МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ ЕЛЕКТРОНІКИ ЗАСОБАМИ VUE.JS ТА LARAVEL»**

Виконав: студент 4 курсу, групи 6.1210-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
А.М. Данілов
(ініціали та прізвище)
Керівник доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 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 Антону Миколайовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет магазину електроніки засобами Vue.js та Laravel

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

затверджені наказом ЗНУ від « 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 розробка інтернет-магазину.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 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.	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 _____ А.М. Данилов
(підпис) (ініціали та прізвище)

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

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет магазину електроніки засобами Vue.js та Laravel»: 64 с., 22 рис., 20 джерел.

БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, КОМПОНЕНТ, КОНТРОЛЕР, МОДЕЛЬ, ІНТЕРНЕТ-МАГАЗИН, ФРЕЙМВОРК, LARAVEL, VUE.JS.

Об’єкт дослідження – процес розробки інтернет-магазину.

Мета роботи: розробка інтернет-магазину.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи програмної інженерії.

Предмет дослідження – розробка інтернет-магазину із застосуванням фреймворків Laravel та Vue.js.

У роботі наведено етапи розробки інтернет-магазину із застосуванням фреймворків Laravel та Vue.js, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення. Описано предметну область. Розроблено інтернет-магазин із застосуванням фреймворків Laravel та Vue.js, який використовуються для реалізації онлайн продажів та комунікації з клієнтами в реальному часі. Зростання попиту на інтерактивні онлайн сервіси робить важливим розв’язок таких задач, розгортання та підтримки інтернет-магазинів. Тому, розробка інтернет-магазину із застосуванням фреймворків Laravel та Vue.js є актуальною задачею.

SUMMARY

Bachelor’s qualifying paper “Development of an Online Electronics Store Using Vue.js and Laravel”: 64 pages, 22 figures, 20 references.

DATABASE, WEB APPLICATION, COMPONENT, CONTROLLER, MODEL, ONLINE STORE, FRAMEWORK, LARAVEL, VUE.JS.

Object of research: the process of developing an online store. Objective: development of an online store.

Research methods: object-oriented programming methods, software engineering methods.

Subject of research: development of an online store using Laravel and Vue.js frameworks.

The thesis presents the stages of developing an online store using Laravel and Vue.js frameworks, identifies key issues, and applies new methods for collecting and analyzing software requirements. The subject area is described. An online store was developed using Laravel and Vue.js frameworks, which is used for implementing online sales and real-time customer communication. The increasing demand for interactive online services makes solving such tasks, as well as deploying and maintaining online stores, important. Therefore, the development of an online store using Laravel and Vue.js frameworks is a relevant task.

ЗМІСТ

ВСТУП

Розвиток технологій значно трансформує різні аспекти нашого життя, включаючи способи комунікації та надання послуг через Інтернет. У сучасному цифровому світі важливо, щоб вебсайти пропонували ефективні та зручні інструменти для онлайн продажів. Сучасні споживачі очікують миттєвої взаємодії та швидких відповідей на свої запити. Це особливо актуально для бізнесів, які прагнуть забезпечити високий рівень обслуговування клієнтів та покращити користувацький досвід на своїх вебресурсах.

Вебсайти сьогодні є ключовою складовою будь-якого бізнесу, слугуючи основним засобом комунікації з клієнтами та відвідувачами. В умовах посиленої конкуренції компаніям необхідні рішення, які дозволяють реальну взаємодію з клієнтами в режимі реального часу, відповідаючи на їхні запитання та надаючи необхідну допомогу. Використання інтерактивних інтернет-магазинів дозволяє досягти цього, забезпечуючи миттєве оновлення вмісту та передачу інформації.

Цей проєкт присвячений розробці інтернет-магазину з використанням потужних фреймворків Laravel та Vue.js. Метою проєкту є створення ефективного інструменту для швидкої та плавної взаємодії з відвідувачами вебсайтів. Laravel та Vue.js забезпечують високу продуктивність, гнучкість та інтерактивність, що значно покращує користувацький досвід, оскільки забезпечує швидку передачу повідомлень та оновлення контенту без необхідності перезавантаження сторінки.

В рамках цього проєкту будуть розроблені необхідні компоненти та інтерфейс для інтернет-магазину. Використання Laravel та Vue.js дозволить забезпечити швидке і ефективне передавання даних між користувачем і сервером. Крім того, будуть розглянуті можливості зберігання даних та інтеграції з базами даних для забезпечення зручного керування товарами та

замовленнями. Використання Laravel забезпечить надійну серверну частину, а Vue.js дозволить створити інтуїтивно зрозумілий та інтерактивний інтерфейс.

Особливу увагу буде приділено питанням безпеки та захисту даних, оскільки інтернет-магазини часто включають обмін конфіденційною інформацією. Впровадження сучасних методів шифрування та аутентифікації допоможе забезпечити безпечне середовище для користувачів.

Цей проєкт має велике значення, оскільки він дозволяє покращити якість обслуговування клієнтів та відвідувачів вебсайту, забезпечуючи швидку та ефективну комунікацію. Використання Laravel та Vue.js дозволяє створити інтерактивне середовище, яке задовольняє потреби користувачів та сприяє підвищенню їх задоволення від взаємодії з вебсайтом. Запровадження такого рішення також може сприяти підвищенню лояльності клієнтів, збільшенню рівня їх задоволення та покращенню загального іміджу компанії.

Таким чином, розробка інтернет-магазину на базі фреймворків Laravel та Vue.js є важливим кроком у напрямку створення ефективних та інноваційних вебзастосунків, що відповідають сучасним вимогам ринку та потребам користувачів.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНІВ

У цьому розділі розглянемо теоретичні основи, пов’язані з розробкою інтернет-магазину, а також заплануємо основні етапи проєкту. Особливу увагу буде приділено історії розвитку технологій, що використовуються в інтернет-магазинах, принципам роботи з базами даних, а також фреймворкам Laravel та Vue.js. Додатково проведемо порівняння з іншими технологіями та фреймворками, які можуть бути використані для реалізації подібних проєктів.

1.1 Огляд сучасних технологій для розробки вебзастосунків

У сучасному світі вебзастосунки стали невід’ємною частиною нашого повсякденного життя, забезпечуючи нам доступ до різноманітних послуг та можливість взаємодії через Інтернет. Постійний розвиток технологій призвів до появи широкого спектру інструментів та платформ для розробки вебзастосунків, які спрощують та прискорюють процес створення.

Однією з провідних технологій у цьому напрямку є фреймворк Laravel. Laravel – це високорівневий фреймворк для роботи з мовою програмування PHP, який надає розробникам гнучкість та зручність у створенні вебзастосунків. Завдяки своїй чистій та елегантній синтаксису, а також широкому спектру вбудованих функцій, Laravel дозволяє швидко та ефективно розробляти як прості, так і складні вебпроєкти [1].

Ще однією важливою технологією є Vue.js – прогресивний JavaScript-фреймворк, який використовується для побудови інтерфейсу користувача вебзастосунків. Vue.js відомий своєю простотою в використанні, швидкодією та гнучкістю. Він дозволяє створювати динамічні та інтерактивні вебінтерфейси без зайвого складності [3].

Подібно до Laravel та Vue.js, існують інші фреймворки та технології, які

також варто розглянути при розробці вебзастосунків. Наприклад, React – інший популярний JavaScript-фреймворк, який використовується для побудови інтерфейсів користувача. Також важливо згадати про Node.js, який дозволяє створювати серверну частину вебзастосунків з використанням JavaScript.

При аналізі вимог та потреб проєкту для вибору найбільш підходящих технологій слід враховувати різноманітні аспекти, які можуть вплинути на цей процес. Одним з ключових аспектів є потреби щодо масштабованості проєкту, його можливої інтеграції з іншими системами та платформами, а також очікуваний обсяг трафіку та навантаження на серверну інфраструктуру [8].

Наведемо приклади деякі з них:

- **функціональні вимоги проєкту –** різні технології можуть мати різний набір функцій та можливостей, перед вибором технологій слід уважно вивчити, які саме функції необхідні для реалізації проєкту та які технології можуть найбільш ефективно це зробити;
- **масштаб проєкту –** розмір та масштаб проєкту можуть вплинути на вибір технологій, наприклад, для великих проєктів можуть підходити рішення, які забезпечують високу масштабованість та продуктивність.
- **команда розробників –** наявність та кваліфікація розробницької команди також може вплинути на вибір технологій, якщо команда має досвід у певних технологіях або мовах програмування, це може бути важливим чинником для вибору;
- **майбутні потреби проєкту –** важливо також розглянути майбутні потреби проєкту та його можливу еволюцію, обрані технології повинні бути гнучкими та здатними адаптуватися до змін у вимогах проєкту з плином часу;
- **екосистема та спільнота розробників –** підтримка та активність у відповідних розробницьких спільнотах може бути важливою для успішного використання технологій у проєкті, наявність розширень,

документації та спільноти розробників може значно полегшити процес розробки та підтримки;

– **вартість розробки та підтримки –** витрати на розробку та підтримку проєкту також можуть вплинути на вибір технологій, деякі технології можуть бути більш витратними у використанні або підтримці, ніж інші.

Загалом, вибір технологій для проєкту є складним та багатогранним процесом, який потребує уважного аналізу різноманітних факторів. Якщо правильно підібрати технології з урахуванням всіх цих аспектів, це може значно сприяти успішності та ефективності проєкту [5].

1.2 Фреймворк Laravel: основні можливості та архітектура

Laravel – це безкоштовний PHP-фреймворк з відкритим кодом, який використовується для створення вебдодатків. Він заснований на шаблоні Model-View-Controller (MVC) і пропонує широкий спектр функцій, які роблять процес розробки більш швидким, простим і зручним [9].

Фреймворк Laravel залишається одним з найпопулярніших PHP-фреймворків завдяки своїй гнучкості, великій спільноті та регулярним оновленням. Він пропонує розробникам зручні інструменти для аутентифікації, маршрутизації, роботи з базами даних та багато іншого, що робить процес розробки швидшим та ефективнішим. Laravel регулярно попадає в списки популярних PHP-фреймворків і часто обирається для корпоративних проєктів та як основа для високоякісних вебдодатків [10].

Сьогодні на Laravel створюють різноманітні проєкти:

– інтернет-магазини;
– сайти-блоги;
– соціальні мережі;
– корпоративні вебсайти.

Тобто ключові переваги використання Laravel – це в першу чергу універсальність та гнучкість, можливість використання фреймворку для реалізації проєктів різної складності. А враховуючи, що фреймворк ще й безкоштовний, його дійсно можна рекомендувати більшості веброзробників, власників стартапів та бізнес-проєктів. Однак є й інші переваги Laravel, про які варто сказати.

Система маршрутизації в Laravel є надзвичайно гнучкою та потужною. Ви можете визначати прості маршрути для обробки HTTP-запитів, а також більш складні маршрути з параметрами, обмеженнями та груповою маршрутизацією. Маршрути можна пов’язувати з контролерами, анонімними функціями або окремими класами. Laravel також підтримує RESTful маршрутизацію для створення API та додатків, які дотримуються принципів RESTful архітектури [4].

Eloquent ORM забезпечує зручний та зрозумілий спосіб взаємодії з базами даних за допомогою PHP-об’єктів. Він абстрагує деталі запитів SQL, дозволяючи працювати з даними на більш високому рівні абстракції. Eloquent пропонує потужні можливості, такі як композитні ключі, м’які видалення, обмеження цілісності, відношення між моделями та багато іншого. Крім того, він забезпечує гнучкий інструментарій для створення складних запитів, включаючи об’єднання, групування, сортування та фільтрацію даних.

Blade – це вбудований шаблонізатор Laravel, який забезпечує зручний та виразний синтаксис для створення динамічних сторінок. Він дозволяє легко вставляти PHP-код безпосередньо в HTML-розмітку, використовуючи спеціальні директиви Blade. Крім того, Blade підтримує розширення функціональності за допомогою компонентів, що робить його ідеальним рішенням для створення складних інтерфейсів користувача.

Міграції в Laravel забезпечують простий та ефективний спосіб керування змінами структури бази даних. Вони дозволяють створювати, оновлювати та видаляти таблиці, стовпці та індекси за допомогою зрозумілих команд PHP. Міграції зберігаються у вигляді файлів, що дозволяє

контролювати зміни в структурі бази даних за допомогою систем контролю версій. Це полегшує розгортання додатків у різних середовищах, забезпечуючи однакову структуру бази даних [6].

Вищезазначені ключові переваги фреймворку Laravel справді роблять його вкрай популярним та затребуваним серед веброзробників. Однак якщо вже ми говоримо про сильні сторони, то обов'язково слід поглянути й на недоліки. Адже вони також є, хоча й відносно небагато (див. рис. 1.1).

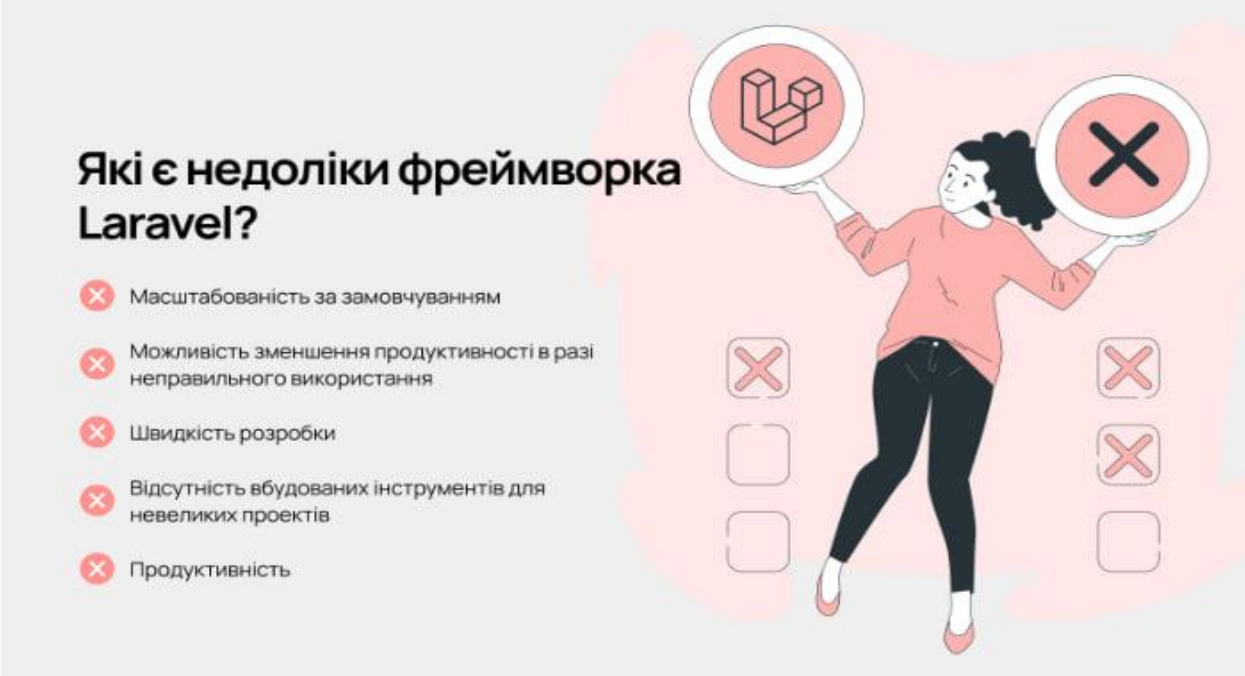

Рисунок 1.1 – Недоліки Laravel

1.3 Фреймворк Vue.js: основні можливості та архітектура

Історія Vue.js починається в 2013 році, коли Еван Ю працював у Google, створюючи багато прототипів прямо в браузері. Для цього Еван використав зручні практики з інших фреймворків, з якими він працював, і офіційно випустив Vue.js у 2014 році.

