, обробки HTTP-запитів та створення RESTful API;
- **Hibernate:** Hibernate – це ORM (Object-Relational Mapping) фреймворк для Java, який дозволяє розробникам легко взаємодіяти з базами даних, використовуючи об’єктно-орієнтований підхід.

1.2 Переваги та недоліки використання комплексного стека

Тепер, коли ми розглянули основні поняття стеку технологій та найкопулярніші фреймворки, розглянемо більш детальніше використання комплексних стеків технологій.

Переваги:

- **швидкість розробки:** використання комплексного стека може значно прискорити процес розробки програмного забезпечення, оскільки багато інструментів та бібліотек вже готові для використання, що дозволяє зосередитися на більш важливих аспектах проєкту;
- **готовість до використання:** багато комплексних стеків містять готові шаблони та рішення для типових задач, що дозволяє розробникам швидко розпочати роботу над проєктом, не витрачаючи час на налаштування середовища розробки;
- **сумісність компонентів:** компоненти стеку зазвичай розробляються та тестуються з урахуванням сумісності один з одним, що дозволяє уникнути проблем з інтеграцією та забезпечити плавну роботу програми;
- **оновлення та підтримка:** багато комплексних стеків мають активну спільноту розробників, яка постійно вдосконалює та оновлює компоненти стеку, а також надає допомогу у вирішенні проблем;
- **єдина екосистема:** використання комплексного стека дозволяє розробникам працювати в єдиній екосистемі, що спрощує управління проєктом та спільну роботу команди;
- **стандартизація:** комплексний стек може встановлювати стандарти та керувати ними у всму проєкті, що сприяє підвищенню якості та однорідності програмного забезпечення.

Недоліки:

- **обмежена гнучкість:** використання комплексного стека може призвести до обмеження гнучкості та свободи вибору технологій, що може бути недоцільним для деяких проєктів;
- **залежність від вендора:** комплексний стек може залежати від вендора або певної компанії, що може вплинути на стратегічні рішення та подальшу розвиток програмного забезпечення;

- **перенесення коду:** існує ризик перенесення коду, який написаний з використанням комплексного стека, до іншого середовища, де може бути складніше його використовувати;
- **масштабованість:** деякі комплексні стеки можуть мати обмеження у масштабованості, що може призвести до проблем при рості проєкту;
- **надмірна складність:** деякі комплексні стеки можуть бути надто складними для використання, особливо для початківців, що може призвести до затримок у розробці або навіть до помилок;
- **продуктивність:** залежно від вибраних компонентів, деякі комплексні стеки можуть мати обмеження на продуктивність або вимагати додаткових оптимізацій для досягнення необхідного рівня продуктивності;
- **зростання витрат:** використання комплексного стека може призвести до зростання витрат на ліцензії, підтримку та інші витрати, пов’язані з використанням певних інструментів чи сервісів;
- **зміна підходу:** іноді комплексний стек може вимагати зміни підходу до розробки або використання нових методів та практик, що може вимагати часу та зусиль для адаптації команди;
- **необхідність навчання:** використання нового комплексного стека може вимагати часу та зусиль для навчання персоналу, особливо якщо вони не мають попереднього досвіду з використання цих технологій;
- **інтеграція з іншими системами:** деякі комплексні стеки можуть мати обмеження або складності у інтеграції з іншими системами чи сервісами, що може ускладнити роботу з розробленим програмним забезпеченням;
- **ризик втрати підтримки:** залежно від вибору компонентів, деякі комплексні стеки можуть мати ризик втрати підтримки чи активного розвитку, що може призвести до проблем з безпекою чи збоєм програмного забезпечення;

- **ліцензійні обмеження:** використання певних компонентів у комплексному стеці може вимагати дотримання ліцензійних умов або обмежень, що може вплинути на вибір та розгортання програмного забезпечення;
- **залежність від зовнішніх сервісів:** деякі комплексні стеки можуть бути залежними від зовнішніх сервісів чи платформ, що може створювати ризик недоступності чи змін у поведінці програмного забезпечення.

Після аналізу переваг та недоліків використання комплексного стека технологій у розробці програмного забезпечення, мій вибір склався на двох ключових технологіях: Vue.js для фронтенду та Laravel для бекенду. Vue.js привернув мою увагу своєю простотою використання та гнучкістю у створенні динамічних інтерфейсів, тоді як Laravel вражає своєю чистотою та ефективністю у розробці серверної частини додатку. Обидва фреймворки володіють активною спільнотою розробників та забезпечують зручні інструменти для реалізації різноманітних функціональних можливостей. Таким чином, вибір Vue.js для фронтенду та Laravel для бекенду виявився оптимальним для виконання мого проєкту, забезпечуючи його якісне та ефективне втілення.

1.3 Фреймворк Vue.js

Vue (вимовляється як (англ.) /vjuː/, (укр) /в’ю/) – це фреймворк, який працює на JavaScript, створений для розробки користувацьких інтерфейсів [1]. Він працює на базі звичайного HTML, CSS та JavaScript, з можливостями декларативно програмувати користувацькі інтерфейси будь-якої складності на основі компонентів. Розглянемо приклад настростіших задач з використанням Vue на рисунку 1.2.

Рисунок 1.2 – Найпростіша реалізація задачі

Наведений вище приклад демонструє дві основні функції Vue:

– **декларативний рендеринг**: Vue розширює стандартний HTML шаблонним синтаксисом, який дозволяє нам декларативно задавати структуру HTML на основі стану описаного у JavaScript;
– **реактивність**: Vue автоматично відстежує зміни стану описаного у JavaScript і з максимальною ефективністю оновлює DOM, коли відбуваються зміни.

Vue – це фреймворк та екосистема, яка охоплює більшість функцій, необхідних для розробки інтерфейсу. Але веб додатки надзвичайно різноманітні – речі, які ми створюємо в рамках вебдодатків, можуть кардинально відрізнятися за формою та масштабом. Зважаючи на це, Vue розроблено таким чином, щоб бути гнучким і адаптивним. Залежно від вашого випадку та задач, Vue можна використовувати різними способами:

– розширення статичного HTML;
– вбудовування як веб компонента на будь-яку сторінку;
– створення одно-сторінкового додатку (SPA);

– фулстек / додаток з рендерингом на стороні серверу (SSR);
– jamstack / генерація статичного додатку (SSG);
– створення десктопних, мобільних, WebGL додатків і навіть додатків для термінала.

