МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Web-орієнтована система підтримки діяльності онлайн-магазину Еко-товарів

Здобувача групи ІТ.м-24 Шульги Дмитра Петровича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро ШУЛЬГА
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доц. кафедри ІТ, к.т.н., доц.Юлія ПАРФЕНЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Шульзі Дмитру Петровичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Web-орієнтована система підтримки діяльності онлайн-магазину Еко-товарів
затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «11» грудня 2023 р.

3 Вхідні дані до кваліфікаційної роботи _____
_____
_____
_____
_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, моделювання та проектування**,** практична реалізація, висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) _____
_____
_____
_____
_____

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів проекту	Примітка
1	Дослідження предметної області	04.09.23-19.09.22	
2	Формування мети і задач	15.09.23-19.09.23	
3	Аналіз аналогів та проблем використання	20.09.23-27.09.23	
4	Виявлення вимог до проекту	28.09.23-02.10.23	
5	Планування робіт та розробка макету	03.10.23-06.10.23	
6	Вибір засобів реалізації	07.10.23-09.10.23	
7	Проектування інформаційної системи	10.10.23-19.10.23	
8	Розробка інформаційної системи	20.10.23-09.11.23	
9	Тестування та завершення роботи	10.11.23-13.11.23	
10	Оформлення пояснювальної записки	14.11.23-01.12.23	

Магістрант _____ Дмитро ШУЛЬГА

Керівник роботи _____ к.т.н., доц. Юлія ПАРФЕНЕНКО

АНОТАЦІЯ

Пояснювальна записка: 105 с., 45 рис., 2 дод., 62 джерел.

Мета кваліфікаційної роботи: розробка web-орієнтованої системи підтримки діяльності онлайн-магазину Еко-товарів, яка включає аналітику продажів та впровадження системи пошуку товарів.

У першому розділі проведено аналіз предметної області, розглянуті актуальні дослідження та публікації, проведено аналіз аналогів і поставлено завдання. Другий розділ визначає мету та завдання дослідження, включаючи вибір інструментів реалізації. У третьому розділі надано опис проектування системи, включаючи побудову діаграм в нотації IDEF0, діаграму варіантів використання функціоналу інформаційної системи та розроблено модель бази даних. У четвертому розділі представлено реалізацію та детальний опис використання web-орієнтованої системи з боку замовника та адміністратора.

Практичне значення полягає у створенні системи, що забезпечує відвідувачам сайту доступ до каталогу товарі, швидкий пошук товарів, можливість створення замовлення та перегляду особистої інформації, що загалом забезпечує комфортне користування онлайн-магазином. Адміністративна панель магазину дає можливість адміністрації магазину надавати актуальну інформацію щодо наявності та вартості товару, обробляти замовлення та на базі статистики продажів та замовлень проводити аналітику.

Результатом проведеної роботи є створена web-орієнтована система підтримки діяльності онлайн-магазину Еко-товарів.

Список ключових слів: web-орієнтована система, інформаційна підтримка, сайт, онлайн-магазин, браузер.

ЗМІСТ

ВСТУП

Сьогоднішня суспільна динаміка та зростаюча свідомість людей щодо важливості збереження навколишнього середовища визначають актуальність та значущість проектів, спрямованих на розвиток екологічно чистої та органічної продукції. Одним із ключових інструментів, що сприяє поширенню екологічних ініціатив та полегшує доступ споживачів до еко-товарів, є створення інтернет-магазинів, спеціалізованих на еко-продукції.

Зростаючий попит на продукти, які мають мінімальний негативний вплив на навколишнє середовище, свідчить про зростаючий інтерес споживачів до сталого розвитку та відповідального споживання. Інтернет-магазин екологічної продукції може стати ефективним засобом об'єднання виробників, постачальників та споживачів, що дозволить забезпечити ефективну та зручну платформу для обміну екологічною інформацією та продукцією.

Проект включатиме в себе детальний аналіз сучасних тенденцій в галузі екологічних інтернет-магазинів, проектування та розробку інтернет-магазину. Надалі, робота буде акцентуватися на важливості розвитку сталого бізнесу та сприянні змінам в споживацьких звичках на користь екологічної продукції.

Проект спрямований на вивчення, аналіз та практичне застосування інформаційних технологій для створення інноваційних рішень в галузі сталого розвитку та споживання, і він має дати вагомий внесок в поліпшення якості життя та збереження навколишнього середовища.

Кваліфікаційна робота магістра присвячена розробленню web-орієнтованої інформаційної системи підтримки продажу екологічної продукції з метою сприяння збереженню природних ресурсів та поліпшення якості життя сучасного суспільства.

Об’єкт дослідження: використання інформаційних технологій у галузі електронної комерції.

Предмет дослідження: інформаційне забезпечення підтримки продажу еко-товарів.

Метою кваліфікаційної роботи є розробка web-орієнтованої інформаційної системи підтримки діяльності онлайн-магазину Еко-товарів, яка включає аналітику продажів та впровадження системи пошуку товарів. Розроблення системи спрямоване на розширення можливостей пошуку товарів за рахунок впровадження пошукового сервісу та підтримки статистики продажів для проведення аналізу менеджером.

Для досягнення мети необхідно вирішити наступні задачі:

− здійснити аналіз інформаційного забезпечення підтримки діяльності онлайн-магазину Еко-товарів;

− провести аналіз існуючих програмних продуктів та визначити вимоги до проекту;

− провести планування етапів розробки web-орієнтованої інформаційної системи та вибрати інструменти для її втілення;

− виконати проектування та моделювання процесу підтримки діяльності онлайн-магазину Еко-товарів;

− розробити і протестувати web-орієнтовану інформаційну систему підтримки діяльності онлайн-магазину Еко-товарів.

У результаті виконання поставлених завдань буде розроблено web-орієнтовану інформаційну систему, яка має спростити пошук товарів та аналіз кількості продажів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження сучасного стану застосування інформаційних технологій для ведення електронної комерції

На початку 90-х років відбулася перша роздрібна транзакція через Інтернет, і відтоді Інтернет постійно змінював спосіб, у який люди здійснюють покупки [1]. Електронна комерція, термін, абсолютно невідомий кілька десятиліть тому, поступово стає фундаментальною частиною повсякденного життя. Сьогодні майже третина користувачів Інтернету в США роблять покупки онлайн принаймні раз на тиждень, тоді як інша третина купує щось онлайн принаймні раз на місяць [2]. Крім того, протягом останніх п’яти років ці продажі постійно зростали приблизно на 15%. Електронна комерція суттєво впливає на купівельну поведінку та місця проживання. На відміну від традиційного способу здійснення покупок, коли людина шукає та купує продукти в магазині, сьогодні 36% купівельних дій (пошук і купівля) здійснюються за допомогою кількох каналів, ще 43% проводяться виключно в Інтернеті, і лише 21% проводяться в магазинах [2]. Хоча було проведено багато досліджень, які свідчать про взаємодоповнюючий ефект між онлайн-торгівлею та фізичною, лише деякі з них припускають ефект заміщення, а також деякі з них припускають асиметричний вплив одного на інший. Наприклад, автори [3] виявили, що онлайн-пошук позитивно вплинув на покупки в магазинах і, у свою чергу, позитивно вплинув на онлайн-покупки. З іншого боку, у роботі [4] автори виявили, що чим частіше люди купують онлайн, тим нижчі їхні шанси здійснити похід за покупками та зробити покупку в магазині. В іншому дослідженні [5] було показано негативний вплив схильності до покупок у магазині на схильність до покупок онлайн, тоді як має місце позитивний ефект з іншого боку, таким чином припускаючи асиметричний ефект.

Ряд авторів аналізували вплив електронної комерції на навколишнє середовище [1, 6, 7]. Ці дослідження показують, що оскільки вантажівки для доставки оптимізують свої маршрути, електронна комерція має потенціал для зменшення

негативного впливу покупок на навколишнє середовище, а тому є набагато більш стійкою, ніж поїздки до магазинів за покупками на особистих автомобілях. Наприклад, автори [1] оцінили потенціал скорочення пройдених відстаней на 54-93% і скорочення викидів парникових газів на 18-84% завдяки електронній комерції.

Тож інтернет-магазини – це форма електронної комерції, яка дозволяє користувачам безпосередньо купувати товари чи послуги у постачальників через Інтернет за допомогою веб-браузера. Це може бути як невеликий місцевий магазин, так і великий роздрібний торговець, інтернет-магазин електронної комерції або навіть особа, яка продає свої товари через сторонній вебсайт, як eBay. Інтернет-магазин може працювати за різними бізнес-моделями та для своєї роботи вимагає каталогу товарів, кошика, панелі адміністратора та менеджера, та інших компонентів. Сьогодні онлайн-торгівля стає популярною як найзручніший спосіб покупки, особливо серед молодого покоління [8].

На відміну від традиційних покупок, онлайн-торгівля є зручним способом покупки, який дозволяє покупцям придбати бажані речі, не виходячи з дому, не проїжджаючи через пробки, не шукаючи місця для паркування, заощаджуючи на дорозі [9], оскільки не потрібно платити за пальне та дорожні збори. Крім того, онлайн-торгівля також економить час, оскільки користувачам потрібно лише здійснити пошук в пошукових системах, таких як Yahoo! і Google без необхідності відвідувати магазини, а також не стояти в черзі біля стійки. Серед інших переваг онлайн-покупок є те, що користувачі можуть робити покупки, не виходячи зі свого комп’ютера чи планшета; легше порівнювати ціни та використовувати запропоновані знижки і робити покупки в будь-який час [9].

Інтернет-магазин можна розглядати як інформаційну систему, засновану на електронній торгівлі. Як і звичайний магазин, інтернет-магазин відображає товари або послуги, обробляє замовлення та здійснює продаж та доставку. Інтернет-магазини поєднують у собі елементи прямого маркетингу і традиційного магазину. Вони можуть запропонувати велику кількість товарів та послуг, а також надати споживачам значну інформацію, необхідну для прийняття рішення про покупку. Використовуючи

комп'ютерні технології, інтернет-магазини можуть персоналізувати пропозиції для кожного клієнта відповідно до їхньої історії покупок та поведінки на сайті [11, 12].

Основним викликом у реалізації інтернет-магазину є поєднання Інтернет-технологій з традиційною торговельною діяльністю. Незважаючи на багато переваг онлайн-покупок, покупці не можуть побачити та сприйняти на дотик товари, перш ніж прийняти рішення про покупку [2]. Візуальної інформації може бути недостатньо, і тут грають важливу роль емоційні та психологічні аспекти. Проблеми також виникають з доставкою товарів, особливо в разі низьких цін.

Електронну комерцію нині прийнято розділяти на низку напрямів, основними з яких вважаються такі [14]:

1) «бізнес – бізнес» (Business-to-Business – B2B);

2) «бізнес – споживач» (Business-to-Customer або Business-to-Client – B2C);

3) «споживач – бізнес» (Consumer-to-Business – C2B);

4) «споживач – споживач» (Consumer-to-Consumer – C2C);

5) «споживач – адміністрація» (Consumer-to-Administration, C2A);

6) «бізнес – адміністрація» (Business-to-Administration, B2A).

Сьогодні найпопулярніша модель взаємодії "бізнес – споживач" (B2C), де виробник або посередник безпосередньо взаємодіє з кінцевим споживачем продукту чи послуги, забезпечуючи прямі продажі. Цей метод сприяє подоланню географічної віддаленості між великими містами та регіонами, роблячи товари та послуги доступними для споживача. Використання B2C дозволяє проводити прямі продажі без багатьох посередників. Відсутність посередників дає можливість встановлювати конкурентоспроможні ціни на місцях і навіть збільшувати їх (за винятком відсотка посередників), що призводить до зростання прибутку. Модель B2C дає змогу порівняти відмінності в доступі до товарів і послуг між споживачами, що живуть у великих містах і віддалених регіонах, за умови вирішення проблеми доступу в Інтернет і можливості оплати у відповідному регіоні [14].

До систем B2C належать:

– web-вітрини (Front Office) торгових компаній для залучення потенційних покупців до продуктів цих компаній;

– інтернет-магазини, які здійснюють тільки продаж товарів і володіють необхідною інфраструктурою (Back Office) для реалізації продажів, управління електронною торгівлею через Інтернет і контролю її;

– інтернет-майданчики торгівлі з повністю інтегрованою з комплексом торгових бізнес-процесів системою електронних продажів (Back Office).

Модель B2B2C (business to business to consumer, або «бізнес – бізнес – споживач») є видом e-commerce, похідним від B2B і B2C. Сутність таких відносин полягає в тому, що компанія B2B платить компанії B2C, щоб остання пропонувала послуги або товари першої своїм клієнтам. У цьому випадку укладається взаємовигідна угода між підприємствами: компанія B2B відкриває нові канали для реалізації своєї продукції та привертає нових потенційних покупців, тоді як компанія B2C отримує переваги у розширенні асортименту продукції та збільшенні обсягів продажів [14].

1.2 Аналіз підходів до розроблення веб-орієнтованих систем для електронної комерції

При розробці інтернет-магазину важливо обрати правильну платформу, оскільки це визначає багато аспектів: скільки коштів потрібно вкласти у створення сайту, наскільки він буде сучасним і чи буде актуальним через п'ять років, а також як великий трафік можна очікувати.

CMS – це серверна програма, яка зберігає текст вебсторінки та відомості про публікацію в базі даних, а не як сторінки HTML [15]. Сторінки не можуть бути завантажені, доки їх не запитає клієнтський браузер і вони будуть створені на вимогу. За допомогою CMS можна організовувати та публікувати багато типів вмісту, а макет, зовнішній вигляд і структуру сайту можна змінювати легко та швидко, оскільки він заснований на шаблонах, а отже вміст відокремлений від презентації - на відміну від звичайного жорстко-кодованого сайту. Додаткові модулі можна використовувати для додавання функцій і можливостей за потреби. Різним відвідувачам може надаватися різний вміст залежно від їхньої мови чи місця розташування. Взаємодію відвідувачів

можна збільшити порівняно зі статичним або HTML-сайтом. Ця динамічна система явно має численні переваги перед більш звичайним статичним веб-сайтом [16].

Три найпопулярніші CMS в Інтернеті – Drupal, Joomla та WordPress. Усі три мають відкритий вихідний код і побудовані на PHP + MySQL і суттєво відрізняються за функціями, можливостями, гнучкістю та простотою використання [18].

WordPress – це проект із відкритим вихідним кодом, над яким працюють сотні людей, і його можна безкоштовно використовувати як для домашньої сторінки, так і для веб-сайту, не сплачуючи жодної плати за ліцензію [18]. Базовий функціонал платформи мало підходить для створення онлайн-магазину. Але плагін WooCommerce або схожий плагін для ecommerce допоможе зробити інтернет-магазин на WordPress. За вибором користувачів можна сказати, що це найкращий безкоштовний CMS для інтернет-магазину. WordPress, спільно з плагіном WooCommerce, займає лідерське положення у світі за кількістю ресурсів для онлайн-торгівлі. Це оптимальний вибір для невеликих проєктів, проте інтернет-магазин з асортиментом понад 10 000 товарів не зможе ефективно працювати через особливості бази даних [19].

Переваги: відкритий код; велика кількість розробників; безліч тем і плагінів для розширення функціоналу; багато тематичних сайтів і форумів з питань адміністрування сайту і проблем, пов’язаних з движком; легко управляти.

Недоліки: немає офіційної техпідтримки; спочатку CMS створена не під інтернет-магазин, тому потрібно попрацювати над функціоналом сайту; велика частина функціоналу плагіна WooCommerce і допомога його техпідтримки коштує грошей; висока вразливість, але це вирішується додатковими налаштуваннями сайту [17].

Drupal - це визнана та безкоштовна платформа для розробки інтернет-магазинів, що спрощує процес вибору шаблонів і розширень для вашого веб-сайту. Наприклад, VirtueMart, Eshop або J2Store - це модулі, які ефективно вирішують різні завдання у сфері електронної комерції [17].

Ця CMS під інтернет-магазин підійде для невеликих електронних ресурсів з асортиментом в кілька сотень товарів. Щоб створити великий магазин, потрібно використовувати платні плагіни, наприклад, RedSHOP.

Переваги: безкоштовність і відкритий код; широкий вибір модулів і тем; простота використання адмін-панелі.

Недоліки: зайвий код, який уповільнює сайти, навантажує процесор і базу даних – потрібний потужний хостинг; відсутність офіційної технічної підтримки; складність налаштування SEO; налаштування вимагає технічних знань [17].

Для створення інтернет-магазину на платформі Drupal, достатньо встановити розширення Drupal Commerce. Крім того, існує багато інших модулів, які дозволять налаштувати функціонал сайту. А дизайн можна легко коригувати за допомогою кастомізації шаблонних тем.

Drupal - це платформа, сприятлива для SEO, що підтримує канонічні URL, автоматично генерує карти сайту та дозволяє налаштовувати метадані сторінок. Додатково, движок підтримує кешування сторінок, покращуючи продуктивність завантаження онлайн-магазину [17].

Переваги включають просту інтеграцію, стабільну сумісність з будь-яким веб-сервером, обширний вибір безкоштовних шаблонів і тем, зручний інтерфейс, регулярні оновлення платформи та більш високий рівень захисту коду порівняно з іншими безкоштовними CMS.

Серед недоліків можна відзначити потребу у потужному хостингу, нескладність адміністративної панелі, обмежений вибір якісних тем дизайну, вимоги до навичок роботи з FTP при встановленні та оновленні модулів, а також високий рівень складності для новачків, оскільки рушій орієнтований на професіоналів у веб-розробці [17].

Іншим підходом до розробки може бути використання фрейморків. Використання фреймворків обумовлено необхідністю швидкої розробки програм [20], сприяючи повторному використанню коду, тестуванню та змінам у програмі. Існує багато популярних і широко використовуваних для розробки фреймворків, написаних різними мовами, які побудовані на одній структурі, що полегшує вивчення

та розуміння фреймворків. Кожен з них має переваги та забезпечує конкретну реалізацію, яку слід брати до уваги, щоб зробити свій вибір [21].

Термін «фреймворк» у випадку веб-фреймворків – це набір бібліотек і інструментів, які можуть покращити дизайн веб-додатків, додаючи додаткові функції та чіткість у розробці. Фреймворк також дозволяє автоматизувати завдання, оскільки він інтегрує низку підпрограм, реалізованих нативно. Таким чином, використання фреймворка гарантує узгоджену архітектуру, де строгість розробки є першочерговою [22, 23].

Фреймворк полегшує веб-програмування та покращує його організацію багатьма способами. Перш за все, фреймворки підвищують продуктивність програмування, тому що написання фрагмента коду, який зазвичай займає години і займає сотні рядків коду, можна виконати за хвилини за допомогою вбудованих функцій фреймворку. По-друге, широко використовувана структура має велику перевагу в безпеці, оскільки її користувачі стають довгостроковими тестувальниками. Якщо користувач виявив проблему безпеки, він може повідомити про це на веб-сайті розробників фреймворку, щоб команда розробників могла її виправити. По-третє, часто більшість популярних фреймворків безкоштовні, і оскільки вони допомагають розробнику швидше писати код, кінцева вартість буде меншою. По-четверте, структура зазвичай постачається разом із командою підтримки, документацією або великими форумами підтримки, де користувачі можуть швидко отримати відповіді. Вибір фреймворку – важливий крок, адже від нього залежить швидкість і якість роботи. Основні аспекти, які слід взяти до уваги, це контекст використання, ліцензія, модель програмного забезпечення, вимоги до хостингу, простота встановлення, основна бібліотека, крива навчання, абстракція БД і ORM, включені бібліотеки JS тощо [24].

Не всі фреймворки задовольняють однакові потреби, і в деяких ситуаціях кілька фреймворків можна використовувати разом, наприклад, фреймворки для веб-додатків (Django, Ruby on Rails, Symfony або Laravel) [25].

Популярними мовами програмування для розробки веб-сайтів є PHP, C#, Ruby, Python і Java. Оскільки PHP є популярною та широко використовуваною мовою, вона

має багато документації в Інтернеті. Об'єктно-орієнтований вихідний код, написаний на PHP, є елегантним і простим у обслуговуванні. Все більше і більше нових проектів використовують фреймворки PHP, і успішне завершення цих проектів зробило фреймворки PHP ще більш популярними (рис. 1.1).

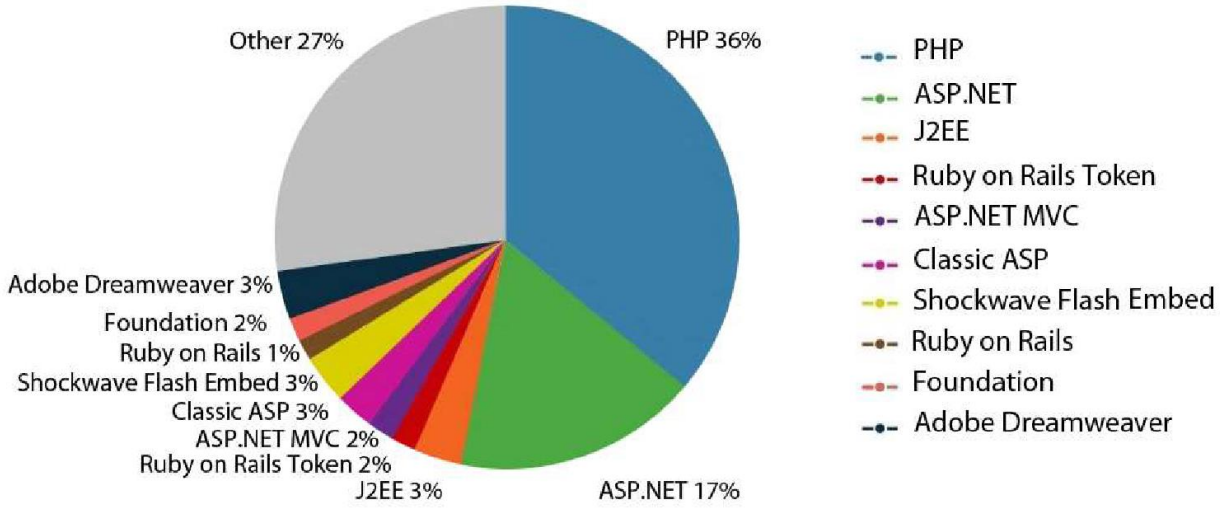


Рисунок 1.1 - Статистика використання фреймворків на 1 мільйоні найкращих вебдодатків
Джерело: [25]

PHP фреймворки спрощують розробку веб-додатків, написаних на PHP, надаючи базову структуру. Іншими словами, PHP фреймворки допомагають сприяти швидкій розробці вебдодатків, заощаджуючи час, допомагаючи створювати більш стабільні додатки та зменшуючи кількість повторюваного кодування [26]. Фреймворки також можуть допомогти початківцям створювати більш стабільні програми, забезпечуючи хорошу взаємодію та кодування бази даних. Це дозволяє витрачати більше часу на створення фактичної веб-програми замість того, щоб витрачати час на написання повторюваного коду.

У роботі [24] було перевірено декілька фреймворків середовища програмування PHP (рис. 1.2).

	CakePHP2	CodeIgniter	Symfony2	Yii	Phalcon
Newest version	3.3.6	3.1.0	3.1.4	2.0.9	3.0.0
Release date	2016 Oct 14	2016 Jul 26	2016 Sep 3	2016 Jul 11	2016 Jul 29
License	MIT	BSD-style	MIT	BSD	New BSD License
Requirements	>= PHP 5.2.8	>= PHP 5.1.6	>= PHP 5.5.9	>= PHP 5.4	>= PHP 5
ORM	Custom	ActiveRecord	Doctrine 2, Propel	Database Access Objects (DAO), Active Record	Phalcon
Code generation	CLI	Combustor	CLI	Yii CLI, Gii (Web based)	Phalcon Developer Tools
Template engine	Custom, bet Smarty/Twig	PHP, Simple template parser "{var_name}"	PHP, Twig	Razor, Smarty, Twig etc.	Volt, PHP
CRUD generation	Scaffolding	Grocery CRUD	SensioGenerator Bundle	Gii	Scaffolding
Web-site	cakephp.org	codeigniter.com	symfony.com	yiiframework.com	phalconphp.com

Рисунок 1.2 - Порівняння фрейморків

Джерело: [24]

Кожен тип PHP фреймворка має свої переваги. Кілька факторів, на які слід звернути увагу, включають: простоту використання, швидку розробку/продуктивність, популярність серед інших розробників, потужні функції та підтримку, форуми. Усі фреймворки дещо відрізняються та мають різні сильні та слабкі сторони [26, 27]. Однак вибір правильного фреймворку поміж різними - іноді може бути складним.

Symfony є PHP фреймворком, який головним чином використовується для створення основи для великих проектів, таких як розробка власних фреймворків або вирішення невеликих завдань для веб-розробників. Symfony складається з набору незалежних компонентів, які можна повторно використовувати в різних проектах [28].

Цей фреймворк дозволяє гнучко налаштовувати практично будь-яку систему під конкретні потреби, хоча в ньому обмежена кількість готових рішень для типових завдань. У мережі Інтернет є багато додатків та бібліотек, створених для Laravel та Yii2, які використовують Symfony.

Переваги: постійно оновлювана та широка документація; безліч незалежних компонентів для повторного використання; система функціональних та модульних тестів для виявлення помилок у веб-додатках; придатність для великих проектів.

Symfony дозволяє встановлювати сторонні пакети, бібліотеки, компоненти і налаштовувати їх за допомогою конфігурацій у форматах YAML, XML, PHP та файлів .env. Symfony надає можливість вести журнал помилок додатків та підключати бібліотеку Monolog для логування. Цей фреймворк часто використовується для довгострокових складних проектів. Він вимагає більше фінансових ресурсів, але надає можливість налаштувати проект з урахуванням індивідуальних потреб.

Недоліки: високий поріг входу; повільне завантаження деяких додатків; більше часу потрібно для тестування. Навіть при наявності обширної документації, початківцю знадобиться час для вивчення великої кількості компонентів та набуття необхідних навичок для їх використання. Велика кількість незалежних компонентів може негативно позначитися на швидкодії деяких додатків, а також, через потребу в попередньому компілюванні коду для повторного використання, тестування може займати більше часу.

Yii – це універсальний фреймворк і може бути задіяний у всіх типах веб-додатків, що використовують PHP. Завдяки його компонентній структурі і відмінній підтримці кешування, фреймворк особливо підходить для розробки таких великих проектів як портали, форуми, системи керування вмістом (CMS), інтернет-магазини або RESTful-додатки [29, 30].

Переваги: низький поріг входу; вбудовані механізми аутентифікації користувачів; вбудовані рішення для створення інтерфейсів; має якісну підсистему відкладеної ініціалізації (тобто код завантажується лише тоді, коли він необхідний); генератори моделей та контролерів. Так як Yii є фреймворком повного стеку, він має безліч перевірених та готових до використання функцій.

Недоліки: обмежена гнучкість при визначенні маршрутів; повільний розвиток; сильний зв'язок між компонентами.

У порівнянні з іншими фреймворками Yii має деякі обмеження у визначенні маршрутів, включаючи як шляхи до контролерів, так і налаштування правил. Крім

того, для групування маршрутів потрібно створювати окремий клас. Багато готових рішень для інтерфейсу безпосередньо вставляють сценарії в тіло сторінок, а їх код містить суміш PHP та HTML. Це може стати джерелом проблем при підтримці та взаємодії зі сторонніми додатками.

Django – доступний веб-фремворк для створення вебдодатків на Python. Django базується на шаблоні Model-Template-View, а також забезпечує поділ версії звіту, що надходить від бізнес-правил, а також інтерфейсу [31]. Сайти на Django будуються з однієї або декількох частин, які рекомендується розробляти модульними. Ця особливість архітектури відрізняє Django від деяких інших фреймворків.

Переваги: багатий функціонал «з коробки»; стандартизована структура; вбудовані інструменти для створення API (Django REST Framework). Django має багатий функціонал (ORM, аутентифікація користувачів, міграції баз даних, адміністраторська панель) для вирішення багатьох завдань веб-розробки.

Django надає фреймворку структуру проекту, що допомагає розробникам зрозуміти, де і як додавати новий функціонал. Завдяки одній і тій же структурі для всіх проектів легше знаходити готові рішення та отримувати допомогу від комьюніті.

Django дозволяє розділити проект на додатки, які додаються в налаштуваннях. Цей підхід спрощує інтеграцію готових рішень. Крім того, фреймворк включає захист від поширених атак, таких як SQL-ін'єкції і міжсайтові запити (CSRF).

Недоліки: монолітна платформа; не підходить для маленьких проектів; django ORM- транзакції відключені за замовчуванням; повільний розвиток; високий вхідний поріг.

Django – це великий і монолітний фреймворк. Це дозволяє спільноті розробників створювати сотні універсальних модулів і додатків, але призводить до сповільнення розвитку самого Django. Крім того, фреймворк має підтримувати зворотну сумісність, тому його розвиток відбувається повільно.

Laravel представляє собою відкритий PHP-фреймворк загального призначення, який використовується для розробки веб-додатків відповідно до шаблону MVC (Model-View-Controller). Він є одним із найбільш використовуваних фреймворків

PHP. Веб-дизайн за допомогою фреймворку Laravel дуже легкий і простий, але потрібно знати основи PHP, щоб зрозуміти структуру фреймворку [32].

Таблиця 1.1 – Порівняння PHP фреймворків

Laravel	- Для новачка це найкращий фреймворк для роботи через широкий спектр документації. - Надійні пакети шифрування. - Модульне тестування - Він пропонує величезну екосистему, яка забезпечує вирішення всіх помилок.
CodeIgniter	- Модульне тестування не є дружнім. - Є лише кілька вбудованих бібліотек. - Немає просторів імен.
CakePHP	- Розробка Restful API не така хороша, як у порівнянні з Laravel.
Zend	- Швидка розробка програм не ідеальна.
Fuel PHP	- Обмежена пропозиція спільноти з відкритим кодом, ніж Laravel. - Недружній для новачка.
Phalcon	- Не має відкритого коду, як Laravel. - Для вирішення проблеми з помилками потрібні розробники.
Symphony	- Структура MVC не підтримується.