Vue.js – це прогресивний фреймворк для JavaScript, який використовується для створення вебінтерфейсів і односторінкових програм.

Vue.js також використовується не лише для вебінтерфейсів, а й для розробки додатків для настільних комп'ютерів і мобільних пристроїв за допомогою фреймворку Electron. Розширення HTML і база JS швидко зробили Vue улюбленим інтерфейсним інструментом, про що свідчить прийняття такими гігантами, як Adobe, Behance, Alibaba, Gitlab і Xiaomi [11].

В інтерв'ю з Еваном він розповідає, що спочатку Vue.js спробою взяти найкраще з Angular і створити спеціальний інструмент, але менший за вагою: «Для мене Angular запропонував щось круте, а саме прив'язування даних і керування даними. Це спосіб роботи з DOM, тому вам не потрібно торкатися DOM самостійно» [2].

Назва фреймворку – Vue – фонетично збігається з англійською мовою view і відповідає традиційній архітектурі Model-View-Controller (MVC). Простіше кажучи, view – це інтерфейс програми/вебсайту, а основна бібліотека Vue.js фокусує рівень перегляду за замовчуванням. Але MVC не означає, що Vue.js не можна використовувати з іншим архітектурним підходом, таким як компонентна архітектура (CBA), яка використовується в React (див. рис. 1.2).

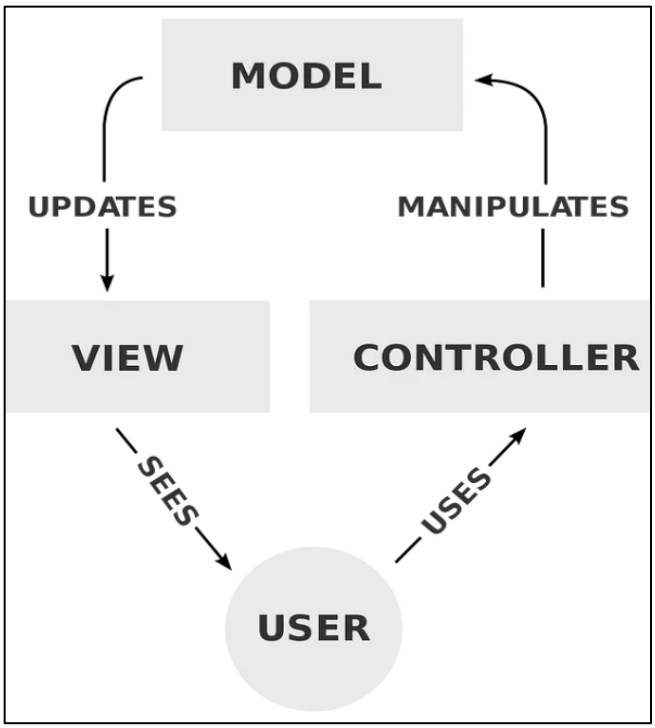

Рисунок 1.2 – Model-View-Controller

На Vue написано багато проєктів, але ось вам невеличкий список популярних:

– Behance;
– Gitlab;
– Trivago;
– Statista;
– 9GAG.

В основному Vue.js було розроблено з ідеєю досягнення позитивних результатів з мінімальними зусиллями. Це дає нам змогу створювати програми або розробляти інтерфейс користувача з невеликою кількістю рядків коду. Vue.js також чудово підходить, коли розробник хоче працювати з компонентами, оскільки це вимагає невеликих витрат – компоненти Vue.js можуть легко зберігати весь ваш код CSS, JavaScript і HTML в одному файлі.

Vue.js є одним із таких фреймворків, який не потребує серйозного навчання. Це корисно для всіх розробників, особливо для початківців – для Vue.js потрібні лише базові знання HTML, CSS і JavaScript. Це протилежність Angular або React. Однією з інших переваг Vue js для розробки додатків є те, що він забезпечує вищу продуктивність. Основний пакет gzipped важить лише 18 Кб. Фреймворк розроблений таким чином, щоб бути продуктивним без додаткової оптимізації . Крім того, він надає вбудовані директиви, такі як v-once та v-memo, які допомагають підвищити продуктивність.

Завдяки простій структурі Vue.js легко зрозуміти, що дозволяє розробникам легко додавати Vue.js до своїх вебпроєктів. Крім того, Vue постачається з чітко визначеною архітектурою, яка дозволяє розробникам зберігати наявну інфраструктуру як є, а також розділяти дані, життєвий цикл і власні методи.

Vue легко інтегрувати, на відміну від інших фреймворків JavaScript. Його можна використовувати не лише для створення програм, але й для включення компонентів у вже існуючі програми. Існують бібліотеки,

створені, щоб допомогти цьому процесу та зробити його ще простішим.

Однією з проблем Vue є те, що його найбільша спільнота розташована в Китаї, що може бути проблематичним у випадку деяких пакетів або досліджень. Є багато пакетів, зроблених китайськими розробниками Vue, які пишуть документацію рідною мовою. Це проблематично для англомовних розробників. Проте ця проблема вже активно вирішується сама по собі, так як спільнота англомовна росте.

Vue пропонує багато свободи у використанні різних підходів. Великі проєкти можуть призвести до проблем із невідповідністю коду. У цій ситуації розробникам Vue потрібно витратити час на узгодження коду, інакше різні підходи можуть призвести до більш серйозних проблем, таких як різні збої в роботі додатків і складніший процес інтеграції нових працівників у розробку проєкту.

Підсумовуючи, Vue є потужним інструментом для створення складних вебрішень. Він може надати вам величезну кількість можливостей для вирішення завдань будь-якої складності.

Порівняно з React, розробники Vue можуть відчувати дещо нестачу плагінів і бібліотек. Найважливіші з них існують для Vue, але пошук чогось специфічного може бути тяжкий , бо відповідних бібліотек ще не створено.

Vue.js побудований на принципі декларативного відображення даних в DOM за допомогою простих шаблонів. Декларативний підхід лежить в основі будь-якого фреймворку і часто є однією з причин його створення [13].

Декларативність означає, що ми говоримо, що хочемо зробити, а фреймворк самостійно виконує необхідне завдання. Декларативність зазвичай протиставляють імперативному стилю «ванільного» JavaScript, коли нам доводиться крок за кроком описувати кожну дію. У JavaScript нам потрібно не просто описати суть завдання, але й конкретно, по кроках, прописати всю логіку виконання:

– робота з html-елементами, звертаючись до них через id, класи, css-селектори;

– додавання та видалення обробників подій через addEventListener / removeEventListener;
– при зміні даних змінювати щось у конкретних DOM-вузлах (наприклад, змінювати текст через textContent) та інше.

Vue.js дуже добре справляється з декларативністю. Наприклад, в нього є набір директив, які дуже просто та легко описують основні операції з html-елементами: v-bind, v-for, v-if, v-show, v-html та інші.

Згідно з дослідженням Stack Overflow, Vue.js входить до топ-5 вебфреймворків за популярністю за підсумками 2021 року (див. рис. 1.3).

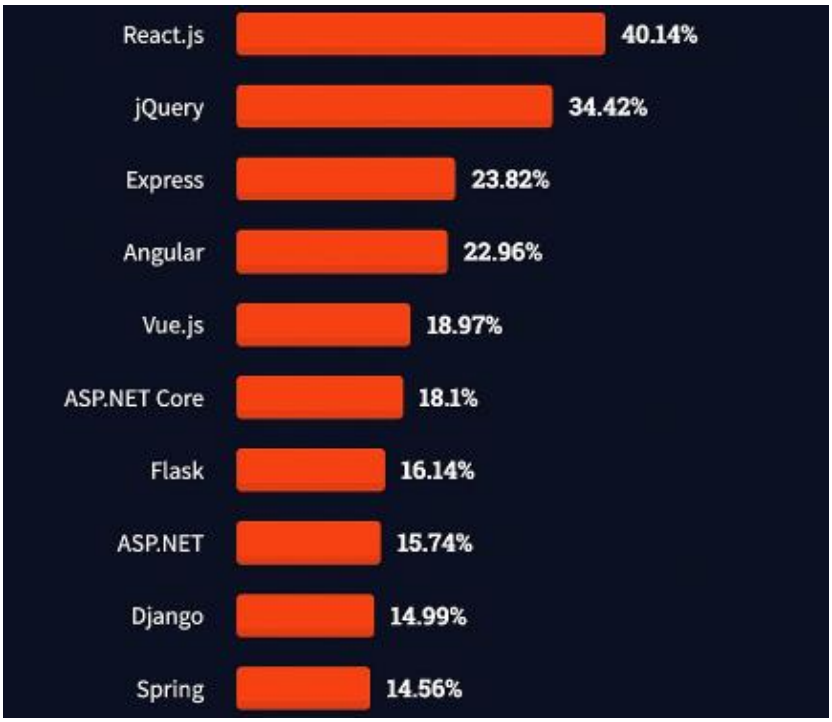

Рисунок 1.3 – Популярність фреймворків за 2021 рік

Що стосується вакансій, React все ще залишається в лідерах. Наприклад: у жовтні 2021 року на hh.ru React зустрічався в вимогах 7636 разів, Angular – 3649 разів, а Vue.js – 2030 разів. У квітні 2022 року на hh.ru React зустрічався 4921 раз, Angular – 2139 раз, Vue.js – 2320 раз (по всіх регіонах і країнах без фільтрів на hh.ru). Якщо взяти період з жовтня 2021 року по квітень 2022 року, то ми побачимо, що Vue.js – єдиний фреймворк, кількість згадок в вакансіях у якого зросла, – по React і Angular ми бачимо зниження.

Взагалі кажучи, дуже складно говорити про Vue.js без порівняння з React. Їх філософія дуже схожа, але у деталях є багато відмінностей. Ось дві основні:

– Vue є більш строгим і декларативним: строгість виявляється у синтаксисі (певні директиви для кожного завдання), а також у тому, як будуються самі компоненти (чітка структура розділення моделі, представлення та стилів, особливо при використанні однофайлових компонентів з розширенням .vue).
– Vue має більш «міцну» інфраструктуру – вибір найважливіших бібліотек часто передбачений наперед.

Такий підхід призводить до того, що новачкам простіше писати зрозумілий код з самого початку знайомства з Vue.js. У вищезазначеному прикладі код на Vue був написаний без знання про існування тернарного оператора і навіть без знання про те, як перебирати елементи в масиві. Було потрібно лише вивчити пару простих директив (v-for і v-show). Решту роботи Vue зробив сам.

На мій погляд, при рівних умовах ймовірність зробити помилку у Vue.js менше, а код його читається простіше. При цьому важливо зауважити, що Vue може бути настільки ж «вільним» у виразності, як і React. І якщо глибоко вивчити документацію, то можна вийти за звичайні рамки – і тоді Vue.js виявиться не менш гнучким, ніж React. І звісно, чим більше і складніше проєкт, тим більше ймовірність, що знадобиться знання вдосконалених технік роботи з Vue.js та його інструментарієм.

Документація – не завжди найпростіший джерело для вивчення якого-небудь інструменту програмування. Але з Vue це не так. Навіть якщо залишити документацію Vue.js як єдиний ресурс для навчання, можна досить непогано в цьому преуспіти, хоча деякі додаткові джерела – курси та відео – можуть допомогти з розумінням певних моментів.

Головне – не забувати якомога більше практикуватися, вирішувати завдання з використанням Vue, копатися в документації, спробовувати різні підходи і постійно збільшувати складність.

1.4 Протоколи та методи передачі даних у вебзастосунках

Мережевий протокол – це комплекс установок, завдяки яким визначається і регулюється процес інформаційного обміну між комп'ютерами, підключеними до інтернету. Протокол в певному сенсі вважається мовою, необхідною машинам для взаємодії. Серед його ключових особливостей – структурованість і стандартизація.

Функціонування мережі ґрунтується на роботі відразу декількох протоколів, наявних на різних рівнях. Виділяють такі рівні протоколів:

- **фізичний** – середовище, де здійснюється обмін даними, на цьому рівні трансформуються в бінарний код електричні імпульси, які далі передаються на більш високі рівні (на даному рівні функціонують медиаконвертери, сигнальні ретранслятори, хаби);
- **канальний** – рівень, на якому дані передаються на хост з метою обробки, а щоб ідентифікувати інформацію, застосовується MAC-адреси;
- мережевий – актуалізуються IP-адреси, завдяки яким в інтернеті ідентифікуються користувачі, дані надходять пакетами;
- **транспортний** – в обов'язковому порядку здійснюється доставка пакетів до адресатів, протокол відстежує цілісність і коректність донесення інформації, для цих цілей використовуються алгоритми фрагментування або об'єднання;
- **сесійний** – протоколи забезпечують підтримку мережевого сеансу, синхронність початку і кінця з'єднання, а також перевіряють дозволи на доступ;
- **репрезентативний** (рівень представлення) – отримана інформація декодується або кодується, файли розпаковуються або стискаються, тобто здійснюється переклад даних на рівень, який підійде конкретному браузеру (з додатком);

– **прикладний** – відбувається регулювання зв’язку користувачів і мережі, даються дозволи на доступ, реалізовується робота протоколів вищого рівня.