У більшості проєктів Vue, що використовують інструменти збірки та потребують етапу збірки, ми створюємо компоненти Vue, використовуючи HTML-подібний формат файлу під назвою Single-File Component (також відомий як файли *.vue, скорочено SFC). Vue SFC, як випливає з назви, інкапсулює логіку компонента (JavaScript), шаблон (HTML) і стилі (CSS) в одному файлі. Ось попередній приклад рисунку 1.2 перероблений у формат SFC та відображений на рисунку 1.3.

```
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">Лічильник: {{ count }}</button>
</template>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

Рисунок 1.3 – Найпростіша реалізація задачі у форматі SFC

Компоненти Vue можна створювати в двох різних стилях API: Опційному API та Композиційному API. Кожен із цих підходів має свої переваги та недоліки, і розробники можуть вибирати між ними в залежності від потреб проєкту та особистих уподобань.

Опційний API є традиційним способом створення компонентів у Vue. Він базується на об’єктно-орієнтованому підході, де різні частини компонента, такі як дані, методи, обчислювані властивості та життєвий цикл, визначаються як окремі опції в межах одного об’єкта.

Основні характеристики Опційного API:

- **простота та читабельність:** опційний API є інтуїтивно зрозумілим і простим для освоєння, особливо для новачків. Код структурований у вигляді об’єкта, де кожна частина компонента визначена чітко і зрозуміло;
- **легкість використання:** опційний API добре підходить для невеликих та середніх проєктів, де не потрібна складна логіка та високий рівень гнучкості;
- **логічне розділення коду:** оскільки код розділений на окремі опції, кожна з яких має свою чітко визначену роль, це полегшує розуміння та підтримку коду.

Композиційний API був представлений у Vue 3 і забезпечує більш гнучкий та модульний підхід до створення компонентів. Він базується на функціях та композиції, що дозволяє більш ефективно розділяти логіку компонентів та повторно використовувати код.

Основні характеристики Композиційного API:

- **гнучкість та повторне використання:** композиційний API дозволяє створювати більш гнучкі та повторно використовувані шматки коду, що особливо корисно для великих та складних проєктів;
- **краща організація коду:** логіка компонента може бути розділена на окремі функції, що робить код більш організованим і легким для підтримки;
- **покращена підтримка TypeScript:** композиційний API забезпечує кращу підтримку TypeScript завдяки своїй функціональній природі;
- **зменшення розміру компонента:** використання композиційного API дозволяє зменшити розмір коду компонента шляхом вилучення логіки в окремі функції.

Vue.js є потужним та гнучким фреймворком для розробки сучасних вебдодатків. Його простота, реактивність та компонентна архітектура роблять його ідеальним вибором як для початківців, так і для досвідчених розробників.

Завдяки підтримці двох стилів API – Опційного та Композиційного – Vue.js надає розробникам можливість вибирати підхід, що найкраще відповідає їхнім потребам та стилю кодування. З широким спектром додаткових інструментів та бібліотек, Vue.js забезпечує все необхідне для створення високоякісних вебдодатків, що легко масштабуються та підтримуються.

1.4 Фреймворк Laravel

Laravel – це один із найпопулярніших і потужних фреймворків для веброзробки, побудований на мові програмування PHP [2, 6]. Він створений для розробки вебдодатків з використанням архітектурного шаблону Model-View-Controller (MVC). Laravel відомий своєю елегантною та виразною синтаксичною структурою, яка робить процес розробки більш простим і приємним.

Laravel був створений Тейлором Отвеллом (Taylor Otwell) у 2011 році як альтернатива фреймворку CodeIgniter, з метою забезпечити сучасніший і багатофункціональний інструмент для веброзробки. З моменту свого випуску Laravel швидко здобув популярність серед розробників завдяки своїм численним функціям, легкості використання та активній спільноті.

Давайте перерахуємо основні можливості та характеристики Laravel.

MVC архітектура: Laravel слідує архітектурному шаблону Model-View-Controller (MVC), що забезпечує розділення бізнес-логіки, презентаційного шару та контролю. Це сприяє кращій організації коду, полегшує його підтримку та тестування.

Елегантний синтаксис: однією з головних переваг Laravel є його елегантний та виразний синтаксис, що робить код більш читабельним та зрозумілим. Це значно знижує поріг входження для нових розробників та підвищує продуктивність команди.

ORM Eloquent: Eloquent – це об’єктно-реляційний мапер (ORM), що

дозволяє взаємодіяти з базою даних за допомогою об’єктів моделей. Завдяки Eloquent, розробники можуть здійснювати складні запити до бази даних за допомогою зрозумілого та інтуїтивного синтаксису.

Міграції баз даних: Laravel включає потужний інструмент для управління версіями баз даних, відомий як міграції. Вони дозволяють створювати та змінювати структуру бази даних програмно, що полегшує спільну роботу в команді та підтримку проєкту.

Система пакетів і модулів: Laravel має вбудовану підтримку для управління пакетами через Composer, що дозволяє легко інтегрувати сторонні бібліотеки та розширювати функціональність додатків. Це забезпечує високу гнучкість та масштабованість проєктів.

Безпека: Laravel включає вбудовані механізми для забезпечення безпеки додатків, такі як захист від SQL-ін’єкцій, XSS-атак та CSRF-атак. Це дозволяє розробникам зосередитися на бізнес-логіці, не турбуючись про основні аспекти безпеки.

Масштабованість: Laravel забезпечує високу масштабованість додатків завдяки підтримці кешування, черг завдань та розподілених систем. Це дозволяє створювати як невеликі додатки, так і великі проєкти з високим навантаженням.

Blade шаблонізатор: Blade – це потужний та гнучкий шаблонізатор, вбудований у Laravel. Він дозволяє створювати динамічні вебсторінки з використанням простого та інтуїтивного синтаксису, що робить процес розробки інтерфейсу більш ефективним.

Підтримка RESTful API: Laravel надає зручні інструменти для створення RESTful API, що робить його ідеальним вибором для розробки сучасних вебдодатків, що взаємодіють з клієнтськими додатками через API.

Розширення функціональності: Laravel легко розширюється за допомогою сторонніх пакетів та модулів, що дозволяє інтегрувати додаткові функції без значних зусиль. Це забезпечує високу гнучкість та адаптивність проєктів.

Laravel є потужним і гнучким фреймворком, який надає безліч можливостей для швидкої та ефективної розробки вебдодатків. Його елегантний синтаксис, модульна архітектура та активна спільнота роблять його ідеальним вибором для багатьох проєктів. Однак, як і будь-який інший інструмент, Laravel має свої переваги та недоліки, які варто враховувати при виборі фреймворку для конкретного проєкту. Завдяки широкому спектру функцій та інструментів, Laravel залишається одним з найпопулярніших фреймворків для веброзробки на PHP.

1.5 Висновки до розділу 1

На цьому етапі було проведено детальний огляд та аналіз стека технологій для розробки онлайн-магазину побутової техніки з використанням фреймворків Vue.js та Laravel. Розглянуто основні особливості, переваги та недоліки цих фреймворків, а також їх інтеграцію. Це дозволило сформувати чітке уявлення про інструменти, які будуть використовуватися в подальшій розробці вебзастосунку, забезпечуючи високу якість та ефективність розробки.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-МАГАЗИНУ

2.1 Концептуальна та логічна модель даних

Розглянемо перші кроки в розробці нашого проєкту. Почнемо з моделювання сутностей, з якими ми будемо працювати у нашому вебзастосунку. Також підготуємо код для міграції кожної з цих сутностей, щоб забезпечити структуру бази даних, необхідну для подальшої розробки.

Для кращого розуміння, спочатку буде представлена загальна модель усіх таблиць та сутностей, яка відображає структуру нашого вебзастосунку онлайн-магазину побутової техніки (див. рис. 2.1).

Давайте розберемо кожну сутність більш детально, та опишемо в більш конкретних прикладах та даних:

Атрибути користувачів (Users):

- ідентифікатор (ID): унікальний номер кожного користувача в системі;
- ім’я (Name): ім’я користувача;
- електронна пошта (Email): адреса електронної пошти користувача;
- пароль (Password): захешований пароль користувача;
- роль (Role): роль користувача у системі (наприклад, адміністратор, клієнт тощо);
- дата реєстрації (Registration Date): дата реєстрації користувача в системі.