Джерело: побудовано на основі даних зі статті [33]

Переваги Laravel включають: масштабованість веб-додатку; зберігання часу, оскільки Laravel кешує та повторно використовує компоненти. Тому додатки, створені на Laravel, володіють вищою продуктивністю порівняно з додатками, створеними на інших фреймворках. Він містить простори імен та інтерфейси, які

допомагають керувати ресурсами. Запит до бази даних можна використовувати за допомогою необробленого SQL, вільного конструктора запитів і Eloquent ORM. Планувальники команд дозволяють вільно визначати розклад команд. Він надає уніфікований API для різних серверних частин черги. Найголовніше, що це дозволяє переносити базу даних з однієї платформи на іншу без зайвих зусиль. Він також містить простий спосіб заповнення бази даних початковими даними. Ця функція Laravel робить його відмінним і ефективним від інших фреймворків PHP. Додаткові особливості включають: легке масштабування додатків; високий рівень безпеки додатків, включаючи захист від SQL-ін'єкцій та міжсайтових атак; шаблони, які дозволяють багаторазово використовувати один і той же код у різних частинах інтерфейсу користувача; Artisan консоль, яка має набір команд для роботи з контролерами, міграціями, моделями та іншими компонентами фреймворку.

Завдяки Laravel легко створити веб-сайт, оскільки для цього не потрібно вводити код з нуля, як це робиться на звичайному PHP. В Laravel більшість роботи зроблено, просто потрібно ввести зміни відповідно до бізнес-вимог. Також велика популярність Laravel допомагає вирішувати питання і розв'язувати проблеми швидше і ефективніше.

Використання фреймворків для розробки web-орієнтованої інформаційної системи може бути перевагою порівняно з використанням систем управління контентом (CMS). Основні переваги фреймворків включають гнучкість, високий рівень контролю, та можливість створення унікальних та індивідуалізованих рішень.

Фреймворки дозволяють розробникам повністю контролювати структуру та поведінку свого коду. Це дає можливість створювати високопродуктивні та гнучкі рішення, спеціально пристосовані під конкретні потреби магазину у порівнянні з CMS, де обмежений функціонал може обмежити можливості налаштувань та внесення змін.

Фреймворки надають розробникам широкий спектр контролю над кожним елементом системи, включаючи базу даних, шаблони, та логіку програми. Це важливо для розширення та оптимізації функціоналу магазину відповідно до конкретних бізнес-потреб.

За допомогою фреймворків розробники можуть легко створювати індивідуальні рішення, які точно відповідають потребам конкретного бізнесу. У той час як CMS може забезпечувати стандартні рішення, вони можуть бути менш гнучкими для вирішення унікальних завдань або потреб.

Фреймворки дозволяють оптимізувати продуктивність шляхом вибору оптимальних рішень для конкретного завдання. Це може призвести до покращення швидкодії та ефективності магазину, особливо в умовах великого обсягу даних чи великої кількості користувачів.

Хоча використання CMS може бути підходящим для простих магазинів, вибір фреймворку стає кращою альтернативою за рахунок високого рівня контролю, гнучкості, та індивідуальності у розробці онлайн-магазину.

1.3 Порівняльна характеристика онлайн-магазинів еко-товарів

Для аналізу були вибрані кілька веб-сайтів з аналогічною тематикою та призначенням. Один із таких ресурсів – це веб-сайт еко-товарів Nowaste Shop https://shop.nowaste.com.ua/ , зображений на рис 1.3.

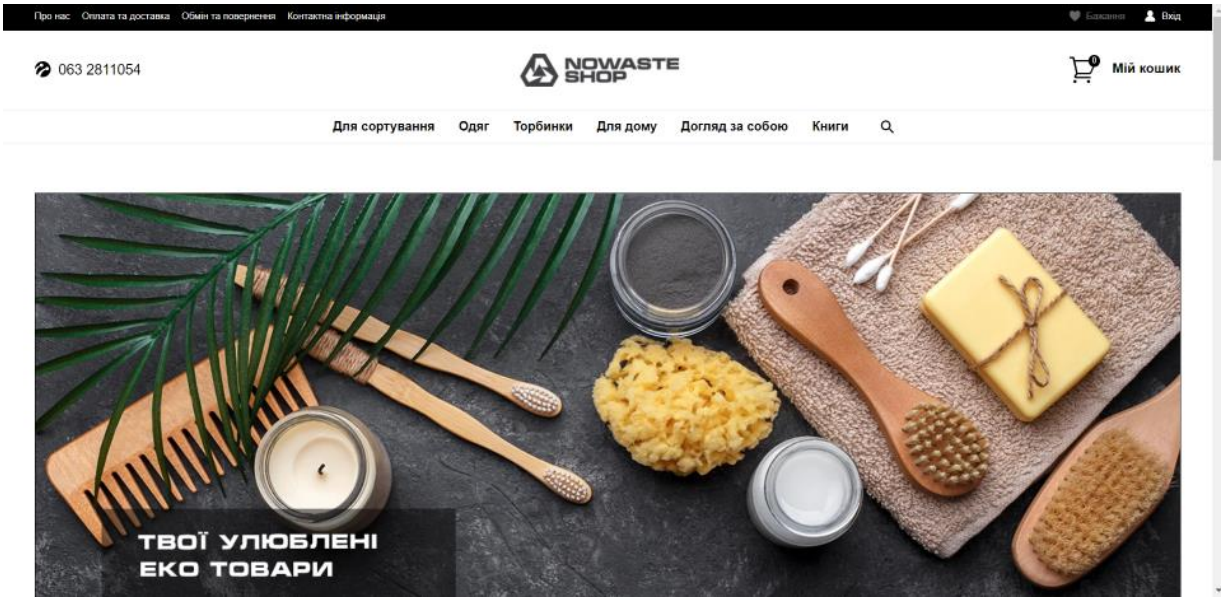


Рисунок 1.3 – Домашня сторінка веб-сайту Nowaste Shop

Джерело: побудовано автором (знімок з екрану)

Оскільки цей магазин спеціалізується на еко-товарах, то особлива увага приділяється опису товару та його призначенню, тобто з яких матеріалів зроблений та чи може бути повторно перероблений. Важливо відзначити, що це також може вплинути на ціну товару. Значущою є також маркетингова складова, яка може використовуватися для стимулювання продажів. До неї може включатися збільшення зображень товару, оскільки збільшення зображень грає важливу роль у прийнятті рішення щодо вибору товару.

До переваг та особливостей цього інтернет-магазину відносяться: зручність оформлення замовлень; наявність мобільної версії вебсайту; широкий асортимент товарів; можливість реєстрації користувачів; можливість створення каталогу улюблених товарів; можливість пошуку товарів.

Недоліками інтернет-магазину Nowaste Shop є неефективність зворотного зв'язку, складність роботи з магазином. Крім того, пошук відбувається тільки за назвою товару, не підтримує синоніми, повільний, немає автодоповнення.

Схожою за тематикою та призначенням є інша платформа UAMET https://uamet.com.ua/ , рис. 1.4. Платформа продає Еко хімію для дому, прибирання в квартирі. Це безфосфатні пральні порошки, засоби для миття посуду, підлоги, вікон і дзеркал, хромованих поверхонь, кухні, ванної кімнати, туалету.

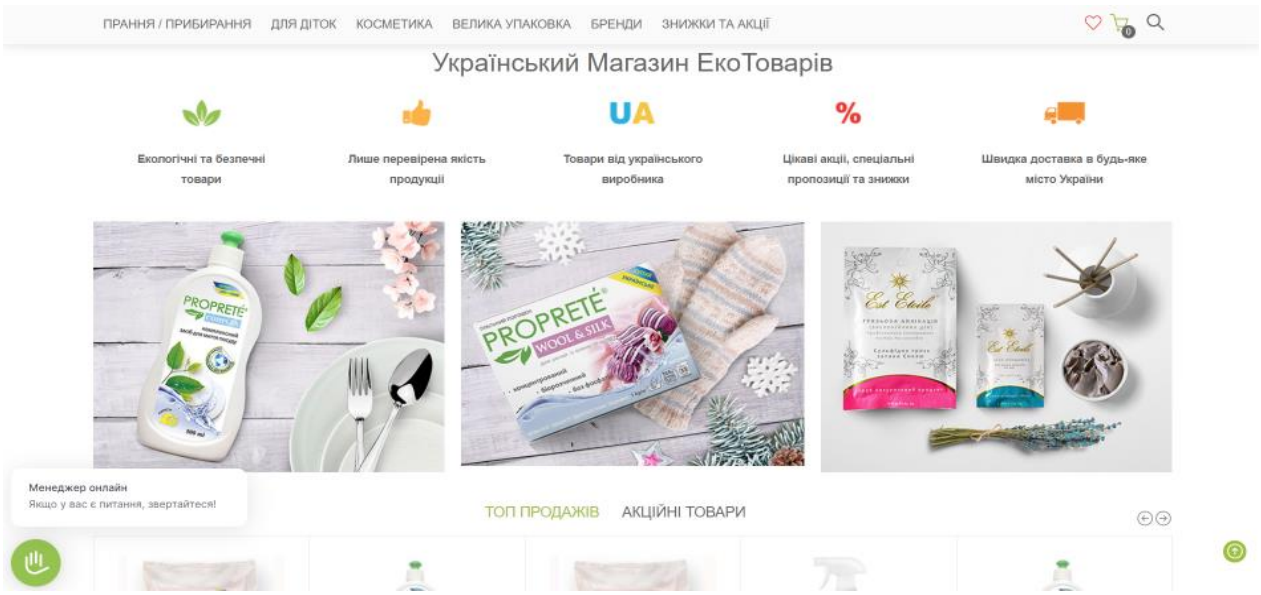


Рисунок 1.4 – Головна сторінка веб-сайту UAMET

Джерело: побудовано автором (знімок з екрану)

Основний акцент робиться на безпечну для здоров'я та навколишнього середовища продукцію, що зберігає природні ресурси.

Плюси цього сайту включають зручний інтерфейс, можливість зворотного зв'язку та функцію збільшення зображення товару. На даному сайті присутній детальний опис товару, що важливо для покупців при прийнятті рішення. Також можна побачити оцінки та відгуки покупців.

Як одним з недоліків можна вважати вузьку спеціалізацію магазину, тільки побутова хімія та косметика. Але в той же час це спрощує пошук та вибір товарів, доступ до замовлень та обробку. Пошук відбувається тільки за назвою товару, не підтримує синоніми. Також у цьому магазині відсутня адаптація інтерфейсу вебсайту для осіб із вадами зору.

Для аналізу був обраний веб-сайт VITO STORE https://vito.store/ , який також спеціалізується на продажу еко-товарів.

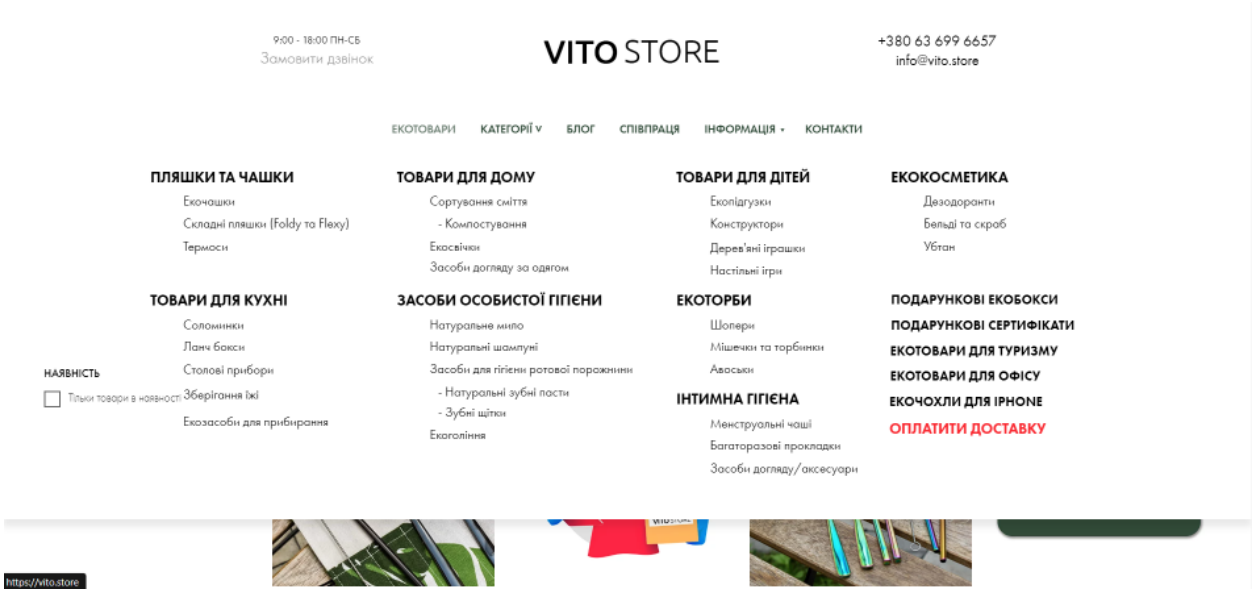


Рисунок 1.5 – Категорії магазину VITO STORE

Джерело: побудовано автором (знімок з екрану)

Цей магазин має свої недоліки, такі як неможливість зареєструватися та дивитися історію замовлень або отримувати особисті бонуси. Він також має простий та бідний дизайн, при цьому є докладний опис товарів.

Проте до переваг цього онлайн-магазину можна віднести зручний пошук, який повертає не тільки товари за назвами але й товари в опису яких зустрічається пошукове слово.

Наступним розглянемо ECOGRIZZLY https://ecogrizzly.shop/

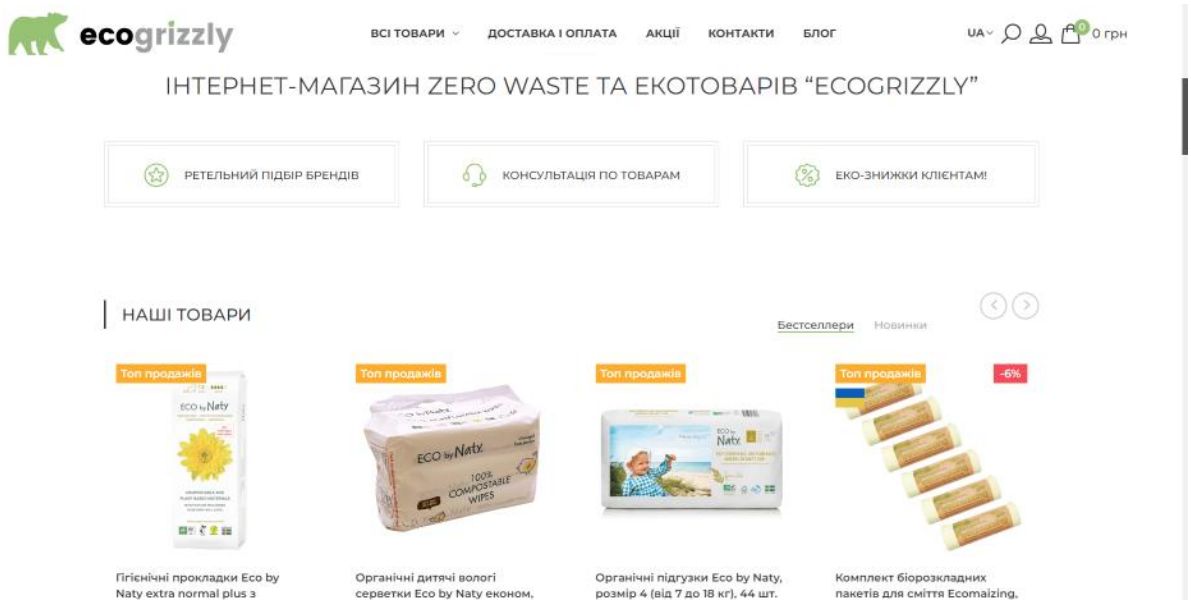


Рисунок 1.6 – Категорії магазину ECOGRIZZLY

Джерело: побудовано автором (знімок з екрану)

Переваги та особливості інтернет-магазину включають: можливість цілодобового замовлення товарів; наявність мобільної версії вебсайту; дуже широкий асортимент; реєстрацію користувачів; можливість пошуку товарів; різноманітні пропозиції для покупця, як то еко-набори, еко-бонуси, еко-еквіваленти, дропшипінг тощо. Хоча пошук відбувається й детальному описі товари, він є повільним.

Таблиця 1.2 – Порівняння програмних продуктів

Критерії	Nowaste Shop	UAMET	VITO STORE	ECOGRIZZLY
Великий вибір товарів	+	-	+	+
Простота використання	-	+	+	+
Актуальна інформація	+	-	+	+
Автоматична обробка замовлень	+	-	+	+
Реєстрація користувачів	+	+	-	+
Можливість замовлення без авторизації	+	+	+	+
Наявність маркетингових пропозицій	-	-	-	+
Повний та швидкий пошук	-	-	+	-

Джерело: побудовано автором

На сьогоднішній день існує значна кількість вебдодатків з аналогічною функціональністю, але вони мають свої недоліки. Аналіз багатьох з них дозволяє виділити певні проблеми, такі як: застаріла інформація або дизайн, невисока швидкість обробки замовлень. Багато з них мають різноманітний асортимент товарів, що може ускладнити вибір покупців тому потребують мати високоякісний пошук товарів. Тому створення спеціалізованої web-орієнтованої інформаційної системи може бути цілком виправданим.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі розробки

Метою кваліфікаційної роботи є розробка web-орієнтованої системи підтримки діяльності онлайн-магазину Еко-товарів. Ця платформа сприятиме ефективному пошуку еко-товарів та їх замовленню, тобто буде спрощувати взаємодію між клієнтами та постачальниками товарів.

Цільовою аудиторією web-орієнтованої системи є, з одного боку, особи, які зацікавлені в придбанні екологічних товарів, тобто потенційні клієнти -замовники, з іншого – менеджери або постачальники товарів, які формуватимуть каталог товарів, оброблятимуть замовлення, тобто працюватимуть з базою даних, а також, на основі зібраної статистики замовлень та продажів, проводитимуть аналітичні прогнози .

Система повинна бути реалізований як вебсайт, що доступний в мережі інтернет, бути простою у використанні як для фахівця, так і для клієнта. Вебсайт має включати в себе авторизацію та розділення на дві ролі: клієнтів та адміністраторів.

Функціональні вимоги до web-орієнтованої інформаційної системи:

- Реєстрація та авторизація користувачів;
- Поділ користувачів на ролі: клієнт, менеджер;
- Можливість оформити доставку;
- Підтримка каталогу товарів з ієрархією за категоріями;
- Відображення детального опису товару;
- Оформлення замовлення за допомогою кошика покупця;
- Розробка пошукової системи товарів;
- Підтримка статистики замовлень та продажів;
- Відображення адміністративного розділу (доступ обмежений за роллю користувача):
    - менеджер додає, редагує та видаляє товари та каталоги товарів,
    - менеджер переглядає та обробляє замовлення;
    - менеджер переглядає статистику замовлень та продажів.

Веб-інтерфейс повинен бути інтуїтивно зрозумілим та відображати структуру інформації на сайті, дозволяючи швидко та логічно переміщатися між розділами. Правильна організація інформації забезпечує користувачам безпроблемне переглядання сайту і впевненість, що вони можуть легко повернутися до попередньо переглянутих сторінок.

Головною особливістю web-орієнтованої системи є можливість ведення статистики замовлень та продажів. Статистика дозволяє проаналізувати кількість переглядів та замовлень певного товару протягом поточного та минулого місяців поденно; проценте відношення замовлень до переглядів товару протягом року помісячно; рейтинг товарів, тобто кількість переглядів товару за весь час, топ 5 та останні 5. Дана аналітика розширює можливості роботи менеджера для прогнозування продажів, замовлень певних товарів або категорій товарів.

Інша відмінність – використання пошукової системи для реалізації швидкого та актуального пошуку товарів.

2.2 Вибір засобів реалізації

Розробка будь-якого програмного продукту розпочинається з вибору інструментів для реалізації проекту. При цьому слід брати до уваги такі критерії, як легкість освоєння, зручність використання і спектр можливостей, які надає обраний інструмент.

З огляду на це було вирішено використовувати Visual Studio Code як редактор коду. Цей редактор відзначається зручним інтерфейсом і можливістю розширення функціональності за допомогою різноманітних плагінів, що сприяє спрощенню процесу розробки.

Для розробки серверної частини був обраний фреймворк Laravel. Цей фреймворк ґрунтується на мові програмування PHP, одній з найпопулярніших мов для генерації сторінок HTML на стороні веб-сервера. Оскільки проект передбачає роботу з великим обсягом даних, було вирішено використовувати систему управління

базами даних MySQL, яка забезпечує широкий функціонал для роботи з динамічними веб-сторінками. В якості пошукової системи обрано MeiliSearch.

Для розробки клієнтської частини проекту був обраний Bootstrap – набір інструментів для створення веб-сайтів. Bootstrap включає в себе CSS-шаблони, HTML-код та підтримує використання мови програмування JavaScript. Для відображення графіків статистики буде використовуватися бібліотека Chart.js.

Visual Studio Code – це редактор вихідного коду, який можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, параметри та встановлювати розширення з VS Code Marketplace, які додають функціональність [34, 35].

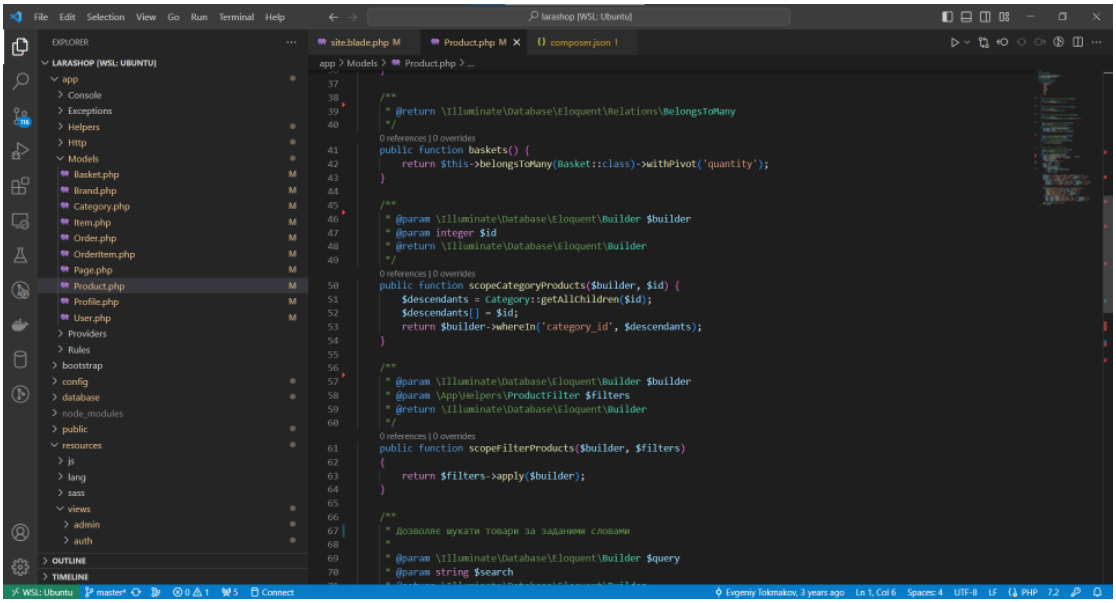


Рисунок 2.1 – Інтерфейс редактору кода

Джерело: побудовано автором (знімок з екрану)

Замість системи проектів цей редактор дозволяє користувачам відкривати один або кілька каталогів, які потім можна зберегти в робочих областях для подальшого використання. Це дозволяє йому працювати як мовно-агностичний редактор коду для

будь-якої мови. Він підтримує багато мов програмування та набір функцій, який відрізняється для кожної мови. Небажані файли та папки можна виключити з дерева проекту за допомогою налаштувань. Багато функцій Visual Studio Code не доступні через меню чи інтерфейс користувача, але доступ до них можна отримати через палітру команд.

Код Visual Studio можна розширити за допомогою розширень, доступних через центральне сховище. Це включає доповнення до редактора та підтримку мови. Примітною функцією є можливість створювати розширення, які додають підтримку нових мов, тем, налагоджувачів, які виконують статичний аналіз коду та додають лінтери коду за допомогою протоколу Language Server [34].

Керування джерелом є вбудованою функцією Visual Studio Code. Він має спеціальну вкладку всередині панелі меню, де користувачі можуть отримати доступ до налаштувань керування версіями та переглянути зміни, внесені до поточного проекту. Щоб використовувати цю функцію, Visual Studio Code має бути зв’язано з будь-якою підтримуваною системою керування версіями (Git, Apache Subversion, Perforce тощо). Це дозволяє користувачам створювати репозиторії, а також робити запити push і pull безпосередньо з програми Visual Studio Code.

Laravel – PHP-фреймворк, який використовується для створення сайтів будь-якої складності. Це програмна платформа з відкритим вихідним кодом, що скорочує час розробки і є одним з найбезпечніших фреймворків у світі. Розробка сайтів на Laravel дуже популярна [36].

Дуже важливою особливістю Laravel для цього проекту є використання архітектури Model-View-Controller (MVC). Ця архітектура розділяє програму на три незалежні частини: модель, контролер та представлення. Вони працюють окремо і можуть використовуватися незалежно один від одного. Це допомагає уникнути ситуацій, коли виправлення помилок у логіці впливає на старий код та призводить до нових помилок у багатьох місцях. Розробникам важко передбачити всі зв'язки та можливі наслідки свого коду.

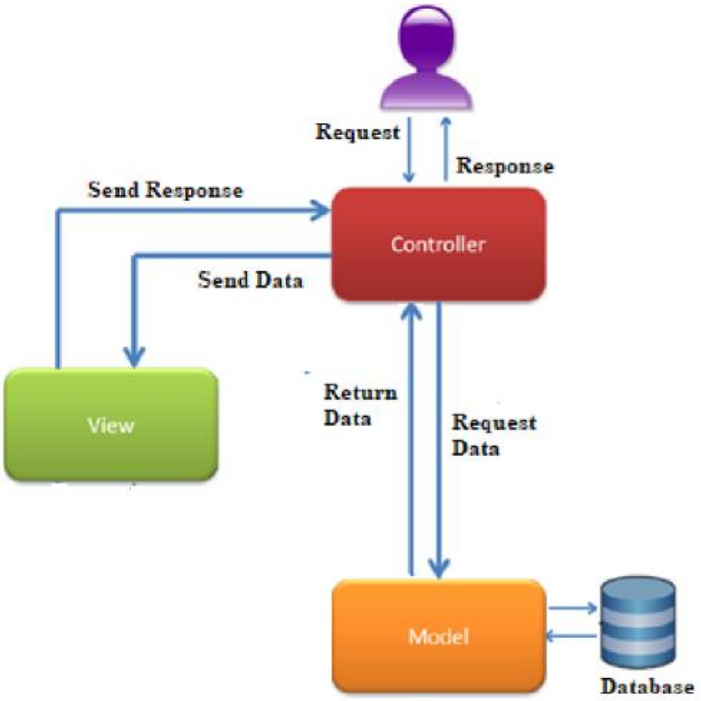


Рисунок 2.2 - Model –View – Controller архітектура
Джерело: [37]

Розробка на Laravel – це універсальне рішення для різних проектів. Вона дозволяє використовувати перевірені рішення, створені для проектів різних напрямків. Це не заважає вам створювати власні рішення з нуля, але, безумовно, заощаджує час, якщо терміни розробки сайту є одним з найважливіших питань у проекті.

MySQL – це система керування базами даних. У реляційній базі даних дані зберігаються не всі разом, а в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів – це найбільш поширена стандартна мова, яка використовується для доступу до баз даних.

MySQL є системою клієнт-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API) [38].

Для безпосередньої роботи з базою даних проєкту використовувався MySQL Workbench. Інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безкоштовне оточення для системи баз даних MySQL [39].

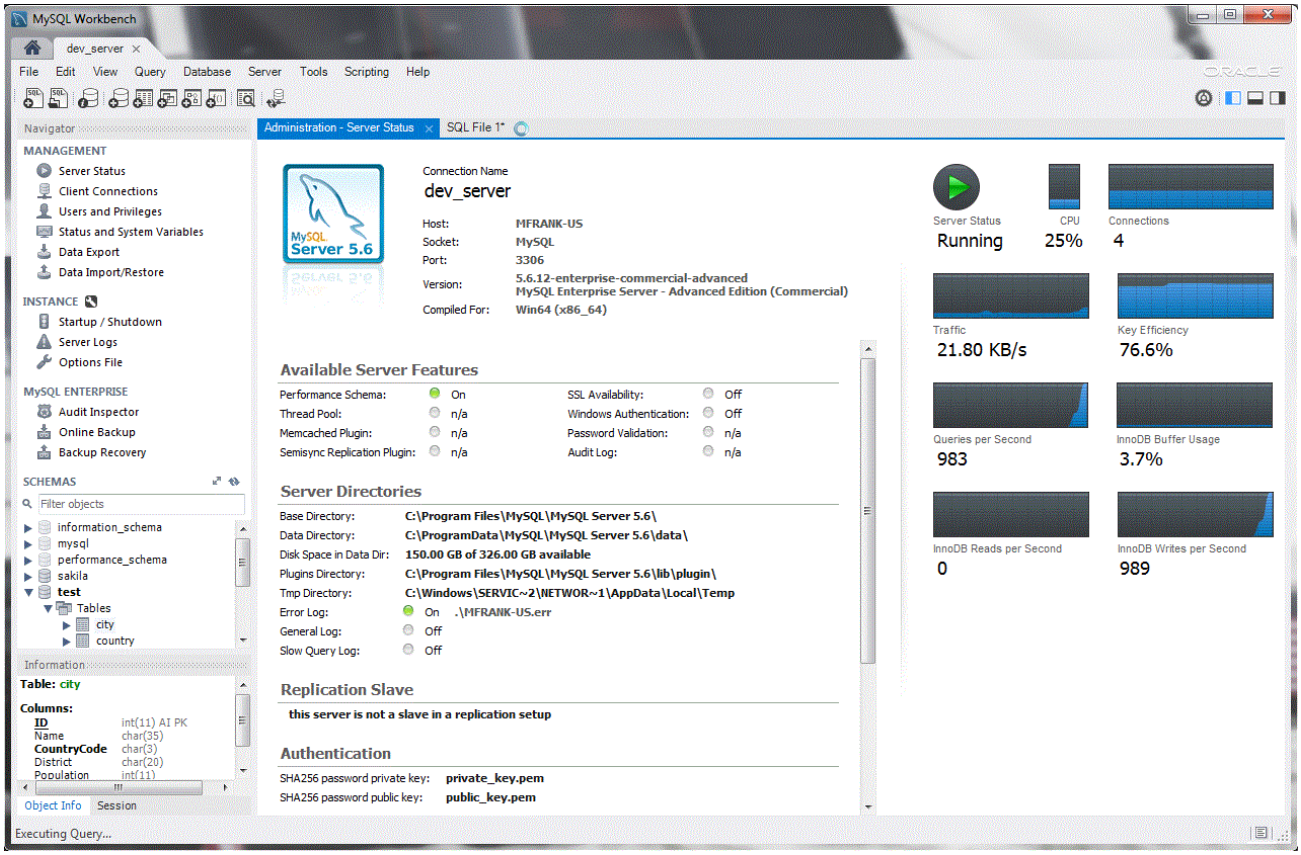


Рисунок 2.3 – Інтерфейс MySQL Workbench

Джерело: побудовано автором (знімок з екрану)

Можливості програми [40]:

- Дозволяє наочно представити модель бази даних в графічному вигляді.
- Наочний і функціональний механізм установки зв'язків між таблицями, в тому числі «багато до багатьох» із створенням таблиці зв'язків.
- Reverse Engineering – відновлення структури таблиць з вже існуючої на сервері БД (зв'язки відновлюються в InnoDB, при використанні MyISAM зв'язки необхідно встановлювати вручну).
- Зручний редактор SQL запитів, що дозволяє відразу ж відправляти їх серверові і отримати відповідь у вигляді таблиці.