Дані типи представлені в порядку ієрархії, тобто з дій, які здійснюються на нижчому рівні, випливають алгоритми нового рівня.

Різниця між мережевими протоколами буде більш наочною, якщо порівнювати найбільш популярні види, розібратися з особливостями і функціями кожного.

Hyper Text Transfer protocol є основним для функціонування всіх інтернет-ресурсів. Завдання даного протоколу – надання можливості запиту ресурсів, які потрібні в віддаленій системі (наприклад, файлів).

HTTP вважається клієнт-серверним протоколом, що означає надсилання даних однією конкретною стороною. Головні особливості HTTP пов’язані з:

– **простотою** – алгоритми легко сприймаються людьми;
– **розширюваністю** – досить навіть звичайного узгодження між клієнтами і серверами, щоб поміняти щось в семантиці;
– **сесійній** – між запитами відсутній послідовний зв’язок, на кожній сесії можна ділитися певним контекстом;
– **транспортним рівнем** – завдяки управлінню на транспортному рівні протокол знаходиться за межами HTTP.

За допомогою HTTP можна управляти кешем, аутентифікацією, проксі, а також сесіями. Однак останнім часом актуальна тенденція переходу сайтів з HTTP на HTTPS, що допомагає підвищити рівень довіри до інтернет-ресурсу.

Internet Protocol є протоколом маршрутизації на мережевому рівні TCP/IP. Завдяки IP стало можливим об’єднати у всесвітню мережу різні комп’ютерні мережі. Головною метою протоколу є доставка пакетів між різноманітним мережевим обладнанням. У числі характерних властивостей:

– **відсутність надійності** – гарантію доставки без помилок забезпечують на більш високих рівнях протоколів, в той час як IP не виключає дублі, пошкодження, порушення порядку та навіть

відсутність надсилання даних;

- **унікальність** – для кожного комп’ютера існують окремі IP-адреси, завдяки яким машини знаходять і ідентифікують один одного в інтернеті;
- **фрагментованість** – протокол проходить через різноманітні канали доставки, і в разі перевищення можливостей певних вузлів передбачено дроблення пакетів.

Даний алгоритм реалізований за допомогою таких видів, як IPv4 і IPv6. У моделі TCP/IP він відноситься до мережевого рівня.

Протокол Secure Shell реалізований на рівні додатків і призначений, щоб дистанційно керувати системою за допомогою захищеного каналу. Даний варіант застосовується в роботі багатьох технологій. Підключення по SSH включають такі особливості:

- **шифрування** – характерна властивість SSH – авторизація по ключу, тобто відбувається кодування всього трафіку, в тому числі паролів, за допомогою різних алгоритмів;
- **безпеку** – це властивість випливає з попереднього, так як завдяки шифруванню збільшується надійність віддаленої роботи;
- **можливість стиснення** – ця особливість актуальна при передачі інформації.

Копіювання файлів по SSH дозволяє підвищити рівень захисту при передачі інформації. Secure Shell вважається протоколом прикладного рівня, і його пряме призначення – забезпечення віддаленого доступу.

File Transfer Protocol – один з найстаріших прикладних варіантів. Протокол FTP служить для доступу до віддалених хостів і передачі програмного забезпечення. Щоб точніше розуміти, що таке FTP, слід розібратися з його особливостями. Так, властивості протоколу мають на увазі:

- **результативність** – протокол гарантує надсилання або видачу помилки, так як застосовується квотна система;
- **варіативність шифрування** – в різних випадках можливі або

анонімні підключення, або передача паролів і логінів відкритим текстом;

- **вбудовану аутентифікацію** – користувачі автентифіковані за замовчуванням;
- **застосовується декілька портів** – FTP-протокол як мінімум застосовує подвійне підключення.

Завдяки бінарному режиму передачі (аналогічна особливість є у HTTP) зменшуються витрати трафіку і час, необхідних для надсилання великих файлів.

Post Office Protocol Version 3 – стандартний варіант на прикладному рівні, який використовується клієнтами email-сервісів. Головне завдання POP3 – забезпечити отримання пошти з віддаленого сервера за допомогою TCP-з'єднання. Крім цього стандарту, щоб отримати електронну пошту, також використовується IMAP. Як правило, на сучасних клієнтах реалізовані обидва варіанти. Особливості POP3:

- постійний доступ до листів, які збережені на комп'ютері (навіть якщо немає зв'язку з мережею);
- відкриття вкладень одночасно з повідомленням;
- постійне звільнення пам'яті ящика – листи перевантажуються на жорсткий диск комп'ютера.

Проте, крім переваг, є істотний недолік – ризик зараження вірусами комп'ютерів користувачів. Також може знадобитися досить частий бекап. Щоб правильно організувати резервне копіювання даних, необхідно заручитися підтримкою надійного хостинг-провайдера.

Media Access Control є протоколом, розміщеним на низькому рівні. Його завдання полягає в тому, щоб ідентифікувати пристрої локальних мереж. Властивості MAC мають на увазі:

- **унікальність** – для кожного пристрою є унікальний MAC-адреса, яка спочатку задана виробником;
- **захист від помилок** – завдяки створенню і перевірці алгоритмів

забезпечується додатковий захист на цьому рівні;

– **контроль** – MAC контролює доступ до фізичного середовища передачі.

Механізми управління доступом до каналу на MAC-рівні реалізовані як метод множинного доступу. Таким чином, відразу кілька станцій можуть застосовувати одне й те ж фізичне середовище.

Вибір правильного протоколу і порту дозволяють оптимізувати роботу системи і убезпечити з’єднання. Якщо протоколи не узгоджені між собою, вебресурси будуть працювати некоректно і давати постійні збої [12].

2 ПРОЄКТУВАННЯ ТА АНАЛІЗ ВИМОГ ДО ОНЛАЙН ЧАТУ

2.1 Аналіз вимог до інтернет-магазину

Процес розробки інтернет-магазину побутової техніки розпочинається з визначення та аналізу вимог. Це ключовий етап, що допомагає встановити чіткі цілі проєкту, зрозуміти потреби користувачів та визначити функціональні можливості майбутнього додатку. Важливо ретельно опрацювати всі вимоги, оскільки це впливає на загальну архітектуру системи, вибір технологій та ефективність подальшої розробки.

Першим кроком є збір вимог від зацікавлених сторін, включаючи клієнтів, майбутніх користувачів та команду розробників. Це включає:

– інтерв’ю з клієнтом: обговорення з клієнтом його бачення проєкту, цілей, яких він прагне досягти, та очікуваних результатів;
– аналіз ринку та конкурентів: дослідження існуючих рішень на ринку інтернет-магазинів побутової техніки для виявлення основних функцій, які користуються попитом, та можливих недоліків, які можна покращити;
– опитування користувачів: вивчення потреб потенційних користувачів, їх уподобань та очікувань від онлайн-магазину.

Перший етап розробки інтернет-магазину побутової техніки розпочався з детального інтерв’ю з клієнтом. Основною метою цього інтерв’ю було визначити ключові вимоги до системи та зрозуміти очікування замовника.

Ключові питання та відповіді для клієнта розглянемо далі.

Цільова аудиторія: інтернет-магазин орієнтований на широке коло споживачів, включаючи як індивідуальних покупців, так і малий та середній бізнес, які потребують побутову техніку для офісів.

Функціональні вимоги: основні функції включають можливість перегляду товарів, фільтрації за категоріями, порівняння товарів, додавання

товарів до кошика, оформлення замовлення, а також відстеження статусу замовлення. Крім того, необхідна можливість реєстрації та авторизації користувачів, залишення відгуків і оцінок для товарів.

Адміністративні функції: адміністратори повинні мати можливість додавати, редагувати та видаляти товари, керувати категоріями, переглядати замовлення та змінювати їх статус, а також отримувати аналітику щодо продажів і відвідуваності.

Дизайн та інтерфейс: інтерфейс має бути зручним та інтуїтивно зрозумілим, з сучасним дизайном. Важливо, щоб сайт був адаптивним і коректно відображався на різних пристроях, включаючи мобільні телефони та планшети.

Безпека: особливу увагу слід приділити безпеці даних користувачів, захисту від зловмисних атак та забезпеченню конфіденційності фінансової інформації під час транзакцій.

Інтеграції: необхідна інтеграція з платіжними системами для прийому онлайн-платежів, а також з сервісами доставки для автоматичного відстеження замовлень.

Масштабованість: система повинна бути здатною до масштабування, щоб витримувати зростаюче навантаження з розширенням бази користувачів та асортименту товарів.

Терміни реалізації: бажано запустити базову версію інтернет-магазину протягом трьох місяців, з подальшим доопрацюванням і додаванням нових функцій за необхідності.

Підтримка та оновлення: необхідно передбачити регулярну технічну підтримку та можливість оновлення функціоналу без суттєвих переривань у роботі магазину.

В результаті інтерв’ю було складено детальний список вимог до інтернет-магазину, що став основою для подальшого моделювання, проєктування та реалізації системи. Клієнт надав необхідну інформацію для

визначення ключових функціональних і нефункціональних вимог, що забезпечило чітке розуміння очікувань та завдань для розробницької команди.

Аналіз ринку та конкурентів є критичним етапом при розробці інтернет-магазину побутової техніки. Він допомагає зрозуміти поточний стан ринку, виявити головних гравців та визначити ключові фактори успіху. Цей аналіз включає вивчення ринку, аналіз конкурентів, а також виявлення їхніх сильних та слабких сторін [14].

Ринок побутової техніки є значним і постійно зростає завдяки зростаючому попиту на сучасні технології та зручність онлайн-шопінгу. Згідно з різними дослідженнями, цей сектор демонструє стабільне зростання з прогнозованим збільшенням обсягу продажів у найближчі роки.

Споживачі все більше надають перевагу покупкам в інтернеті через зручність, широкий вибір товарів та конкурентоспроможні ціни. Важливими факторами при виборі магазину стають якість обслуговування клієнтів, швидкість доставки та можливість легко повернути товар. Зростає популярність екологічно чистої та енергоефективної техніки, що враховують більшість сучасних споживачів при виборі побутових приладів.

Споживацька поведінка:

– покупці часто використовують мобільні пристрої для здійснення покупок, що підкреслює важливість мобільної адаптації сайту;
– важливим аспектом є наявність відгуків та рейтингів товарів, які допомагають приймати рішення про покупку.

В ході аналізу конкурентів було ідентифіковано кілька провідних інтернет-магазинів побутової техніки, які є основними конкурентами: “Rozetka”, “Foxtrot”, “Comfy” та “Eldorado” (див. рис. 2.1 – 2.4).

Сильні сторони конкурентів:

– широкий асортимент: більшість конкурентів пропонують великий вибір побутової техніки різних брендів та цінових категорій;
– брендова репутація: довготривалий досвід на ринку та позитивні відгуки клієнтів допомагають формувати довіру до бренду;

– програми лояльності: наявність програм лояльності та знижок для постійних клієнтів є додатковою перевагою;
– сервіс та підтримка: швидка та якісна підтримка клієнтів, зручні умови повернення товарів та швидка доставка.

Слабкі сторони конкурентів:

– погана мобільна адаптація: деякі конкуренти мають сайти, які не завжди добре працюють на мобільних пристроях;
– відсутність індивідуального підходу: великі магазини іноді не можуть запропонувати персоналізовані рекомендації та підтримку, що може відштовхувати певну категорію клієнтів;
– складність у навігації: деякі інтернет-магазини мають складну структуру та заплутану навігацію, що ускладнює пошук потрібного товару.

Можливості для нового інтернет-магазину:

– покращений користувацький досвід: забезпечення зручної та інтуїтивно зрозумілої навігації, швидкого завантаження сторінок та якісної мобільної адаптації;
– персоналізація: використання рекомендаційних систем для персоналізованих пропозицій товарів, базуючись на інтересах та поведінці користувачів;
– покращена підтримка клієнтів: впровадження цілодобової онлайн-підтримки, чат-ботів для швидкої відповіді на запити та гнучкої системи повернення товарів;
– інноваційні маркетингові стратегії: використання соціальних мереж, програм лояльності та партнерських програм для залучення нових клієнтів та утримання постійних.

Підсумовуючи, аналіз ринку та конкурентів надав цінну інформацію для розробки стратегії створення інтернет-магазину побутової техніки, що дозволить запропонувати споживачам унікальний продукт, який виділяється серед конкурентів.