Атрибути продуктів (Products):

- ідентифікатор (ID): унікальний номер кожного продукту;
- назва (Name): назва продукту;
- ціна (Price): вартість продукту;
- опис (Description): опис характеристик та властивостей продукту;

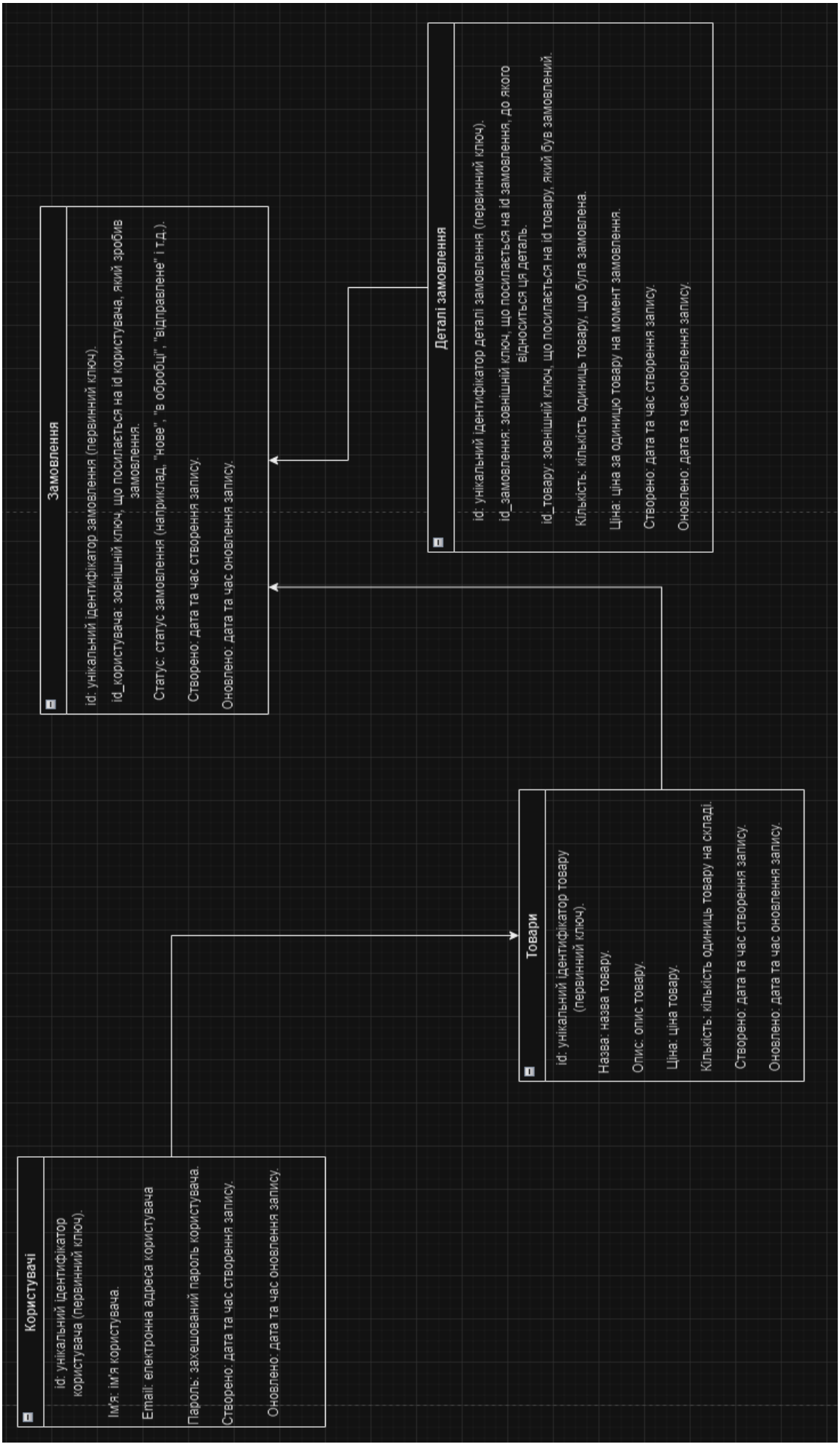


Рисунок 2.1 – Схема вебзастосунку

– категорія (Category): категорія, до якої відноситься продукт;
– фотографія (Photo): зображення або фотографія продукту.

Атрибути замовлень (Orders):

– ідентифікатор (ID): унікальний номер кожного замовлення;
– ідентифікатор користувача (User ID): посилання на користувача, який зробив замовлення;
– статус (Status): поточний статус замовлення (наприклад, «очікує оплати», «в обробці», «відправлено» тощо);
– дата створення (Creation Date): дата створення замовлення;
– загальна вартість (Total Price): загальна вартість усіх продуктів у замовленні;
– адреса доставки (Shipping Address): адреса, на яку потрібно доставити замовлення.

Кожна сутність має свої унікальні атрибути, які визначають її властивості та характеристики [3]. При цьому вони взаємодіють між собою, утворюючи складну систему, яка дозволяє здійснювати різноманітні функції та операції у вебзастосунку.

2.2 Проєктування структури вебзастосунку

Проєктування структури вебзастосунку є критичним етапом у розробці будь-якого проєкту. На цьому етапі визначається архітектура застосунку, його модульна структура та взаємозв’язки між різними компонентами. Нижче подано детальний опис проєктування структури вебзастосунку з використанням фреймворку Vue.js та Laravel.

На основі фреймворка Vue.js ми будемо побудовані структуру фронтенду нашого вебзастосунку. Ця структура включає низку компонентів, кожен з яких відповідає за певну частину функціоналу та відображення на сторінці:

- головна сторінка (Home.vue): ця сторінка буде вітати користувачів при завантаженні вебзастосунку (тут можуть бути відображені загальні оголошення, акції, новини чи інші важливі повідомлення);
- каталог товарів (Products.vue): цей компонент буде відповідати за відображення списку товарів або послуг, які доступні для покупки (користувачі зможуть переглядати категорії товарів, застосовувати фільтри, переглядати деталі товару та додавати їх до корзини);
- сторінка товару (ProductDetail.vue): тут буде детальна інформація про окремий товар або послугу (користувачі зможуть переглядати зображення товару, опис, характеристики та відгуки інших користувачів);
- корзина покупок (Cart.vue): цей компонент дозволить користувачам переглядати товари, які вони додали до корзини, і виконувати різні операції з ними, такі як зміна кількості товарів або їх видалення.

Кожен з цих компонентів грає важливу роль у вебзастосунку, забезпечуючи зручну та ефективну взаємодію користувачів з додатком. Їх розподіл дозволяє зберігати код чистим і організованим, що полегшує розвиток та підтримку нашого проєкту.

Додатково можливе використання Vuex, для керування станом додатку у Vue.js, яка дозволяє нам зберігати, оновлювати та спільно використовувати дані між різними компонентами [4]. Використання Vuex рекомендується для складних додатків зі значною кількістю взаємодіючих компонентів, де потрібно ефективно керувати станом та його змінами.

Це дозволяє уникнути проблем з передачею даних через вкладені компоненти та робить код більш чистим та організованим.

- дії (Actions): функції, які викликаються компонентами для виконання певних операцій або взаємодії зі стором (дії можуть виконувати асинхронні операції та викликати мутації для зміни стану);
- мутації (Mutations): функції, які змінюють стан в сховищі (вони є синхронними та дозволяють змінювати стан у відповідності до певної логіки);

– модулі (Modules): підсховища даних та функціональності, які можуть бути організовані у вигляді окремих модулів (це дозволяє структурувати сховище та розділяти логіку між різними частинами додатку).

За допомогою Vuex ми можемо створити централізований сховище даних, яке буде зберігати всі необхідні дані для роботи нашого додатку.

Взаємодія між фронтендом і бекендом нашого вебзастосунку буде здійснюватися через RESTful API (Application Programming Interface). RESTful API є стандартом для побудови вебсервісів, який дозволяє взаємодіяти з сервером за допомогою HTTP-запитів і отримувати або надсилати дані у вигляді JSON або XML.

Коли фронтенд потребує отримати дані з сервера або зберегти зміни, він виконує HTTP-запити до відповідних ендпоінтів на бекенді. Наприклад, для отримання списку товарів фронтенд може зробити GET-запит до ендпоінту /api/products, а для збереження нового замовлення – POST-запит до /api/orders.

Бекенд, у свою чергу, обробляє ці запити, взаємодіє з базою даних, виконує потрібні операції і повертає фронтенду необхідні дані або підтвердження операцій.

Використання RESTful API дозволяє нам розділити фронтенд та бекенд на два окремі компоненти, що спрощує розробку, розширення та підтримку нашого вебзастосунку. Крім того, цей підхід робить наш додаток більш масштабованим та гнучким, оскільки дозволяє незалежно масштабувати фронтенд та бекенд, використовуючи різні технології та інфраструктуру.

Структура бази даних буде забезпечена за допомогою міграцій. Міграції дозволяють створювати та оновлювати таблиці та їх структуру, забезпечуючи цілісність та узгодженість даних. Використання міграцій дозволяє легко відстежувати зміни в структурі бази даних, відновлювати попередні версії та забезпечувати стабільність додатку під час розробки.

Для керування користувачами та доступом до ресурсів буде використовуватися вбудована система автентифікації Laravel. Ця система

забезпечує надійний захист та управління доступом до різних частин додатку. Вона дозволяє реалізувати функціональність реєстрації, входу, виходу, скидання пароля та інші важливі функції, пов’язані з автентифікацією користувачів. Авторизація забезпечує контроль доступу до різних ресурсів на основі ролей та прав користувачів, що дозволяє забезпечити безпеку та захищеність даних.

Отже, проєктування структури вебзастосунку з використанням фреймворків Vue.js та Laravel є комплексним процесом, що включає різні аспекти, такі як побудова фронтенду на основі компонентів, керування станом за допомогою Vuex, взаємодія з бекендом через RESTful API, створення та управління базою даних за допомогою міграцій, а також забезпечення автентифікації та авторизації користувачів. Ретельне планування та інтеграція цих компонентів дозволяє створити надійний, масштабований та гнучкий вебзастосунок, який забезпечить зручну та ефективну взаємодію користувачів з додатком, а також полегшить підтримку та розширення функціональності в майбутньому.

2.3 Інтеграція з базою даних

Інтеграція з базою даних є критично важливим аспектом розробки вебзастосунку. У нашому проєкті ми будемо використовувати фреймворк Laravel, який забезпечує зручний та потужний інструментарій для роботи з базою даних, включаючи ORM (Object-Relational Mapping) Eloquent, міграції та фабрики для створення тестових даних. Нижче подано детальний опис кроків інтеграції з базою даних у нашому проєкті.

У нашому проєкті ми будемо використовувати MySQL, реляційну базу даних, яка є однією з найпопулярніших і найбільш використовуваних у світі [5]. MySQL забезпечує високу продуктивність, надійність та масштабованість, що робить її ідеальним вибором для більшості вебзастосунків.

Для налаштування з’єднання з базою даних у Laravel, необхідно налаштувати файл конфігурації .env, де зазначаються параметри підключення, такі як назва бази даних, ім’я користувача, інші необхідні дані (див. рис. 2.2).

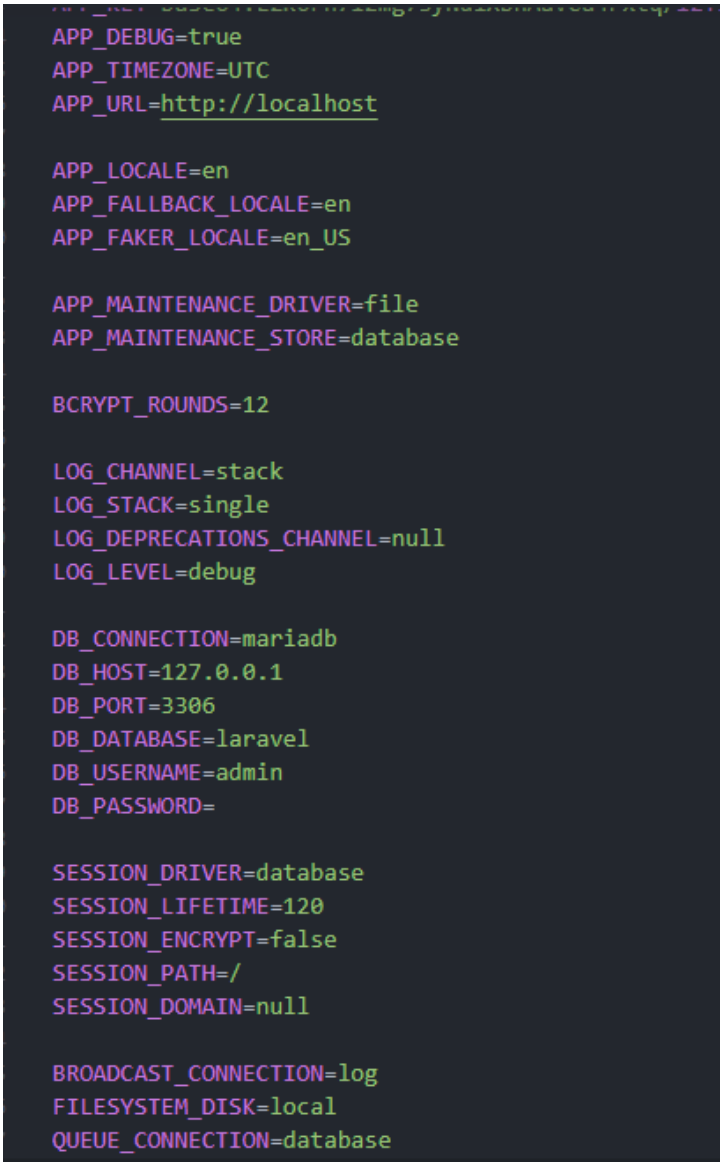


Рисунок 2.2 – Налаштування підключення Laravel

Міграції в Laravel використовуються для створення та оновлення структури бази даних. Вони дозволяють визначати структуру таблиць, індекси, зв’язки та інші елементи бази даних за допомогою PHP-коду. Міграції зберігаються у директорії database/migrations і можуть бути виконані за допомогою команди «php artisan migrate».

На рисунку 2.3 продемонстровано приклад міграції таблиці реєстрації.

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

Рисунок 2.3 – Приклад міграції таблиці реєстрації

Eloquent – це ORM для Laravel, що забезпечує зручний інтерфейс для роботи з базою даних, дозволяючи взаємодіяти з таблицями як з об’єктами. Кожна таблиця у базі даних відповідає моделі у Laravel, яка визначається у директорії app/Models (див. рис. 2.4).