- Можливість редагування даних у таблиці в візуальному режимі.

Bootstrap – це безкоштовний відкритий фреймворк для HTML, CSS та JS, який використовується для швидкого створення адаптивних дизайнів для веб-сайтів і веб-додатків [41]. Цей фреймворк включає в себе готові HTML та CSS шаблони для створення веб-форм, кнопок, навігаційних меню та інших веб-інтерфейсних компонентів, а також JavaScript-розширення.

Після підключення Bootstrap до проекту доступні різноманітні класи та готові компоненти, які допомагають швидко створювати сучасний адаптивний дизайн для веб-сайту. Класи Bootstrap можна розділити на групу для створення адаптивної сітки сторінки; групу для стилізації контенту, такого як текст, код, зображення, таблиці тощо; група для вирішення допоміжних завдань, таких як вирівнювання, управління відображенням, додавання кордонів тощо.

Крім класів, у Bootstrap також є готові компоненти, такі як кнопки, форми, навігаційні меню, випадаючі списки, спливаючі панелі та інші. Використання Bootstrap також полегшує адаптацію веб-сайту для відображення на різних пристроях.

Для побудови діаграм обрана бібліотека Chart.js. Серед багатьох JavaScript бібліотек діаграм для розробників додатків Chart.js наразі є найпопулярнішою згідно [42]. Chart.js надає набір часто використовуваних типів діаграм, плагінів і параметрів налаштування. Окрім розумного набору вбудованих типів діаграм, можна використовувати додаткові типи діаграм, які підтримуються спільнотою. Крім того, можна об’єднати кілька типів діаграм у змішану діаграму (по суті, змішуючи кілька типів діаграм в одну на одному полотні).

Chart.js легко налаштовується за допомогою користувальницьких плагінів для створення анотацій, масштабування або функцій перетягування. Chart.js постачається зі стандартною конфігурацією за замовчуванням, що спрощує початок і отримання програми, готової до виробництва.

Візуалізація Canvas робить Chart.js дуже продуктивним, особливо для великих наборів даних і складних візуалізацій, які інакше вимагали б тисячі вузлів SVG у дереві DOM. У той же час візуалізація полотна не дозволяє використовувати стилі CSS, тому доводиться використовувати для цього вбудовані параметри або створити

спеціальний плагін або тип діаграми, щоб відобразити все на свій смак. Зрештою, рендеринг canvas, який використовує Chart.js, зменшує навантаження на DOM дерево порівняно з рендерингом SVG.

2.3 Огляд пошукових сервісів

Для реалізації швидкого інтелектуального пошуку товарів зробимо огляд існуючих пошукових сервісів. Загальні пошукові системи можна розділити на три широкі категорії, які шукають веб-сайти, включаючи Google, Yandex, Yahoo і Baidu; соціальні мережі, такі як WeChat, Facebook і Instagram, а також соціальні мережі, наприклад Netflix, Youtube і Tiktok. Проте є деякі обмеження, які мають комерційні пошукові системи. Вони мають близьке джерело, важко налаштовувати функції пошуку та мають менше функцій для розробників. Пошукові системи з відкритим кодом сприяють популярності та розвитку пошукових технологій, водночас змушуючи все більше і більше людей розуміти пошукові технології та сприяти їх використанню. Використання пошукових систем з відкритим вихідним кодом може значно скоротити цикл побудови пошукових програм і створити персоналізовані пошукові програми відповідно до вимог або навіть побудувати системи пошукових систем, які відповідають конкретним потребам [43].

MeiliSearch – це пошуковий RESTful API, який пропонує ретельно підготовлене рішення для тих, хто хоче надати своїм кінцевим споживачам швидкий і актуальний пошук. Крім того, MeiliSearch може підтримувати низку мов, включаючи англійську, японську та романську. MeiliSearch може задовольнити потреби більшості. Він надає функції швидкого пошуку, такі як виправлення друкарських помилок, фільтри, спеціальні ранги тощо.

Нижче наведено деякі з основних атрибутів MeiliSearch. По-перше, він виконує пошук, поки користувачі вводять текст, що зазвичай називають «швидким пошуком». Дослідники можуть отримати результати, ввівши вміст, який вони хочуть запитати. Крім того, відображені результати змінюватимуться зі збільшенням вмісту введеного тексту. Друга особливість полягає в наявності фільтрів. У Meilisearch користувачі

можуть встановлювати фільтри, щоб додатково обмежити та покращити обсяг своїх результатів пошуку.

Meilisearch надає клієнтам можливість сортувати результати запиту, щоб вони могли вибрати, які результати відображати першими. Крім того, правила асоціації MeiliSearch пропонують простий пошук із кількома параметрами. Навіть якщо люди введуть неправильне слово під час пошуку, MeiliSearch усе одно зможе знайти бажані результати. Нарешті, використання синонімів допомагає користувачам спростити та змінити процес пошуку [43].

Apache Solr – це корпоративний пошуковий сервер із відкритим кодом на основі Java, який легко розширювати та змінювати. Він використовує бібліотеку пошуку Lucene Java для повнотекстового індексування та пошуку. Існує три етапи роботи Apache Solr – це індексування, запит та ранжування результатів.

Перший крок – індексування. Завдяки обробнику користувачі можуть безпосередньо завантажувати файли у форматах JSON, XML/XSLT або CSV у Solr. Електронні листи, RSS-канали, XML-дані, звичайні текстові файли та інші джерела також можна використовувати для імпорту даних. Використовуючи імена стовпців як імена полів документа, плагін DataImport Handler від Solr може витягувати та індексувати дані з бази даних. Наступним кроком є запит. Наприклад, люди можуть шукати ключові слова, зображення або інформацію про місцезнаходження. Коли вони надсилають запит Solr, обробник обробляє його. Обробник запиту виконує подібні завдання до обробника індексу, але замість завантаження документів він повертає їх з індексу Solr. Наступним кроком Solr упорядкує результати на основі релевантності та відображає найбільш релевантні результати вгорі для перегляду користувачами, коли пошуковий вміст, який ми надаємо, збігається з індексним документом [44].

Подібно до інших пошукових систем з відкритим вихідним кодом, Apache Solr надає низку функцій, включаючи повнотекстовий пошук і підтримку XML, JSON і звичайного зв'язку HTTP. Крім того, він надзвичайно стійкий до збоїв і масштабований. Він підтримує англійську, німецьку, китайську та японську мови.

Однак він має деякі обмеження. Apache Solr можна використовувати лише в приватних мережах, оскільки він не має автентифікації та авторизації. Найважливіше

те, що головний вузол Solr потрібно кожного разу переналаштувати, якщо він виходить з ладу.

ElasticSearch – це бібліотека Lucene та система повнотекстового пошуку, як і інші відкриті пошукові системи. Механізм розподіленого пошуку та аналізу знаходиться в центрі Elastic стека [45]. Управління та отримання документо-орієнтованих напівструктурованих даних – це те, як працює ElasticSearch. Обернене індексування – це техніка, яка швидко знаходить документи, які містять певне ключове слово, зіставляючи кожне окреме «слово» (тег) зі списком документів, які його містять. Один або кілька розділів містять інформацію про індекс (також відомий як фрагмент). ElasticSearch має здатність тиражувати та динамічно розподіляти фрагменти по вузлах кластера [46].

ElasticSearch пропонує дві переваги: він масштабований і стійкий. Додатковими функціями є висока доступність і кластеризація. Кластер складається з одного або кількох вузлів. Коли вузол виходить з ладу, кластери ElasticSearch використовують основні сегменти та шарди-репліки для забезпечення відновлення після відмови. Горизонтальна масштабованість, автоматичне відновлення вузла, автоматичне перебалансування даних, крос-кластерна реплікація та реплікація між центрами обробки даних є додатковими функціями ElasticSearch [46].

Для реалізації пошуку у web-орієнтованій системі онлайн-магазину Еко-товарів було обрано MeiliSearch.

Переваги MeiliSearch для розробників включають його масштабованість, зручність обслуговування та адаптивність. Meilisearch вимагає лише кількох налаштувань для запуску та роботи. Для більшості проектів достатньо стандартних параметрів Meilisearch. З іншого боку, пошук продовжує бути гнучким. Крім того, процес пошуку має бути простим, щоб користувачі могли зосередитися на результатах пошуку. Meilisearch прагне запропонувати простий процес пошуку з часом відповіді менше п’ятдесяти мілісекунд. Meilisearch додатково прискорює та оптимізує процес пошуку, дозволяючи споживачам зосередитися на результатах пошуку [43].

Незважаючи на численні переваги, не слід забувати про його обмеження. Кожен пошуковий запит перевіряє максимум десять термінів. У запит MeiliSearch можна ввести лише десять слів. Навіть якщо їх більше десяти, їх не шукатимуть, оскільки відповіді на запити з багатьма пошуковими термінами можуть зайняти більше часу. Крім того, база даних за замовчуванням може мати розмір до 100 ГБ. Meilisearch використовує дві бази даних: одну для роботи та іншу для зберігання даних. Документи із занадто великою кількістю полів створюють занадто велику внутрішню структуру даних, що призводить до великої бази даних на диску та повільної роботи пошуку. В індексі зберігання документів також обмежено. Загалом він може вмістити 4294967296 документів. Це максимальна кількість документів, збережених в індексі, оскільки внутрішня ідентифікація документів у системі Meilisearch використовує номери без знаку. Тим не менш більшість людей можуть скористатися швидким і зручним пошуком завдяки невеликому розміру корисного навантаження [43].

3 ПРОЕКТУВАННЯ WEB-ОРІЄТОВАНОЇ СИТЕМИ

3.1 Структурно-функціональне моделювання

Модель будується для того, щоб краще розуміти розроблювану систему. Моделювання дозволяє вирішити чотири різні завдання [47]:

1. Візуалізувати систему в її поточному або бажаному для замовника стані.
2. Описати структуру або поведінку системи.
3. Отримати шаблон, що дозволяє сконструювати систему.
4. Документувати прийняті рішення, використовуючи отримані моделі.

Чим більша і складніша система, тим більшого значення набуває моделювання при її розробленні. Без моделі складну систему неможливо сприйняти як одне ціле. Моделювання системи чи об’єкта дозволяє звузити проблему, зосередивши увагу в кожен момент тільки на певних її аспектах. Складне завдання завжди легше вирішити, якщо розділити його на менші. Моделювання підсилює можливості людського інтелекту. Правильно обрана модель дозволяє створювати проекти на вищих рівнях абстракції [47].

IDEF0 – нотація графічного моделювання, що використовується для створення функціональної моделі, що відображає структуру та функції системи, а також потоки інформації та матеріальних об'єктів, що пов'язують ці функції. Методологія IDEF0 одна із популярних підходів до опису бізнес-процесів. До її особливостей можна віднести: використання контекстної діаграми; підтримка декомпозиції; домінування; виділення 4 типів стрілок [48].

Структурно-функціональне моделювання процесу створення web-орієнтованої системи підтримки діяльності онлайн-магазину Еко-товарів можна представити за допомогою контекстної діаграми (рис. 3.1). Це найвища діаграма, на якій об'єкт моделювання представлений єдиним блоком із граничними стрілками. Ця діаграма називається A-0 (А мінус нуль). Стрілки на цій діаграмі відображають зв'язки об'єкта моделювання з довкіллям. Діаграма A-0 встановлює область моделювання та її межу [49].

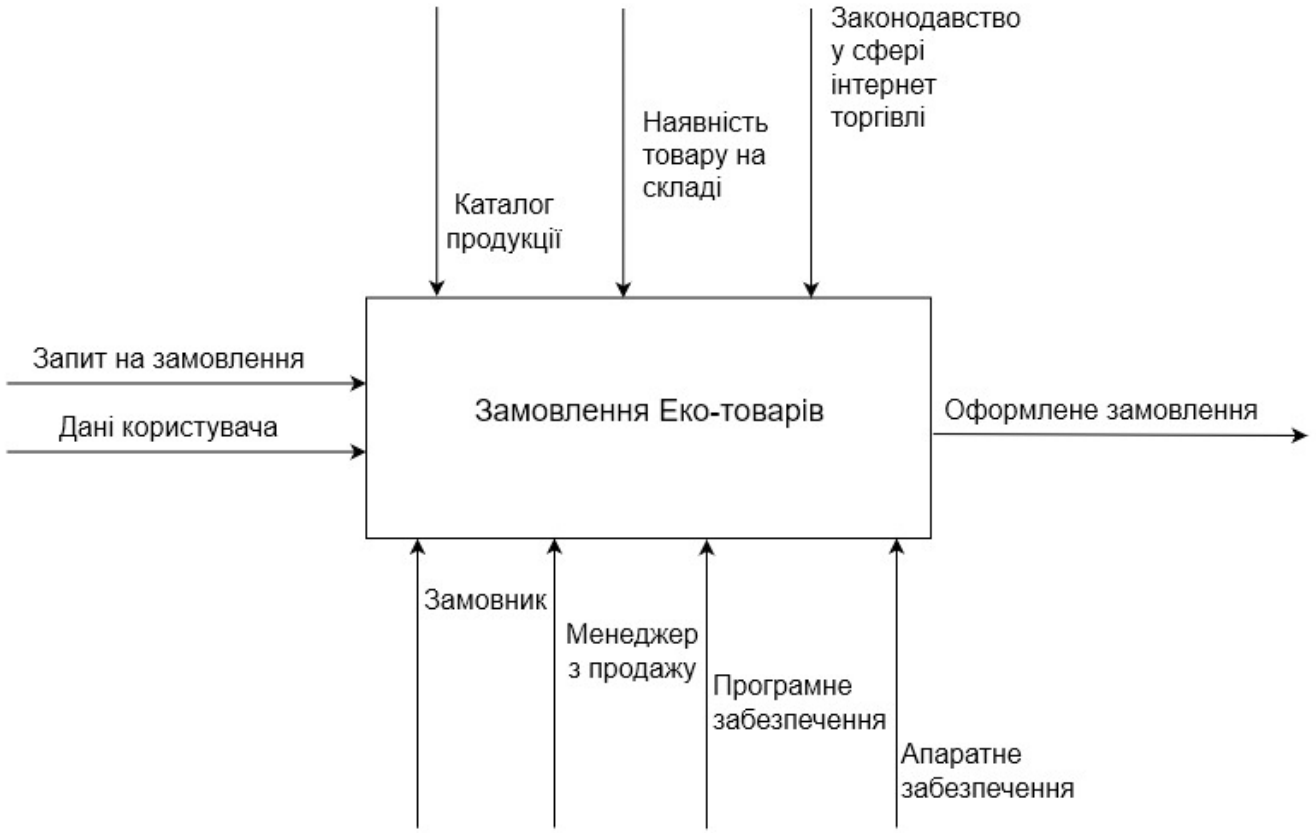


Рисунок 3.1 – Контекстна діаграма процесу замовлення Еко-товарів

Джерело: побудовано автором

Виділяються такі типи стрілок: "Вхід", "Вихід", "Механізм", "Керування". Входи перетворюються або витрачаються процесом, щоб створити те, що з'явиться на його виході. Управління визначають умови, необхідні процесу, щоб зробити правильний вихід. Виходи – дані чи матеріальні об'єкти, вироблені процесом. Механізми ідентифікують засоби, що підтримують виконання процесу. Таким чином, блок IDEF0 показує перетворення входу у вихід за допомогою механізмів з урахуванням впливів, що управляють [49].

Після успішного створення контекстної діаграми, необхідно провести ретельний аналіз основних завдань проекту. Обрані завдання служитимуть основою для створення діаграми на першому рівні. Результат виконання кожного завдання повинен служити вихідними даними для наступного завдання.

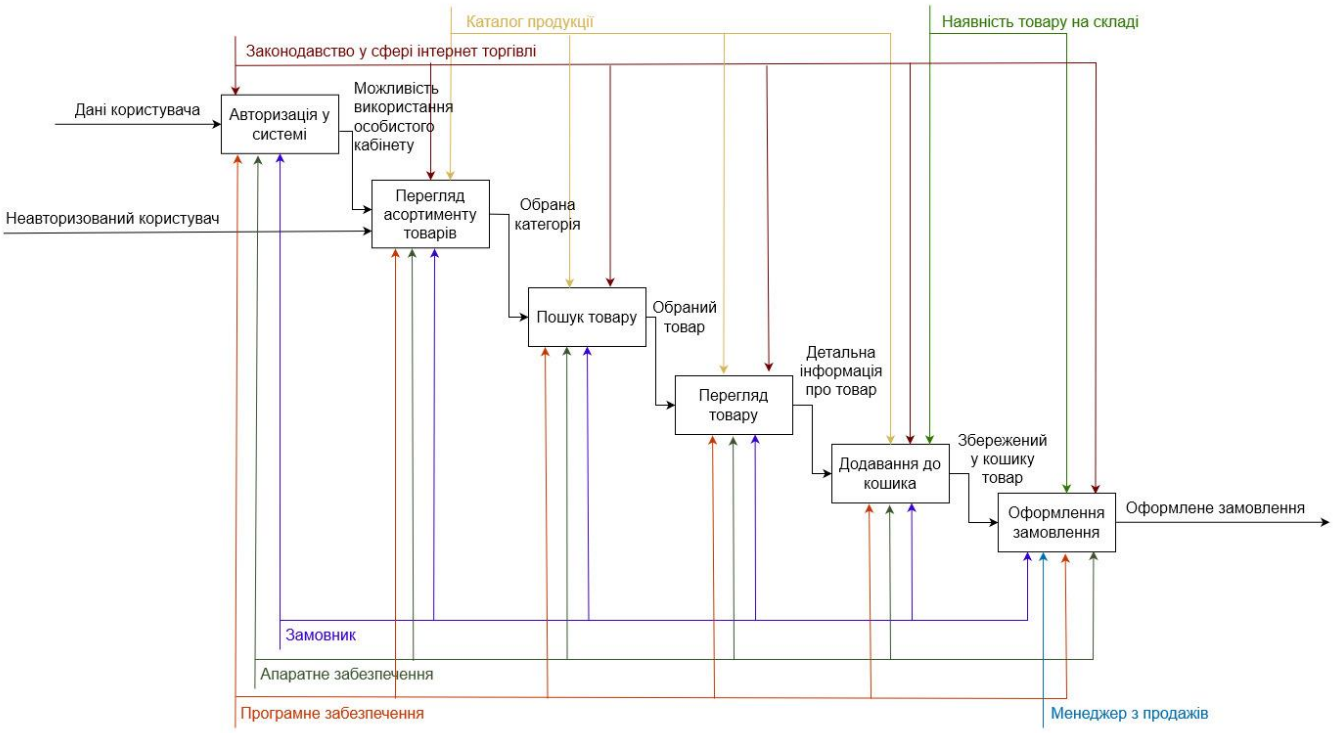


Рисунок 3.2 – Діаграма декомпозиції

Джерело: побудовано автором

3.2 Моделювання варіантів використання

Діаграми варіантів використання є основним видом діаграм при моделюванні поведінки системи. Мета діаграм варіантів використання полягає у відображенні динамічних аспектів системи. Кожна з діаграм показує набір варіантів використання і акторів у їхній взаємодії. UML діаграми варіантів використання використовуються для візуалізації поведінки системи, щоб користувач зміг зрозуміти, як використовувати той чи інший елемент, а розробник – як його реалізувати[47].

Актор – це будь-яка сутність, яка виконує роль впливу на систему, моделюючу розробником. Варіант використання виконують функцію опису сервісів, які надаються актору системою [47].

Взаємодія акторів з системою зображено на рис. 3.3.

Актори:

Менеджер з продажів – особа, яка має доступ до адміністративної частини системи, має право змінювати каталог товарів, коригувати бази даних, приймати замовлення та обробляти їх.

Клієнт – особа, яка використовує систему, ознайомлюючись з інформацією, має можливість зареєструватися, передивитись каталог товарів, здійснити замовлення.

База даних – сукупність даних, яка взаємодіє з іншими акторами системи при роботі з даними.

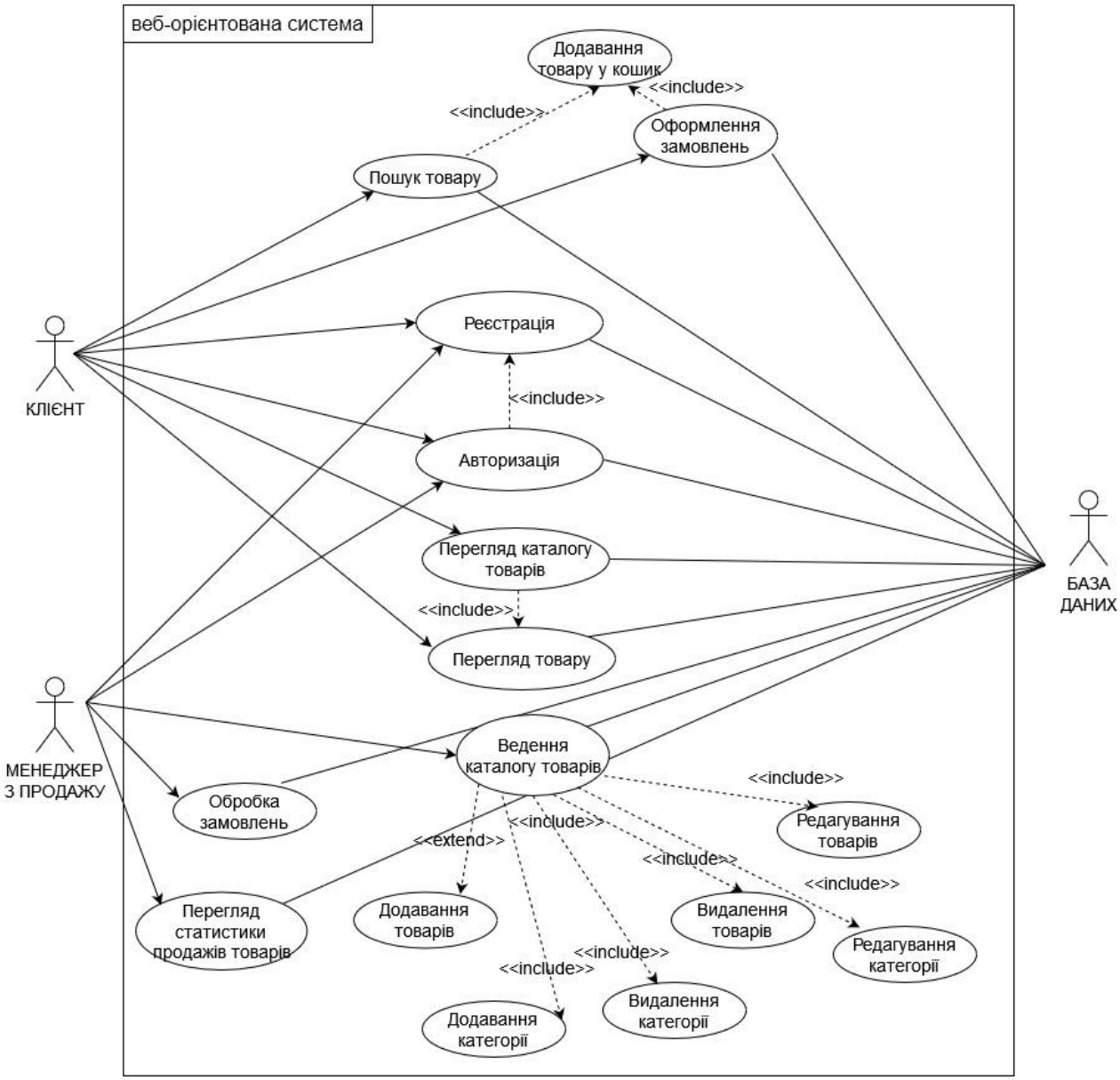


Рисунок 3.3 – Діаграма варіантів використання

Джерело: побудовано автором

Табл. 3.1 – Опис варіантів використання

№	Назва варіанту використання	Опис
1	Реєстрація	Дозволяє зареєструватися новому користувачеві
2	Перегляд інформації про магазин	Дозволяє отримати загальну інформацію про магазин
3	Перегляд каталогу товарів	Категорії, бренди, товари
4	Перегляд товару	Детальний опис товару
5	Перегляд замовлень	Зроблені користувачем замовлення
6	Особистий кабінет	Особиста інформація користувача, адрес та ін
7	Авторизація	Вхід в систему
8	Пошук товарів	Використання системи пошуку
9	Вибір товару	Додавання товару в кошик
10	Оформлення замовлень	Оформлення кошику як замовлення на адресу
11	Обробка замовлень	Менеджер змінює статуси засмолення в залежності від оплати, доставки та ін
12	Перегляд статистики продажів	Дозволяє отримати статистику продажів певних товарів
13	Ведення каталогу товарів	Додавання, редагування, видалення товарів та категорій товарів

Джерело: побудовано автором

3.3 Проектування моделі бази даних

База даних представляє собою сховище інформації, яке організоване та централізоване для конкретної сфери з доступом для різних користувачів. Основним завданням є збереження великих обсягів інформації та надання доступу до неї [50].

Розглянемо реляційну базу даних у контексті проектування моделі даних. Реляційні бази даних спроектовані для швидкого зберігання та отримання значних обсягів інформації. База даних складається з набору таблиць (сутностей). Таблиця містить колонки (атрибути) та рядки (кортежі). Рядки включають значення колонок.

Всі рядки в одній таблиці мають однакову структуру. Кожен рядок у таблиці може бути позначений унікальним ідентифікатором, який називається первинним ключем, а рядки з кількох таблиць можуть бути пов'язані за допомогою зовнішніх ключів [51].

При розробці бази даних було створено такі таблиці:

- users: дані про зареєстрованих користувачів, використовуються для ідентифікації через електронну пошту та містять інформацію для авторизації;

- profiles: загальна інформація про користувачів, така як ім'я, телефон, адреса та інші особисті дані;

- orders: інформація про замовлення, адресу доставки, вартість товарів;

- order_items: проміжна таблиця для реалізації зв'язку "багато-до-багатьох" між таблицями products та orders, також зберігає ціну та кількість товарів;

- products: основні дані про товар;

- brands: бренд до якого належить товар;

- categories: категорія до якої належить товар, також можуть зберігатися дочірні підкатегорії;

- baskets: кошик користувача;

- basket_product: проміжна таблиця для реалізації зв'язку "багато-до-багатьох" між таблицями products та baskets, також зберігає кількість товарів у кошику;

- statistics: інформація про кількість переглядів та замовлень товарів;

Для кращого розуміння бази даних застосовується ER-діаграма, яка ілюструє головні сутності, властивості та зв'язки між цими сутностями. На рис. 3.4 представлено діаграму для бази даних, яка була створена.

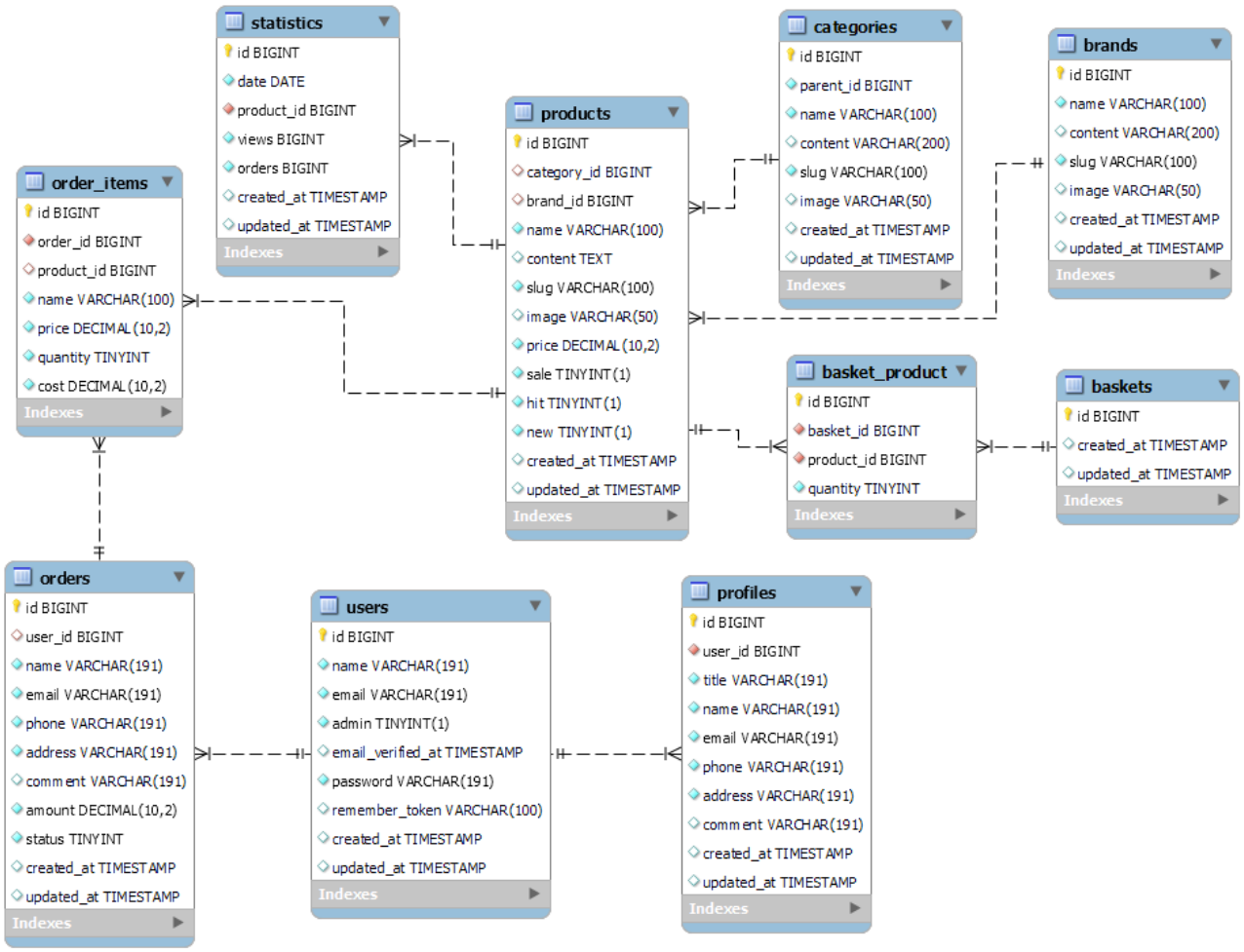


Рисунок 3.4 – ER-діаграма бази даних

Джерело: побудовано автором

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ

4.1 Програмна реалізація

Програмна реалізація web-орієнтованої системи підтримки діяльності онлайн-магазину Еко-товарів була виконана за допомогою php-фреймворку Laravel. Розглянемо детальніше структуру системи (рис.4.1).