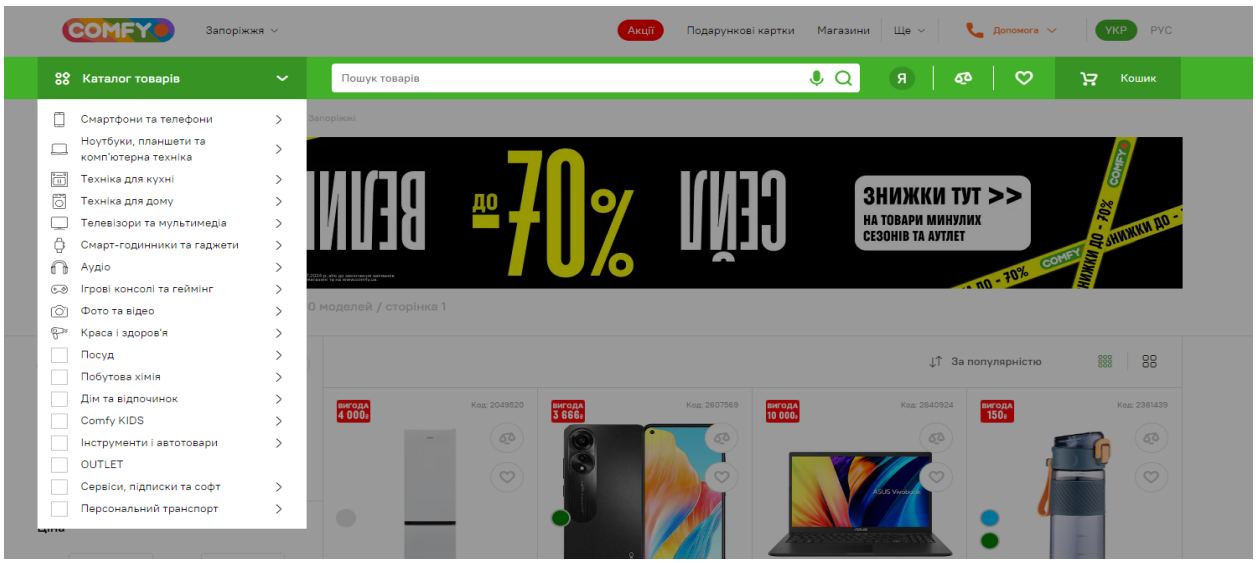

Рисунок 2.1 – Comfy

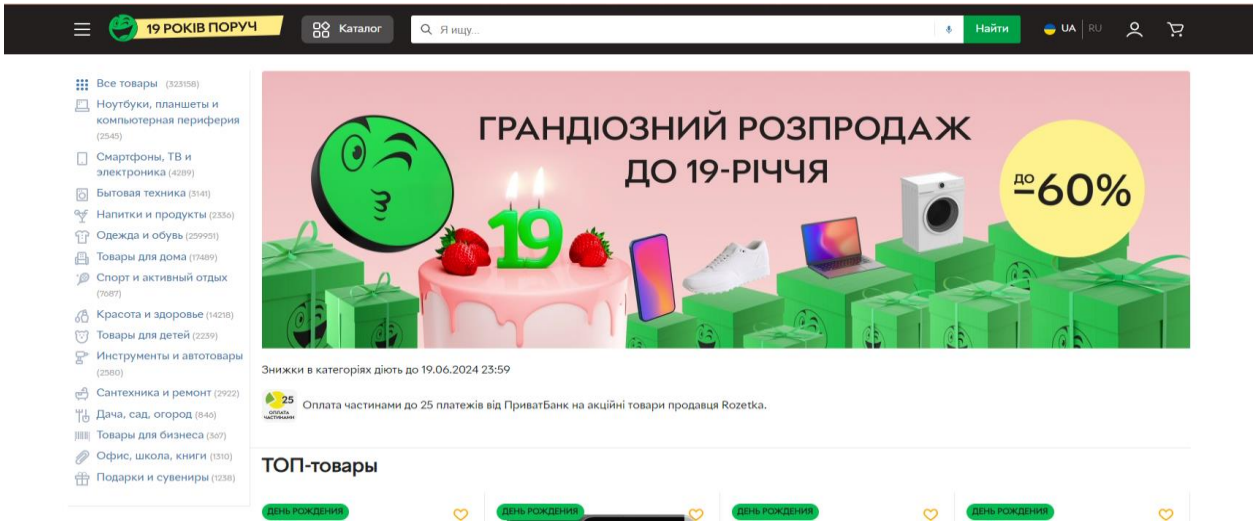

Рисунок 2.2 – Розетка

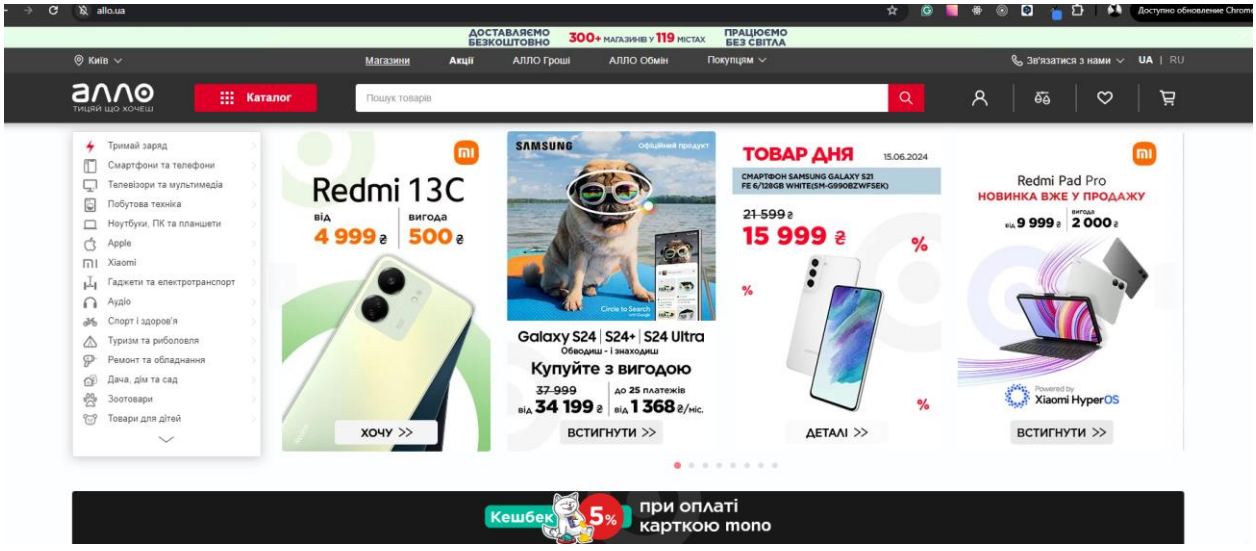

Рисунок 2.3 – Алло

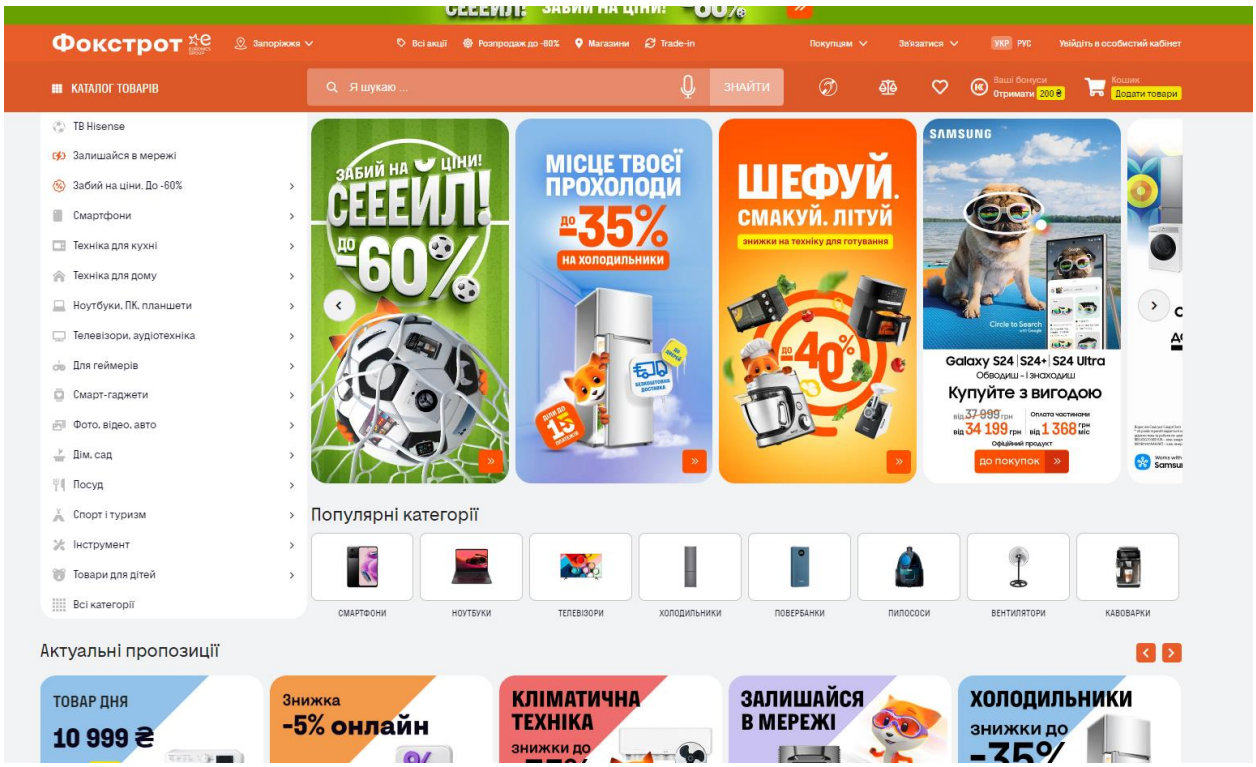

Рисунок 2.4 – Фокстрот

На основі зібраної інформації формулюються функціональні вимоги, які включають:

– каталог товарів: можливість перегляду товарів з різними категоріями та фільтрами;
– картка товару: детальна інформація про товар, включаючи опис, характеристики, фотографії, відгуки;
– кошик: можливість додавання товарів до кошика, редагування кількості, видалення товарів;
– замовлення: процес оформлення замовлення, включаючи вибір способу оплати та доставки;
– аутентифікація та авторизація: реєстрація, вхід, відновлення пароля, різні ролі користувачів (адміністратор, покупець);
– адміністративна панель: управління товарами, замовленнями, користувачами, відгуками.

Крім функціональних вимог, важливо визначити нефункціональні вимоги, які включають:

– продуктивність: час завантаження сторінок, обробка запитів, масштабованість системи;
– безпека: захист від атак, шифрування даних, безпечна аутентифікація;
– юзабіліті: інтуїтивно зрозумілий інтерфейс, адаптивний дизайн для різних пристроїв;
– надійність: висока доступність, резервне копіювання даних, стійкість до збоїв;
– підтримка та обслуговування: легкість оновлення, документація, технічна підтримка.

2.2 Моделювання бізнес-процесів та користувацьких сценаріїв

Моделювання бізнес-процесів та користувацьких сценаріїв є критичним етапом у розробці інтернет-магазину побутової техніки. Воно дозволяє візуалізувати основні функціональні аспекти системи, визначити ролі користувачів та забезпечити безперервний потік інформації між різними компонентами системи. Бізнес-процеси охоплюють всі ключові операції, необхідні для функціонування інтернет-магазину. До них відносяться наступні.

Управління товарами:

– додавання нових товарів: адміністратори додають нові товари до каталогу, вказуючи назву, опис, ціну, наявність на складі та інші характеристики;
– оновлення інформації про товари: адміністратори можуть змінювати інформацію про наявні товари, включаючи ціну, опис, зображення тощо;
– видалення товарів: адміністратори видаляють товари, які більше не доступні або не актуальні.

Управління користувачами:

– реєстрація користувачів: користувачі створюють облікові записи для доступу до персоналізованих функцій магазину;
– вхід та вихід: користувачі входять до своїх облікових записів для перегляду замовлень, особистих даних тощо;
– управління профілем: користувачі оновлюють особисті дані, такі як адреса доставки, контактна інформація та інше.

Обробка замовлень:

– створення замовлень: користувачі додають товари до кошика та оформлюють замовлення, вказуючи деталі оплати та доставки;
– обробка платежів: система перевіряє та обробляє платежі за допомогою інтегрованих платіжних шлюзів;
– підтвердження замовлень: користувачі отримують підтвердження замовлення з детальною інформацією про доставку та оплату;
– доставка товарів: логістичний відділ забезпечує доставку товарів згідно з зазначеними термінами;

Обробка повернень та обмінів:

– запит на повернення: користувачі можуть ініціювати запит на повернення або обмін товарів;
– обробка запитів: адміністратори перевіряють запити на повернення та обмін, забезпечуючи їх відповідність політикам магазину;
– відшкодування та обмін: адміністратори проводять відшкодування коштів або обмін товарів згідно з запитами користувачів.

Для створення схеми моделювання бізнес-процесів та користувацьких сценаріїв інтернет-магазину побутової техніки, зосередимося на основних процесах, таких як реєстрація користувача, вибір та покупка товару, обробка замовлення, та обслуговування користувачів. Скористаємося UML-діаграмами для відображення цих процесів [15].

На рисунках 2.5 – 2.7 наведено діаграми для проєктування функціональних компонентів інтернет-магазину.

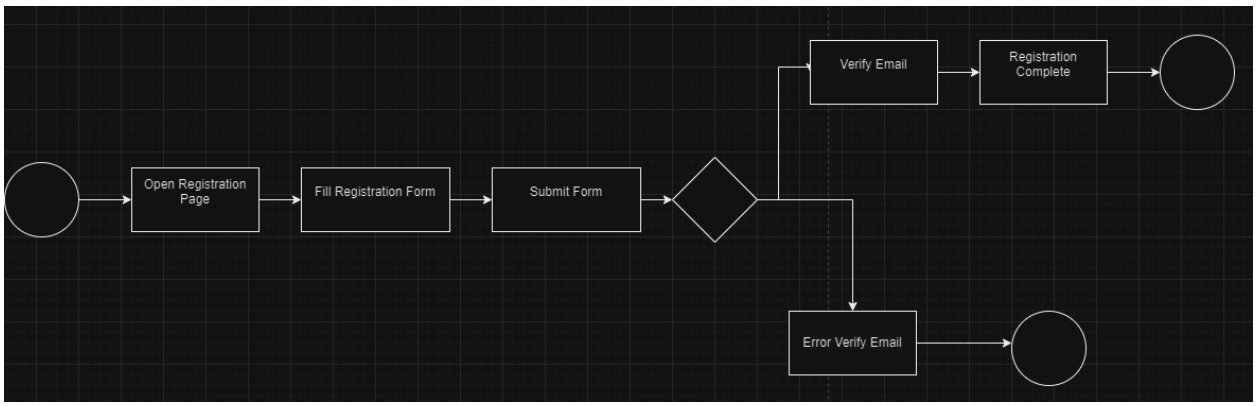

Рисунок 2.5 – Register User

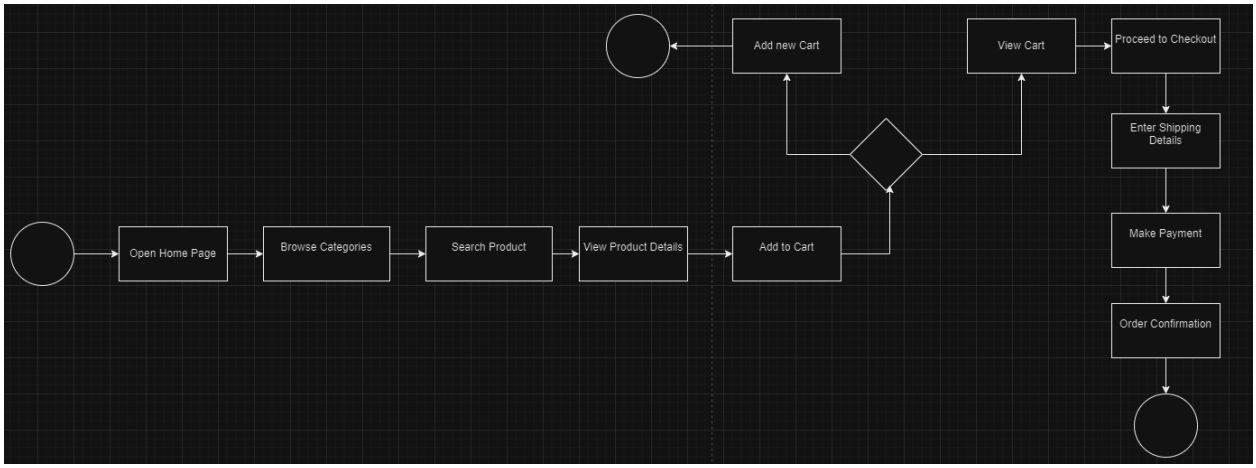

Рисунок 2.6 – Browse and Purchase Product

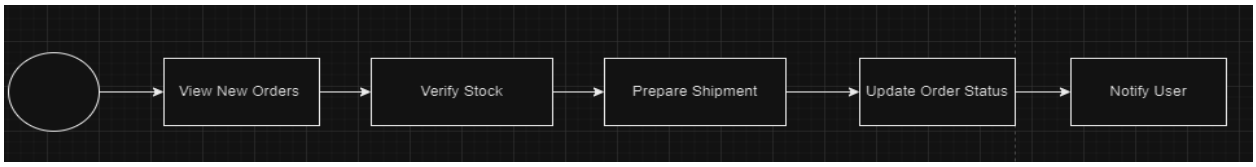

Рисунок 2.7 – Process Order

Розпишемо кожну діаграмму Activity Diagram детальніши для побудови загальної картини.