```php
use App\Models\Category;
use App\Models\Product;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function index()
    {
        $query = Product::query();

        return $this->renderProducts($query);
```

Рисунок 2.4 – Приклад моделі таблиці products

Контролери у Laravel відповідають за обробку запитів та взаємодію з моделями. Приклад використання моделі Product у контролері наведений на рисунку 2.5.

```php
class ProductController extends Controller
{
    public function index()
    {
        $query = Product::query();

        return $this->renderProducts($query);
```

Рисунок 2.5 – Приклад моделі таблиці products у контролері

Фабрики у Laravel використовуються для генерації тестових даних. Вони дозволяють легко створювати фіктивні записи у базі даних для тестування (див. рис. 2.6).

```php
class UserFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition()
    {
        return [
            'name' => fake()->name(),
            'email' => fake()->safeEmail(),
            'email_verified_at' => now(),
            'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheW(
            'remember_token' => Str::random(10),
        ];
    }
```

Рисунок 2.6 – Приклад фабрики моделі користувача

Інтеграція з базою даних за допомогою Laravel забезпечує ефективний і структурований підхід до управління даними, що спрощує розробку, тестування та підтримку вебзастосунку.

Також для побудови нашого вебзастосунку ми визначили кілька ключових сутностей, які будуть використовуватися для зберігання та управління даними в базі даних. Нижче наведено таблицю 2.1 з основними сутностями та їх атрибутами.

Таблиця 2.1 – Таблиця сутностей для бази даних

users	– id; – name; – email; – password; – created_at; – updated_at.	– Int; – String; – String; – String; – Timestamp; – Timestamp.	Таблиця користувачів. Зберігає дані про користувачів, включаючи їх ім’я, електронну пошту та пароль.
products	– id; – name; – description; – price; – created_at; – updated_at.	– Int; – String; – Text; – Decimal; – Timestamp; – Timestamp.	Таблиця товарів. Містить інформацію про товари, включаючи назву, опис та ціну.
orders	– id, – user_id, – total, – created_at, – updated_at.	– Int; – Int; – Decimal; – Timestamp; – Timestamp.	Таблиця замовлень. Зберігає інформацію про замовлення, включаючи ID користувача, що здійснив замовлення, та загальну суму замовлення.
order_itemsorder_items	– id, – order_id, – product_id, – quantity, – price, – created_at, – updated_at.	– Int; – Int; – Int; – Int; – Decimal; – Timestamp; – Timestamp.	Таблиця позицій замовлення. Містить дані про окремі товари у складі замовлення, включаючи їх кількість та ціну.
categories	– id, – name, – created_at, – updated_at.	– Int, – String, – Timestamp, – Timestamp.	Таблиця категорій. Зберігає назви категорій товарів.

Продовження таблиці 2.1

product_category	– id, – product_id, – category_id, – created_at, – updated_at.	– Int, – Int, – Int, – Timestamp, – Timestamp.	Таблиця зв’язків товарів та категорій. Зберігає інформацію про те, які товари належать до яких категорій.

2.4 Висновки до розділу 2

У другому розділі було детально розглянуто проєктування та моделювання вебзастосунку онлайн-магазину. Спочатку ми створили концептуальну та логічну модель даних, яка визначила основні сутності та їх взаємозв’язки. Це дало нам змогу чітко зрозуміти структуру даних, з якими буде працювати наш вебзастосунок, та забезпечити їх ефективне зберігання й обробку.

Далі ми перейшли до проєктування структури вебзастосунку. Було визначено основні компоненти системи та їхню взаємодію, що дозволило розробити зручну та інтуїтивно зрозумілу архітектуру. Це включало розробку клієнтської та серверної частин, а також інтеграцію з сторонніми сервісами та інструментами для забезпечення повноцінної роботи застосунку.

Останнім етапом була інтеграція з базою даних. Ми налаштували підключення до бази даних, створили необхідні таблиці та налаштували відповідні запити для зберігання і отримання даних. Це забезпечило надійне збереження інформації та швидкий доступ до неї, що є критично важливим для будь-якого онлайн-магазину.

Таким чином, у цьому розділі ми заклали міцний фундамент для подальшої розробки вебзастосунку, забезпечивши його стійкість, масштабованість та ефективність. Ретельно спланована структура та належна інтеграція з базою даних гарантують стабільну роботу системи та високу якість обслуговування користувачів.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-МАГАЗИНУ

3.1 Опис функціональних вимог

Функціональні вимоги до вебзастосунку онлайн-магазину визначають основні можливості та функції, які система повинна підтримувати для забезпечення її коректної роботи та задоволення потреб користувачів. Наведемо основні функціональні вимоги, які повинні бути реалізовані у вебзастосунку.

Реєстрація та автентифікація користувачів:

– реєстрація користувачів: користувачі повинні мати можливість створювати облікові записи, надаючи основну інформацію (ім’я, електронну пошту, пароль);
– вхід в систему: користувачі повинні мати можливість входити до системи за допомогою своїх облікових даних;
– відновлення пароля: користувачі повинні мати можливість відновлювати забуті паролі через електронну пошту.

Управління профілем користувача:

– редагування профілю: користувачі повинні мати можливість переглядати та змінювати інформацію у своєму профілі;
– зміна пароля: користувачі повинні мати можливість змінювати свій пароль.

Каталог товарів:

– перегляд каталогу: користувачі повинні мати можливість переглядати каталог товарів, які доступні для покупки;
– фільтрація та сортування: користувачі повинні мати можливість фільтрувати та сортувати товари за різними критеріями (ціна, категорія, рейтинг тощо);

– перегляд деталей товару: користувачі повинні мати можливість переглядати детальну інформацію про окремий товар.

Кошик покупок:

– додавання товарів до кошика: користувачі повинні мати можливість додавати товари до кошика;
– перегляд вмісту кошика: користувачі повинні мати можливість переглядати товари, додані до кошика;
– зміна кількості товарів: користувачі повинні мати можливість змінювати кількість кожного товару в кошику;
– видалення товарів з кошика: користувачі повинні мати можливість видаляти товари з кошика.

Оформлення замовлення:

– перегляд замовлення: користувачі повинні мати можливість переглядати своє замовлення перед оформленням;
– вибір способу доставки: користувачі повинні мати можливість обирати спосіб доставки;
– вибір способу оплати: користувачі повинні мати можливість обирати спосіб оплати (кредитна картка, PayPal тощо);
– підтвердження замовлення: користувачі повинні мати можливість підтверджувати замовлення і отримувати підтвердження про успішну обробку замовлення.

Управління замовленнями:

– перегляд історії замовлень: користувачі повинні мати можливість переглядати історію своїх замовлень;
– перегляд статусу замовлення: користувачі повинні мати можливість переглядати поточний статус своїх замовлень.

Адміністрування:

– управління товарами: адміністратори повинні мати можливість додавати, редагувати та видаляти товари з каталогу;
– управління категоріями: адміністратори повинні мати можливість

створювати, редагувати та видаляти категорії товарів;

– управління замовленнями: адміністратори повинні мати можливість переглядати та змінювати статус замовлень.

Пошук:

– пошук товарів: користувачі повинні мати можливість шукати товари за ключовими словами.

Відгуки та рейтинги:

– додавання відгуків: користувачі повинні мати можливість залишати відгуки про товари;
– перегляд відгуків: користувачі повинні мати можливість переглядати відгуки інших користувачів;
– рейтинг товарів: користувачі повинні мати можливість оцінювати товари за допомогою рейтингової системи.

Сповіщення:

– електронні сповіщення: користувачі повинні отримувати електронні сповіщення про статус замовлень, нові товари, акції та інші важливі події.

Ці функціональні вимоги забезпечують комплексний набір можливостей для користувачів та адміністраторів вебзастосунку онлайн-магазину, що дозволяє ефективно управляти процесом покупки та продажу товарів через інтернет.

3.2 Розробка клієнт-сервера

Процес розробки клієнт-серверної архітектури для вебзастосунку онлайн-магазину включає в себе декілька етапів, серед яких проєктування, реалізація та тестування серверної та клієнтської частин. Клієнт-серверна модель дозволяє створювати розподілені системи, де клієнт взаємодіє з сервером для отримання даних, виконання операцій та відображення

інформації користувачу. У нашому випадку серверна частина реалізується за допомогою Laravel, а клієнтська – за допомогою React.

Компонент Categories представляє інтерфейс для управління категоріями в додатку, який використовує Vue 3. Він включає функціональність для відображення списку категорій, додавання нових категорій та редагування існуючих категорій через модальні вікна (див. рис. 3.1).

Рисунок 3.1 – Компонент Categories

Компонент надає користувачу можливість переглядати список категорій, додавати нові категорії та редагувати існуючі. Кнопка «Додати нову категорію» відкриває модальне вікно для введення даних нової категорії. При редагуванні існуючої категорії, дані вибраної категорії завантажуються у модальне вікно, де їх можна змінити.

Цей підхід забезпечує зручне управління категоріями з використанням компонентного підходу Vue.js, що дозволяє легко масштабувати і підтримувати додаток.

Компонент CategoriesTable представляє таблицю категорій з можливістю сортування, редагування та видалення категорій, а також додавання нових категорій через модальне вікно (див. рис. 3.2).