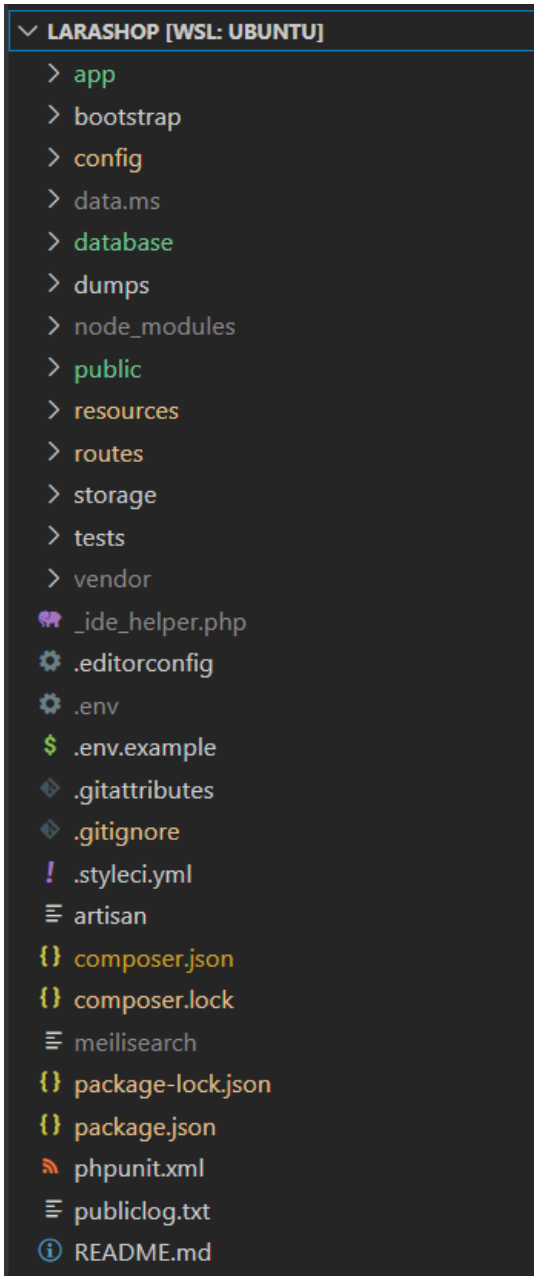


Рисунок 4.1 – Структура проекту

Джерело: побудовано автором (знімок з екрану)

Короткий опис проекту представлено в табл. 4.1

Таблиця 4.1 – Опис структури проекту

№	пакет	короткий опис
1	«app»	основний код програми, що розробляється, контролери, моделі та ін.
2	«bootstrap»	містить файл app.php, який завантажує фреймворк, також містить каталог cache, з файлами, згенерованими фреймворком для оптимізації продуктивності.
3	«config»	конфігураційні файли
4	«database»	міграції баз даних, фабрики моделей та наповнювачі
5	«public»	index.php - вхідна точка для всіх запитів, ресурси (зображення, JavaScript, CSS)
6	«resources»	шаблони, а також нескомпільовані ресурси, наприклад, JavaScript або CSS.
7	«routes»	містить всі визначення маршрутів для програми
8	«storage»	журнали (логи), скомпіловані шаблони Blade, файли сесій, кеша та інші файли, створені фреймворком
9	«tests»	автоматизовані тести

Джерело: побудовано автором

Web-орієнтована інформаційна система складатиметься з двох частин: адміністративної та користувацької.

Адміністративна частина надасть можливість менеджерам:

1. Створювати категорії товарів,
2. Додавати, змінювати та видаляти товари,
3. Переглядати статистику продажів.

Користувацька частина дозволить клієнтам онлайн-магазину швидко та логічно переходити між різними розділами та сторінками. Буде складатися:

1. Каталог товарів – включає категорії та бренди товарів.

2. Вхід/Реєстрація – дозволить користувачам увійти до свого особистого кабінету або зареєструватися.
3. Особистий кабінет – цей розділ буде доступний авторизованим користувачам і міститиме інформацію про користувача та його замовлення.
4. Кошик – ця сторінка покаже користувачам, скільки товарів вони вже додали до свого кошика для покупок.

4.2 Розробка компонентів інформаційної системи

При використанні Eloquent кожна таблиця БД має відповідну «Модель», яка використовується для взаємодії з цією таблицею. Тому одним з важливих етапів у створенні проекту є створення моделей (рис. 4.2). Крім отримання записів із таблиць БД, моделі Eloquent також дозволяють створювати, оновлювати та видаляти записи з таблиць [52, 53]. У цих моделях вказується fillable (поля, які можуть бути заповнені масово), відношення між моделями наприклад замовлення може бути пов'язано з користувачем, який його розмістив [54]. Моделі функціонують як посилання на таблиці (для спрощення викликів до бази даних).

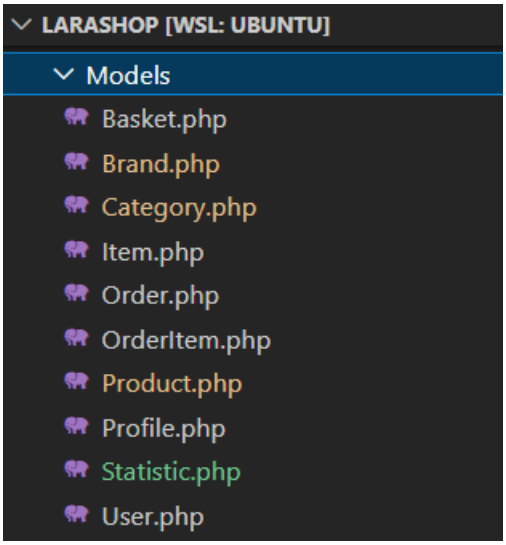


Рисунок 4.2 – Моделі проекту

Джерело: побудовано автором (знімок з екрану)

У розробці було використано міграції. Міграції схожі на контроль версій для бази даних. Вони дозволяють команді розробників визначати схеми бази даних проекту та спільно використовувати їх визначення. Якщо коли-небудь доводилося вручну вносити зміни у схему локальної бази даних після застосування змін у системі керування версіями, ці проблеми вирішує міграція бази даних [55]. Для прикладу приведемо міграцію для таблиці users (рис. 4.3)

```php
<?php

use Illuminate\Database\Migrations\Migration;      You, 1 minute ago
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

0 references | 0 implementations | You, 4 minutes ago | 1 author (You)
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    0 references | 0 overrides
    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();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    0 references | 0 overrides
    public function down()
    {
        Schema::dropIfExists('users');
    }
```

Рисунок 4.3 – Приклад файлу міграції

Джерело: побудовано автором (знімок з екрану)

Також Laravel включає механізм наповнення бази даних початковими даними (seeding) за допомогою спеціальних класів. Усі такі класи зберігаються у директорії database/seeds. За промовчанням у Laravel вже визначено клас DatabaseSeeder. З цього класу можна викликати метод call() для підключення інших класів з даними, що дозволить контролювати порядок виконання. Файли фабрик моделей зберігаються в директорії database/factories, і вже є один готовий файл UserFactory.php — це фабрика для моделі User [56].

У Laravel є дуже простий і виразний метод визначення маршрутів. Найпростіші маршрути Laravel приймають URI та замикання, забезпечуючи нетрудомісткий та виразний метод визначення маршрутів та поведінки без складних конфігураційних файлів маршрутизації. Всі маршрути Laravel мають бути визначені у файлах маршрутів, що знаходяться у каталозі routes. Ці файли автоматично завантажуються постачальником App\Providers\RouteServiceProvider програми. Файл routes/web.php визначає маршрути для веб-інтерфейсу [57]. Цим маршрутам призначається група посередників web, яка забезпечує такі функції, як стан сесії та захист від CSRF. Маршрути в routes/api.php не зберігають стану і призначається група посередників api. До маршрутів, визначених у routes/web.php, можна отримати доступ, ввівши URL-адресу певного маршруту у браузері.

Маршрутизатор дозволяє реєструвати маршрути для будь-якого HTTP-запиту [57]:

- Route::get($uri, $callback);
- Route::post($uri, $callback);
- Route::put($uri, $callback);
- Route::patch($uri, $callback);
- Route::delete($uri, $callback);
- Route::options($uri, $callback).

routes > web.php > ...

```php
Route::group([
    'as' => 'catalog.',
    'prefix' => 'catalog',
], function () {
    Route::get('index', 'CatalogController@index')
        ->name('index');

    Route::get('category/{category:slug}', 'CatalogController@category')
        ->name('category');

    Route::get('brand/{brand:slug}', 'CatalogController@brand')
        ->name('brand');

    Route::get('product/{product:slug}', 'CatalogController@product')
        ->name('product');

    Route::get('search', 'CatalogController@search')
        ->name('search');
});
```

Рисунок 4.4 – Приклад маршрутів проекту

Джерело: побудовано автором (знімок з екрану)

Замість того, щоб визначати всю логіку обробки запитів як замикання у файлах маршрутів, можна організувати цю поведінку за допомогою класів «контролерів». Контролери можуть згрупувати пов'язану логіку обробки запитів до одного класу. Наприклад, клас UserController може обробляти всі вхідні запити, що стосуються користувачів, включаючи відображення, створення, оновлення та видалення користувачів. За замовчанням контролери зберігаються у каталозі app/Http/Controllers. Контролер розширює базовий клас контролера App\Http\Controllers\Controller, включений до Laravel [58].

Фрагмент програмного коду контролера з обробки запитів користувача наведено далі в лістингу:

```
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
```

```
class UserController extends Controller {

    /**
     * @return \Illuminate\View\View
     */
    public function index() {
        $users = User::paginate(5);
        return view('admin.user.index', compact('users'));
    }

    /**
     * @param  \App\Models\User  $user
     * @return \Illuminate\View\View
     */
    public function edit(User $user) {
        return view('admin.user.edit', compact('user'));
    }

    /**
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Models\User  $user
     * @return \Illuminate\Http\RedirectResponse
     * @throws \Illuminate\Validation\ValidationException
     */
    public function update(Request $request, User $user) {
        $this->validator($request->all(), $user->id)->validate();

        if ($request->change_password) {
            $request->merge(['password' => Hash::make($request->password)]);
            $user->update($request->all());
        } else {
            $user->update($request->except('password'));
        }

        return redirect()
            ->route('admin.user.index')
            ->with('success', 'Профіль користувача оновлено');
    }

    /**
     * @param array $data
     * @return \Illuminate\Contracts\Validation\Validator|\Illuminate\Validation\Validator
     */
    private function validator(array $data, int $id) {
        $rules = [
            'name' => [
                'required',
                'string',
                'max:255'
            ],
            'email' => [
```

```
                'required',
                'string',
                'email',
                'max:255',
                'unique:users,email,'.$id.',id',
            ],
        ];
        if (isset($data['change_password'])) {
            $rules['password'] = ['required', 'string', 'min:8', 'confirmed'];
        }
        return Validator::make($data, $rules);
    }
}
```

У даному проекті використовується декілька контролерів (рис. 4.5) і кожен з них включає в себе набір функцій для роботи з частиною функціоналу сайту.

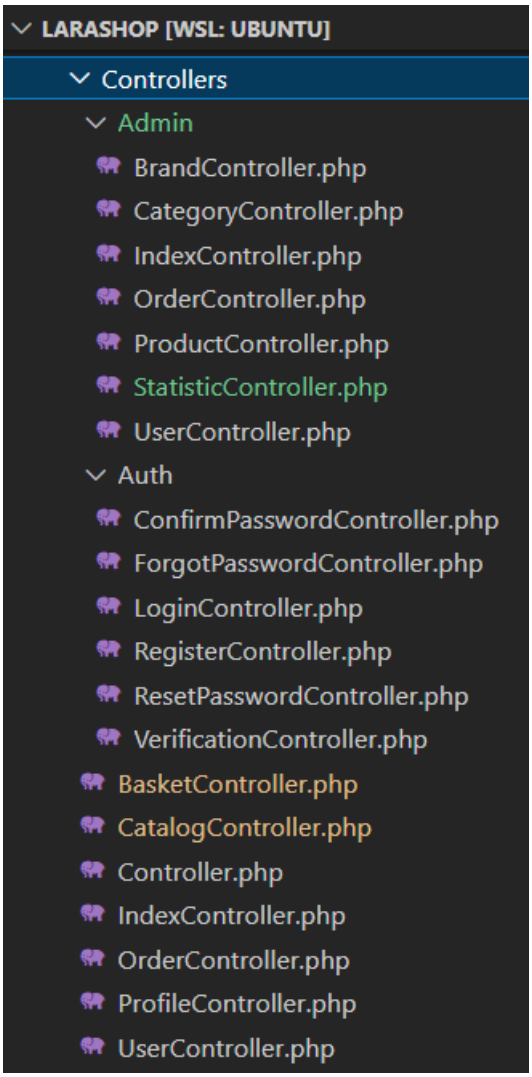


Рисунок 4.5 – Список контролерів проекту

Джерело: побудовано автором (знімок з екрану)

Посередники у Laravel надають зручний спосіб для фільтрації HTTP-запитів. Наприклад, один з основних посередників відповідає за перевірку аутентифікації користувача: якщо користувач не має аутентифікації, він буде перенаправлений на сторінку входу в систему; якщо ж користувач аутентифікований, посередник дозволить йому продовжити роботу. Посередник може бути написаний для виконання різних завдань. Наприклад, посередник для ведення журналу може реєструвати всі запити вашої програми. До складу фреймворку Laravel вже входять кілька посередників, включаючи посередник для автентифікації та посередник для захисту від CSRF. Всі ці посередники знаходяться у каталозі app/Http/Middleware [59].

У проекті реалізуємо посередника, який буде перевіряти, чи є користувач адміністратором, щоб дати йому доступ до сторінки або сповістити що такої сторінки не існує.

Blade – це простий, але потужний шаблонізатор, що входить до складу Laravel [60]. Усі шаблони Blade компілюються у звичайний PHP-код і кешуються до тих пір, поки не будуть змінені, що означає, що Blade додає фактично нульове навантаження додатку. Файли шаблонів Blade використовують розширення файлу .blade.php і зазвичай зберігаються у каталозі resources/views [60].

Крім успадкування шаблонів та відображення даних, Blade також містить зручні псевдоніми для загальних структур управління PHP, таких як умовні оператори та цикли. Ці директиви забезпечують дуже чистий та лаконічний спосіб роботи зі структурами управління PHP, але при цьому залишаються схожими зі своїми аналогами PHP.

Фрагмент програмного коду представлення користувачів на адміністративній панелі наведено далі в лістингу

```
@extends('layout.admin', ['title' => 'Всі користувачі'])

@section('content')
   <h1 class="mb-4">Всі користувачі</h1>

   <table class="table table-bordered">
      <tr>
         <th>#</th>
         <th width="25%">Дата реєстрації</th>
         <th width="25%">Ім'я, Прізвище</th>
```

```
            <th width="25%">Адреса пошти</th>
            <th width="20%">Кількість замовлень</th>
            <th><i class="fas fa-edit"></i></th>
        </tr>
        @foreach($users as $user)
            <tr>
                <td>{{ $user->id }}</td>
                <td>{{ $user->created_at->format('d.m.Y H:i') }}</td>
                <td>{{ $user->name }}</td>
                <td><a href="mailto:{{ $user->email }}">{{ $user->email }}</a></td>
                <td>{{ $user->orders->count() }}</td>
                <td>
                    <a href="{{ route('admin.user.edit', ['user' => $user->id]) }}">
                        <i class="far fa-edit"></i>
                    </a>
                </td>
            </tr>
        @endforeach
    </table>
    {{ $users->links() }}
@endsection
```

4.3 Реалізація пошуку товарів

Для пошуку товарів обрано MeiliSearch, оскільки MeiliSearch має ліцензію MIT та його можна розгорнути локально, без жодних складнощів, які є наприклад у Elasticsearch. А також його легко інтегрувати в Laravel-додаток за допомогою драйвера для Laravel Scout було вирішено використовувати MeiliSearch в кваліфікаційній роботі для пошуку товарів.

Laravel Scout надає просте рішення на основі драйверів для додавання повнотекстового пошуку до моделей Eloquent. Використовуючи спостерігачі (observers) моделей, Scout автоматично синхронізуватиме пошукові індекси з даними моделей Eloquent [61].

У даний час Scout поставляється з драйверами Algolia та MeiliSearch. Крім того, Scout включає пошуковий драйвер «колекцій», призначений для використання в локальній розробці і не потребує зовнішніх залежностей або сторонніх сервісів. Крім того, написати власні драйвери просто, і ви можете вільно розширювати Scout власними реалізаціями пошуку.

Установка Scout доволі проста, через менеджер пакетів Composer: composer require laravel/scout

Після інсталяції Scout необхідно опублікувати файл конфігурації Scout за допомогою Artisan-команди vendor:publish та додати Laravel\Scout\Searchable трейт в моделі, серед яких потрібно проводити пошук. Цей трейт зареєструє спостерігача моделі, який автоматично синхронізуватиме модель з драйвером пошуку.

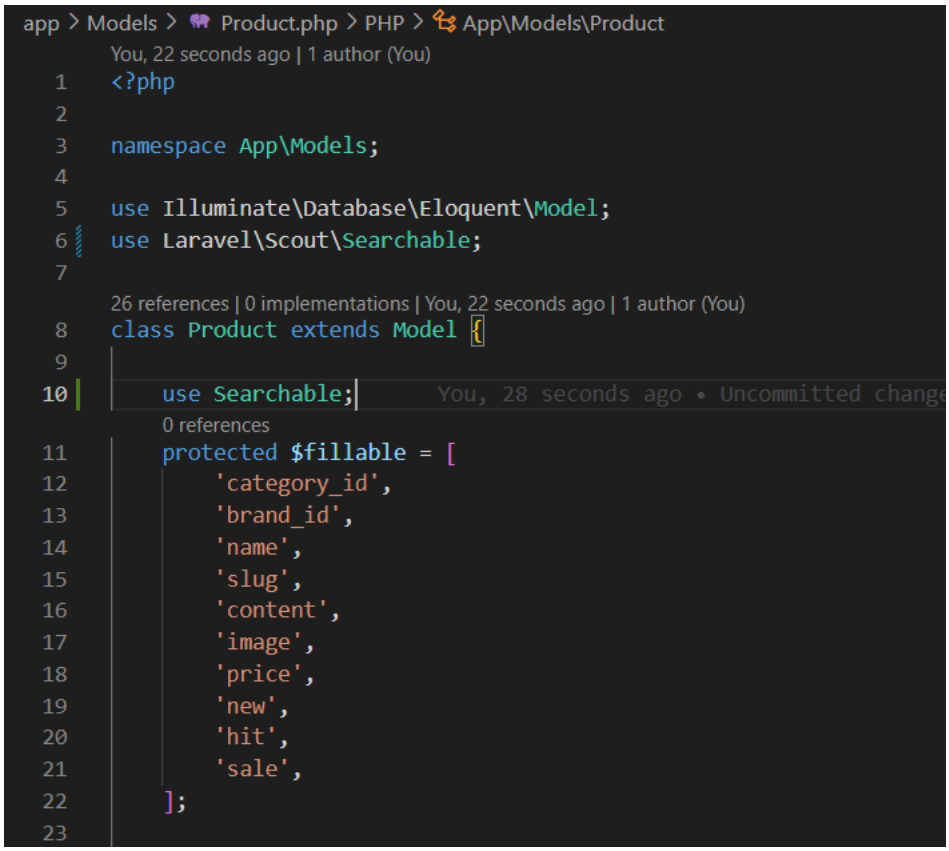


Рисунок 4.6 – Код моделі, в яку додано Searchable трейт

Джерело: побудовано автором (знімок з екрану)

Наступним кроком слід налаштувати драйвер черги перед використанням бібліотеки MeiliSearch. Запуск обробника черги дозволить Scout ставити в чергу всі операції, що синхронізують інформацію моделі з пошуковими індексами, забезпечуючи набагато кращий час відгуку для веб-інтерфейсу програми [62].

Після того, як додали в модель трейт Searchable, все, що потрібно зробити, це викликати метод save або create на екземплярі моделі, і вона буде автоматично додана

в пошуковий індекс. Якщо Scout налаштовано для використання черг, ця операція буде виконуватися у фоновому режимі обробником черги.

Кожна модель Eloquent синхронізується із заданим пошуковим індексом, який містить всі доступні для пошуку записи для цієї моделі. Іншими словами, можна думати про кожен індекс як таблицю MySQL. За замовчуванням, кожна модель буде збережена в індексі, що відповідає типовому «табличному» імені моделі [62].

Пошук за моделлю можна виконати, використовуючи метод search. Метод search приймає рядок як пошуковий запит, який буде використовуватися для пошуку за моделлю. Потім викликається метод get або paginate, щоб отримати модель Eloquent як результат заданого пошукового запиту:

```
$search = $request->input('query');
$products = Product::search($search)->paginate(15);
return view('catalog.search', compact('products', 'search'));
```

Наступним кроком треба завантажити та встановити Meilisearch. Далі запускаємо Meilisearch, виконавши команду ./meilisearch у терміналі (рис. 4.7):

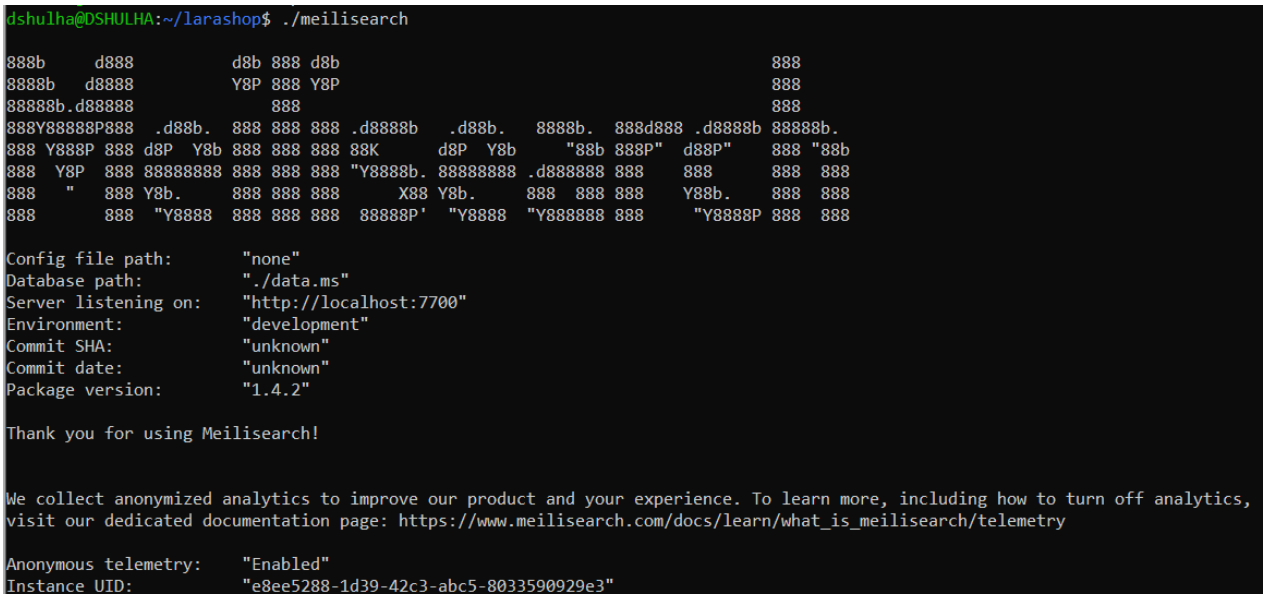


Рисунок 4.7 – Зображення працюючої програми Meilisearch

Джерело: побудовано автором (знімок з екрану)

Тепер у вікні терміналу працює екземпляр Meilisearch. Зараз можна перейти по URL *http://localhost:7700/* до панелі керування та побачити імпортовані дані (рис 4.8).

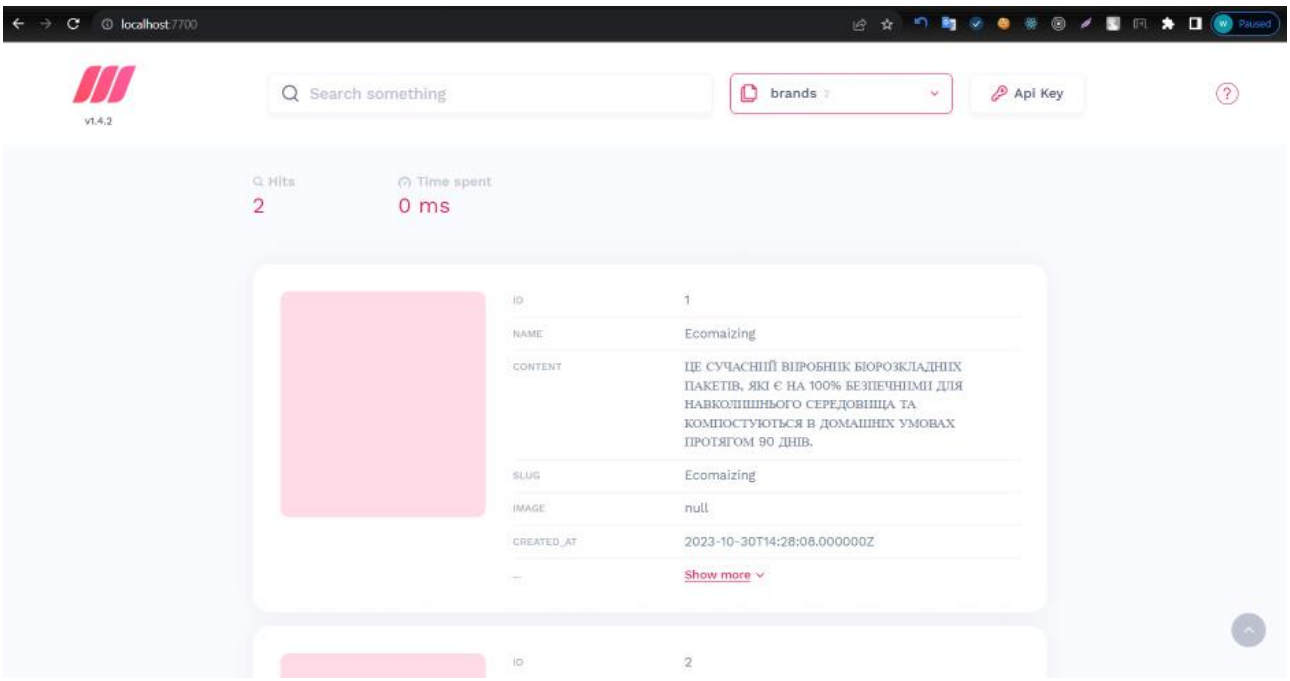


Рисунок 4.8 – Зображення панелі керування Meilisearch

Джерело: побудовано автором (знімок з екрану)

4.4 Використання системи клієнтом

Система підтримки діяльності онлайн-магазину має дві основні складові. Зокрема, існує клієнтська частина, призначена для перегляду інформації, яка доступна усім користувачам. Також існує адміністративна панель з можливістю додавання і редагування інформації, яка стосується тільки менеджерів онлайн-магазину.

Робота онлайн-магазину Еко-товарів розпочинається з головної сторінки, де користувач має можливість подивитися інформацію про новинки та лідерів продажу, зареєструватися або авторизуватися (рис.4.9-4.11).