Register User.

Open Registration Page:

– Description: користувач переходить на сторінку реєстрації на вебсайті інтернет-магазину;
– Details: користувач натискає на посилання «Реєстрація» або відповідну кнопку в навігаційному меню.

Fill Registration Form:

– Description: користувач заповнює форму реєстрації, вводячи необхідні дані, такі як ім’я, електронна пошта, пароль та інші контактні дані;
– Details: форма містить поля для введення імені, електронної пошти, пароля, підтвердження пароля та, можливо, додаткові поля, такі як номер телефону або адреса.

Submit Form:

– Description: користувач надсилає заповнену форму реєстрації, натискаючи на кнопку «Зареєструватися» або аналогічну;
– Details: після натискання на кнопку «Зареєструватися» дані форми відправляються на сервер для обробки.

Validate Form Data:

– Description: система перевіряє, чи всі обов’язкові поля заповнені та чи відповідають введені дані вимогам (наприклад, правильний формат електронної пошти, сильний пароль);
– Details: у разі помилок користувачеві відображаються відповідні повідомлення про помилки.

Create User Account:

– Description: якщо всі дані коректні, система створює новий обліковий запис користувача у базі даних;
– Details: включає збереження введених даних та хешування пароля для безпеки.

Send Verification Email:

– Description: система надсилає користувачеві електронний лист для підтвердження електронної пошти;
– Details: лист містить посилання, перейшовши за яким користувач зможе підтвердити свою електронну пошту.

Verify Email:

– Description: користувач переходить за посиланням у верифікаційному листі;

– Details: система перевіряє, чи посилання дійсне і чи відповідає воно даним у базі.

Registration Complete:

– Description: після успішної верифікації електронної пошти система підтверджує завершення реєстрації користувача;
– Details: користувач отримує повідомлення про успішну реєстрацію та може увійти на сайт.

Browse and Purchase Product.

Open Home Page:

– Description: користувач відкриває домашню сторінку інтернет-магазину;
– Details: домашня сторінка містить основну інформацію про магазин, популярні товари та навігаційне меню.

Browse Products:

– Description: користувач переглядає список товарів;
– Details: користувач може використовувати фільтри, сортування та пошук для знаходження потрібних товарів.

View Product Details:

– Description: користувач натискає на товар для перегляду детальної інформації;
– Details: сторінка товару містить зображення, опис, ціну, відгуки та іншу інформацію.

Add Product to Cart:

– Description: користувач додає вибраний товар до кошика;
– Details: користувач може вибрати кількість товару перед додаванням до кошика.

View Cart:

– Description: користувач переглядає свій кошик;
– Details: кошик містить список доданих товарів, їх кількість, ціну та загальну суму.

Proceed to Checkout:

– Description: користувач переходить до оформлення замовлення;
– Details: користувач вводить необхідну інформацію для доставки та оплати.

Enter Payment Details:

– Description: користувач вводить дані для оплати;
– Details: включає вибір способу оплати (кредитна картка, PayPal тощо) та введення відповідної інформації.

Confirm Order:

– Description: користувач підтверджує замовлення;
– Details: після підтвердження замовлення користувач отримує повідомлення про успішне оформлення.

Receive Order Confirmation:

– Description: користувач отримує підтвердження замовлення на електронну пошту;
– Details: лист містить деталі замовлення та очікувану дату доставки.

Process Order (Admin).

Receive New Order Notification:

– Description: адміністратор отримує повідомлення про нове замовлення;
– Details: повідомлення може надходити на електронну пошту або відображатися в адміністративній панелі.

View Order Details:

– Description: адміністратор переглядає деталі замовлення;
– Details: включає інформацію про товари, доставку та платіж.

Verify Payment:

– Description: адміністратор перевіряє, чи була оплата успішно проведена;
– Details: перевірка може здійснюватися через платіжну систему або банк.

Prepare Order for Shipment:

– Description: адміністратор готує товари до відправки;
– Details: включає пакування товарів та підготовку супровідних документів.

Update Order Status:

– Description: адміністратор оновлює статус замовлення у системі;
– Details: користувач отримує повідомлення про оновлення статусу замовлення.

Ship Order:

– Description: адміністратор відправляє замовлення користувачу;
– Details: включає передачу товарів кур’єру або поштовій службі.

Confirm Shipment:

– Description: адміністратор підтверджує відправлення замовлення;
– Details: користувач отримує повідомлення про відправлення замовлення разом з трекінг-номером.

Handle Post-Delivery Issues:

– Description: адміністратор вирішує будь-які питання, що можуть виникнути після доставки;
– Details: включає обробку повернень, обмінів та скарг.

2.3 Проєктування архітектури системи

Проєктування архітектури системи для інтернет-магазину з використанням фреймворків Laravel та Vue.js є ключовим етапом, що визначає загальну структуру та організацію програмного забезпечення. Основні аспекти проєктування архітектури системи включають наступні елементи:

– Frontend (Vue.js);
– Backend (Laravel);
– База даних (MySQL).

Frontend на Vue.js для інтернет-магазину включає в себе різноманітні компоненти, які взаємодіють з backend через API, реалізоване на Laravel. Основні компоненти та їх функціональність можуть бути наступними:

Компонент ProductList.vue.

Опис: відображення списку продуктів;

Функціонал:

– запит до API для отримання списку продуктів;
– відображення кожного продукту у вигляді карточки;
– можливість сортування та фільтрації продуктів;
– пагінація для навігації між сторінками продуктів.

Компонент ProductDetail.vue.

Опис: детальна інформація про конкретний продукт.

Функціонал:

– відображення основної інформації про продукт: назва, опис, ціна, зображення;
– відображення коментарів користувачів до цього продукту;
– можливість додавання нового коментаря;
– кнопки для додавання продукту до кошика.

Компонент Cart.vue.

Опис: кошик покупок.

Функціонал:

– відображення усіх продуктів, які користувач вибрав для покупки;
– можливість зміни кількості продуктів у кошику;
– підрахунок загальної суми замовлення;
– кнопки для оформлення замовлення та очищення кошика.

Компонент Authentication.vue.

Опис: форма авторизації та реєстрації користувача.

Функціонал:

– форма для введення електронної пошти та пароля для входу;
– можливість реєстрації нового користувача з обов’язковим підтвердженням електронної пошти;

– відновлення пароля через електронну пошту.

Компонент AdminDashboard.vue.

Опис: панель адміністратора.

Функціонал:

– відображення загальних статистичних даних про магазин;

– можливість додавання, редагування та видалення продуктів;

– керування категоріями продуктів та управління користувачами.

Фронтенд на Vue.js взаємодіє з backend на Laravel через HTTP запити до API. Наприклад:

– отримання продуктів: Vue компонент ProductList.vue робить GET запит до /api/products, щоб отримати список продуктів;

– додавання продукту в кошик: після клікання кнопки «Додати в кошик» на ProductDetail.vue, відбувається POST запит до /api/cart/add, що додає продукт до кошика користувача;

– оформлення замовлення: компоненті Cart.vue після натискання кнопки «Оформити замовлення» відбувається POST запит до /api/orders/create, що створює нове замовлення.

Ці компоненти та їх взаємодія з backend дозволяють створити функціональний та ефективний інтернет-магазин на основі Vue.js та Laravel [17].

Laravel забезпечує ефективне керування серверною логікою для інтернет-магазину. Основні аспекти, включаючи архітектуру, маршрутизацію, контролери, моделі та взаємодію з базою даних, детально розглянемо далі [16].

Архітектура Laravel базується на шаблоні MVC (Model-View-Controller), що дозволяє чітко розділити логіку додатку на моделі для роботи з базою даних, контролери для обробки запитів та представлення для відображення даних користувачам.

Сервіс-орієнтована архітектура (SOA) є підходом до розробки програмного забезпечення, який базується на концепції надання функцій через незалежні, повторно використовувані сервіси. Ці сервіси співпрацюють між

собою для виконання комплексних бізнес-функцій і можуть бути розподілені на різні системи та платформи.

Моделі (Models):

– моделі можуть мати відношення (відношення один-до-одного, один-до-багатьох, багато-до-багатьох) з іншими моделями;
– в нашому випадку, основні моделі включають Product, Category, Order, User.

Контролери (Controllers):

– обробляють вхідні HTTP-запити, взаємодіють з моделями та повертають відповідь;
– основні контролери включають ProductController, CartController, OrderController, AuthController.

Маршрути (Routes):

– визначають, які контролери та методи обробляють певні URL-запити;
– використовуються файли для вебмаршрутів та API-маршрутів.

2.4 Проєктування бази даних

Проєктування бази даних для інтернет-магазину на базі фреймворків Laravel та Vue.js є критично важливим етапом, оскільки вона визначає структуру даних та їх взаємозв’язки, що використовуються в додатку. Основні аспекти проєктування бази даних включають наступні елементи [18].

Першим кроком є ретельний аналіз вимог до даних, який включає:

– типи продуктів: категорії товарів, їх характеристики та атрибути;
– користувачі та ролі: автентифікація, авторизація та облікові записи користувачів;
– замовлення та оплата: інформація про замовлення, статуси, історія та оплата.

На основі аналізу вимог розробляється концептуальна модель, яка описує основні сутності та взаємозв’язки між ними:

– сутності: таблиці бази даних для продуктів, користувачів, замовлень тощо;
– взаємозв’язки: зв’язки між таблицями для забезпечення консистентності даних.

Логічна модель визначає структуру таблиць та поля для кожної сутності:

– таблиці і поля: визначення атрибутів для кожної сутності, включаючи унікальні ідентифікатори, зовнішні ключі та індекси;
– нормалізація: оптимізація структури для зменшення дублікації та забезпечення ефективного зберігання даних.

Фізична модель включає специфікації технічних аспектів бази даних:

– типи даних і обмеження: визначення типів даних для кожного поля та обмежень їх значень;
– індексація і оптимізація: використання індексів для прискорення запитів і пошуку даних.

Після створення бази даних проводяться тести, оптимізація та внесення необхідних змін:

– тестування: перевірка на правильність роботи, відповідність вимогам та швидкодію;
– оптимізація: покращення швидкодії та ефективності запитів;
– супровід і скальпельне розширення: підтримка бази даних під час росту обсягів даних та функціональних вимог.

Ефективне проєктування бази даних для інтернет-магазину забезпечує надійність, продуктивність та гнучкість системи. Вона відображає вимоги бізнесу та технічні аспекти додатку, забезпечуючи оптимальну роботу всіх компонентів системи [19].

Для інтернет-магазину побутової техніки на базі Laravel і Vue.js, ми розробимо базу даних, яка міститиме таблиці для користувачів, продуктів, замовлень та інших сутностей, необхідних для функціонування магазину.

Нижче наведено опис таблиць та їх взаємозв’язків, а також URL діаграму для наочності (див. рис. 2.8).

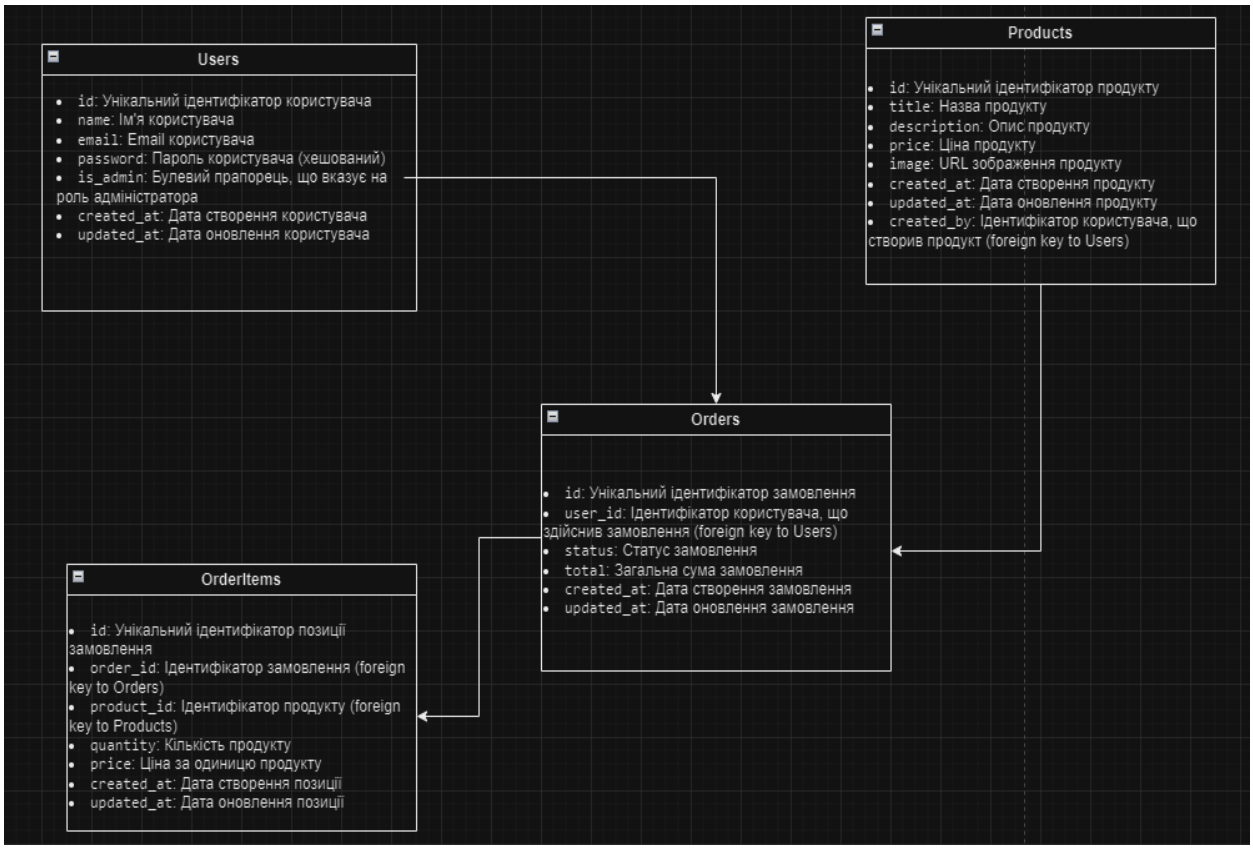

Рисунок 2.8 – URL-діаграма бази даних

2.5 Висновок та результати проєктування системи

В процесі проєктування системи інтернет-магазину із застосуванням фреймворків Laravel та Vue.js було проведено ряд важливих етапів, що дозволили створити логічну та ефективну структуру бази даних, а також розробити архітектуру системи, яка задовольняє всі вимоги замовника.