```
<table class="table-auto w-full">
  <thead>
  <tr>
    <TableHeaderCell field="id" :sort-field="sortField" :sort-direction="sortDirection"
                     @click="sortCategories('id')">
      ID
    </TableHeaderCell>
    <TableHeaderCell field="name" :sort-field="sortField" :sort-direction="sortDirection"
                     @click="sortCategories('name')">
      Назва
    </TableHeaderCell>
    <TableHeaderCell field="slug" :sort-field="sortField" :sort-direction="sortDirection"
                     @click="sortCategories('slug')">
      Slug
    </TableHeaderCell>
    <TableHeaderCell field="active" :sort-field="sortField" :sort-direction="sortDirection"
                     @click="sortCategories('active')">
      Активний
    </TableHeaderCell>
    <TableHeaderCell field="parent_id" :sort-field="sortField" :sort-direction="sortDirection"
                     @click="sortCategories('parent_id')">
      Батьківський елемент
    </TableHeaderCell>
    <TableHeaderCell field="created_at" :sort-field="sortField" :sort-direction="sortDirection"
                     @click="sortCategories('created_at')">
      Дата створення
    </TableHeaderCell>
    <TableHeaderCell field="actions">
      Дії
    </TableHeaderCell>
  </tr>
  </thead>
```

Рисунок 3.2 – Компонент CategoriesTable

Розпишемо логіку роботи компонента компонента.

Обчислення категорій:

– categories обчислює стан категорій з магазину.

Реактивні змінні:

– sortField визначає поле сортування;

– sortDirection визначає напрямок сортування;

– showCategoryModal керує видимістю модального вікна.

Методи:

– getCategories(url): завантажує категорії з API з урахуванням сортування;

– sortCategories(field): сортує категорії за вказаним полем;

– deleteCategory(category): видаляє вибрану категорію після підтвердження;
– editCategory(category): викликає подію clickEdit для редагування категорії.

Цей компонент забезпечує зручне управління категоріями з можливістю сортування, додавання, редагування та видалення категорій. Використання модальних вікон та інтеграція з магазинами Vuex забезпечує ефективне управління станом та взаємодію з бекендом.

Компонент Customers відповідає за відображення списку клієнтів та управління їх редагуванням через таблицю та модальне вікно (див. рис. 3.3).

Рисунок 3.3 – Компонент Customers

Цей компонент забезпечує основний інтерфейс для управління клієнтами, включаючи їх відображення у вигляді таблиці та редагування через модальне вікно. Використання магазину Vuex для управління станом дозволяє ефективно отримувати та оновлювати дані клієнтів.

Компонент Products відповідає за відображення списку продуктів та надає можливість додавання нового продукту через посилання на відповідну сторінку. Нижче надається детальний опис цього коду (див. рис. 3.4).

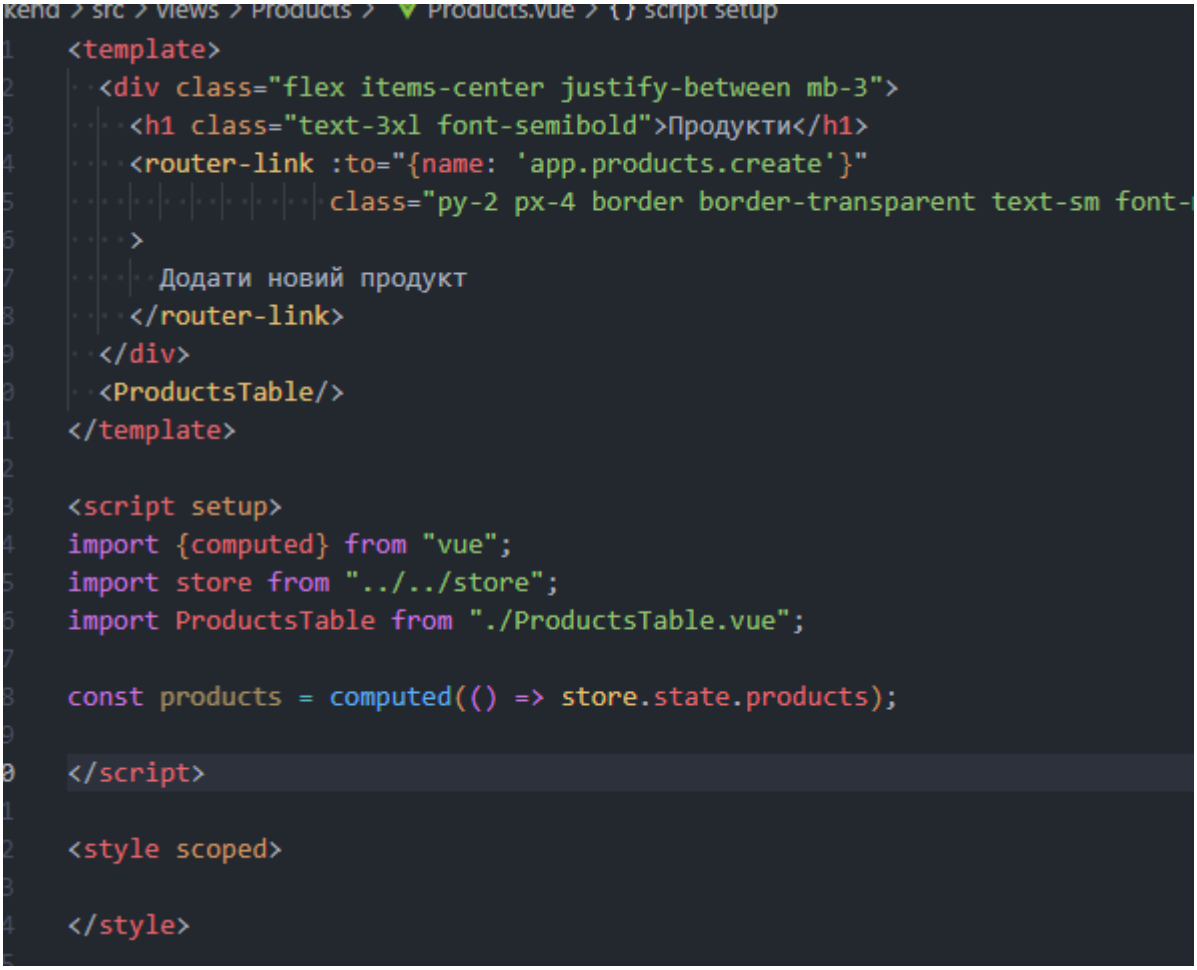
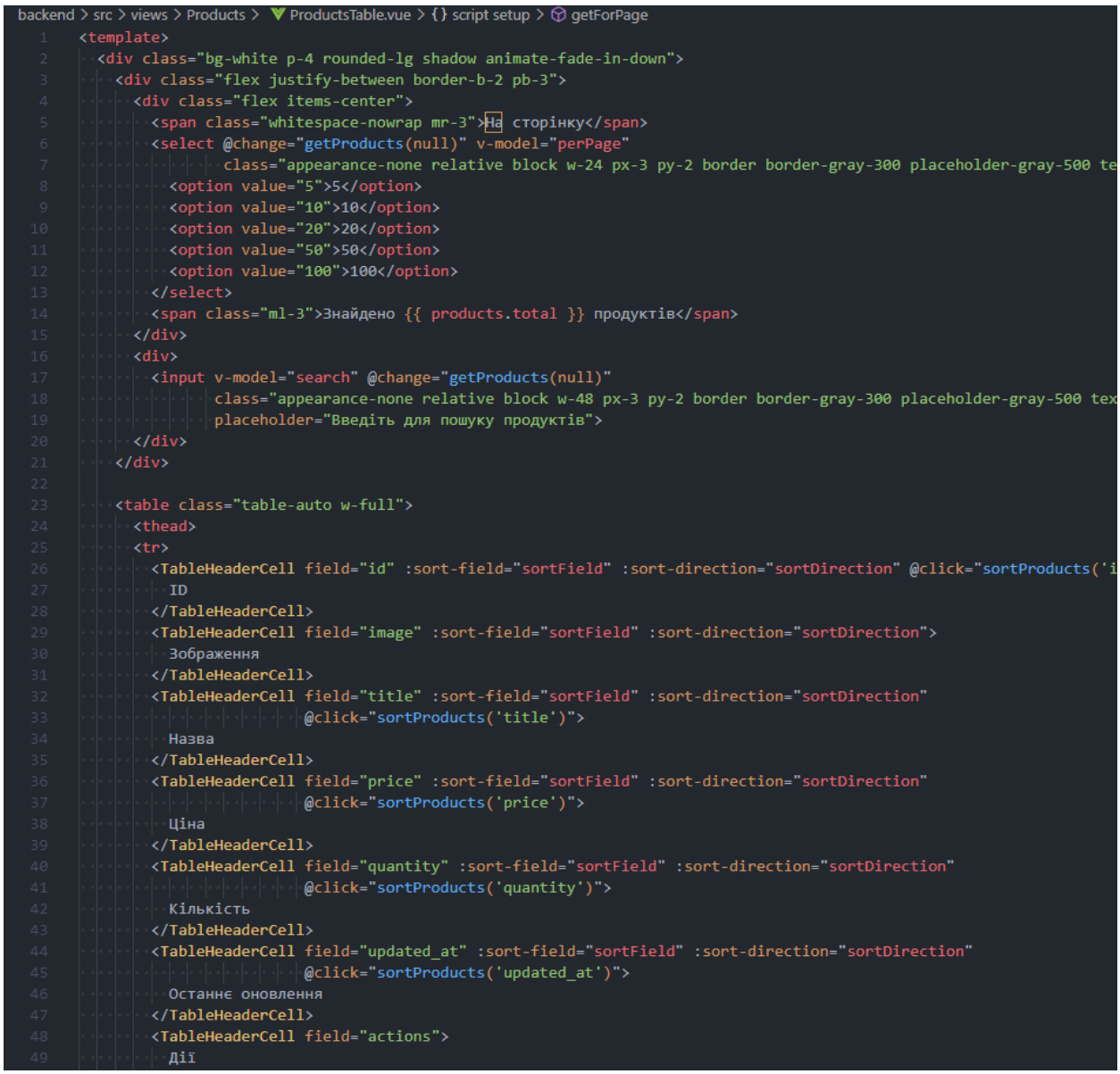


Рисунок 3.4 – Компонент Products

Цей компонент забезпечує основний інтерфейс для управління продуктами, включаючи їх відображення у вигляді таблиці та можливість додавання нових продуктів через перенаправлення на сторінку створення нового продукту. Використання магазину Vuex для управління станом дозволяє ефективно отримувати та оновлювати дані продуктів, забезпечуючи зручний та інтуїтивний інтерфейс для користувачів.

Компонент ProductsTable є частиною вебзастосунку для керування продуктами. Він надає функціонал для перегляду, сортування, пошуку, пагінації та видалення продуктів у зручному інтерфейсі (див. рис. 3.5).



```
backend > src > views > Products > ProductsTable.vue > {} script setup > getForPage
<template>
  <div class="bg-white p-4 rounded-lg shadow animate-fade-in-down">
    <div class="flex justify-between border-b-2 pb-3">
      <div class="flex items-center">
        <span class="whitespace-nowrap mr-3">На сторінку</span>
        <select @change="getProducts(null)" v-model="perPage"
                class="appearance-none relative block w-24 px-3 py-2 border border-gray-300 placeholder-gray-500 te
          <option value="5">5</option>
          <option value="10">10</option>
          <option value="20">20</option>
          <option value="50">50</option>
          <option value="100">100</option>
        </select>
        <span class="ml-3">Знайдено {{ products.total }} продуктів</span>
      </div>
      <div>
        <input v-model="search" @change="getProducts(null)"
               class="appearance-none relative block w-48 px-3 py-2 border border-gray-300 placeholder-gray-500 tex
               placeholder="Введіть для пошуку продуктів">
      </div>
    </div>

    <table class="table-auto w-full">
      <thead>
      <tr>
        <TableHeaderCell field="id" :sort-field="sortField" :sort-direction="sortDirection" @click="sortProducts('i
          ID
        </TableHeaderCell>
        <TableHeaderCell field="image" :sort-field="sortField" :sort-direction="sortDirection">
          Зображення
        </TableHeaderCell>
        <TableHeaderCell field="title" :sort-field="sortField" :sort-direction="sortDirection"
                         @click="sortProducts('title')">
          Назва
        </TableHeaderCell>
        <TableHeaderCell field="price" :sort-field="sortField" :sort-direction="sortDirection"
                         @click="sortProducts('price')">
          Ціна
        </TableHeaderCell>
        <TableHeaderCell field="quantity" :sort-field="sortField" :sort-direction="sortDirection"
                         @click="sortProducts('quantity')">
          Кількість
        </TableHeaderCell>
        <TableHeaderCell field="updated_at" :sort-field="sortField" :sort-direction="sortDirection"
                         @click="sortProducts('updated_at')">
          Останнє оновлення
        </TableHeaderCell>
        <TableHeaderCell field="actions">
          Дії
```

Рисунок 3.5 – Компонент Products

Реактивні змінні, створені за допомогою ref(), включають perPage, search, products, sortField, sortDirection і product. Ці змінні забезпечують автоматичне оновлення інтерфейсу користувача при їх зміні.

Використання життєвого циклу onMounted дозволяє автоматично отримувати список продуктів при завантаженні компонента.

Функції getProducts(url), sortProducts(field) і deleteProduct(product) взаємодіють з глобальним станом за допомогою store.dispatch, щоб отримувати, сортувати та видаляти продукти. Функція getForPage(ev, link) використовується для отримання продуктів за певною сторінкою під час пагінації.

Список продуктів відображається за допомогою директиви v-for, яка ітерується по кожному продукту із списку, а також містить розмітку для пагінації, що включає інформацію про загальну кількість продуктів.

Кожний продукт має свій власний компонент Menu, який містить кнопки дій «Редагувати» і «Видалити». Ці кнопки взаємодіють з відповідними методами editProduct і deleteProduct.

Цей компонент забезпечує повний функціонал управління та відображення списку продуктів у вебдодатку, забезпечуючи користувачам зручний інтерфейс для взаємодії з даними.

Компонент Login створює сторінку для входу в особистий кабінет користувача. Основні компоненти та функціонал представлені таким чином (див. рис. 3.6, 3.7).