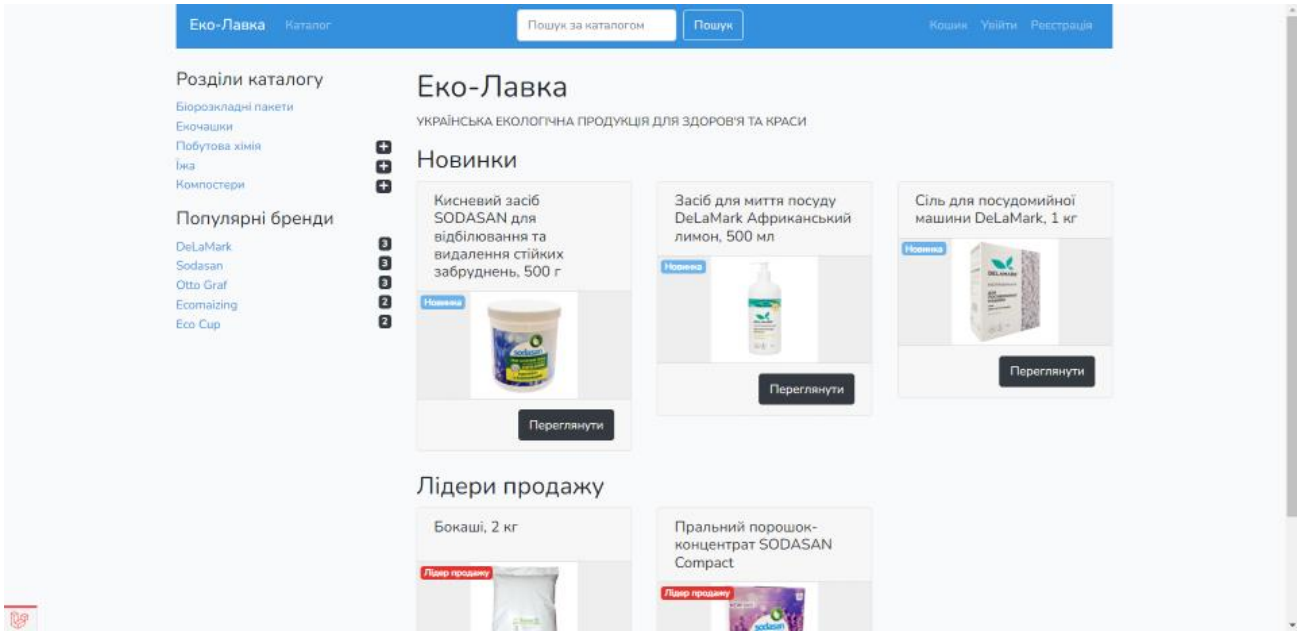


Рисунок 4.9 – Головна сторінка сайту

Джерело: побудовано автором (знімок з екрану)

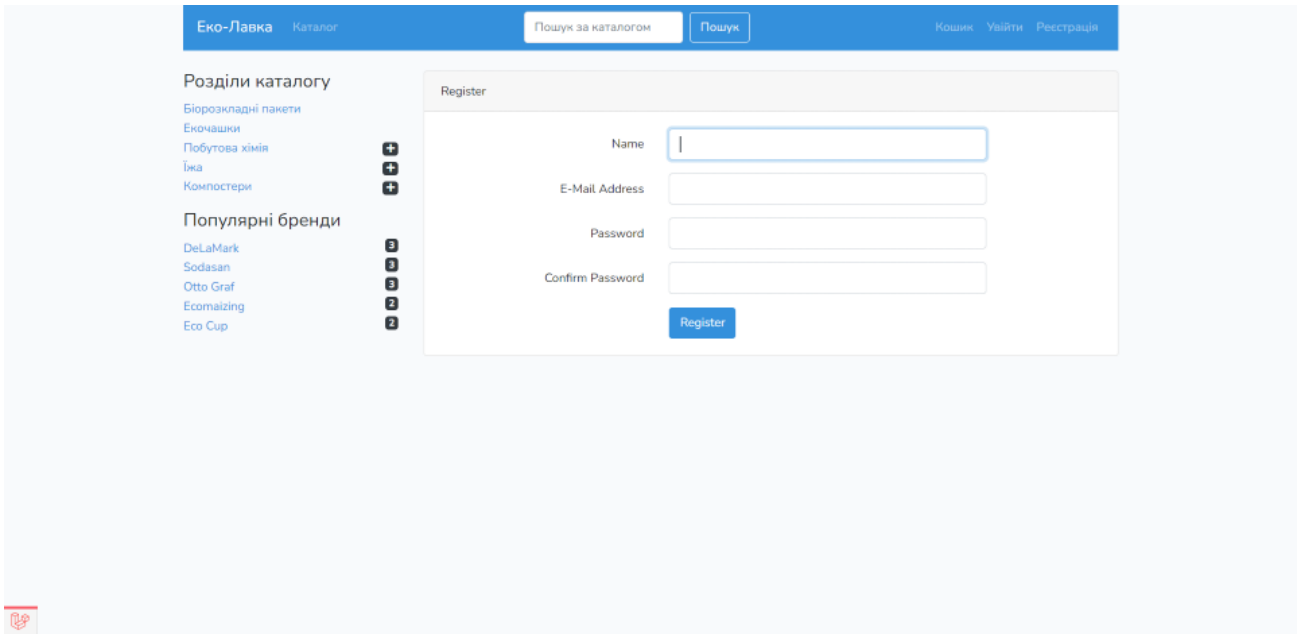


Рисунок 4.10 – Сторінка реєстрації

Джерело: побудовано автором (знімок з екрану)

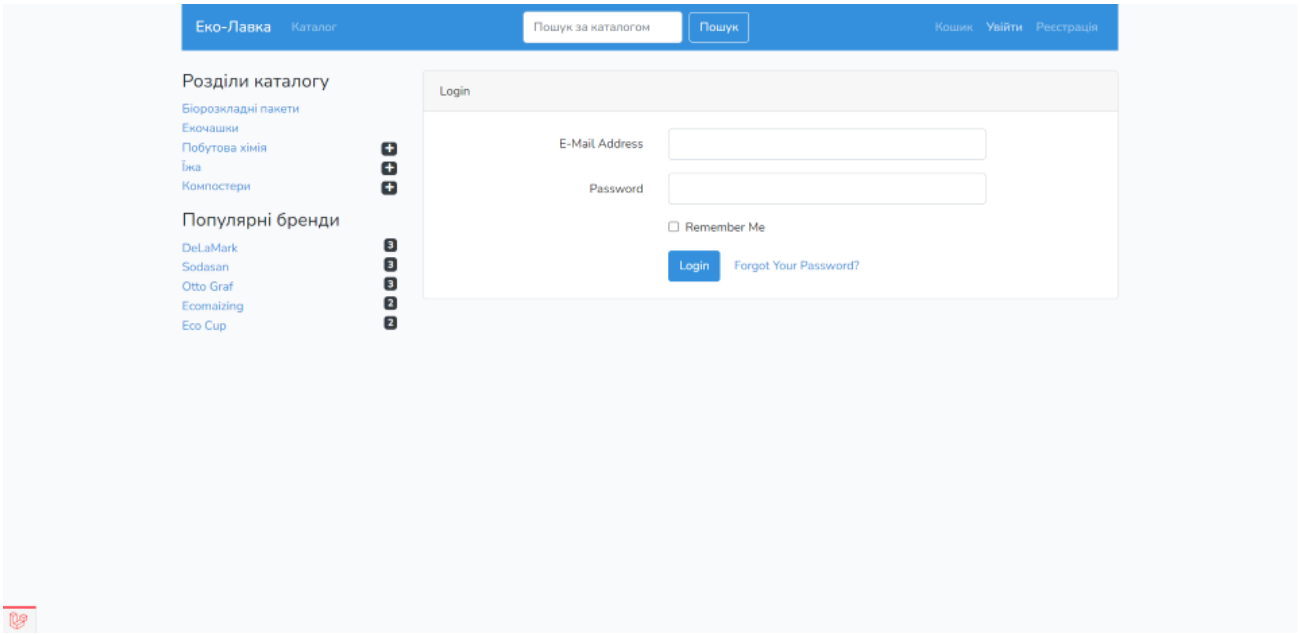


Рисунок 4.11 – Сторінка авторизації

Джерело: побудовано автором (знімок з екрану)

Після авторизації та реєстрації відбувається перехід до особистого кабінету користувача (рис. 4.12).

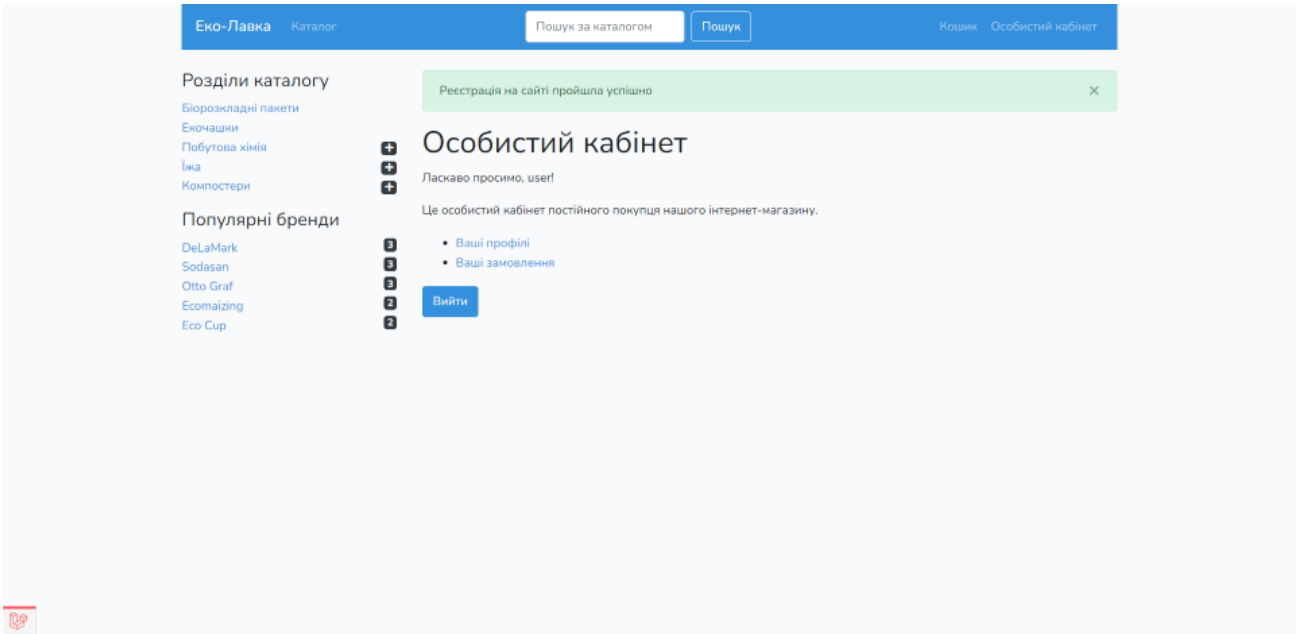


Рисунок 4.12 – Особистий кабінет

Джерело: побудовано автором (знімок з екрану)

Особиста сторінка користувача дає можливість перейти до профілів або списку замовлень. Користувач може завести кілька профілів, наприклад з домашнім адресом або адресою батьків, й ці профілі будуть доступні при оформленні замовлення, щоб заповнити дані отримувача

Переходячи до каталогу, користувач може обрати категорію товару (рис. 4.13, 4.14) та наступним етапом подивитися детальну інформацію про товар (рис 4.15).

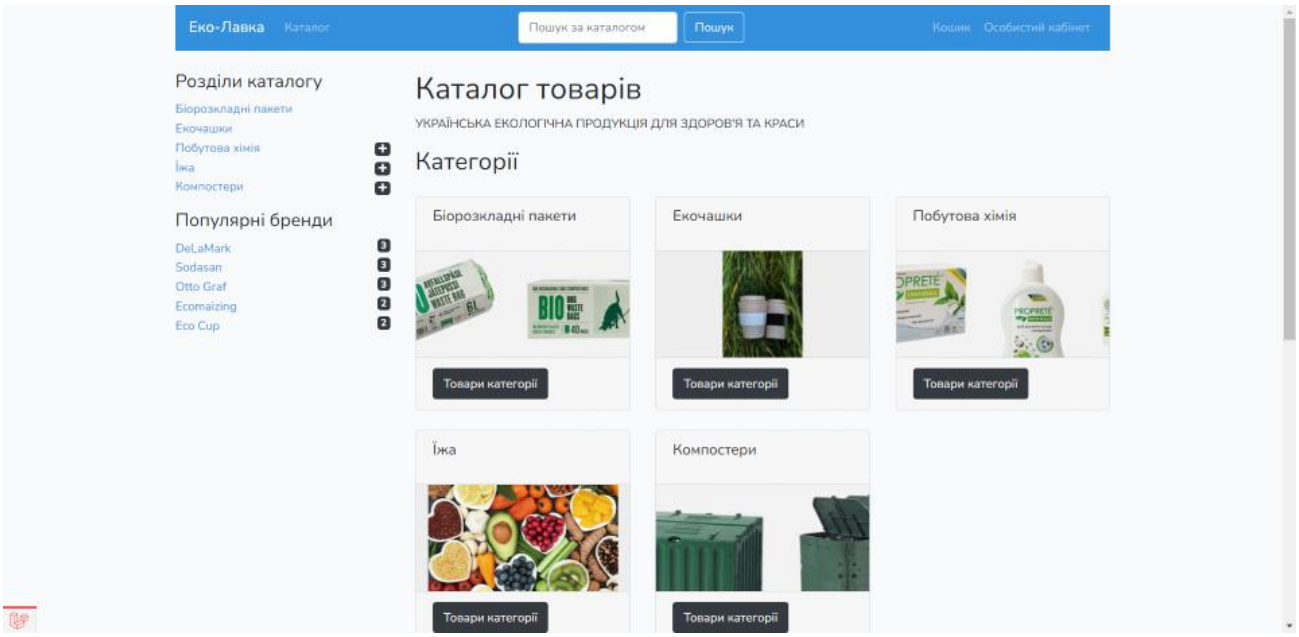


Рисунок 4.13 – Каталог

Джерело: побудовано автором (знімок з екрану)

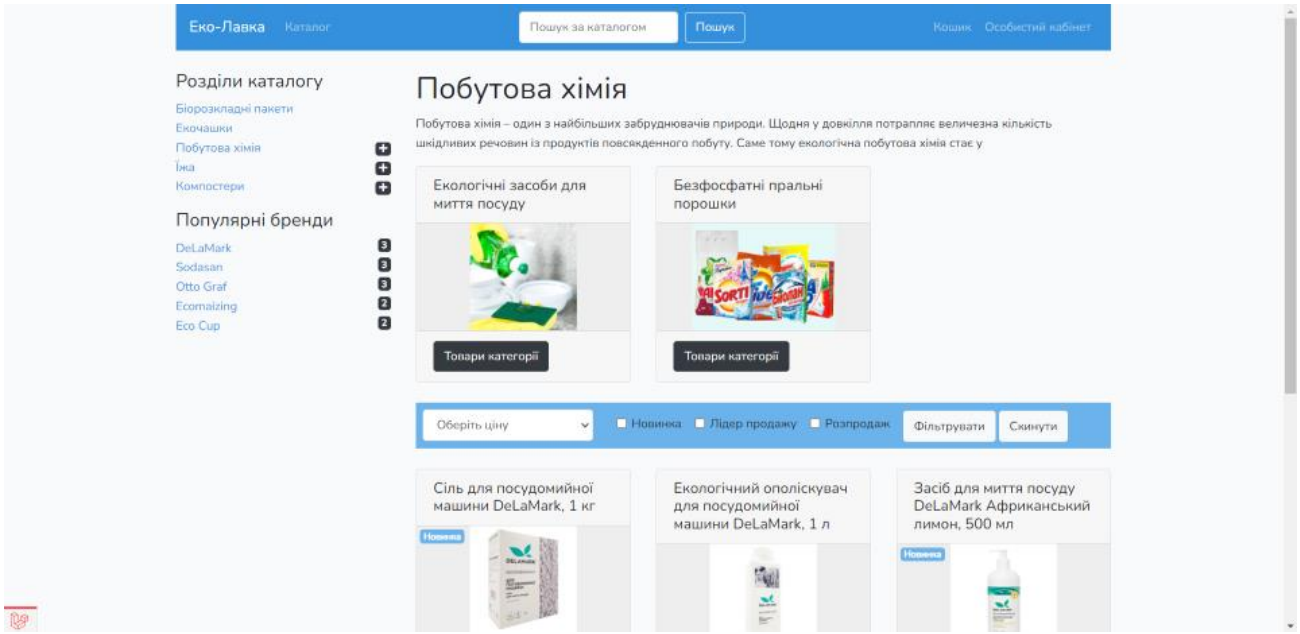


Рисунок 4.14 – Категорія

Джерело: побудовано автором (знімок з екрану)

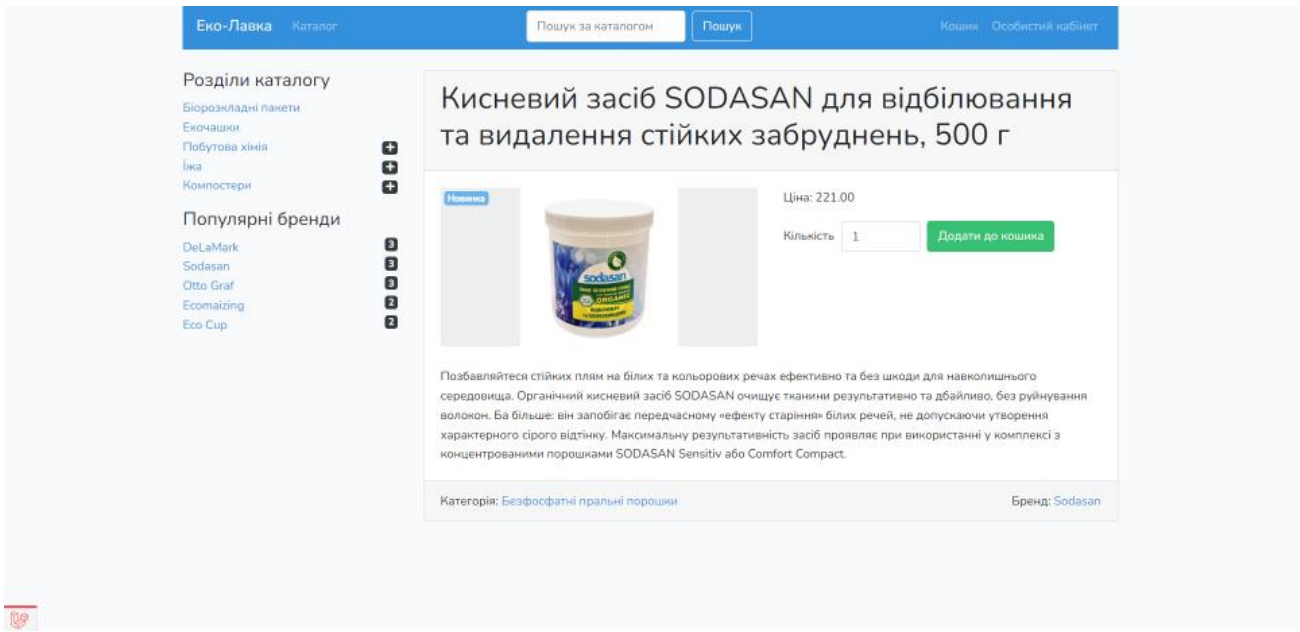


Рисунок 4.15 – Сторінка товару

Джерело: побудовано автором (знімок з екрану)

На сторінці товару можна вказати кількості для замовлення та додати товар до кошику.

Для переходу до кошика (рис 4.16) слід натиснути на відповідну іконку, яка шапці сайту.

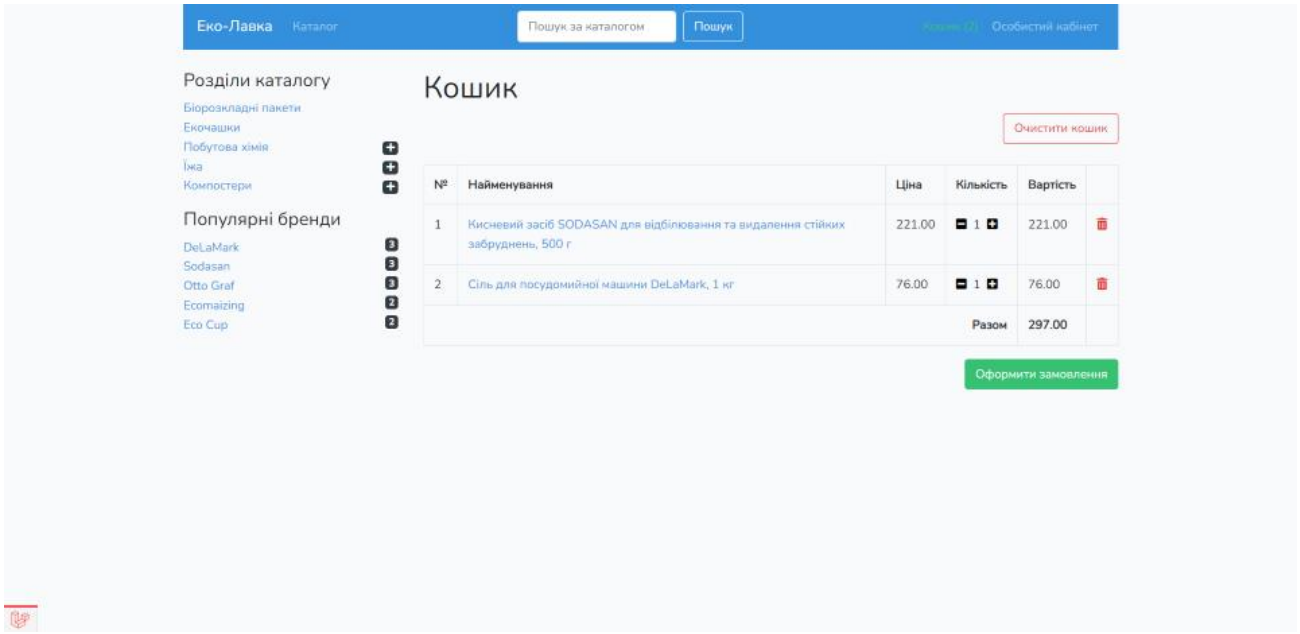


Рисунок 4.16 – Кошик

Джерело: побудовано автором (знімок з екрану)

Сторінка кошика містить список вибраного товару, який можна в будь-який момент видалити або змінити кількість. Під списком товарів вказана загальна вартість замовлення та розташована кнопка для оформлення замовлення. Але замовлення можна робити й без реєстрації. Користувачу достатньо додати товари до кошику.

Сторінка для оформлення замовлення містить форму для введення контактних даних замовника та можливості додати коментар за бажанням (рис 4.17). Після збереження замовлення з'явиться повідомлення про успішне оформлення та деталі про нього (рис 4.18).

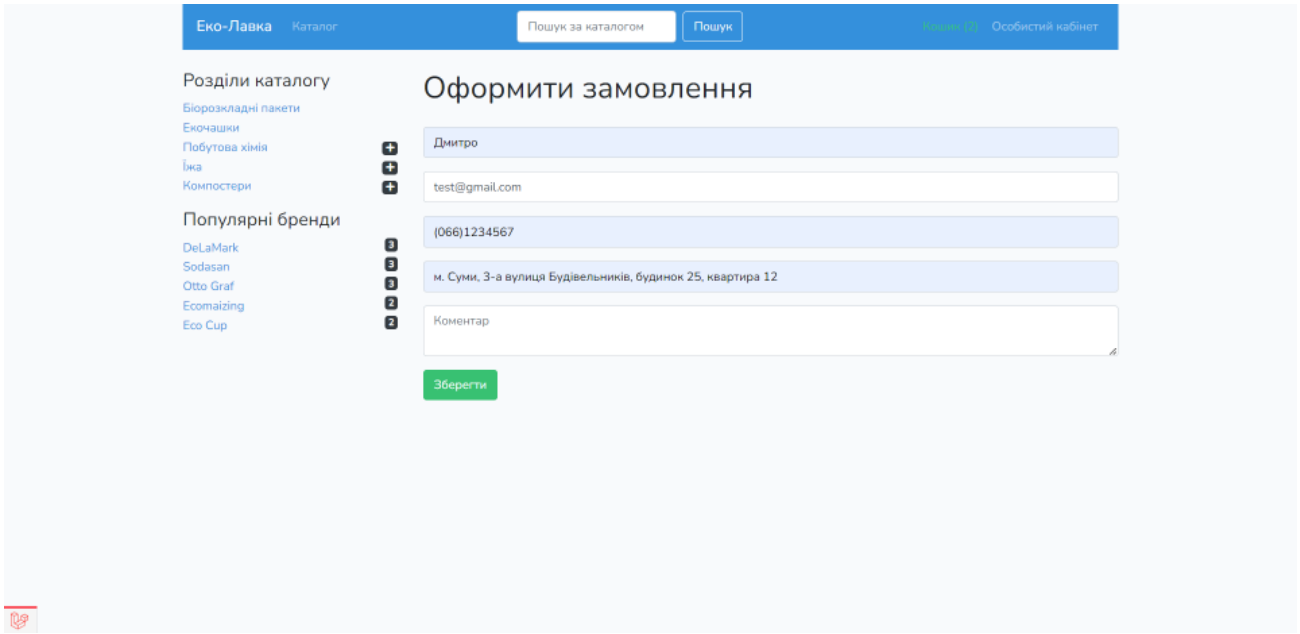


Рисунок 4.17 – Форма оформлення замовлення

Джерело: побудовано автором (знімок з екрану)

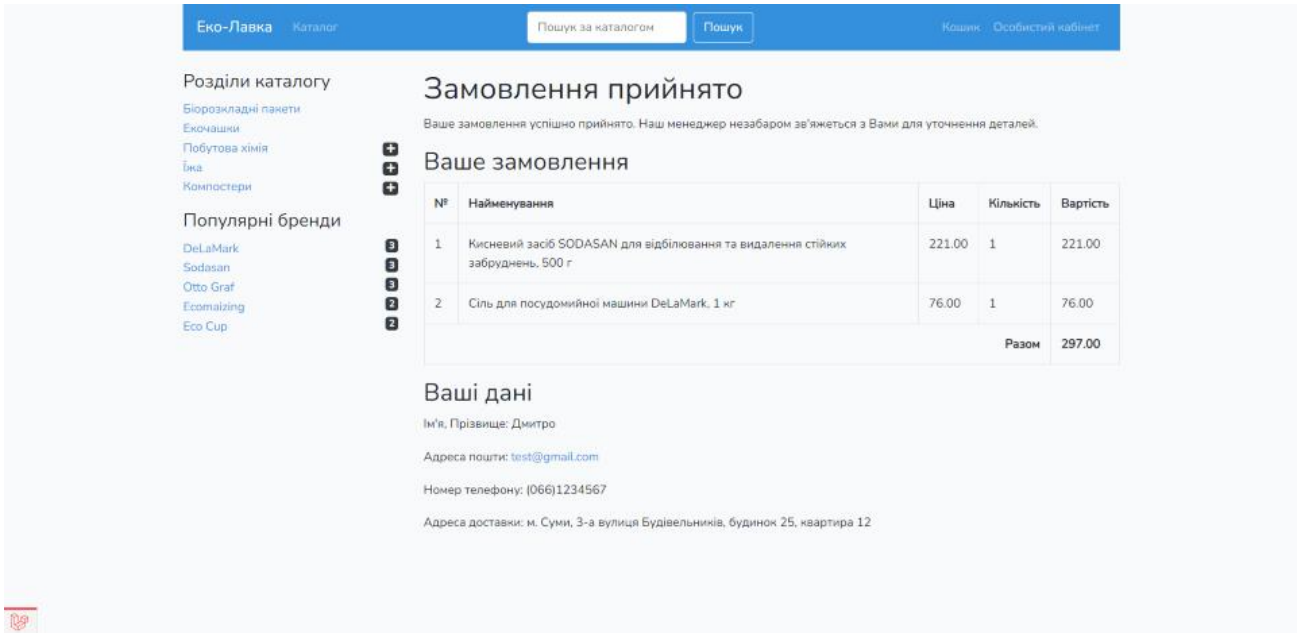


Рисунок 4.18 – Повідомлення про оформлення замовлення

Джерело: побудовано автором (знімок з екрану)

Окремо слід сказати про пошук товарів. Зазвичай сервіс пошуку складається з двох компонент: пошукова система та індексатор. Індексатор отримує текст на вхід, робить обробку тексту (вирізання закінчень, незначних слів тощо) і зберігає все в індексі. Зазвичай індексатор буде ігнорувати стоп-слова (такі як «або» та «і»), які є

загальними і недостатньо значущими, щоб бути корисними при пошуку. Деякі індексатори також використовують мовні скорочення слів, які індексуються. Наприклад, слова «копав», «копала», і «копали» будуть занесені в індекс під єдиною концепцією слова «копати». Структура такого індексу дозволяє проводити по ньому дуже швидкий пошук. Пошуковик – інтерфейс пошуку за індексом – приймає від клієнта запит, обробляє фразу та шукає її в індексі [62].

Припустимо у нас є каталог товарів та введемо пошуковий запит «куп» до «Пошук за каталогом». Знайдено п’ять товарів (рис. 4.19).