На першому етапі було проведено детальний аналіз вимог, що включав інтерв’ю з клієнтом, дослідження ринку та конкурентів, а також опитування користувачів. Цей етап дозволив чітко визначити функціональні та нефункціональні вимоги до системи, що є основою для подальшого проєктування.

Далі, в процесі моделювання бізнес-процесів та користувацьких сценаріїв, були визначені основні операції, які повинен підтримувати інтернет-магазин. Зокрема, це реєстрація користувачів, перегляд і покупка товарів, а також обробка замовлень адміністратором. Кожен з цих процесів був детально описаний та візуалізований за допомогою діаграм активностей (Activity Diagrams), що допомогло чітко розуміти взаємодію між різними компонентами системи.

Проєктування архітектури системи включало вибір технологічного стеку, який складається з Laravel для бекенду та Vue.js для фронтенду. Цей вибір дозволив забезпечити гнучкість, масштабованість та високу продуктивність системи. Також було розроблено архітектурну схему, що показує взаємодію між основними компонентами системи.

Проєктування бази даних було проведено з урахуванням всіх функціональних вимог, що дозволило створити оптимальну структуру даних для зберігання інформації про користувачів, продукти, замовлення та їх деталі. Розроблена база даних забезпечує цілісність даних та підтримує всі необхідні зв’язки між таблицями.

В результаті проєктування було створено детальну та обґрунтовану документацію, яка включає всі необхідні схеми, діаграми та описи. Ця документація слугуватиме основою для подальшої реалізації та тестування системи, забезпечуючи високу якість кінцевого продукту.

Загалом, проведене проєктування системи дозволяє стверджувати, що розроблений інтернет-магазин буде відповідати всім вимогам замовника та забезпечить користувачів зручним, надійним та ефективним інструментом для здійснення покупок онлайн [20].

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Розробка імплементації функціоналу

У цьому підрозділі ми описуємо процес розробки користувацького інтерфейсу для інтернет-магазину з використанням фреймворку Vue.js. Основними завданнями є створення привабливого та зручного інтерфейсу для користувачів, який дозволяє їм легко переглядати товари, додавати їх у кошик та оформляти замовлення.

OrderForm призначений для оформлення замовлення. Він включає наступні частини.

Шаблон містить HTML-код для відображення форми введення даних замовлення:

– поля введення для прізвища, імені, мобільного телефону, електронної пошти та міста;
– загальна сума замовлення;
– кнопка для підтвердження замовлення.

Скрипт компонента містить:

– імпорт бібліотеки axios для виконання HTTP-запитів;
– оголошення даних компонента, які включають об’єкт form для зберігання значень полів форми та total для загальної суми замовлення;
– метод submitOrder, який викликається при відправці форми. Цей метод відправляє дані форми на сервер за допомогою POST-запиту та обробляє відповідь сервера (у разі успішного відправлення форми, поля очищаються).

CSS стилі для компонента OrderForm, які відповідають за зовнішній вигляд форми:

– контейнер форми з максимальним шириною, відступами та рамкою;

– відступи та вирівнювання для груп елементів форми;
– стилі для кнопки підтвердження замовлення, включаючи кольори, відступи та ефект наведення.

Цей компонент забезпечує зручний інтерфейс для введення та відправки даних замовлення, а також обробляє взаємодію з сервером для збереження замовлення (див. рис. 3.1).

```vue
<div class="order-form">
  <h2>Оформление заказа</h2>
  <form @submit.prevent="submitOrder">
    <div class="form-group">
      <label for="lastName">Фамилия</label>
      <input type="text" id="lastName" v-model="form.lastName" required>
    </div>
    <div class="form-group">
      <label for="firstName">Имя</label>
      <input type="text" id="firstName" v-model="form.firstName" required>
    </div>
    <div class="form-group">
      <label for="phone">Мобильный телефон</label>
      <input type="tel" id="phone" v-model="form.phone" required>
    </div>
    <div class="form-group">
      <label for="email">Электронная почта</label>
      <input type="email" id="email" v-model="form.email" required>
    </div>
    <div class="form-group">
      <label for="city">Город</label>
      <input type="text" id="city" v-model="form.city" required>
    </div>
    <div class="total">
      <p>Итого: {{ total }} ₽</p>
    </div>
    <button type="submit">Заказ подтверждаю</button>
  </form>
 </div>
</template>

<script>
import axios from 'axios';

export default {
  name: 'OrderForm',
  data() {
    return {
      form: {
        lastName: '',
        firstName: '',
        phone: '',
        email: '',
        city: '',
      },
      total: 799, // примерная сумма заказа
    };
  },
```

Рисунок 3.1 – Компонент OrderForm

Компонент Home використовується для відображення головної сторінки вебсайту. Він включає в себе наступні частини.

Шаблон містить структуру HTML, яка відображає бокову панель з категоріями, банер та список товарів зі знижками:

– бокова панель (<aside class="sidebar">): містить список категорій товарів;
– основний контент (<main class="content">): містить банер і розділ зі знижками на товари;
– банер ( ): містить зображення банера;
– розділ зі знижками ( ): містить заголовок і список товарів зі знижками, кожен товар представлений зображенням, назвою і ціною.

Скрипт компонента містить:

– ім’я компонента (name: ‘Home’);
– дані компонента (data()): повертає об’єкт з двома масивами;
– categories: список категорій товарів;
– discountItems: список товарів зі знижками, кожен товар представлений об’єктом з властивостями name, image та price.

CSS стилі компонента забезпечують візуальне оформлення сторінки:

– home: контейнер для всієї сторінки, розміщує бокову панель і основний контент поруч один з одним;
– sidebar: стилі для бокової панелі, встановлюють ширину і відображення списку категорій;
– sidebar ul: стилі для списку категорій, видаляють маркери списку і відступи;
– sidebar li: стилі для елементів списку, додають відступи між категоріями;
– content: стилі для основного контенту, встановлюють ширину і відступи;
– banner img: стилі для зображення банера, встановлюють його ширину на 100% контейнера;

– discounts: стилі для розділу зі знижками, додають верхній відступ;
– discount-item: стилі для елементів зі знижками, розміщують їх поруч один з одним і центрують текст;
– discount-item img: стилі для зображень товарів, встановлюють їх розмір.

Цей компонент Home забезпечує структуру і стиль для відображення головної сторінки магазину з категоріями товарів, банером і списком товарів зі знижками (див. рис. 3.2).

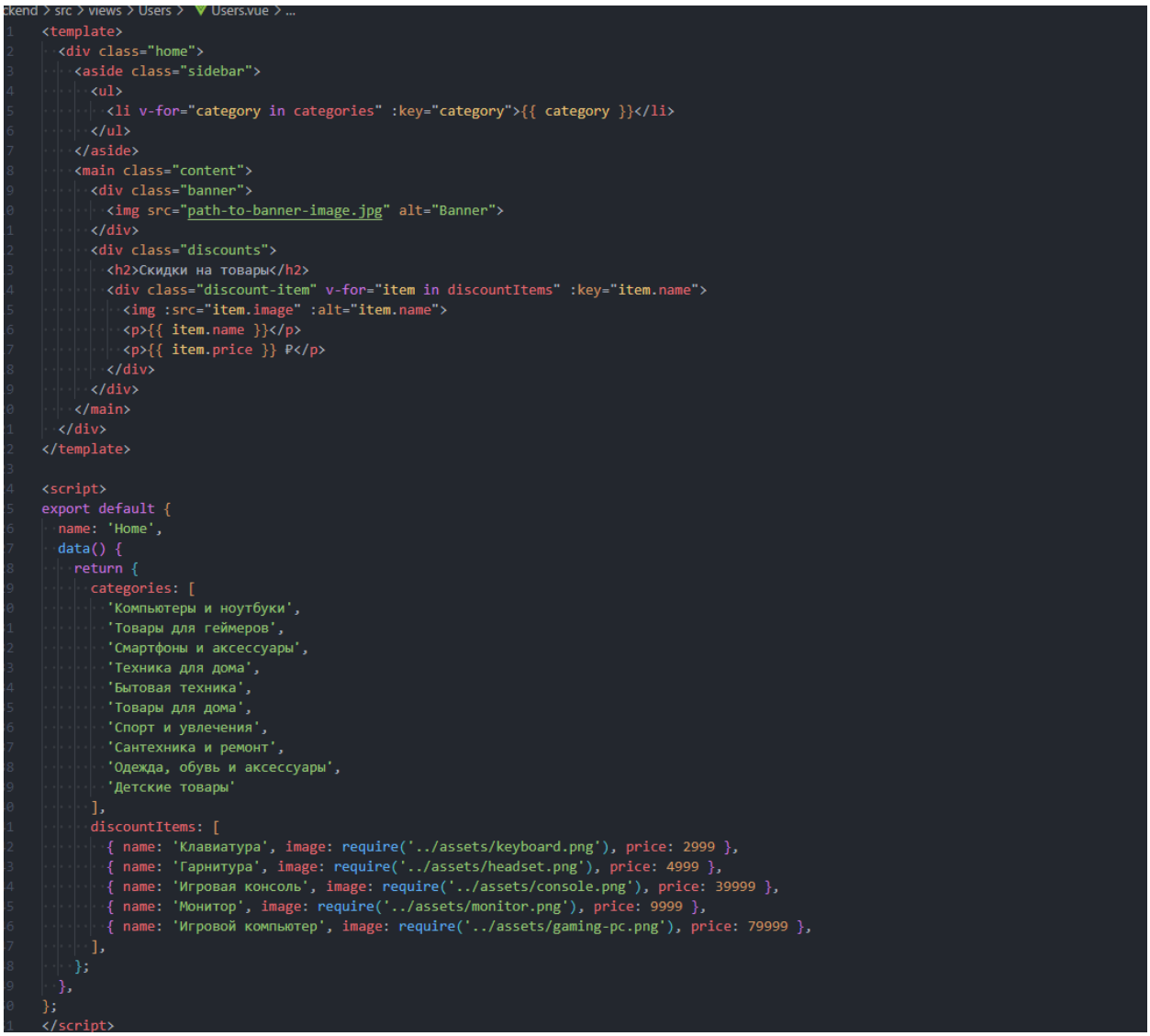

Рисунок 3.2 – Компонент Home

Компонент Categories використовується для відображення списку категорій товарів. Він включає в себе наступні частини.

Шаблон містить структуру HTML для відображення категорій товарів:

– контейнер div з класом categories, який обгортає всі категорії;
– внутрішній div з класом category, який створюється для кожної категорії з масиву categories за допомогою циклу v-for;
– для кожної категорії відображається зображення (img) та назва категорії (<p>).

Скрипт компонента містить:

– ім’я компонента (name: ‘Categories’);
– дані компонента (data()): повертає об’єкт з масивом categories, який містить об’єкти з властивостями name (назва категорії) та image (шлях до зображення категорії).

Цей компонент Categories відображає список категорій товарів у вигляді сітки. Кожна категорія представлена зображенням і назвою. Використання v-for дозволяє динамічно створювати елементи категорій на основі даних, що зберігаються в масиві categories. Стилі забезпечують привабливе і зручне відображення категорій на сторінці (див. рис. 3.3).

```
<template>
  <div class="categories">
    <div class="category" v-for="category in categories" :key="category.name">
      <img :src="category.image" :alt="category.name">
      <p>{{ category.name }}</p>
    </div>
  </div>
</template>
<script>
export default {
  name: 'Categories',
  data() {
    return {
      categories: [
        { name: 'Ноутбуки', image: require('../assets/laptop.png') },
        { name: 'Компьютеры', image: require('../assets/computer.png') },
        { name: 'Мониторы', image: require('../assets/monitor.png') },
        { name: 'Компьютерные комплектующие', image: require('../assets/computer-parts.png') },
        { name: 'Дополнительные девайсы', image: require('../assets/accessories.png') },
        { name: 'Сетевое оборудование', image: require('../assets/networking.png') },
        { name: 'Планшеты', image: require('../assets/tablets.png') },
        { name: 'Наушники и аксессуары', image: require('../assets/headphones.png') },
        { name: 'Акустические системы', image: require('../assets/speakers.png') },
        { name: 'Клавиатуры и мыши', image: require('../assets/keyboard-mouse.png') },
        { name: 'Микрофоны', image: require('../assets/microphones.png') },
        { name: 'Аксессуары для электроники', image: require('../assets/electronics-accessories.png') },
      ],
    };
  },
};
</script>
<style scoped>
.categories {
  display: flex;
  flex-wrap: wrap;
```

Рисунок 3.3 – Компонент Categories

3.2 Реалізація серверної частини застосунку

Laravel – популярний фреймворк PHP, який забезпечує зручну структуру для побудови вебзастосунків. Включає в себе різноманітні інструменти та функції, що спрощують розробку, такі як маршрутизація, ORM (Eloquent), шаблонізатор Blade та інші. В цьому розділі наведемо приклади основних компонентів, необхідних для реалізації серверної частини інтернет-магазину: маршрути, контролери, моделі та міграції.