```
const user = {
  email: '',
  password: '',
  remember: false
}

function login() {
  loading.value = true;
  store.dispatch('login', user)
    .then(() => {
      loading.value = false;
      router.push({name: 'app.dashboard'})
    })
    .catch(({response}) => {
      loading.value = false;
      errorMsg.value = response.data.message;
    })
}
```

Рисунок 3.6 – Компонент Login

```
<input type="hidden" name="remember" value="true"/>
<div class="rounded-md shadow-sm -space-y-px">
  <div>
    <label for="email-address" class="sr-only">Email адрес</label>
    <input id="email-address" name="email" type="email" autocomplete="email" required="" v-model="user.email"
           class="appearance-none rounded-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-t-md focus:outline-none focu
           placeholder="Email адрес"/>
  </div>
  <div>
    <label for="password" class="sr-only">Пароль</label>
    <input id="password" name="password" type="password" autocomplete="current-password" required=""
           v-model="user.password"
           class="appearance-none rounded-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-b-md focus:outline-none focu
           placeholder="Пароль"/>
  </div>
</div>

<div class="flex items-center justify-between">
  <div class="flex items-center">
    <input id="remember-me" name="remember-me" type="checkbox" v-model="user.remember"
           class="h-4 w-4 text-indigo-600 focus:ring-indigo-500 border-gray-300 rounded"/>
    <label for="remember-me" class="ml-2 block text-sm text-gray-900"> Запам'ятати мене </label>
  </div>

  <div class="text-sm">
    <router-link :to="{name: 'requestPassword'}" class="font-medium text-indigo-600 hover:text-indigo-500"> Забули
      пароль?
    </router-link>
  </div>
</div>
```

Рисунок 3.7 – Шаблон компоненту Login

GuestLayout: це компонент розмітки для гостьової сторінки, який приймає параметр title для відображення заголовку сторінки. Використовується для обгортання основного вмісту сторінки, щоб забезпечити однорідний зовнішній вигляд.

Form: форма для входу користувача, яка відправляє дані методом POST із використанням @submit.prevent для перехоплення події відправки форми.

Поле помилок (errorMsg): виводиться під формою у випадку, якщо виникає помилка під час входу. Використовується для відображення повідомлення про помилку, яке приходить з сервера.

Реактивні дані:

– user: об’єкт, що містить введені користувачем дані для email, password і remember-me checkbox;
– loading: реактивна змінна, яка використовується для відображення анімації завантаження під час відправки форми;
– errorMsg: реактивна змінна для зберігання тексту помилки під час спроби входу.

Метод login(): викликається при натисканні кнопки «Увійти». Встановлює loading в true, щоб активувати анімацію завантаження. Викликає дію login через store.dispatch, яка намагається виконати вхід, передаючи дані користувача. Якщо вхід успішний, перенаправляє користувача на сторінку app.dashboard. У випадку помилки встановлює loading в false і зберігає повідомлення про помилку у errorMsg.

Кнопка «Запам’ятати мене2 і посилання «Забули пароль?»: надають можливість користувачам вибирати, чи хочуть вони, щоб їхні дані входу залишалися в системі («Запам’ятати мене»). Посилання «Забули пароль?» веде на сторінку для відновлення пароля.

Фонове зображення: використовується фонове зображення для створення атмосфери або ідентифікації бренду на гостровій сторінці входу.

Цей компонент створений для забезпечення простого, але функціонального інтерфейсу для входу користувача з можливістю відправки форми, управління станом завантаження та обробки помилок.

3.3 Розробка серверної частини

Серверна частина відповідає за обробку запитів від клієнтів, управління даними та забезпечення бізнес-логіки застосунку. Вона взаємодіє з базою даних для зберігання та отримання інформації та забезпечує відповідну обробку запитів, таких як реєстрація користувачів, управління товарами, обробка замовлень тощо.

Для реалізації серверної частини використовується Laravel – популярний PHP-фреймворк, який надає зручний інструментарій для розробки вебзастосунків. Laravel забезпечує MVC-архітектуру, що сприяє розділенню бізнес-логіки, відображення та обробки даних [7].

Клас ProductFactory використовується для генерації тестових даних моделі Product у Laravel, використовуючи механізм фабрик даних (Data Factories) (див. рис. 3.8).

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

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Product>
 */
class ProductFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition()
    {
        return [
            'title' => fake()->text(),
            'image' => fake()->imageUrl(),
            'description' => fake()->realText(2000),
            'price' => fake()->randomFloat(2, 2, 5),
            'created_at' => now(),
            'updated_at' => now(),
            'created_by' => 1,
            'updated_by' => 1,
        ];
    }
}
```

Рисунок 3.8 – Фабрика тестування ProductFactory

Детальне пояснення функціоналу нашого класу:

- Namespace: клас ProductFactory знаходиться в просторі імен Database\Factories;
- клас ProductFactory: цей клас розширює базовий клас Factory, який надає Laravel для створення фабрик моделей;
- метод definition(): цей метод визначає стандартний стан моделі, яка буде створюватись фабрикою (він повертає масив з визначеними атрибутами моделі Product);
- ‘title’ => $this->faker->text(): генерує випадковий текст для заголовка продукту за допомогою об’єкта $this->faker (faker використовується для створення фіктивних даних в тестових середовищах);
- ‘image’ => $this->faker->imageUrl(): генерує випадкове URL зображення за допомогою Faker;
- ‘description’ => $this->faker->realText(2000): генерує випадковий текст опису продукту довжиною до 2000 символів;
- ‘price’ => $this->faker->randomFloat(2, 2, 5): генерує випадкове дійсне число (з плаваючою комою) в діапазоні від 2 до 5 з двома знаками після коми;
- ‘created_at’ та ‘updated_at’: встановлюють поточну дату та час для полів created_at і updated_at в базі даних;
- ‘created_by’ та ‘updated_by’: встановлюють ідентифікатор користувача, який створив і оновив запис (у цьому випадку завжди встановлено на значення 1, але у реальному додатку вони можуть бути встановлені залежно від авторизованого користувача).

Фабрики в Laravel використовуються для створення тестових даних під час розробки та тестування. Наприклад, ми можемо використовувати цю фабрику для створення декількох фейкових записів продуктів для наповнення вашої бази даних під час тестування або розробки (див. рис. 3.9).

Клас UserFactory, який використовується для генерації тестових даних моделі User у Laravel за допомогою фабрики даних (Data Factory).

```php
class UserFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition()
    {
        return [
            'name' => fake()->name(),
            'email' => fake()->safeEmail(),
            'email_verified_at' => now(),
            'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password
            'remember_token' => Str::random(10),
        ];
    }
```

Рисунок 3.9 – Фабрика тестування UserFactory

Детальне пояснення функціоналу нашого класу:

– Namespace: клас UserFactory знаходиться в просторі імен Database\Factories;
– клас UserFactory: цей клас розширює базовий клас Factory, який надає Laravel для створення фабрик моделей;
– метод definition(): цей метод визначає стандартний стан моделі, яка буде створюватись фабрикою (він повертає масив з визначеними атрибутами моделі User);
– ‘name’ => $this->faker->name(): генерує випадкове ім’я за допомогою об’єкта $this->faker (faker використовується для створення фіктивних даних, таких як імена, електронні адреси тощо);
– ‘email’ => $this->faker->safeEmail(): генерує випадкову безпечну (неправильну) електронну адресу (це забезпечує генерацію валідних електронних адрес, які можна використовувати у тестових цілях);
– ‘email_verified_at’ => now(): встановлює поточну дату і час для підтвердження електронної адреси (у цьому випадку завжди встановлено поточний момент часу);
– ‘password’=>‘$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi’: зафіксований пароль, який представляє хешований пароль ‘password’ (у реальному застосунку варто генерувати безпечні

випадкові паролі, а не використовувати фіксовані значення);

– ‘remember_token’ => Str::random(10): генерує випадковий токен для «запам’ятати мене» (використовується для аутентифікації на основі токенів у Laravel);
– метод unverified(): цей метод створює стан, де адреса електронної пошти користувача не підтверджена (атрибут email_verified_at встановлюється на null) (використовується для створення тестових даних, коли потрібно тестувати різні стани підтвердження електронної адреси).

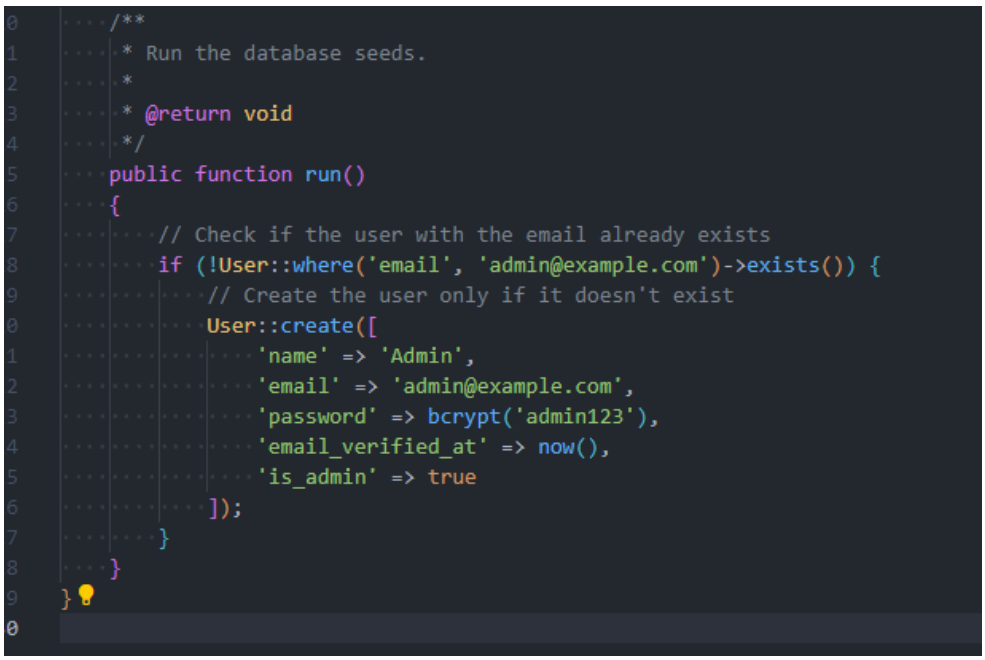
Клас CreateLaravelSessionsTable представляє собою міграцію Laravel для створення таблиці laravel_sessions в базі даних (див. рис. 3.10).

```php
class CreateLaravelSessionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('laravel.sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->constrained();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->text('payload');
            $table->integer('last_activity');
        });
    }
```

Рисунок 3.10 – Міграції для створення таблиці в БД

Цей код міграції дозволяє Laravel автоматично створити таблицю laravel_sessions у вашій базі даних при виконанні команди міграції. Такий підхід є частиною системи міграцій Laravel, яка забезпечує контроль версій структури бази даних та спрощує її синхронізацію між розробниками.