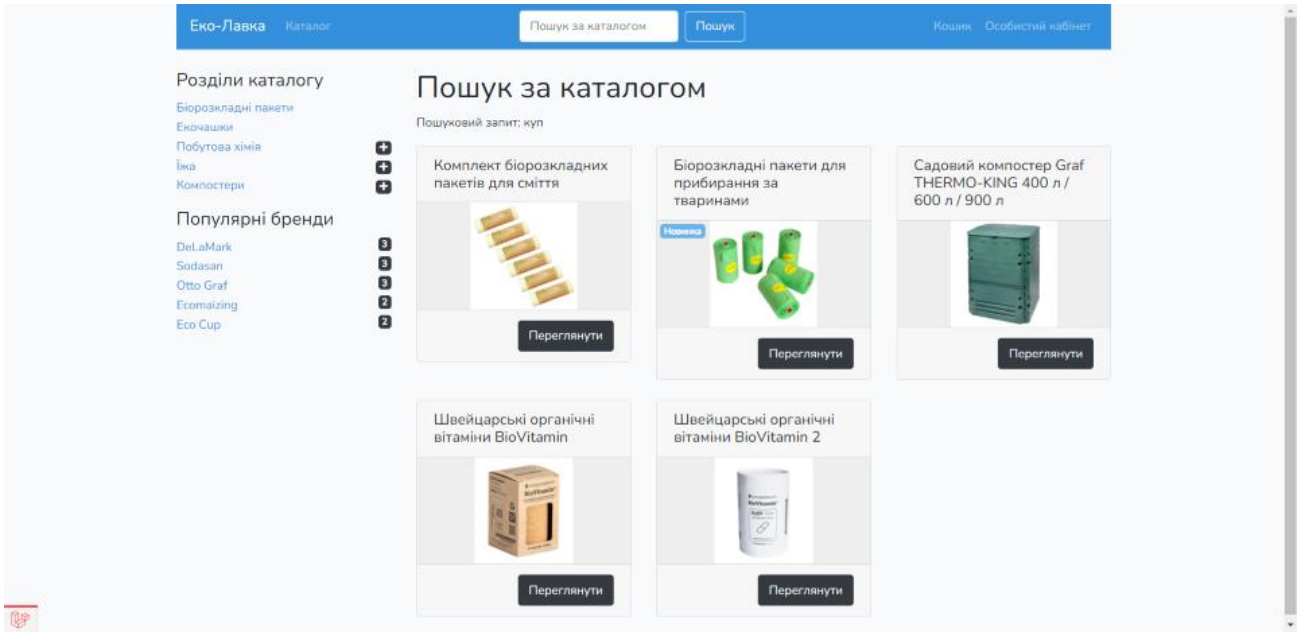


Рисунок 4.19 – Результат пошуку за словом «куп»

Джерело: побудовано автором (знімок з екрану)

В інтерфейсі MeiliSearch ми бачимо ці проіндексовані товари. Задаємо в пошуку «куп» та також бачимо п’ять товарів (рис 4.20)

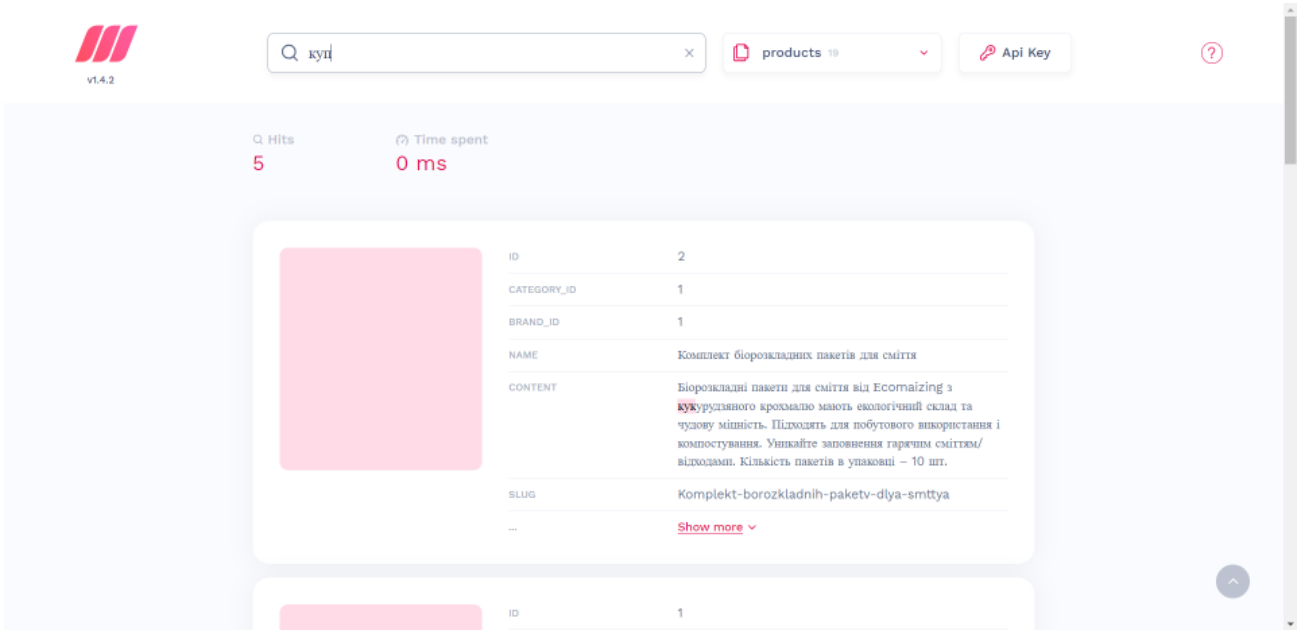
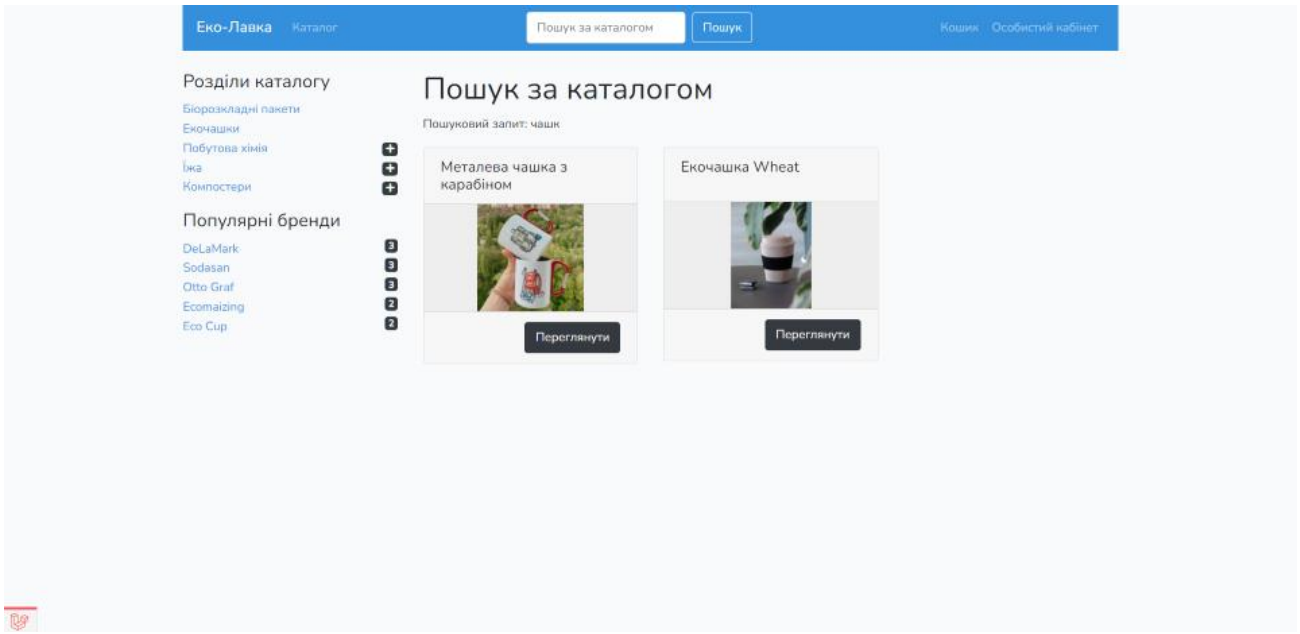


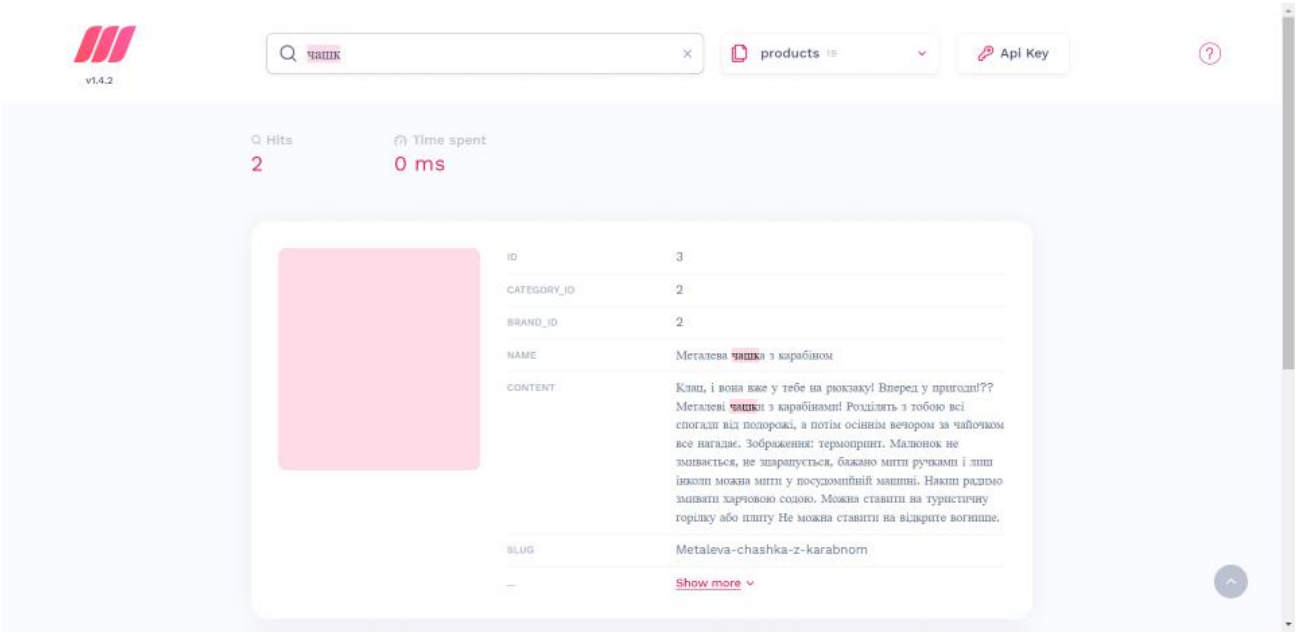
Рисунок 4.20 – Результат пошуку за словом «куп» в інтерфейсі MeiliSearch

Джерело: побудовано автором (знімок з екрану)

Інший приклад – пошук «чашк» (рис. 4.21). Введемо це слово до «Пошуку за каталогом» та до пошукового запиту MeiliSearch.



а)



б)

Рисунок 4.21 – Результат пошуку за словом «чашк»

Джерело: побудовано автором (знімок з екрану)

4.5 Використання системи менеджером

Для управління онлайн магазином, додавання товарів, обробки замовлень, перегляду статистики за товарами доступна адміністративна панель. Для входу в неї необхідно пройти авторизацію, ввести email та пароль користувача. Після авторизації як адміністратор користувач потрапляє на головну сторінку адміністративної панелі, де у верхній частині панелі можна обрати розділ для перегляду.

Перед додаванням товарів слід створити категорії, оскільки це необхідно для подальшого відношення продукту до певної категорії. Категорії мають ієрархічну структуру. На сторінці адміністрування категорій (рис. 4.22) можна побачити всі категорії, відредагувати існуючу категорію або видалити її.

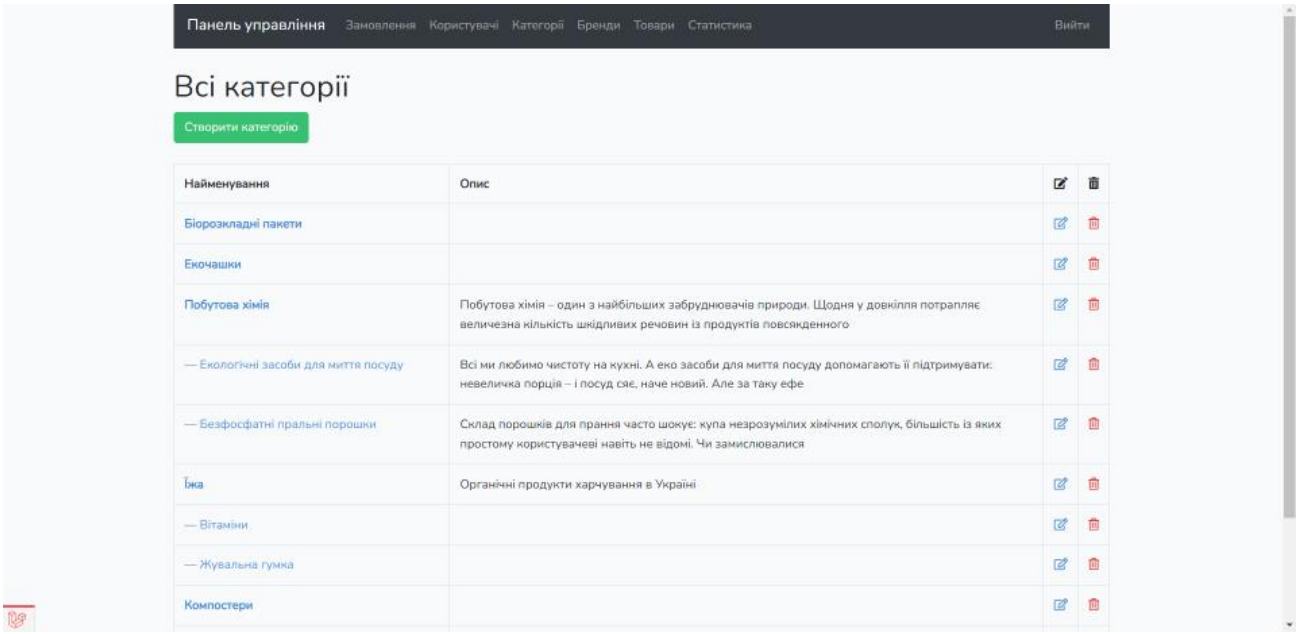


Рисунок 4.22 – Сторінка адміністрування категорій

Джерело: побудовано автором (знімок з екрану)

Сторінка для створення категорії представляє собою форму для введення найменування категорії, короткого опису, зображення та де, за потреби, можна обрати батьківську категорію.

Для додавання товару слід обрати вкладку «Товари» (рис. 4.23). На ній відображаються товари, можна створити новий товар, відредагувати існуючий або видалити його.

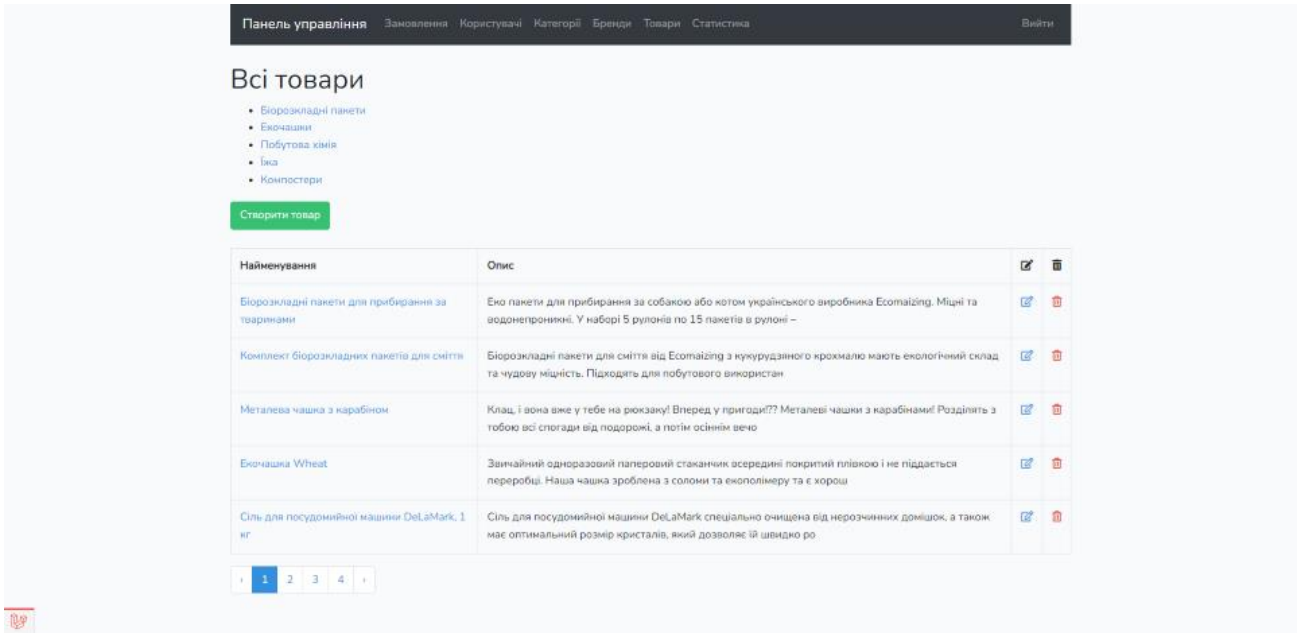


Рисунок 4.23 – Сторінка адміністрування товарів

Джерело: побудовано автором (знімок з екрану)

Сторінка для створення товару представляє собою форму для введення найменування, ціни, опису, додавання зображення та вибору категорії та бренду (рис 4.24). Також можна поставити тег, чи товар є новинкою або лідером продажів.

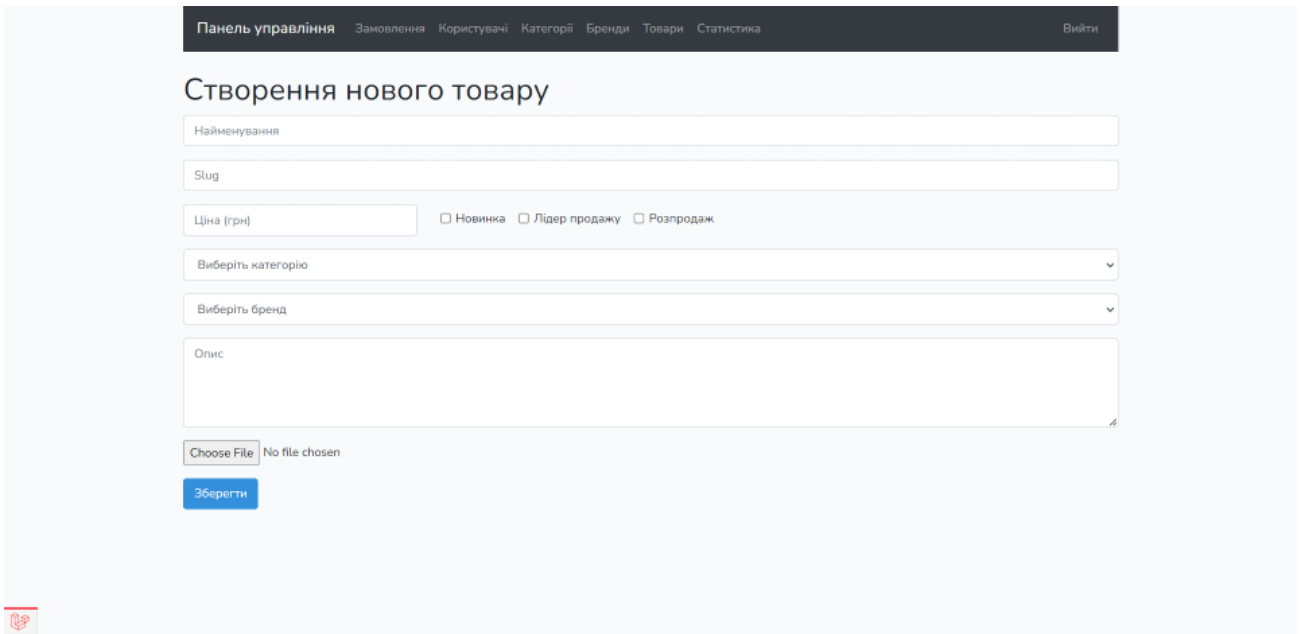


Рисунок 4.24 – Сторінка створення нового товару

Джерело: побудовано автором (знімок з екрану)

Для обробки замовлень існує вкладка «Замовлення» (рис. 4.25). На цій вкладці можна переглядати замовлення та змінювати його статус в процесі обробки (рис 4.26).

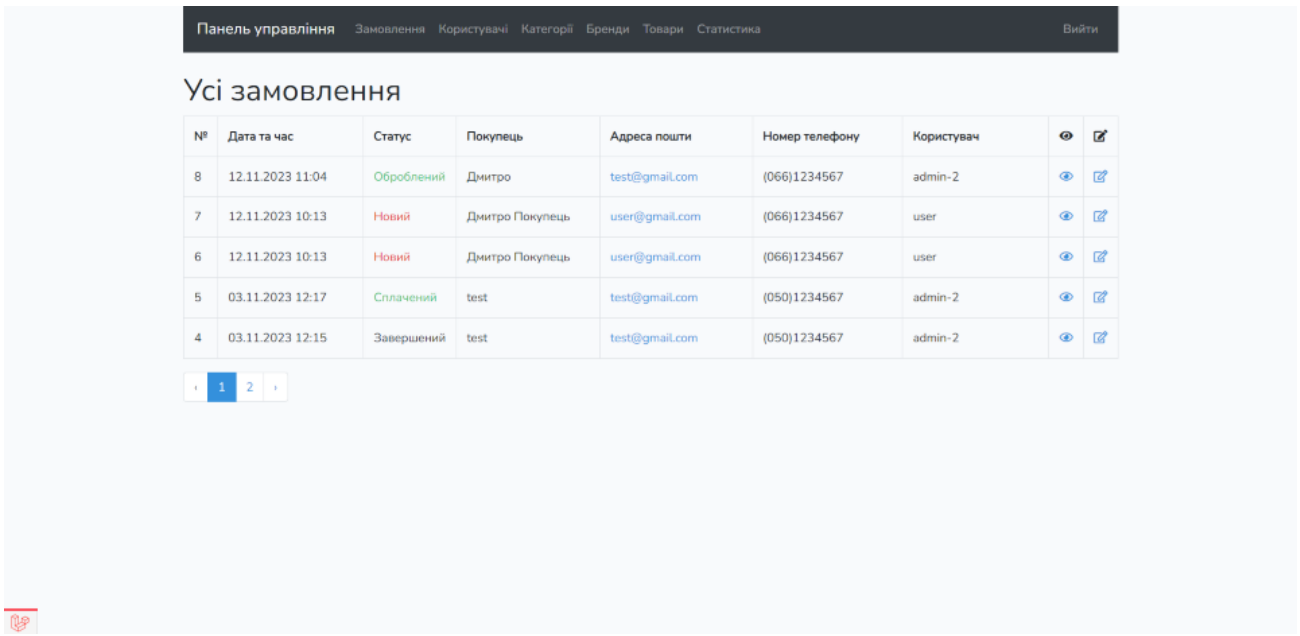


Рисунок 4.25 – Сторінка адміністрування замовлень

Джерело: побудовано автором (знімок з екрану)

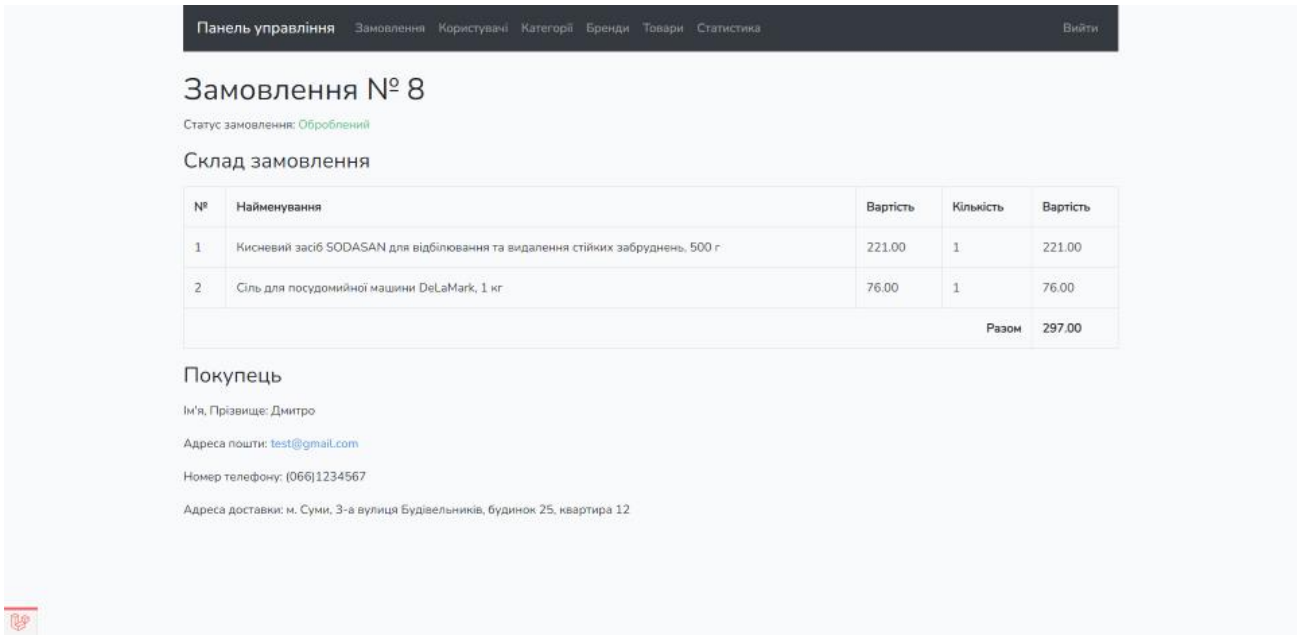


Рисунок 4.26 – Сторінка замовлення

Джерело: побудовано автором (знімок з екрану)

Вкладка «Користувачі» представляє короткі дані по користувачам, кількість замовлень та дату реєстрації (рис. 4.27).

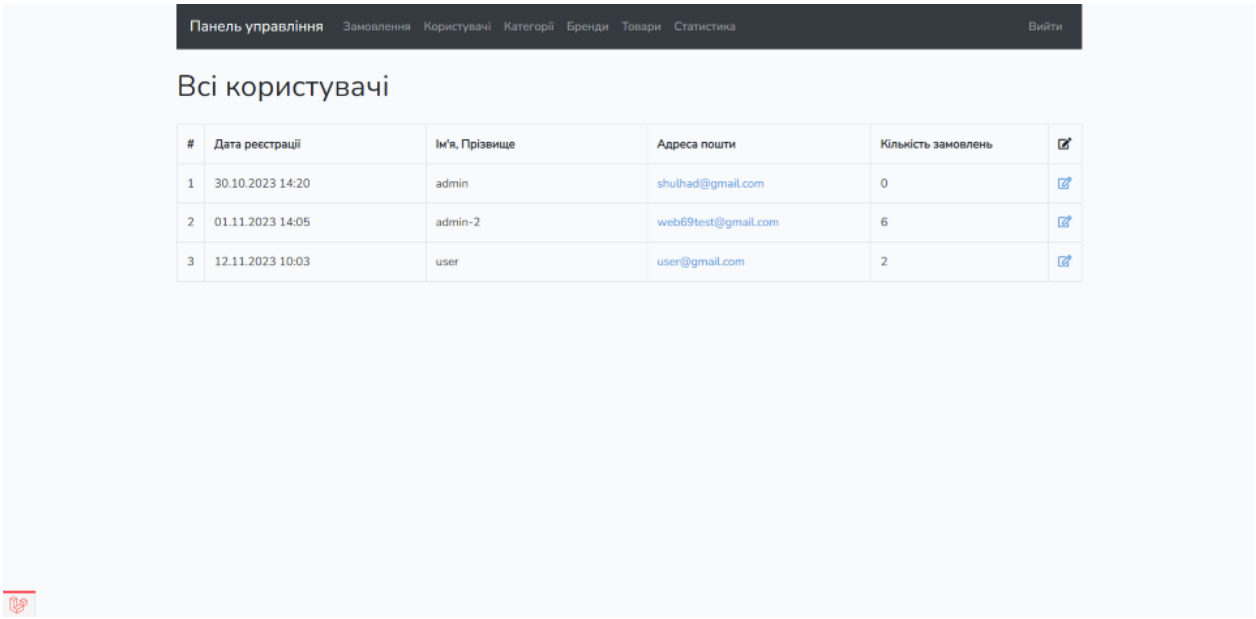


Рисунок 4.27 – Сторінка адміністрування користувачів

Джерело: побудовано автором (знімок з екрану)

Сторінка «Статистика» містить детальну статистику продажів товарів на сайті (рис. 4.28).

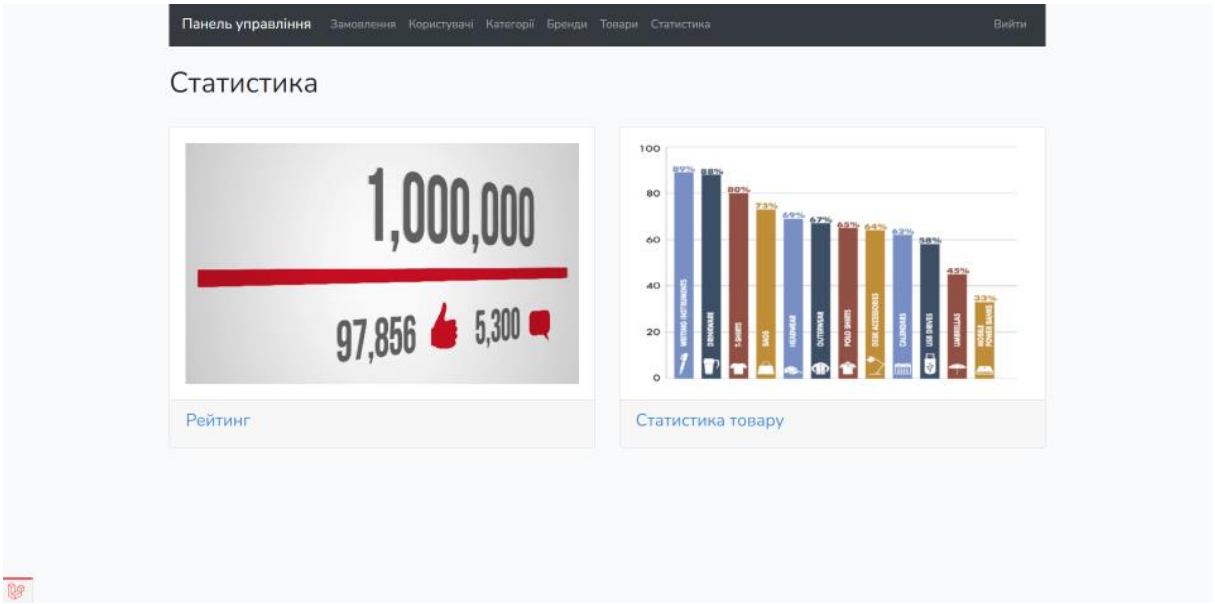


Рисунок 4.28 – Сторінка вибору статистики

Джерело: побудовано автором (знімок з екрану)

Одним із видів статистики є рейтинг товарів, тобто кількість переглядів товару за весь час, топ 5 та останні 5.