Для реалізації маршрутизації в інтернет-магазині на Laravel потрібно налаштувати маршрути для основних дій користувача, таких як перегляд каталогу товарів, перегляд деталей товару, додавання товарів до кошика, обробка замовлень та аутентифікація користувачів. Нижче наведено приклад маршрутів, які можуть бути використані для інтернет-магазину (див. рис. 3.4).

```php
namespace Tests\Feature\Auth;

use App\Http\Controllers\ProductController;
use App\Http\Controllers\CartController;
use App\Http\Controllers\OrderController;
use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\Auth\RegisterController;

Route::get('/', [ProductController::class, 'index'])->name('home');
Route::get('/product/{id}', [ProductController::class, 'show'])->name('product.show');
Route::post('/cart', [CartController::class, 'addToCart'])->name('cart.add');
Route::get('/cart', [CartController::class, 'viewCart'])->name('cart.view');
Route::post('/order', [OrderController::class, 'store'])->name('order.store');

Route::get('/login', [LoginController::class, 'showLoginForm'])->name('login');
Route::post('/login', [LoginController::class, 'login']);
Route::post('/logout', [LoginController::class, 'logout'])->name('logout');

Route::get('/register', [RegisterController::class, 'showRegistrationForm'])->name('register');
Route::post('/register', [RegisterController::class, 'register']);
```

Рисунок 3.4 – Приклад маршрутизаторів

Опис функціоналу маршрутизатора.

Головна сторінка:

– Route::get(‘/’, [ProductController::class, ‘index’])->name(‘home’) – відповідає за відображення головної сторінки, де користувачі можуть переглядати список всіх доступних продуктів, викликає метод index в ProductController.

Перегляд деталей продукту:

– Route::get(‘/product/{id}’, [ProductController::class, ‘show’])->name(‘product.show’) – відповідає за відображення деталей конкретного продукту, коли користувач переходить на сторінку продукту за його ідентифікатором, викликає метод show в ProductController.

Додавання продукту до кошика:

– Route::post(‘/cart’, [CartController::class, ‘addToCart’])->name(‘cart.add’) – відповідає за додавання продукту до кошика покупок, коли користувач натискає кнопку «Додати до кошика», викликається метод addToCart в CartController.

Перегляд кошика:

– Route::get(‘/cart’, [CartController::class, ‘viewCart’])->name(‘cart.view’) – відповідає за відображення кошика покупок, де користувачі можуть переглядати всі товари, додані до їхнього кошика, викликає метод viewCart в CartController.

Оформлення замовлення:

– Route::post(‘/order’, [OrderController::class, ‘store’])->name(‘order.store’) – відповідає за оформлення замовлення, після підтвердження кошика користувачі можуть перейти до оформлення замовлення, викликає метод store в OrderController.

Аутентифікація користувачів:

– Route::get(‘/login’, [LoginController::class, ‘showLoginForm’])->name(‘login’) – відображає форму для входу;
– Route::post(‘/login’, [LoginController::class, ‘login’]) – обробляє запит на вхід користувача;
– Route::post(‘/logout’, [LoginController::class, ‘logout’])->name(‘logout’) – вихід з облікового запису.

Реєстрація:

– Route::get(‘/register’, [RegisterController::class,

‘showRegistrationForm’])->name(‘register’) – відображає форму для реєстрації;

– Route::post(‘/register’, [RegisterController::class, ‘register’]) – обробляє запит на реєстрацію нового користувача.

Контролери в Laravel відповідають за обробку HTTP-запитів та управління логікою додатку. Вони реалізовані у вигляді класів, які містять методи, що обробляють конкретні запити і повертають відповіді (див. рис. 3.5).

```php
namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    /**
     * Відображення списку продуктів.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        // Отримання усіх продуктів з бази даних
        $products = Product::all();

        // Повернення виду, який відображає список продуктів
        return view('products.index', compact('products'));
    }

    /**
     * Відображення деталей конкретного продукту.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        // Отримання конкретного продукту за його ідентифікатором
        $product = Product::findOrFail($id);

        // Повернення виду, який відображає деталі продукту
        return view('products.show', compact('product'));
    }
}
```

Рисунок 3.5 – Приклад контролера

В даному коді:

– index(): метод index() отримує всі продукти з бази даних за допомогою моделі Product і передає їх у шаблон виду products.index, в цьому шаблоні буде відображений список усіх доступних продуктів;

– show($id): метод show($id) отримує ідентифікатор конкретного продукту як параметр, знаходить цей продукт у базі даних за допомогою findOrFail($id) і передає його у шаблон виду products.show – цей шаблон відображає детальну інформацію про обраний продукт.

Цей контролер демонструє базові операції зчитування даних з бази даних і передачу їх у відповідні коомопненти для подальшого відображення користувачам

В Laravel модель є основним класом, що представляє таблицю в базі даних. Основні поняття моделі включають:

– зв’язки: модель визначає взаємозв’язки між таблицями бази даних (в Laravel це здійснюється за допомогою методів, які визначають типи зв’язків: hasOne, hasMany, belongsTo, belongsToMany і morphTo);
– масове заповнення: модель дозволяє виконувати масове заповнення атрибутів, що дозволяє ефективно зберігати дані у базі даних – це досягається через властивість $fillable або $guarded;
– запити до бази даних: модель дозволяє легко створювати запити до бази даних за допомогою Eloquent ORM;
– аксесори та мутатори: модель дозволяє визначати методи для отримання або зміни атрибутів перед їх збереженням у базі даних;
– таблиці та міграції: модель відображає таблицю в базі даних і може бути створена або змінена за допомогою міграцій Laravel.

Всі ці концепції роблять модель в Laravel потужним інструментом для роботи з даними у базі даних, забезпечуючи зручний спосіб взаємодії з інформацією та її зберіганням (див. рис. 3.6).

Опис моделі Product включає наступне:

– використання трейту HasFactory: цей трейт дозволяє використовувати фабрики для генерації тестових даних із моделі;
– $fillable: властивість $fillable масив, що вказує на те, які атрибути моделі можна заповнювати масово через методи, як create() або update();

– $casts: властивість $casts конвертує атрибути моделі до певних типів, в даному випадку, поле price конвертується до типу float;
– методи createdBy() та updatedBy(): ці методи визначають відносини між моделями, наприклад, createdBy() повертає об’єкт користувача, який створив цей продукт, використовуючи зв’язок belongsTo до моделі User.

```php
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;

    /**
     * Атрибути моделі, які можна заповнювати масово.
     *
     * @var array
     */
    protected $fillable = ['title', 'description', 'price', 'image'];

    /**
     * Атрибути, які мають бути конвертовані до певних типів.
     *
     * @var array
     */
    protected $casts = [
        'price' => 'float',
    ];

    /**
     * Отримати користувача, який створив цей продукт.
     */
    public function createdBy()
    {
        return $this->belongsTo(User::class, 'created_by');
    }

    /**
     * Отримати користувача, який оновив цей продукт останній раз.
     */
    public function updatedBy()
    {
        return $this->belongsTo(User::class, 'updated_by');
    }
}
```

Рисунок 3.6 – Приклад моделей

Модель Product використовується для взаємодії з таблицею products у базі даних. Вона дозволяє здійснювати операції створення, читання, оновлення та видалення даних про продукти, а також встановлення зв’язків між продуктами та користувачами, що їх створили чи оновили.

3.3 Інтеграція компонентів та налаштування системи

Інтеграція компонентів та налаштування системи в проєкті на Laravel включає в себе кілька ключових аспектів, які дозволяють забезпечити правильну роботу системи із врахуванням всіх її компонентів. Наведемо основні аспекти цього процесу.

Інтеграція фронтенду (Vue.js) із бекендом (Laravel):

– налаштування маршрутів і API ендпоінтів у Laravel для взаємодії з фронтенд-компонентами;
– використання Axios або Fetch API для зроблення HTTP-запитів із фронтенду до бекенду;
– використання Laravel Mix для збірки фронтенд-ресурсів (JavaScript, CSS) та їх інтеграції в Laravel.

Конфігурування середовища:

– налаштування середовища розробки, тестування та продуктивного середовища в файлі .env;
– налаштування міграцій, фабрик, сідерів для автоматизованого наповнення бази даних тестовими даними.

Інтеграція сторонніх сервісів і пакетів:

– використання пакетів Composer для додаткової функціональності (наприклад, автентифікація через OAuth, робота з платіжними системами тощо);
– налаштування інтеграції з зовнішніми API сервісами для обміну даними.

Настройка маршрутизації і політик доступу:

– визначення маршрутів для різних частин системи (наприклад, адміністративна панель, публічні сторінки);
– використання політик доступу для визначення правил доступу до різних ресурсів і функціоналу.

Тестування і відлагодження:

– налаштування тестових середовищ для автоматизованого тестування, включаючи Unit та Feature тести;
– відлагодження коду через інструменти Laravel Debugbar або вбудований дебагер.

Оптимізація і масштабування:

– впровадження кешування (наприклад, Redis) для підвищення продуктивності та швидкодії системи;
– налаштування масштабовування бази даних та вебсервера для обробки більшого обсягу запитів.

Ці аспекти дозволяють не лише інтегрувати окремі компоненти системи, а й забезпечують її ефективну роботу та готовність до впровадження у виробництво.

3.4 Тестування функціональності

Тестування функціональності в проєкті на Laravel включає ряд підходів і інструментів для перевірки правильності роботи окремих функцій, модулів або системи в цілому. Наведемо основні аспекти тестування функціональності.

Unit-тести:

– мета: перевірка правильності роботи окремих функцій, методів і класів в ізоляції від інших компонентів;
– інструменти: використання PHPUnit для написання та запуску тестів. Mock-об’єкти для імітації залежностей.

Feature-тести:

– мета: перевірка правильності взаємодії між компонентами (наприклад, контролерами і моделями) та внутрішньої логіки програми;
– інструменти: використання Laravel Dusk або Laravel HTTP тестів для емулювання HTTP-запитів і перевірки сторінок.

Інтеграційні тести:

– мета: перевірка правильності взаємодії між різними компонентами системи, такими як база даних, зовнішні сервіси, API;
– інструменти: використання Laravel Dusk або власних тестів для тестування інтеграції з базою даних та зовнішніми API.

Тестування відмовостійкості:

– мета: перевірка правильності реакції системи на відмови, помилки або некоректне введення користувачів;
– інструменти: вручні або автоматизовані тести для симуляції відмов та перевірки поведінки системи в таких ситуаціях.

Тестування безпеки:

– мета: перевірка системи на вразливості та відповідність стандартам безпеки;
– інструменти: використання спеціалізованих інструментів для сканування вразливостей, написання тестів для перевірки XSS, CSRF атак.

Автоматизовані тести вебінтерфейсу:

– мета: перевірка коректності відображення інтерфейсу користувача та взаємодії з ним;
– інструменти: використання Laravel Dusk або інших автоматизованих інструментів для емуляції дій користувача та перевірки відповіді системи**.**

На рисунках 3.7 та 3.8 наведено приклади використання Unit-теста для нашого інтрент-магазину.

Цей Unit-тест призначений для перевірки коректності функціоналу додавання продукту до кошика через контролер CartController (рис. 3.7). Розглянемо, як він працює.

Arrange (Підготовка):

– створюється тестовий продукт у базі даних за допомогою фабрики Product;

– цей продукт має унікальний ідентифікатор (id), який передається у запиті на додавання до кошика.

Act (Дія):

– викликається метод post для маршруту /cart з передачею параметрів у масиві [‘product_id’ => $product->id];
– очікується, що контролер CartController правильно обробить цей запит і відповість зі статусом 302 (перенаправлення).

Assert (Перевірка):

– метод assertStatus(302) перевіряє, що HTTP-відповідь має статус 302, що підтверджує успішне додавання продукту до кошика;
– метод assertDatabaseHas перевіряє, що в базі даних існує запис у таблиці cart_items з відповідним product_id, що підтверджує додавання продукту до кошика.

Цей тест допомагає впевнитися, що функціонал додавання товару до кошика працює правильно і коректно взаємодіє з базою даних через контролер CartController.

```php
namespace Tests\Unit\Controllers;

use Tests\TestCase;
use App\Http\Controllers\CartController;
use App\Models\Product;
use Illuminate\Http\Request;
use Illuminate\Foundation\Testing\RefreshDatabase;

class CartControllerTest extends TestCase
{
    use RefreshDatabase;

    public function test_addToCart_method()
    {
        // Arrange: Створення тестового продукту у базі даних
        $product = Product::factory()->create();

        // Act: Виклик методу addToCart контролера CartController з параметрами продукту
        $response = $this->post('/cart', ['product_id' => $product->id]);

        // Assert: Перевірка, що продукт додано до кошика і створено відповідний запис
        $response->assertStatus(302); // Перенаправлення після успішного додавання
        $this->assertDatabaseHas('cart_items', [
            'product_id' => $product->id,
        ]);
    }
}
```

Рисунок 3.7 – Приклад тестування для додавання продукту в кошик

Цей Unit-тест для моделі Product перевіряє метод canAddToCart(), який визначає можливість додавання продукту до кошика (рис. 3.8). Розглянемо, як працює цей код.

```php
namespace Tests\Unit\Models;

use Tests\TestCase;
use App\Models\Product;
use Illuminate\Foundation\Testing\RefreshDatabase;

class ProductTest extends TestCase
{
    use RefreshDatabase;

    public function test_canAddToCart()
    {
        // Arrange: Створення тестового продукту
        $product = Product::factory()->create();

        // Act: Виклик методу addToCart моделі Product
        $canAddToCart = $product->canAddToCart();

        // Assert: Перевірка, що метод повертає true (можливість додавання до кошика)
        $this->assertTrue($canAddToCart);
    }
}
```

Рисунок 3.8 – Приклад тестування методу контролера для перегляду кошика

Arrange (підготовка): створюється тестовий продукт у базі даних за допомогою фабрики Product.

Act (дія): викликається метод canAddToCart() на об’єкті $product, щоб перевірити, чи можна додати цей продукт до кошика.

Assert (перевірка): використовується метод $this->assertTrue($canAddToCart), щоб перевірити, що метод canAddToCart() повертає true, тобто підтверджується, що продукт можна додати до кошика.

Цей тест допомагає перевірити коректність роботи методу canAddToCart() моделі Product і впевнитися, що логіка перевірки можливості додавання продукту до кошика працює правильно.