Клас AdminUserSeeder реалізує сідер (Seeder) для створення адміністративного користувача в базі даних за допомогою фреймворку Laravel (див. рис. 3.11).



```php
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // Check if the user with the email already exists
        if (!User::where('email', 'admin@example.com')->exists()) {
            // Create the user only if it doesn't exist
            User::create([
                'name' => 'Admin',
                'email' => 'admin@example.com',
                'password' => bcrypt('admin123'),
                'email_verified_at' => now(),
                'is_admin' => true
            ]);
        }
    }
}
```

Рисунок 3.11 – Сідер (Seeder) для створення адміністративногокористувача

3.4 Тестування вебзастосунку

Тестування вебдодатків є важливою частиною розробки для переконання в правильності роботи програмного забезпечення перед виходом його в продакшн. Вебдодатки зазвичай тестуються на різних рівнях, включаючи модульне тестування, інтеграційне тестування та енд-ту-енд (E2E) тестування. Давайте розглянемо кожен з цих видів тестування для вебдодатків на прикладі PHP-фреймворка Laravel.

Модульне тестування є одним з основних видів тестування програмного забезпечення і використовується для перевірки коректності окремих модулів або компонентів програми. Основна ідея полягає в тому, щоб тестувати кожний окремий «модуль» або функцію програми, щоб переконатися, що він працює правильно в ізоляції від інших частин системи (див. рис. 3.12).

Основні аспекти модульного тестування:

– ізоляція компонентів: модульні тести зазвичай тестуються в ізоляції від інших компонентів програми – це означає, що кожен тест оцінює лише одну конкретну функціональність або модуль, не взаємодіючи

з реальною базою даних, зовнішніми API або іншими системами;

– автоматизація: тести пишуться таким чином, щоб їх можна було виконувати автоматично – це дозволяє швидко виявляти помилки та забезпечувати стабільність коду під час його змін;
– маленькі обсяги коду: кожен модульний тест зазвичай охоплює невеликий обсяг коду, що робить їх легкими для написання, розуміння та підтримки;
– загальна покриття: модульне тестування допомагає досягти високого рівня покриття коду тестами, оскільки кожна функція або модуль може бути випробувано окремо;
– швидкість виконання: тести в ізоляції зазвичай виконуються швидше, оскільки вони не залежать від зовнішніх ресурсів або інфраструктури.

```php
namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

class ExampleTest extends TestCase
{
    /**
     * A basic test example.
     *
     * @return void
     */
    public function test_the_application_returns_a_successful_response()
    {
        $response = $this->get('/');

        $response->assertStatus(200);
    }
}
```

Рисунок 3.12 – Модульне тестування

Компоненти модульного тестування:

– фікси та асерти: використовуються для створення тестових умов і перевірки очікуваних результатів;
– виготовлення даних: функції, що створюють тестові дані або ініціалізують стан системи перед виконанням тесту;

– спостереження за подіями: деякі тести можуть перевіряти, чи спрацювали певні події або функції, викликані в процесі тестування.

Переваги модульного тестування:

– раннє виявлення помилок: допомагає виявити та виправити помилки на ранніх етапах розробки, що знижує витрати на усунення дефектів пізніше;
– забезпечення стабільності: дозволяє переконатися, що навіть після внесення змін в код існуючі функції працюють коректно;
– підтримка коду: тести стають частиною документації коду і допомагають розробникам швидко розуміти, як повинен вести себе певний модуль або функція;
– модульне тестування є ключовим елементом підтримки якості програмного забезпечення і допомагає забезпечити стабільність та надійність програм.

Інтеграційне тестування – це вид тестування програмного забезпечення, який спрямований на перевірку взаємодії між різними компонентами системи [8]. Основна мета інтеграційного тестування полягає в тому, щоб переконатися, що окремі компоненти програми правильно працюють разом, а їх взаємодія не порушує функціональність системи в цілому(див. рис. 3.13).

Основні аспекти інтеграційного тестування:

– тести на рівні інтерфейсів: це тести, які перевіряють, як різні модулі або компоненти спілкуються між собою (наприклад, це може бути перевірка, що дані, що введені в форму користувачем, правильно передаються і обробляються базою даних);
– тести інтеграції сервісів: ці тести перевіряють взаємодію між різними сервісами, які можуть бути розгорнуті на різних серверах або в середовищах;
– тести інтеграції з базою даних: вони використовуються для перевірки, як програма взаємодіє з реальною або тестовою базою даних (ці тести переконуються, що запити до бази даних виконуються коректно та ефективно);

– тести інтеграції API: перевіряють, як API взаємодіє з клієнтськими програмами або іншими сервісами через HTTP-запити та відповіді.

```php
use App\Models\User;
use App\Providers\RouteServiceProvider;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

class AuthenticationTest extends TestCase
{
    use RefreshDatabase;

    public function test_login_screen_can_be_rendered()
    {
        $response = $this->get('/login');

        $response->assertStatus(200);
    }

    public function test_users_can_authenticate_using_the_login_screen()
    {
        $user = User::factory()->create();

        $response = $this->post('/login', [
            'email' => $user->email,
            'password' => 'password',
        ]);

        $this->assertAuthenticated();
        $response->assertRedirect(RouteServiceProvider::HOME);
    }

    public function test_users_can_not_authenticate_with_invalid_password()
    {
        $user = User::factory()->create();

        $this->post('/login', [
            'email' => $user->email,
            'password' => 'wrong-password',
        ]);

        $this->assertGuest();
    }
}
```

Рисунок 3.13 – Інтеграційне тестування

Ключові переваги інтеграційного тестування:

– виявлення помилок інтеграції: допомагає виявити помилки, які можуть виникати при взаємодії різних компонентів системи, наприклад, неправильна передача даних або невідповідність форматів;

– підтвердження стабільності системи: забезпечує, що після внесення змін в код чи конфігурацію система залишається стабільною і працездатною;

– зменшення ризику помилок в продакшені: знижує ризик виникнення непередбачених проблем в реальному середовищі завдяки передбачуваній реакції системи на різні сценарії взаємодії.

Інструменти для інтеграційного тестування:

– фреймворки для тестування: наприклад, PHPUnit для PHP, Jest для JavaScript, PyTest для Python і т. д;

– засоби автоматизації: як правило, інтеграційні тести запускаються автоматично, щоб забезпечити швидкість виконання та повторюваність результатів;

– моки та стаби: іноді використовуються для симуляції частин системи, що ще не готові або щоб ізолювати тести від зовнішніх залежностей.

Інтеграційне тестування є важливою частиною процесу розробки програмного забезпечення, оскільки воно допомагає забезпечити, що всі компоненти системи працюють разом із заданим функціоналом та не викликають непередбачуваних проблем в реальних умовах експлуатації.

На рисунку 3.14 зображена головна сторінка вебзастосунку, яка є початковою точкою для користувачів онлайн-магазину побутової техніки.

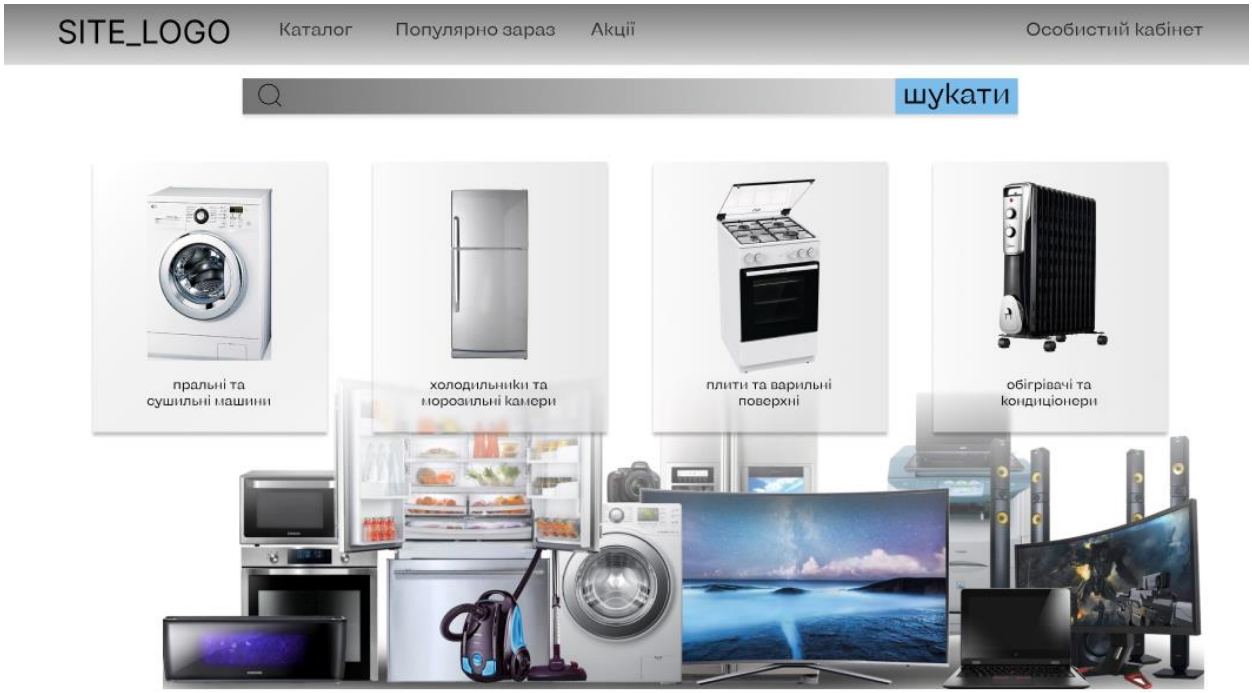


Рисунок 3.14 – Головна сторінка

Інтерфейс цієї сторінки реалізований за допомогою фреймворків Vue та Laravel. Основні елементи сторінки включають логотип, навігаційне меню, пошуковий рядок та категорії товарів. Розглянемо детально основні компоненти.