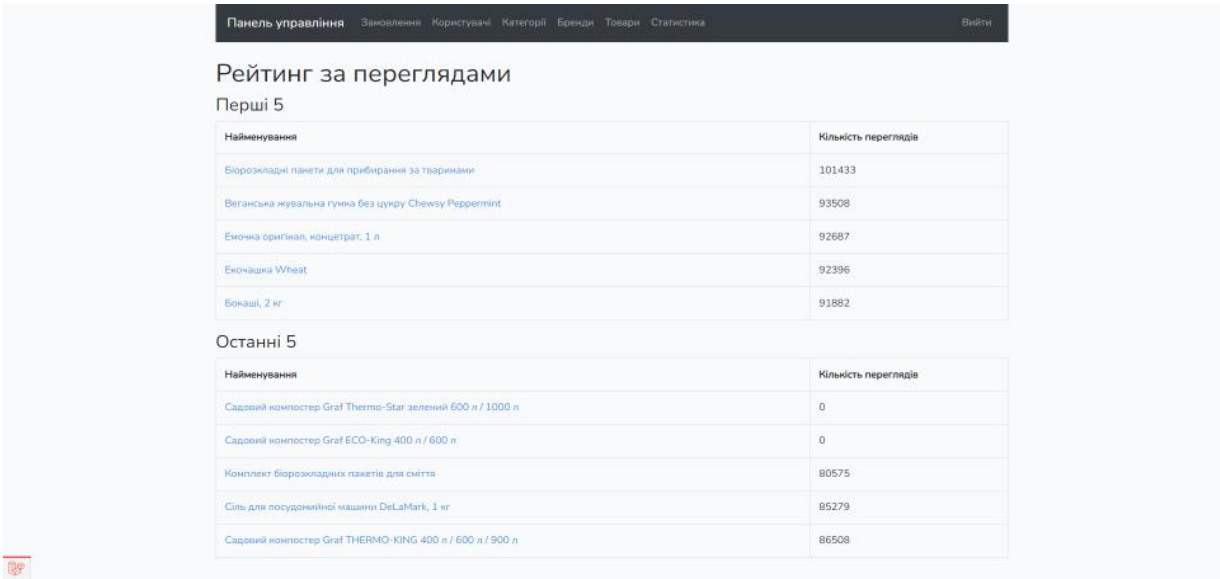


Рисунок 4.29 – Сторінка рейтингу товарів за переглядами

Джерело: побудовано автором (знімок з екрану)

Другий вид аналізу продажів певного товару можна обрати, перейшовши на «Статистика товару». Ця сторінка представляє собою список товарів з двома посиланнями (рис. 4.30).

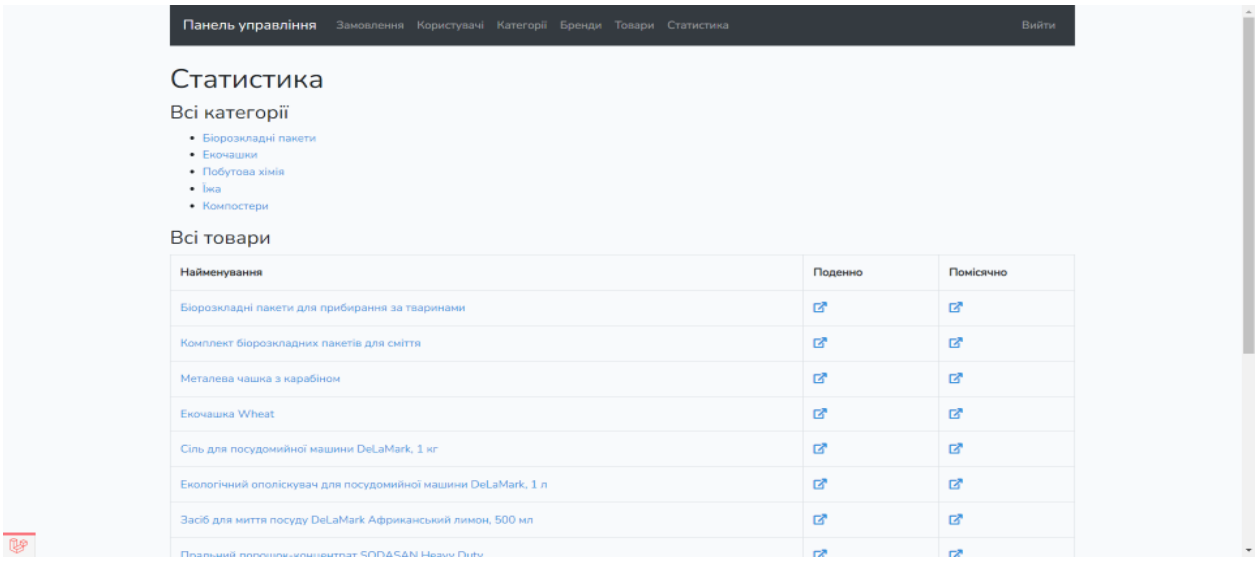


Рисунок 4.30 – Сторінка вибору статистики для певного товару

Джерело: побудовано автором (знімок з екрану)

Статистика товару дозволяє проаналізувати кількість переглядів та замовлень певного товару протягом поточного та минулого місяців поденно (рис. 4.31); проценте відношення замовлень до переглядів товару протягом року помісячно (рис. 4.32).

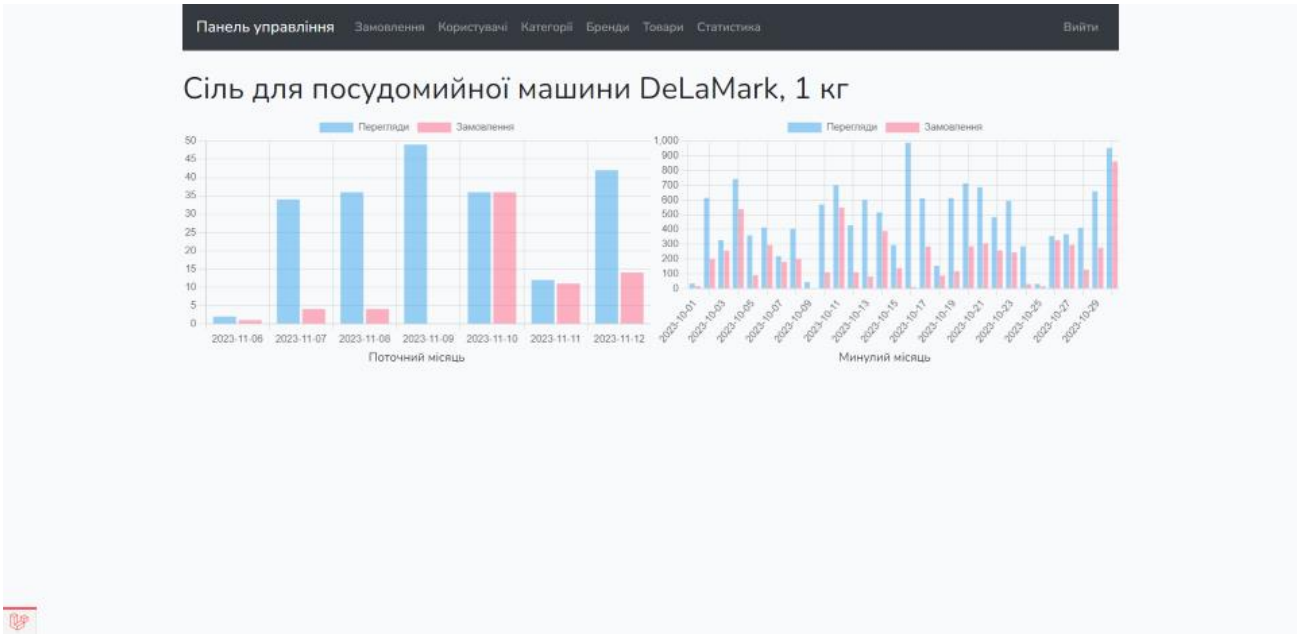


Рисунок 4.31 – Кількість переглядів та замовлень певного товару поденно

Джерело: побудовано автором (знімок з екрану)

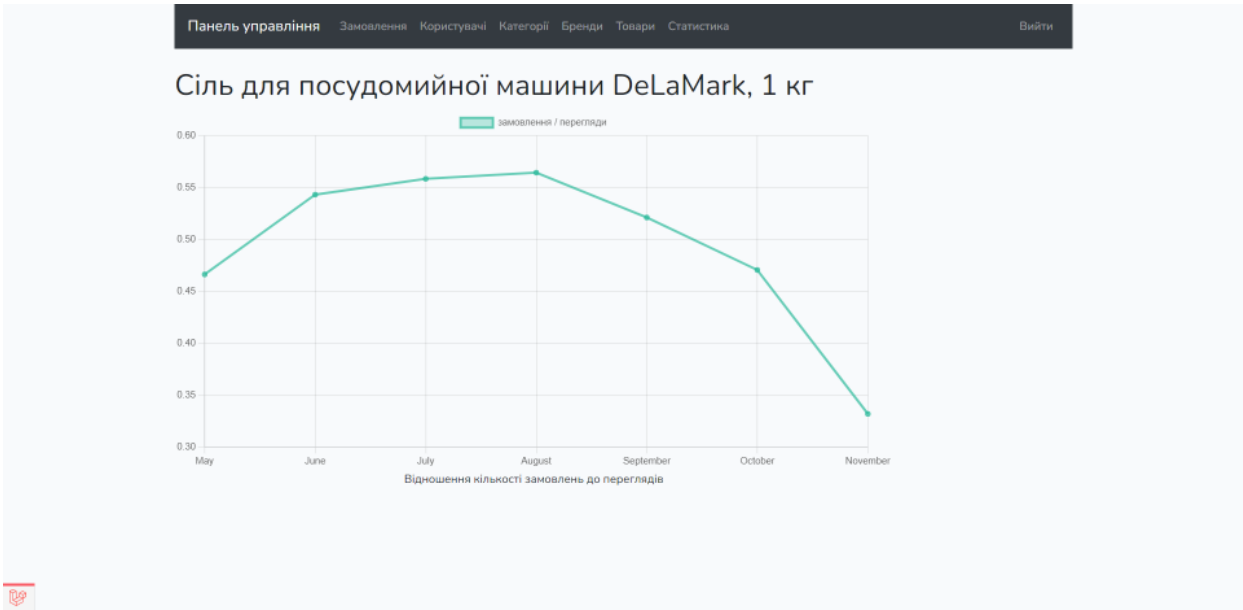


Рисунок 4.32 – Співвідношення замовлень до переглядів товару протягом року помісячно у відсотках

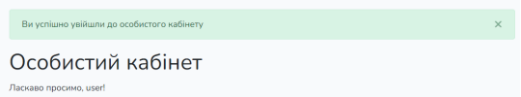
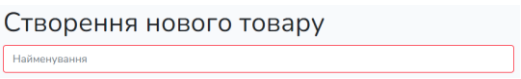
Джерело: побудовано автором (знімок з екрану)

4.6 Тестування та розгортання веб-орієнтованої системи

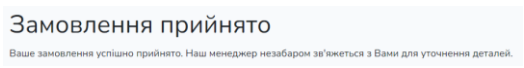
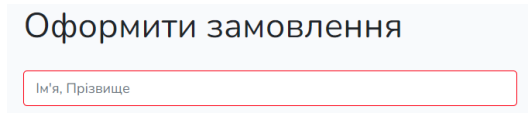
Для перевірки ефективності реалізованих функцій у веб-орієнтованій системі необхідно провести функціональне тестування. Це включає в себе перевірку введених даних у формах створення, редагування та пошуку товарів. Також важливо протестувати роботу гіперпосилань та основних функцій веб-додатку.

Тестування функціоналу веб-орієнтованої системи проводилося за допомогою методу "black box", що передбачає перевірку працездатності системи без доступу до її структури та коду. Результати тестування представлені в таблиці 3.1.

Таблиця 4.1 – Тестування на валідні та невалідні дані

№	Назва тесту	Очікуваний результат	Фактичний результат	0/1
1	Реєстрація користувача	Користувач успішно створений.		1
2	Авторизація користувача	Користувач має доступ до особистого кабінету		1
3	Перевірка форми створення товару введенням коректних даних	Товар успішно створений	Новий товар відображається на сторінці відповідного розділу каталогу	1
4	Перевірка форми створення товару введенням некоректних даних	Не активна кнопка «Зберегти», поле підсвічене червоним кольором		1

Продовження табл. 4.1

5	Перевірка форми оформлення замовлення товару введенням коректних даних	Замовлення прийнято. Відображаються всі подробиці замовлення		1
6	Перевірка форми оформлення замовлення товару введенням некоректних даних	Не активна кнопка «Зберегти», поле підсвічене червоним кольором		1
7	Перевірка пошуку товару	Згідно пошукового запиту відображаються товари	Знайдено товари згідно пошукового запиту	1
8	Перевірка збирання статистики при перегляді товару	Змінюється кількість переглядів певного товару	Відображається зміна на одиницю на графіку кількісті переглядів певного товару поденно.	1
9	Перевірка збирання статистики при замовленні товару	Змінюється кількість замовлень певного товару	Відображається зміна на одиницю на графіку кількісті замовлень певного товару поденно.	1

Джерело: побудовано автором

У ході цього процесу було показано, що інформація повністю та коректно відображається. Товари успішно додаються і належним чином відображаються у відповідних розділах каталогу. Пошук товарів відповідає критеріям пошуку. У результаті проведеного тестування не виявлено блокуючих дефектів. Увесь функціонал працює коректно.

Розгортання проекту розглянемо на прикладі Heroku. Це хмарна платформа для розгортання, управління та масштабування проектів. Щоб розгорнути систему в Heroku треба виконати наступні кроки:

1. Створити Procfile, який повідомляє Heroku, яку команду використовувати для запуску веб-сервера з правильними налаштуваннями. За замовчуванням Heroku запускає веб-сервер Apache разом із PHP для обслуговування програм із кореневого каталогу проекту. Однак кореневим каталогом документа системи є підкаталог public/, тому потрібно створити Procfile, який налаштовує правильний кореневий каталог документа.

2. Створити новий застосунок на Heroku. Щоб його створити, треба використати команду: $ heroku create. У цьому випадку для проекту встановлюється випадкова назва, яку можна змінити в майбутньому.

3. Налаштувати ключ шифрування Laravel. Ключ шифрування програми використовується Laravel для шифрування сеансів користувачів та іншої інформації. Його значення буде зчитано зі змінної середовища APP_KEY.

4. Завантажити систему на Heroku. Це можна зробити за допомогою команди: $ git push heroku main

Можна перевіряти роботу системи в браузері за URL-адресом, який надав Heroku.

ВИСНОВКИ

У рамках дослідження предметної області були розглянуті аналоги програмного забезпечення для підтримки діяльності онлайн-магазину Еко-товарів. Визначено функціональні вимоги, сформульовано цілі та завдання для їх реалізації. Також проведено планування, створено структуру проекту, розроблено календарний план і проаналізовано ризики. Здійснено структурно-функціональне моделювання, створено діаграму IDEF0 та її декомпозицію, а також діаграму варіантів використання.

Проведений огляд та аналіз існуючих технологій розробки web-орієнтованих інформаційних систем дозволив обрати ефективний та швидкий PHP-фреймворк Laravel 8.

В ході реалізації була створена модель та розроблена база даних. Також розроблена серверна частина, яка відповідає за обробку запитів і реалізацію функціоналу онлайн-магазину. Додатково впроваджено систему швидкого та повного пошуку товарів. Додано збір статистики продажів товарів за певні періоди для проведення аналізу продажів та збільшення їх кількості в майбутньому.

Результатом кваліфікаційної роботи магістра є готова веб-орієнтована інформаційна система для підтримки діяльності онлайн-магазину Еко-товарів, яка дозволяє швидко та повно шукати товари, а менеджеру – ефективно проводити аналітику продажів. В майбутньому інформаційна система може бути вдосконалена та оснащена додатковими функціями для підтримки роботи менеджера з продажів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Cao X. The relationships between e-shopping and store shopping in the shopping process of search goods. *Transportation Research Part A: Policy and Practice*. 2012. Vol 46, №7. P. 993-1002

2 Miguel Jaller, Anmol Pahwa. Evaluating the environmental impacts of online shopping: A behavioral and transportation approach. *Transportation Research Part D.* 2020. Vol. 80. P.102223

3 Farag S., Schwanen T., Dijst M., Faber J. Shopping online and/or in-store? A structural equation model of the relationships between e-shopping and in-store shopping. *Transport. Res. Part A.* 2007. Vol.41. P. 125–141.

4 Weltevreden J.W.J., Rietbergen T.V. E-shopping versus city centre shopping: the role of perceived city centre attractiveness. *J. Econ. Social Geogr.* 2007. Vol.98 (1). P. 68–85.

5 Zhou Y., Wang X. Explore the relationship between online shopping and shopping trips: an analysis with the 2009 NHTS data. *Transport. Res. Part A.* 2014. Vol.70, 1–9.

6 Brown J.R., Guiffrida A.L. Carbon emission comparison of last mile delivery versus customer pickup. *Int. J. Logist. Res. Appl.* 2014. Vol. 17 (6). P. 503–521.

7 Durand, B., Feliu, J.G. Urban logitics and e-grocery: have proximity delivery services a positive impact on shopping trips? *Social Behav. Sci.* 2012. Vol. 39. P. 510–520.

8 Aziz N. N. A., Wahid N. A. Why Consumers are Hesitant to Shop Online: The Major Concerns towards Online Shopping. International Journal of Academic *Research in Business and Social Sciences.* 2018. Vol.8, № 9. P. 1175–1185.

9 Al-Debei, M. M., Akroush, M. N., Ashouri, M. I. Consumer attitudes towards online shopping: The effects of trust, perceived benefits, and perceived web quality. *Internet Research*. 2015 Vol.25, № 5. P. 707–733.

10 Huseynov, F., Yildirim, S. O. Internet users' attitudes toward business-to-consumer online shopping: A survey. *Information Development.* 2016. Vol. 32, № 3. P. 452–465.

11 Aziz N. N. A., Wahid N. A. Factors influencing online purchase intention among university students. *International Journal of Academic Research in Business and Social Sciences.* 2018 Vol.8, № 7. P. 702– 717.

12 Lim Y. J., Osman A., Salahuddin S. N., Romle A. R., Abdullah S. (2016). Factors influencing online shopping behavior: the mediating role of purchase intention. *Procedia Economics and Finance.* 2016. Vol.35. P. 401–410.

13 Nwaizugbo I. C., Ifeanyichukwu C. D. (2016). Understanding consumers' behaviour towards online shopping: A study of online shoppers in Anambra State. *International Journal of Sales, Retailing and Marketing*. 2016. Vol. 5, № 2. P. 28–38.

14 Виноградова О.В., Євтушенко Н.О., Крючок І.С. Електронна комерція в епоху диджиталізації. *Причорноморські економічні студії*. 2020. Випуск 53. С. 55-61

15 Patel Savan K., Rathod V.R., Prajapati Jigna B. Performance Analysis of Content Management Systems- Joomla, Drupal and WordPress. *International Journal of Computer Applications*. 2011. Vol. 21, No.4. 0975 – 8887

16 Hembram M. Comparative Study of Open Source Content Management Systems (CMS) in Digital Era. *Asian Journal of Electrical Sciences*. 2022. Vol. 11, No 1. P. 12-16

17 Яку CMS вибрати для інтернет-магазину [Електронний ресурс] – Режим доступу до ресурсу: https://hostiq.ua/blog/ukr/ecommerce-cms/ (дата звернення: 05.09.2023)

18 Fernandes S., Vidyasagar A. Digital Marketing and Wordpress. *Indian Journal of Science and Technology*. 2015. Vol 8, S4. P. 61-68.

19 WordPress.ORG. About WordPress. [Електронний ресурс] – Режим доступу до ресурсу: http://wordpress.org/about/ (дата звернення: 10.09.2023)

20 Lakshmi D. R., Mallika S. S. A Review on Web Application Testing and its Current Research Directions. *International Journal of Electrical and Computer Engineering (IJECE)*. 2017. Vol. 7, No. 4. P. 2132.

21 Kivunja C. Distinguishing between theory, theoretical framework, and conceptual framework: A systematic review of lessons from the field. 2018. *International Journal of Higher Education*. Vol.7, No.6. P. 44-53

22 Norhaidah A. Haris, Nurdatillah Hasim. PHP frameworks usability in web application development. 2019. *International Journal of Recent Technology and Engineering*. 2019. Vol.8, No.3. P.109-116

23 Laaziri M., Benmoussa K., Khoulji S., Kerkeb M. A Comparative study of PHP frameworks performance. *Procedia Manufacturing*. 2019. Vol. 32. P. 864-871

24 N. Prokofyeva, V. Boltunova. Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. *Procedia Comput. Sci.* 2017. Vol. 104. P. 51–56.

25 Laaziri M., Benmoussa K., Khoulji S., Larbi K. M. A comparative study of laravel and symfony PHP frameworks. *International Journal of Electrical and Computer Engineering*. 2019. Vol. 9, No. 1. P. 704–712.

26 Das R., Saikia D. Prasad. Comparison of Procedural PHP with Codeigniter and Laravel Framework. *Int. J. Curr. Trends Eng. Res. Sci. J. Impact Factor*. 2016. Vol. 2, No.6. P. 42–48.

27 Olanrewaju R. F., Islam T., Ali N. An Empirical Study Of The Evolution Of PHP MVC Framework. *Lecture Notes in Electrical Engineering.* 2015. Vol. 315. P. 399–410.

28 Symfony Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://symfony.com/doc/current/index.html (дата звернення: 12.09.2023)

29 Yii documentation [Електронний ресурс] – Режим доступу до ресурсу: https://yiiframework.com.ua/uk/doc/guide/2/ (дата звернення: 13.09.2023)

30 Yii documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.yiiframework.com/doc/guide/2.0/en (дата звернення: 13.09.2023)

31 Thoutam V. A Study On Python Web Application Framework. *Journal of Electronics, Computer Networking and Applied Mathematics.* 2021. Vol. 11. P. 48-55

31 Django documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/ (дата звернення: 14.09.2023)

32 Zoltán S. Web-development with Laravel framework. *Gradus.* 2021. Vol 8, No 1 P. 211-218

33 Yadav N., Dhakad S. K. LARAVEL: A PHP Framework for E-Commerce Website. *Fifth International Conference on Image Information Processing (ICIIP).* 2019. P. 56-62

34 Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 20.09.2023)

35 Visual Studio Code Docs [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs (дата звернення: 21.09.2023)

36 Що таке Laravel? Розробка сайтів на Ларавел [Електронний ресурс] – Режим доступу до ресурсу: https://it-rating.ua/scho-take-laravel-rozrobka-saytiv-na-laravel (дата звернення: 25.09.2023)

37 Huynh T. S., Tran D. T., Vu Q. H., Nguyen L. Design and Implementation of Web Application Based on MVC Laravel Architecture. *European Journal of Electrical Engineering and Computer Science*. 2022. Vol 6, Issue 4. P.32-38

38 База даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 26.09.2023)

39 MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/products/workbench/ (дата звернення: 26.09.2023)

40 MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MySQL_Workbench (дата звернення: 27.09.2023)

41 Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/

42 Chart.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.chartjs.org/docs/latest/ (дата звернення: 28.09.2023)

43 Xinyuan Changa. The Analysis of Open Source Search Engines, *Highlights in Science, Engineering and Technology EMIS 2022*. 2023. Vol. 32. P.125-129.

44 Kwiatkowska M., Norman G., Parker D. Stochastic model checking. *Proceedings of the Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM)*, 2007. Berlin: Springer. P. 220–270.

45 Forejt V., Kwiatkowska M., Norman G., Parker D. Automated verification techniques for probabilistic systems. *Proceedings of the Formal Methods for Eternal Networked Software Systems (SFM).* 2011. Berlin: Springer. P. 53–113.

46 Clarke E., Grumberg O., Jha S. Counterexample-guided abstraction refinement. *Computer Aided Verification*. 2000. Berlin: Springer. P. 154–169.

47 Петрик М.Р., Петрик О.Ю. Моделювання програмного забезпечення: науково-методичний посібник. Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2015. – 200 с.

48 Automation of strategy using IDEF0 - A proof of concept / Waissi GDemir MHumble J et al. *Operations Research Perspectives.* 2015. Vol. 2. P.106-113

49 Нотація IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0 (дата звернення: 01.10.2023)

50 Харів Н. О.. Бази даних та інформаційні системи: навчальний посібник Рівне: НУВГП, 2018. – 127 с.

51 Корнієнко С. К. Системи баз даних: організація та проектування: Навч. Посібник. Запоріжжя: ЗНТУ, 2006. – 252 с.

52 Магазин на Laravel 7 [Електронний ресурс] – Режим доступу до ресурсу: https://tokmakov.msk.ru/blog/tags/314 (дата звернення: 05.10.2023)

53 Laravel Eloquent: Getting Started [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/eloquent (дата звернення: 06.10.2023)

54 Laravel Eloquent: Relationships [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/eloquent-relationships (дата звернення: 08.10.2023)

55 Laravel Migrations [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/migrations (дата звернення: 10.10.2023)

56 Laravel Seeding [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.comdocs/8.x/seeding (дата звернення: 13.10.2023)

57 Laravel Routing [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/routing (дата звернення: 15.10.2023)

58 Laravel Controllers [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/controllers (дата звернення: 16.10.2023)

59 Laravel Middleware [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/middleware (дата звернення: 16.10.2023)

60 Laravel Blade [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/blade (дата звернення: 17.10.2023)

61 Laravel Scout [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/8.x/scout (дата звернення: 18.10.2023)

62 Getting started with Laravel Scout and Meilisearch [Електронний ресурс] – Режим доступу до ресурсу: https://laravel-news.com/getting-started-laravel-scout-meilisearch (дата звернення: 20.10.2023)

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART.

Для конкретизації мети проекту застосовано методологію SMART. Це інноваційний підхід до формулювання працюючих цілей. Використання системи SMART-цілей на етапі визначення мети дозволяє узагальнити всю необхідну інформацію, встановити прийнятні терміни виконання, визначити наявність достатніх ресурсів та сформулювати чіткі та конкретні завдання. Суть конкретизації мети проекту з використанням SMART-методу випливає із розкодування термінів, які складають його назву: конкретна (Specific), вимірювана (Measurable), досяжна (Achievable), реалістична (Relevant), обмежена у часі (Time-framed). Розглянемо детальніше кожен з критеріїв SMART для мети проекту в таблиці А.1 дипломної роботи за технологією SMART.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробка web-орієнтованої системи підтримки діяльності онлайн-магазину Еко-товарів
Measurable (вимірювана)	Створити високоякісний програмний виріб, використовуючи обмежену кількість ресурсів
Achievable (досяжна)	Реалізація інформаційної системи виконується за допомогою фреймворку Laravel.
Relevant (реалістична)	Розробник володіє необхідним рівнем компетентності та має необхідні програмні та апаратні засоби для втілення та подальшої підтримки веб-орієнтованої інформаційної системи.
Time-framed (обмежена у часі)	Робота повинна бути виконана у терміни згідно календарного плану.

Джерело: побудовано автором

Планування змісту структури робіт IT-проекту (WBS).

WBS (Work Breakdown Structure) − це узгоджена ієрархічна декомпозиція завдань, яку необхідно виконати для досягнення мети проекту та створення конкретних результатів. Основна мета − визначити та структурувати всі елементи проекту. Іншими словами, WBS включає в себе всі заплановані роботи в межах проекту. Переваги декомпозиції включають:

- ефективне відображення обсягу проекту;
- фокус на очікуваному результаті, а не на процесі;
- забезпечення єдиного розуміння результатів проекту для розробників і замовників;
- наочне обгрунтування потреби в ресурсах;
- допомога в управлінні ризиками та змінами;
- визначення і погодження контрольних точок проекту.

Таким чином, для ефективного управління проектом була створена WBS-діаграма, яка представлена на рис. А.1.

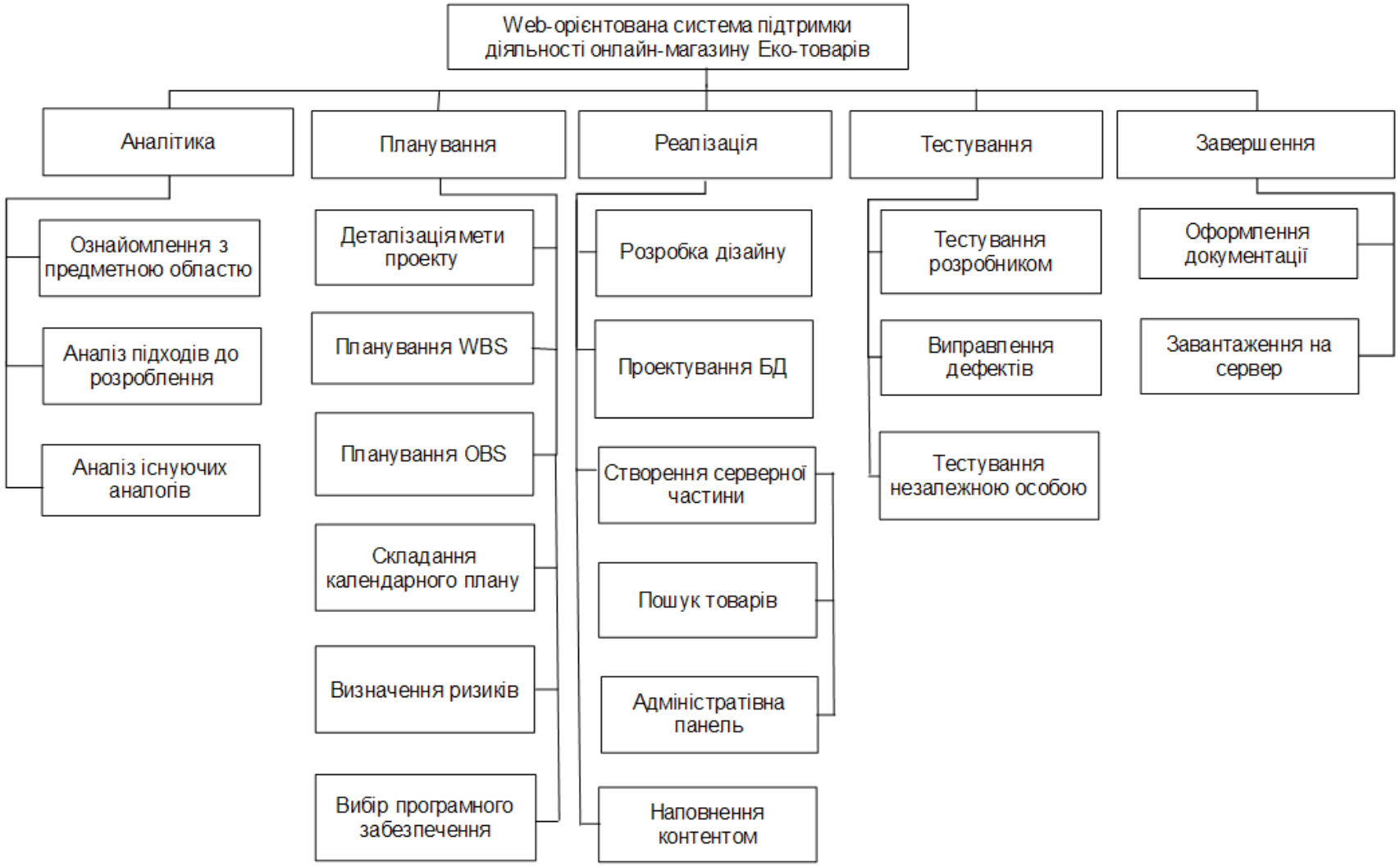


Рисунок А.1 – Таблиця WBS (work breakdown structure)

Джерело: побудовано автором

Організаційна структура проекту (OBS).