На рисунку 3.9 ображено головну сторінку інтернет-магазину. У верхній частині сторінки присутнє горизонтальне меню з логотипом та рядком пошуку. У лівій частині екрану розташоване вертикальне меню з різними категоріями товарів:

– комп’ютери та ноутбуки;
– товари для геймерів;
– смартфони та аксесуари;
– техніка для дому;
– побутова техніка;
– товари для дому;
– спорт і захоплення;
– сантехніка та ремонт;
– одяг, взуття та аксесуари;
– дитячі товари.

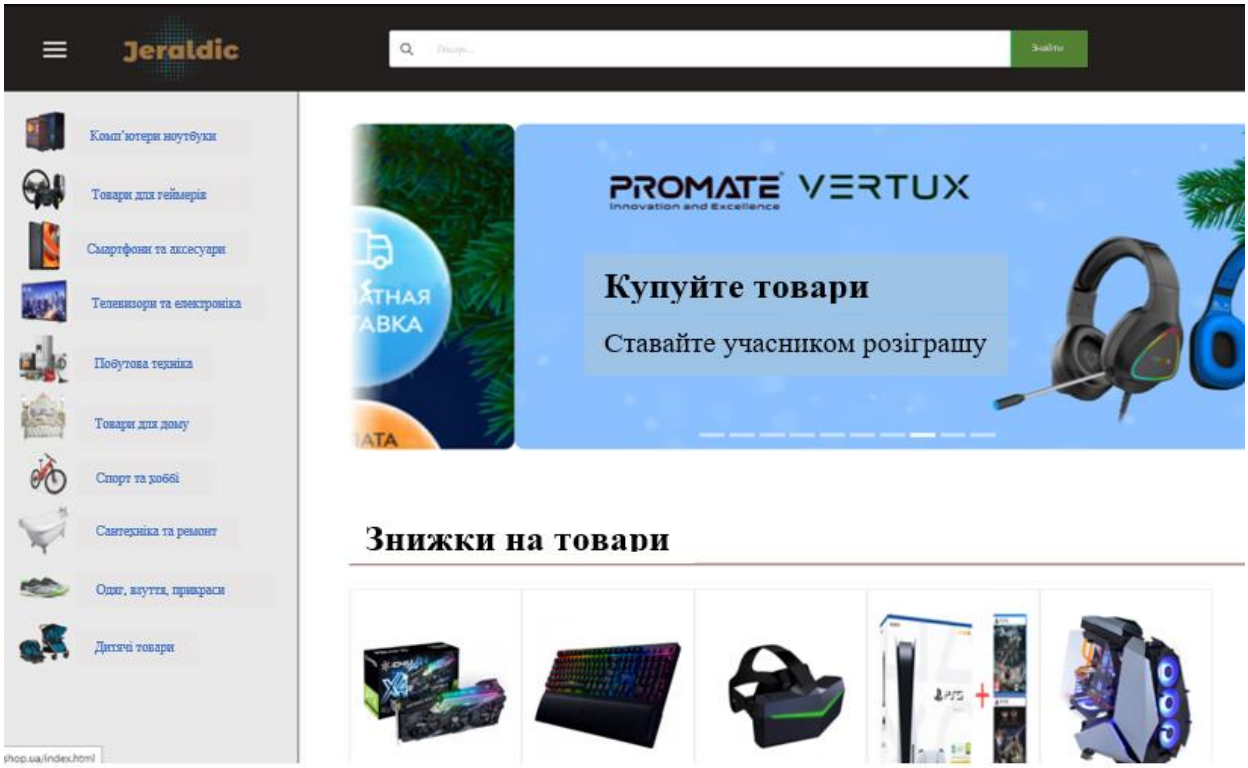

Рисунок 3.9 – Головна сторінка

На рисунку 3.10 зображено сторінку категорій товарів інтернет-магазину. Тут можна побачити різні категорії товарів, представлені за допомогою зображень і підписів під ними. Категорії включають:

– ноутбуки;
– комп’ютери;

– монітори;
– комп’ютерні комплектуючі;
– додаткові девайси;
– мережеве обладнання;
– планшети;
– навушники та аксесуари;
– акустичні системи;
– клавіатури та миші;
– мікрофони;
– аксесуари для електроніки.

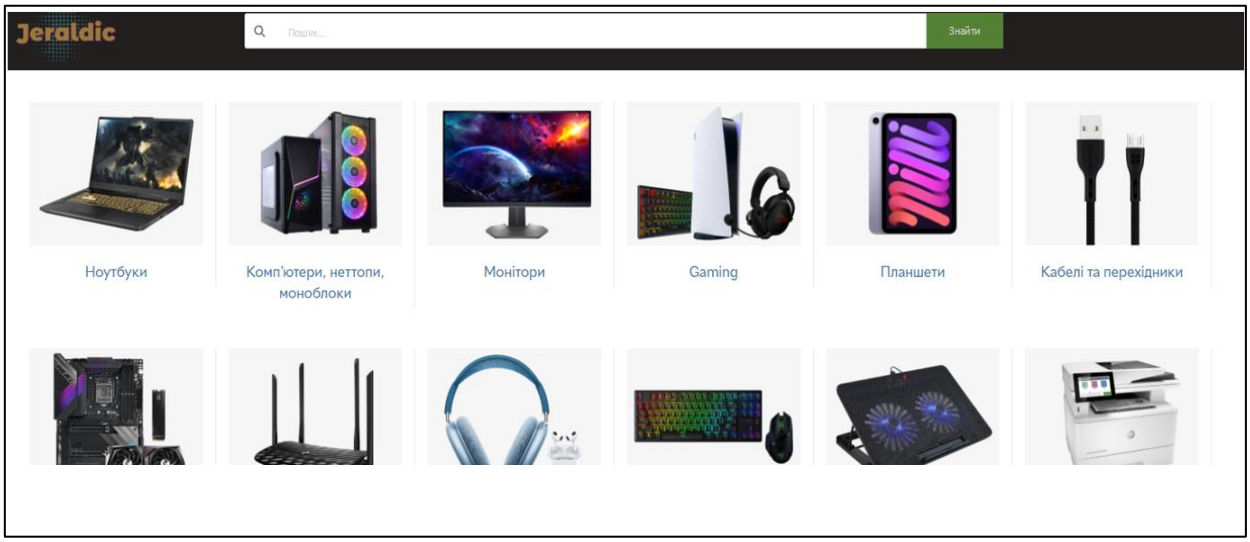

Рисунок 3.10 – Каталог товарів

На рисунку 3.11 зображено сторінку оформлення замовлення. Користувач вводить свої контактні дані, такі як прізвище, ім’я, мобільний телефон та електронна пошта.

Також вказується адреса доставки.

Праворуч на екрані відображається інформація про замовлення, включаючи вартість товарів, доставку та підсумкову суму до оплати. Є можливість ввести промокод для отримання знижки.

Внизу розташована кнопка для підтвердження замовлення.

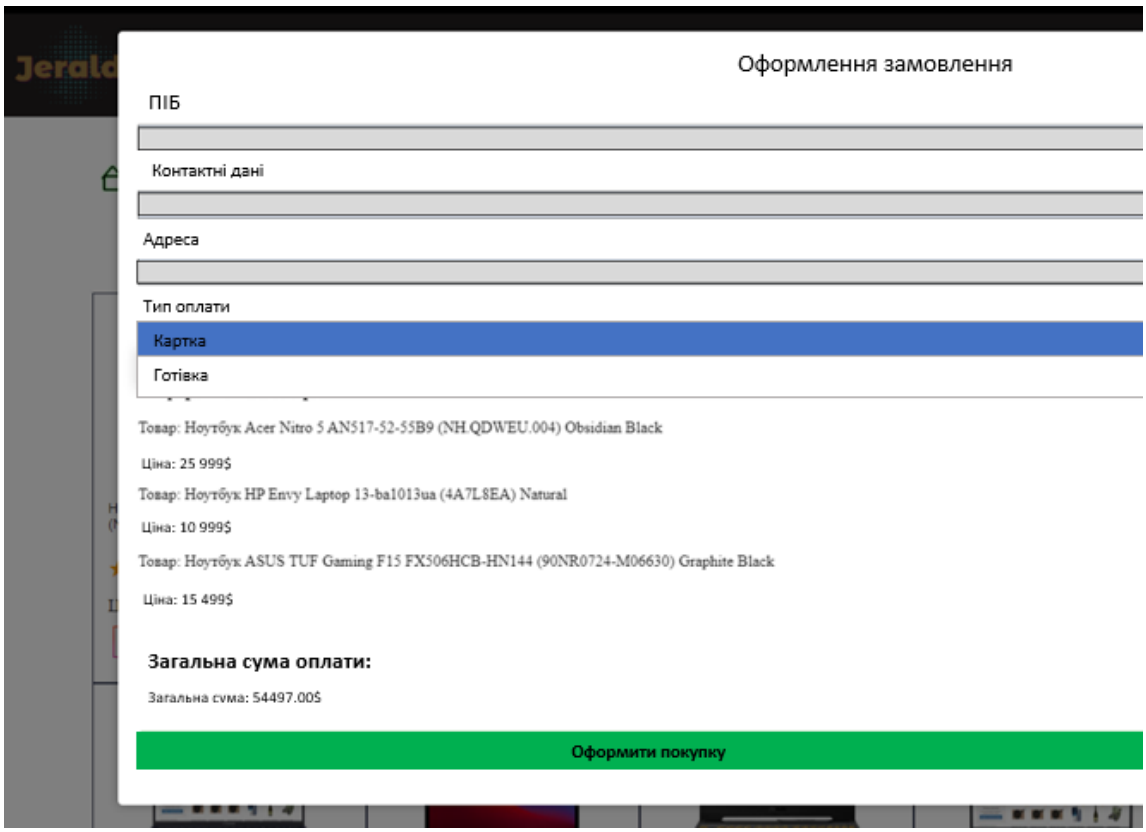

Рисунок 3.11 – Каталог товарів

3.5 Результати тестування та реалізації проєкту

Результати тестування та реалізації проєкту показали високу стабільність і функціональність програмного забезпечення. Під час тестування було виявлено та виправлено всі виявлені помилки, що сприяло покращенню загальної надійності системи. Оптимізація коду та алгоритмів дозволила досягти покращення швидкодії системи, зменшення часу відповіді та ефективне використання ресурсів сервера.

Проєкт успішно реалізував всі основні функціональні вимоги, включаючи каталог продуктів з можливістю швидкого пошуку і фільтрації, кошик покупок з додаванням і видаленням товарів, а також зручний процес оформлення замовлень. Інтерфейс користувача із застосуванням Vue.js забезпечує зручність взаємодії із системою, а адміністративна панель Laravel забезпечує зручний доступ до управління контентом і налаштуваннями.

Загалом проєкт відповідає всім вимогам та очікуванням замовника, що підтверджується успішним завершенням всіх етапів від розробки до випробувань, а також позитивними відгуками від користувачів під час випробування та після випуску в експлуатацію.

ВИСНОВКИ

Команда, яка працювала над розробкою вебзастосунку для нашого інтернет-магазину побутової техніки на базі фреймворків Laravel та Vue, успішно завершила проєкт, відповідно до всіх вимог та очікувань. Процес розробки був підтриманий комплексним аналізом вимог та деталізацією архітектурних рішень на кожному етапі проєкту.

Особлива увага приділялась вирішенню питання мобільної адаптації, що забезпечило високий рівень зручності користувачів незалежно від типу пристрою. Додатково була успішно інтегрована адміністративна панель для зручного управління контентом та користувачами системи.

Вибір технологічного стеку, включаючи Laravel для серверної частини та Vue для клієнтської, показав свою ефективність у вирішенні ключових завдань, таких як валідація даних, управління ролями користувачів і базові операції з даними. Проєкт включав передові практики розробки, такі як CI/CD для автоматизованого тестування, що забезпечило високу якість програмного забезпечення і надійність.

Загальний висновок підкріплює, що розроблений вебзастосунок відповідає всім вимогам, що були поставлені на етапі аналізу, і є функціональним продуктом, здатним ефективно задовольнити потреби користувачів і забезпечити їхнє задоволення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Laravel офіційна документація. URL: https://laravel.com/docs (дата звернення: 06.02.2024).
2. Vue.js офіційна документація. URL: https://vuejs.org/v2/guide/ (дата звернення: 10.02.2024).
3. Курси на Laravel: Laracasts. URL: https://laracasts.com/ (дата звернення: 26.02.2024).
4. Курси на Vue.js: Vue Mastery. URL: https://www.vuemastery.com/ (дата звернення: 15.02.2024).
5. Блог Laravel News. URL https://laravel-news.com/ (дата звернення: 21.02.2024).
6. Vue.js DevTools. URL: https://github.com/vuejs/vue-devtools (дата звернення: 29.02.2024).
7. Laravel Forge для деплою на сервер. URL: https://forge.laravel.com/ (дата звернення: 02.03.2024).
8. Офіційна документація з Laravel Valet s. URL: https://laravel.com/docs/8.x/valet (дата звернення: 03.03.2024).
9. Відеокурси на Laracasts про Vue.js. URL https://laracasts.com/series/learn-vue-2-step-by-step (дата звернення: 14.03.2024).
10. Конференції та заходи з Laravel. URL: https://laracon.net/ (дата звернення: 15.04.2024).
11. Vue Router офіційна документація. URL: https://laracon.net/ (дата звернення: 19.04.2024).
12. Laravel Sanctum для аутентифікації. URL: https://laravel.com/docs/8.x/sanctum (дата звернення: 23.04.2024).
13. Axios для HTTP запитів в Vue.js. URL: https://axios-http.com/docs/intro (дата звернення: 27.04.2024).
14. Віджети та компоненти для Vue.js. URL: https://vuetifyjs.com/ (дата звернення: 29.04.2024).

15. Курси з Laravel на Udemy. URL: https://www.udemy.com/courses/search/?q=laravel (дата звернення: 03.05.2024).
16. Laravel Echo для реального часу. URL: https://laravel.com/docs/8.x/broadcasting (дата звернення: 10.05.2024).
17. Vuex для стану додатку в Vue.js. URL: https://vuex.vuejs.org/ (дата звернення: 10.05.2024).
18. Тема по Laravel на Reddit. URL: https://www.reddit.com/r/laravel/ (дата звернення: 10.05.2024).
19. Тема по Vue.js на Reddit. URL: https://www.reddit.com/r/vuejs/ (дата звернення: 10.05.2024).
20. Laravel: Up & Running. URL: https://www.amazon.com/Laravel-Up-Running-Matt-Stauffer/dp/1492041215 (дата звернення: 10.05.2024).