Логотип та навігація:

– у верхньому лівому куті розташований логотип сайту “SITE_LOGO”;
– основне навігаційне меню містить посилання на «Каталог», «Популярно зараз», «Акції» та «Особистий кабінет». Ці посилання дозволяють користувачам швидко переміщатися між різними розділами сайту.

Пошуковий рядок: у центрі верхньої частини сторінки розташований пошуковий рядок з кнопкою «шукати», яка дозволяє користувачам швидко знайти необхідний товар.

Категорії товарів: нижче пошукового рядка розміщені основні категорії товарів у вигляді великих зображень з підписами. Це такі категорії як «пральні та сушильні машини», «холодильники та морозильні камери», «плити та варильні поверхні», «обігрівачі та кондиціонери».

На рисунку 3.15 зображена сторінка результатів пошуку або розділу каталогу.

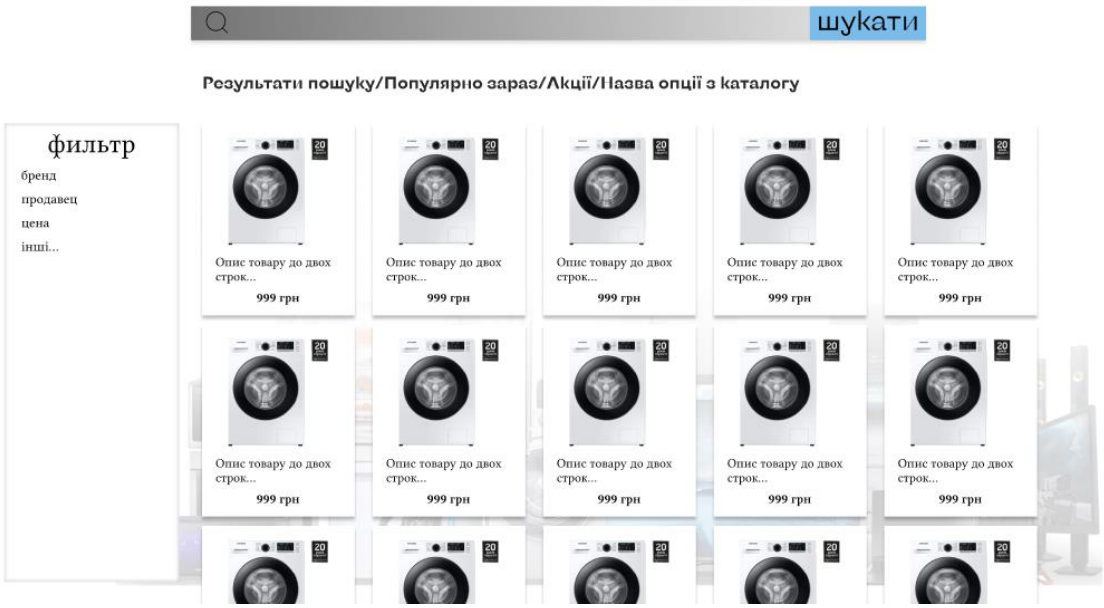


Рисунок 3.15 – Результат пошуку

Ця сторінка дозволяє користувачам переглядати товари за певними критеріями або категоріями.

Фільтри пошуку: ліворуч розташовані фільтри, які включають такі параметри як «бренд», «продавец», «цена» та «інші...». Це дозволяє користувачам звузити пошук та знайти найбільш відповідні товари.

Список товарів: основна частина сторінки заповнена карточками товарів. Кожна карточка містить зображення товару, короткий опис та ціну. Наприклад, на сторінці відображені пральні машини з ціною «999 грн».

На рисунку 3.16 зображена сторінка авторизації в особистий кабінет. Ця сторінка надає користувачам можливість увійти в свій аккаунт або зареєструвати новий.

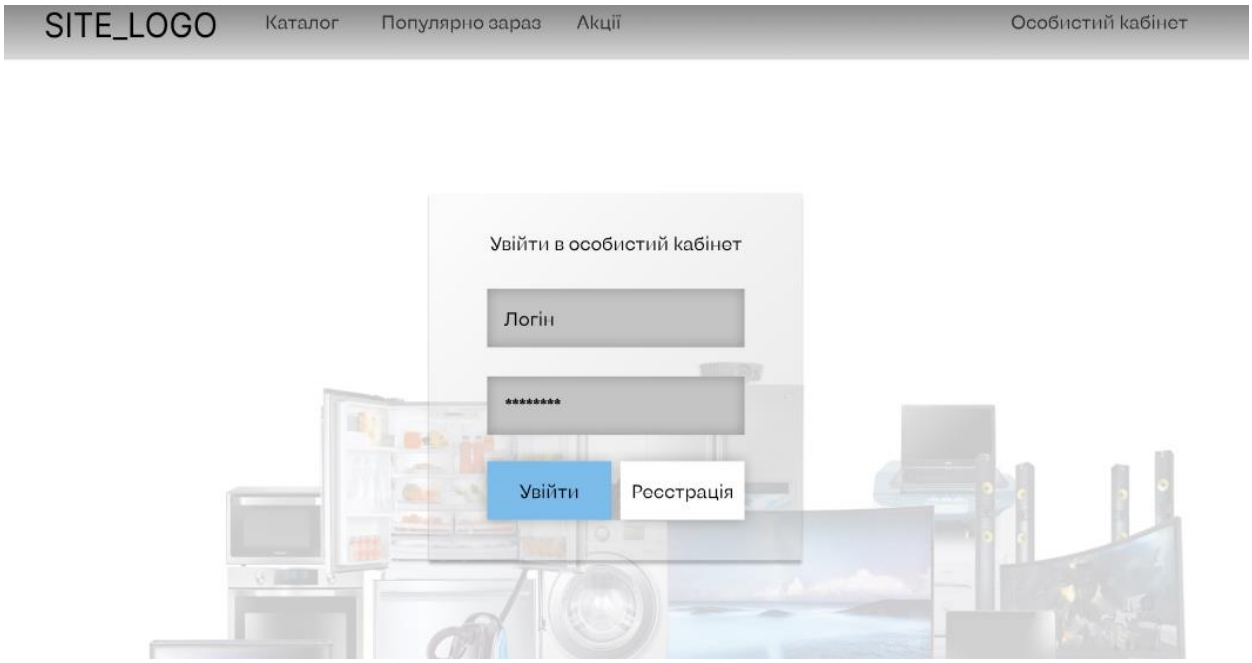


Рисунок 3.16 – Авторизація в особистому кабінеті

У центрі сторінки розташована форма з полями «Логін» та «Пароль». Під формою розміщені кнопки «Увійти» та «Реєстрація», які дозволяють користувачам увійти до свого особистого кабінету або створити новий аккаунт.

3.5 Висновки до розділу 3

У третьому розділі дослідження ми представили детальний опис програмної реалізації кожного етапу нашого проєкту. Це охоплювало розробку як серверної, так і клієнтської частини, а також докладний аналіз архітектурних рішень для обох компонентів системи. Ми систематично розглянули кожен етап реалізації, пояснили його деталі та обґрунтували вибір конкретних рішень. Представлені приклади ілюстрували практичне застосування кожного етапу та виокремлювали ключові аспекти нашого програмного продукту.

Крім того, ми розробили і реалізували систему CI/CD для автоматизованого тестування серверної частини, що значно сприяло забезпеченню якості нашого продукту. В розділі ми також надали знімки екрану реалізованого інтерфейсу, таблиці з прикладами запитів до сервера та API-маршрутів, що дозволило детальніше зрозуміти та оцінити функціональні можливості нашого програмного забезпечення.

ВИСНОВКИ

У рамках розробки вебзастосунку для онлайн-магазину побутової техніки з використанням фреймворків Vue та Laravel було проведено комплексне вивчення вимог та архітектурних рішень на кожному етапі проєкту. Високий рівень деталізації аналізу дозволив успішно реалізувати всі функціональні та технічні вимоги, що були поставлені перед командою розробників.

Додатково було успішно інтегровано адміністративну панель для зручного управління контентом та користувачами системи.

Ретельний аналіз технічного завдання та вибір оптимального технологічного стеку, зокрема Vue для клієнтської частини та Laravel для серверної, сприяв ефективній реалізації ключових аспектів, включаючи валідацію даних, управління ролями користувачів і базові операції з даними (CRUD).

Застосування передових практик у розробці, таких як CI/CD для автоматизованого тестування серверної частини, забезпечило високу якість програмного забезпечення. Практичні приклади реалізації ілюструють не лише теоретичні знання, а й їхнє практичне застосування у вирішенні конкретних завдань проєкту.

Загальний висновок підкріплює те, що створений вебзастосунок відповідає усім вимогам, поставленим на етапі аналізу, і є функціональним продуктом, готовим задовольнити потреби користувачів і забезпечити їхнє задоволення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційна документація Vue.js. URL: https://vuejs.org/ (дата звернення: 10.02.2024).
2. Офіційна документація Laravel. URL: https://laravel.com/docs/ (дата звернення: 20.02.2024).
3. Поняття сутності, атрибута, ключа, зв'язку – UA5.org. URL: https://ua5.org/database/1844-ponyattya-sutnosti-atrybuta-klyucha-zvyazku.html (дата звернення: 08.03.2024).
4. Vuex для керування станом додатка в Vue.js. URL: https://vuex.vuejs.org/ (дата звернення: 15.03.2024).
5. MySQL – офіційна документація. URL: https://dev.mysql.com/doc/ (дата звернення: 01.03.2024).
6. Розширені можливості Laravel: робота зі зберіганням сеансів (Sessions). URL: https://laravel.com/docs/session (дата звернення: 15.04.2024).
7. Модель-вид-контролер – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80 (дата звернення: 18.04.2024).
8. Інтеграційне тестування. URL: https://qalight.ua/baza-znaniy/integratsijne-testuvannya/ (дата звернення: 20.04.2024).