Основою контролю за проектом є не лише визначення робіт, але також встановлення загальних підходів до їх виконання. Додатково до WBS використовується організаційна структура розподілу (OBS), яка забезпечує необхідні ресурси для виконання проектних завдань. Ця структура стосується тільки внутрішньої організаційної структури проекту і не зачіпає відносин проектних груп або учасників з батьківськими організаціями. Інтеграція OBS і WBS гарантує, що всі аспекти проекту враховуються, і кожен елемент роботи призначається на відповідальний рівень для планування, відстеження прогресу, витрат та звітності. Тому після розробки WBS-діаграми необхідно створити OBS-діаграму для організації процесу розробки проекту через розподіл обов'язків між учасниками проекту.

Діаграму OBS наведено на рис.А.2.

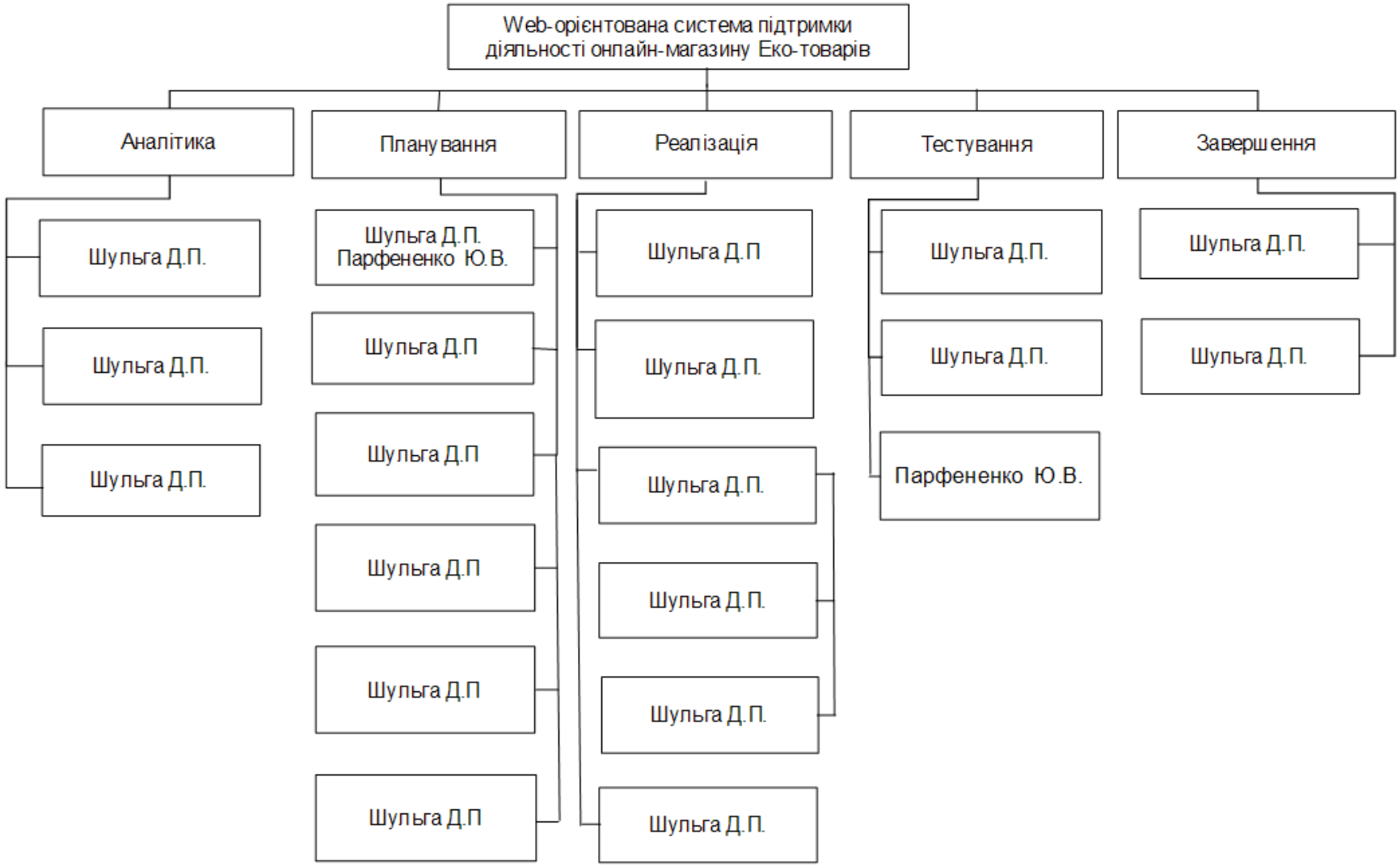


Рисунок А.2 – Таблиця OBS (Organization Breakdown Structure)

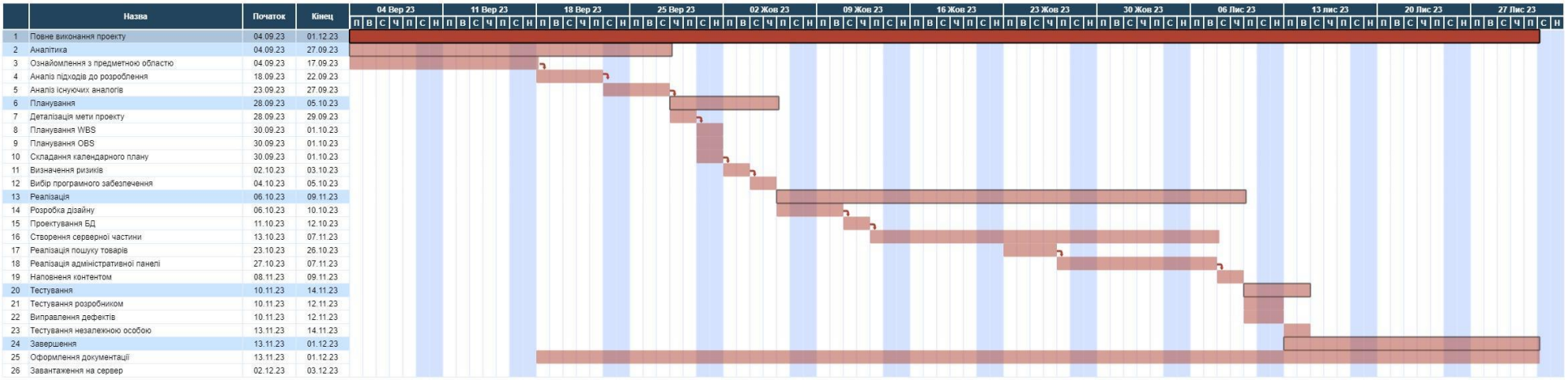
Джерело: побудовано автором

Побудова календарного графіка виконання ІТ-проекту.

Календарне планування включає в себе визначення та коригування термінів виконання всього проекту і його окремих завдань, відображення взаємозв'язків між різними елементами проекту, оптимізацію розподілу ресурсів у часі. Горизонтальна шкала, використовувана для візуалізації плану робіт за проектом у відношенні до часу, відома як діаграма Ганта.

За допомогою цього інструменту керівники проектів і менеджери продукту розбивають завдання на робочі елементи для полегшення управління, збереження порядку та виділення залежностей між ними.

Використання діаграм Ганта дозволяє спростити проект, адже цей інструмент наочно та ефективно узагальнює велику кількість інформації. Велика кількість учасників, команд або зацікавлених осіб може без проблем вносити завдання та адаптуватися до змін обсягу роботи за допомогою цієї діаграми. Додатковою перевагою є те, що діаграма Ганта дає загальне уявлення про проект, включаючи контрольні точки та терміни виконання. Використання діаграми Ганта може бути розглянуте як ефективний інструмент для попередження можливих проблем у ранній стадії. Деталізуємо створену діаграму Ганта для заданої інформаційної системи (рис. А.3).



	Назва	Початок	Кінець
1	Повне виконання проекту	04.09.23	01.12.23
2	Аналітика	04.09.23	27.09.23
3	Ознайомлення з предметною областю	04.09.23	17.09.23
4	Аналіз підходів до розроблення	18.09.23	22.09.23
5	Аналіз існуючих аналогів	23.09.23	27.09.23
6	Планування	28.09.23	05.10.23
7	Деталізація мети проекту	28.09.23	29.09.23
8	Планування WBS	30.09.23	01.10.23
9	Планування OBS	30.09.23	01.10.23
10	Складання календарного плану	30.09.23	01.10.23
11	Визначення ризиків	02.10.23	03.10.23
12	Вибір програмного забезпечення	04.10.23	05.10.23
13	Реалізація	06.10.23	09.11.23
14	Розробка дізайну	06.10.23	10.10.23
15	Проектування БД	11.10.23	12.10.23
16	Створення серверної частини	13.10.23	07.11.23
17	Реалізація пошуку товарів	23.10.23	26.10.23
18	Реалізація адміністративної панелі	27.10.23	07.11.23
19	Наповнення контентом	08.11.23	09.11.23
20	Тестування	10.11.23	14.11.23
21	Тестування розробником	10.11.23	12.11.23
22	Виправлення дефектів	10.11.23	12.11.23
23	Тестування незалежною особою	13.11.23	14.11.23
24	Завершення	13.11.23	01.12.23
25	Оформлення документації	13.11.23	01.12.23
26	Завантаження на сервер	02.12.23	03.12.23

Рисунок А.3 – Діаграма Ганта проекту

Джерело: побудовано автором

Управління ризиками.

Ризик – це подія, ймовірність якої може призвести до негативних або позитивних наслідків для проекту. Керування ризиками – це процес реагування на події та зміни в ризиках під час виконання проекту. Важливо активно вести моніторинг ризиків у цьому процесі.

Розробка стратегії взаємодії з ризиками – це процес створення плану дій і визначення шляхів для збільшення можливостей та зменшення загроз для досягнення цілей проекту. Цей етап розпочинається після проведення якісного і кількісного аналізу можливих ризиків. Під час аналізу для визначення числових значень ймовірності та ступеня впливу часто використовується емпірична шкала оцінки ризику: 5 балів – критичний ризик; 4 бали – максимальний ризик; 3 бали – високий ризик; 2 бали – нормальний ризик ; 1 бал – малий ризик.

Аналізуючи процес розробки проекту було сформовано можливі ризики впливу на якість та ефективність кінцевого продукту (табл. А.1).

Таблиця А.1. Ймовірність виникнення і величина ризику

№	Ризики	Виникнення	Втрати
1	Поверхневе ознайомлення з предметною областю	3	2
2	Затримка у фінансування	3	2
3	Поява позапланових вимог до проекту	4	4
4	Зміни пріоритетів з боку керівництва	2	3
5	Нестабільність програмного забезпечення	2	4
6	Нестача кваліфікованих спеціалістів на проекті	2	5
7	Недотримання графіку робіт	2	4
8	Відсутність тестування	3	4

Джерело: побудовано автором

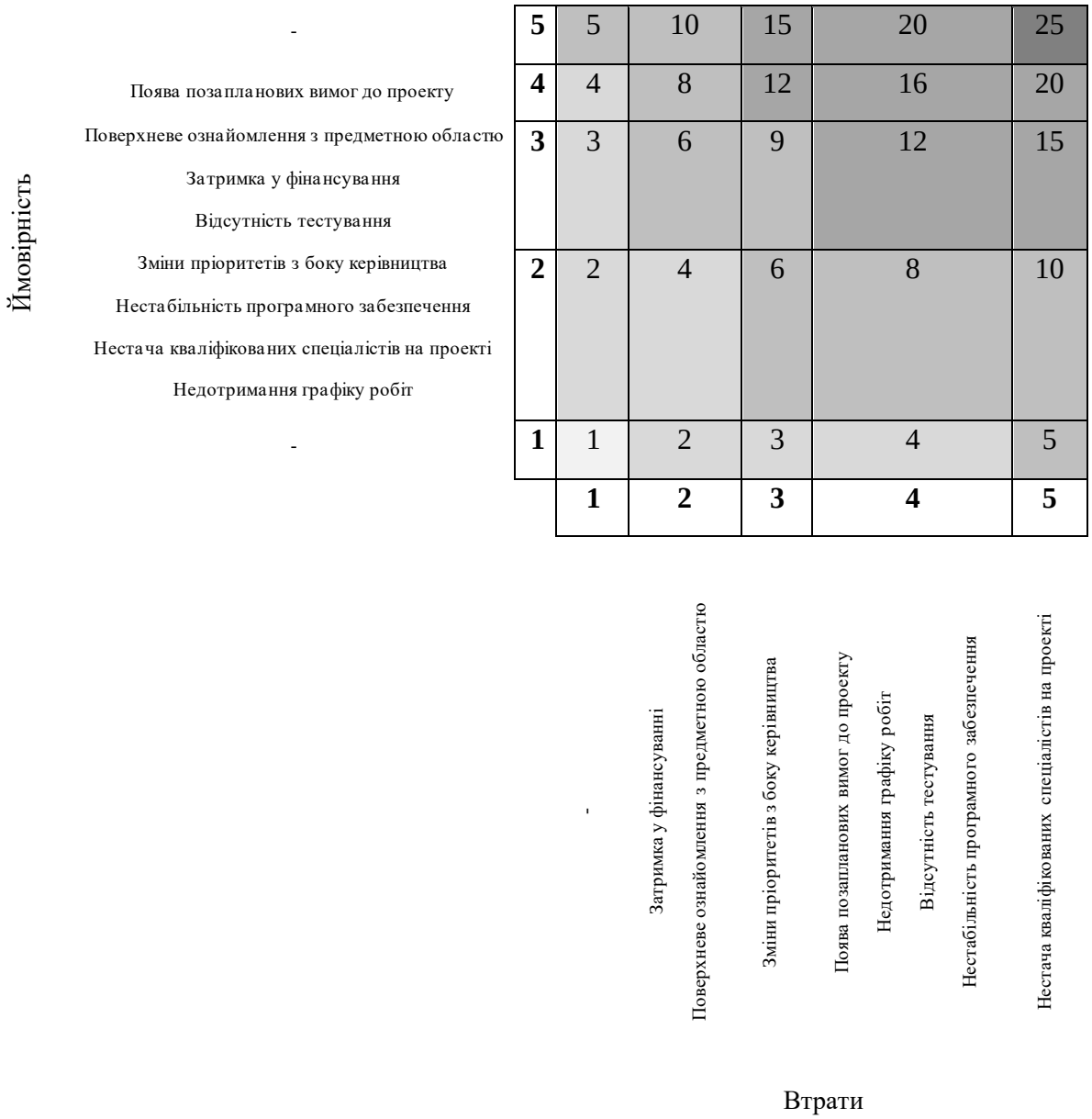


Рисунок А.4 – Матриця «Ймовірність – Втрати»

Джерело: побудовано автором

Таблиця А.2 – Класифікація за ступенем впливу та за рівнем ризику

Ризик	Ступінь впливу	Рівень ризику
Поверхневе ознайомлення з предметною областю	Виправдані	Помірні
Затримка у фінансуванні	Виправдані	Незначні
Поява позапланових вимог до проекту	Неприпустимі	Істотні
Зміни пріоритетів з боку керівництва	Виправдані	Незначні
Нестабільність програмного забезпечення	Виправдані	Незначні
Нестача кваліфікованих спеціалістів на проекті	Виправдані	Помірні
Недотримання графіку робіт	Виправдані	Незначні
Відсутність тестування	Виправдані	Істотні

Джерело: побудовано автором

ДОДАТОК Б

Файл маршрутів - CatalogController.php

```
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', 'IndexController')->name('index');

Route::group([
    'as' => 'catalog.',
    'prefix' => 'catalog',
], function () {
    Route::get('index', 'CatalogController@index')
        ->name('index');

    Route::get('category/{category:slug}', 'CatalogController@category')
        ->name('category');

    Route::get('brand/{brand:slug}', 'CatalogController@brand')
        ->name('brand');

    Route::get('product/{product:slug}', 'CatalogController@product')
        ->name('product');

    Route::get('search', 'CatalogController@search')
        ->name('search');
});

Route::group([
    'as' => 'basket.',
    'prefix' => 'basket',
], function () {
    Route::get('index', 'BasketController@index')
        ->name('index');
```

```
    Route::get('checkout', 'BasketController@checkout')
        ->name('checkout');

    Route::post('profile', 'BasketController@profile')
        ->name('profile');

    Route::post('saveorder', 'BasketController@saveOrder')
        ->name('saveorder');

    Route::get('success', 'BasketController@success')
        ->name('success');

    Route::post('add/{id}', 'BasketController@add')
        ->where('id', '[0-9]+')
        ->name('add');

    Route::post('plus/{id}', 'BasketController@plus')
        ->where('id', '[0-9]+')
        ->name('plus');

    Route::post('minus/{id}', 'BasketController@minus')
        ->where('id', '[0-9]+')
        ->name('minus');

    Route::post('remove/{id}', 'BasketController@remove')
        ->where('id', '[0-9]+')
        ->name('remove');

    Route::post('clear', 'BasketController@clear')
        ->name('clear');
});

Route::name('user.')->prefix('user')->group(function () {
    Auth::routes();
});

Route::group([
    'as' => 'user.',
    'prefix' => 'user',
    'middleware' => ['auth']
], function () {
    Route::get('index', 'UserController@index')->name('index');

    Route::resource('profile', 'ProfileController');

    Route::get('order', 'OrderController@index')->name('order.index');
```

```
        Route::get('order/{order}', 'OrderController@show')->name('order.show');
    });

    Route::group([
        'as' => 'admin.',
        'prefix' => 'admin',
        'namespace' => 'Admin',
        'middleware' => ['auth', 'admin']
    ], function () {
        Route::get('index', 'IndexController')->name('index');

        Route::resource('category', 'CategoryController');

        Route::resource('brand', 'BrandController');

        Route::resource('product', 'ProductController');

        Route::get('product/category/{category}', 'ProductController@category')
            ->name('product.category');

        Route::resource('order', 'OrderController', ['except' => [
            'create', 'store', 'destroy'
        ]]);

        Route::resource('user', 'UserController', ['except' => [
            'create', 'store', 'show', 'destroy'
        ]]);

        Route::get('statistic/index', 'StatisticController@index')->name('statistic.index');
        Route::get('statistic/product/index', 'StatisticController@indexProduct')-
>name('statistic.product.index');
        Route::get('statistic/product/rating', 'StatisticController@showRating')-
>name('statistic.product.rating');
        Route::get('statistic/product/ratio/{product}', 'StatisticController@showRatio')-
>name('statistic.product.ratio');
        Route::get('statistic/product/{product}', 'StatisticController@showProduct')-
>name('statistic.product.show');
        Route::get('statistic/product/category/{category}', 'StatisticController@category')-
>name('statistic.product.category');
    });
```

Контролер для роботи з каталогом товарів - CatalogController.php

```
<?php

namespace App\Http\Controllers;

use App\Helpers\ProductFilter;
use App\Models\Brand;
use App\Models\Category;
use App\Models\Product;
use Illuminate\Http\Request;

class CatalogController extends Controller {
   public function index() {
      $roots = Category::where('parent_id', 0)->get();
      $brands = Brand::popular();
      return view('catalog.index', compact('roots', 'brands'));
   }

   public function category(Category $category, ProductFilter $filters) {
      $products = Product::categoryProducts($category->id)
         ->filterProducts($filters)
         ->paginate(6)
         ->withQueryString();
      return view('catalog.category', compact('category', 'products'));
   }

   public function brand(Brand $brand, ProductFilter $filters) {
      $products = $brand
         ->products()
         ->filterProducts($filters)
         ->paginate(6)
         ->withQueryString();
      return view('catalog.brand', compact('brand', 'products'));
   }

   public function product(Product $product) {
      $product->increaseViews();
      return view('catalog.product', compact('product'));
   }

   public function search(Request $request) {
      $search = $request->input('query');
      $products = Product::search($search)->paginate(15);
      return view('catalog.search', compact('products', 'search'));
   }
}
```

Контролер для роботи з статистикою - StatisticController.php

```
<?php

namespace App\Http\Controllers\Admin;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Http\Controllers\Controller;
use App\Models\Product;
use App\Models\Category;
use App\Models\Statistic;
use Carbon\Carbon;

class StatisticController extends Controller
{
    /**
     * @return \Illuminate\View\View
     */
    public function index() {
        return view('admin.statistic.index');
    }

    /**
     * @return \Illuminate\View\View
     */
    public function indexProduct() {
        $roots = Category::where('parent_id', 0)->get();
        $products = Product::paginate(15);
        return view('admin.statistic.index_product', compact('products', 'roots'));
    }

    /**
     * @return \Illuminate\View\View
     */
    public function showRating() {
        $topProducts = Statistic::topProducts();
        $lastProducts = Statistic::lastProducts();
        return view('admin.statistic.product_rating', compact('topProducts', 'lastProducts'));
    }

    /**
     * @return \Illuminate\View\View
     */
    public function category(Category $category) {
        $products = $category->products()->paginate(10);
        return view('admin.statistic.category', compact('category', 'products'));
    }
```

```
    private function createChart($collection, $chartName) {
        $labels = [];
        $views = [];
        $orders = [];
        foreach($collection as $collection) {
            $labels[] = $collection->date;
            $views[] = $collection->views;
            $orders[] = $collection->orders;
        }

        return app()->chartjs
        ->name($chartName)
        ->type('bar')
        ->size(['width' => 400, 'height' => 200])
        ->labels($labels)
        ->datasets([
            [
                "label" => "Перегляди",
                'data' => $views
            ],
            [
                "label" => "Замовлення",
                'data' => $orders
            ]
        ])
        ->options([]);
    }

    // кількість щоденних продажів означеного товару протягом місяцу (продаж товару по дням)
    public function showProduct(Product $product) {
        $collection = $product->statistics()->whereMonth('date', Carbon::now()->month)->orderBy('date', 'ASC')->get();
        $currentMonthChart = $this->createChart($collection, 'currentMonthChart');

        $collection = $product->statistics()->whereMonth('date', Carbon::now()->subMonth()->month)->orderBy('date', 'ASC')->get();
        $prevMonthChart = $this->createChart($collection, 'prevMonthChart');

        return view('admin.statistic.product_stat', compact('product', 'currentMonthChart', 'prevMonthChart'));
    }

    // проценте відношення замовлень до переглядів товару протягом року помісячно
    public function showRatio(Product $product) {
        $collection = $product->statistics()
        ->select(
```

```
            DB::raw('sum(views) as sum_views'),
            DB::raw('sum(orders) as sum_orders'),
            DB::raw('YEAR(date) year, MONTH(date) month'))
        ->groupby('year','month')
        ->orderBy('month', 'ASC')
        ->get();

        $labels = [];
        $data = [];
        foreach($collection as $collection) {
            $dateObj = \DateTime::createFromFormat('!m', $collection->month);
            $monthName = $dateObj->format('F');
            $labels[] = $monthName;
            $data[] = (int)$collection->sum_orders/(int)$collection->sum_views;
        }

        $chartjs = app()->chartjs
        ->name('lineChart')
        ->type('line')
        ->size(['width' => 400, 'height' => 200])
        ->labels($labels)
        ->datasets([
            [
                "label" => "замовлення / перегляди",
                'backgroundColor' => "rgba(38, 185, 154, 0.31)",
                'borderColor' => "rgba(38, 185, 154, 0.7)",
                "pointBorderColor" => "rgba(38, 185, 154, 0.7)",
                "pointBackgroundColor" => "rgba(38, 185, 154, 0.7)",
                "pointHoverBackgroundColor" => "#fff",
                "pointHoverBorderColor" => "rgba(220,220,220,1)",
                'data' => $data,
            ]
        ])
        ->options([
            'legend' => [
                'display' => false,
            ]
        ]);

        return view('admin.statistic.product_ratio', compact('product', 'chartjs'));
    }
}
```

Модель статистики - Statistic.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;
use Illuminate\Database\Query\JoinClause;

class Statistic extends Model
{
    protected $fillable = [
        'date',
        'product_id',
        'views',
        'orders',
    ];

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function product() {
        return $this->belongsTo(Product::class);
    }

    public static function topProducts() {
        return self::with('product')
        ->groupBy('product_id')
        ->selectRaw('sum(views) as sum, product_id')
        ->orderBy('sum', 'DESC')
        ->take(5)->get();
    }

    public static function lastProducts() {
        $subQuery = DB::table('statistics')
            ->select('product_id', DB::raw('SUM(views) sum'))
            ->groupBy('product_id');

        return DB::table('products')
            ->leftJoinSub($subQuery, 'last_products', function (JoinClause $join) {
                $join->on('products.id', '=', 'last_products.product_id');
            })->orderBy('sum', 'ASC')
            ->take(5)->get();
    }
}
```

Модель товару - Product.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Laravel\Scout\Searchable;
use Carbon\Carbon;

class Product extends Model {

    use Searchable;

    protected $fillable = [
        'category_id',
        'brand_id',
        'name',
        'slug',
        'content',
        'image',
        'price',
        'new',
        'hit',
        'sale',
    ];

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function category() {
        return $this->belongsTo(Category::class);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function brand() {
        return $this->belongsTo(Brand::class);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsToMany
     */
    public function baskets() {
        return $this->belongsToMany(Basket::class)->withPivot('quantity');
    }
```

```
    /**
     * @return \Illuminate\Database\Eloquent\Relations\HasMany
     */
    public function statistics() {
        return $this->hasMany(Statistic::class);
    }

    private function increase($fieldName) {
        $today = Carbon::now()->format('Y-m-d');
        $statistic = Statistic::where([['product_id', $this->id], ['date', $today]]);
        if (!$statistic->count()) {
            $statistic = new Statistic;
            $statistic->product_id = $this->id;
            $statistic->date = $today;
            $statistic->save();
        }
        $statistic->increment($fieldName);
        return $statistic->first();
    }
    public function increaseViews() {
        return $this->increase('views');
    }

    public function increaseOrders() {
        return $this->increase('orders');
    }

    /**
     * @param \Illuminate\Database\Eloquent\Builder $builder
     * @param integer $id
     * @return \Illuminate\Database\Eloquent\Builder
     */
    public function scopeCategoryProducts($builder, $id) {
        $descendants = Category::getAllChildren($id);
        $descendants[] = $id;
        return $builder->whereIn('category_id', $descendants);
    }

    /**
     * @param \Illuminate\Database\Eloquent\Builder $builder
     * @param \App\Helpers\ProductFilter $filters
     * @return \Illuminate\Database\Eloquent\Builder
     */
    public function scopeFilterProducts($builder, $filters)
    {
        return $filters->apply($builder);
    }
}
```

Файли відображення статистики

- index.blade.php

```
@extends('layout.admin', ['title' => 'Статистика'])

@section('content')
   <h1>Статистика</h1>
   <div class="row">
      <div class="col-6">
         <a href="{{ route('admin.statistic.product.rating') }}">
            <div class="card mt-4 mb-4">
               <div class="card-body">
                  <img src="{{ asset('img/rating.jpg') }}" alt="" class="img-fluid">
               </div>
               <div class="card-footer">
                  <h4>Рейтинг</h1>
               </div>
            </div>
         </a>
      </div>
      <div class="col-6">
         <a href="{{ route('admin.statistic.product.index') }}">
            <div class="card mt-4 mb-4">
               <div class="card-body">
                  <img src="{{ asset('img/stat.png') }}" alt="" class="img-fluid">
               </div>
               <div class="card-footer">
                  <h4>Статистика товару</h1>
               </div>
            </div>
         </a>
      </div>
   </div>
@endsection
```

- index_product.blade.php

```
@extends('layout.admin', ['title' => 'Статистика'])

@section('content')
   <h1>Статистика</h1>
   <h3>Всі категорії</h3>
   <ul>
   @foreach ($roots as $root)
      <li>
         <a href="{{ route('admin.statistic.product.category', ['category' => $root->id]) }}">
            {{ $root->name }}
```

```
            </a>
        </li>
    @endforeach
    </ul>
    <h3>Всі товари</h3>
    @include('admin.statistic.part.table', ['products' => $products])
@endsection
```

- product_rating.blade.php

```
@extends('layout.admin', ['title' => 'Статистика'])

@section('content')
    <h1>Рейтинг за переглядами</h1>
    <h3>Перші 5</h3>
    <table class="table table-bordered">
        <tr>
            <th width="75%">Найменування</th>
            <th>Кількість переглядів</th>
        </tr>
        @foreach ($topProducts as $item)
        <tr>
            <td>
                <a href="{{ route('admin.product.show', ['product' => $item->product->id]) }}">
                    {{ $item->product->name }}
                </a>
            </td>
            <td>
                {{ $item->sum }}
            </td>
        </tr>
        @endforeach
    </table>
    <h3>Останні 5</h3>
    <table class="table table-bordered">
        <tr>
            <th width="75%">Найменування</th>
            <th>Кількість переглядів</th>
        </tr>
        @foreach ($lastProducts as $item)
        <tr>
            <td>
                <a href="{{ route('admin.product.show', ['product' => $item->id]) }}">
                    {{ $item->name }}
                </a>
            </td>
            <td>
```

```
            {{ $item->sum ?? 0}}
          </td>
        </tr>
        @endforeach
    </table>
@endsection
```

- product_ratio.blade.php

```
@extends('layout.admin', ['title' => 'Статистика'])

@section('content')
    <h1>{{$product->name}}</h1>
    <div style="width:80%;">
        {!! $chartjs->render() !!}
        <p class="text-center">Відношення кількості замовлень до переглядів</p>
    </div>
@endsection
```

- product_stat.blade.php

```
@extends('layout.admin', ['title' => 'Статистика'])

@section('content')
    <h1>{{$product->name}}</h1>
    <div class="d-flex">
        <div style="width:50%;">
            {!! $currentMonthChart->render() !!}
            <p class="text-center">Поточний місяць</p>
        </div>
        <div style="width:50%;">
            {!! $prevMonthChart->render() !!}
            <p class="text-center">Минулий місяць</p>
        </div>
    </div>
@endsection
```