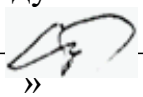
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
___  _ проф. Приходько С.Б.
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Виконав: студент групи 4151
 Гурмаза М.В.
(підпис, ПІБ)

Керівник роботи:
к.ф-м.н., доцент
(посада, науковий ступень вчене звання)
 Латанська Л.О.
(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «магістр»

Студенту Гурмазі Микиті В'ячеславовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Керівник роботи Латанська Л.О.,

Затверджені наказом ректора №153 від 08.04.2025 р.

2. Термін подання роботи: 02.06.2025 р. _____

3. Вихідні дані по роботі: _____

_____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1.Тема 2. Мета роботи, 3. Функції програмного забезпечення, 4. Діаграма варіантів використання програмного застосунку «BioGild». 5. Діаграма діяльності варіанта використання «Ввести дані користувача», 6. Діаграма діяльності варіанта використання «Додати продукт». 7. Інформаційна модель програмного застосунку «BioGild» у вигляді діаграми «Сутність-Зв'язок». 8. Діаграма класів програмного застосунку «BioGild», 9. Діаграма послідовності для варіанта використання «Ввести дані користувача», 10. Діаграма послідовності для варіанта використання «Додати продукт», 11. Логічна модель даних програмного застосунку «BioGild», 12. Вибір програмних засобів для розробки програмного застосунку «BioGild», 13. Фізична модель даних програмного застосунку «BioGild», 14. Діаграма компонентів програмного застосунку «BioGild», 15. Діаграма розгортання програмного застосунку «BioGild», 16. Результати розробки програмного застосунку «BioGild», 17. Висновоки. 18. Висновки.Продовження. _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Гурмаза М В.
(підпис) (ПІБ)

Керівник роботи _____ Латанська Л. О.

АНОТАЦІЯ

Дана робота покликана вирішити практичну задачу інженерії програмного забезпечення, що полягає у розробці програмного забезпечення інтернет-магазину екологічних продуктів «BioGild».

Метою роботи є створення спеціалізованого веб-застосунку, який поєднує функціональність традиційного інтернет-магазину з інструментами для роботи з екологічними характеристиками товарів, такими як сертифікація, екорейтинг та фільтрація за екологічними критеріями.

У кваліфікаційній роботі проведено аналіз існуючих платформ для електронної комерції, виявлено їхні недоліки щодо роботи з екопродуктами та сформульовано вимоги до розробки спеціалізованого рішення. Робота містить повний проєкт програмного забезпечення та розділ з охорони праці.

Робота викладена на 145 сторінках друкованого тексту, містить 24 рисунка, 4 таблиці, 5 додатків та список використаних джерел з 12 найменувань.

ABSTRACT

This work is devoted to solving a practical problem of software engineering which stands in developing the software for the online store of ecological products «BioGild».

The aim of the work is to create a specialized web application that combines the functionality of a traditional online store with tools for working with environmental characteristics of products, such as certification, eco-rating, and filtering by environmental criteria.

The qualification work analyzes existing e-commerce platforms, identifies their shortcomings in working with eco-products, and formulates requirements for the development of a specialized solution. The work contains a complete software project and labor protection section.

The work is presented on 145 pages of printed text, contains 24 figures, 4 tables, 5 appendices, and a list of references with 12 titles.

ЗМІСТ

ВСТУП

У сучасному світі, де екологічна свідомість стає все більш значущою, попит на екологічно чисті продукти стрімко зростає. Споживачі частіше віддають перевагу товарам, які не лише задовольняють їхні потреби, але й мінімізують шкоду для довкілля. Однак, незважаючи на зростання популярності екопродуктів, багато покупців стикаються з труднощами у пошуку надійних джерел, перевірці сертифікації товарів та зручному способі їх придбання.

Традиційні інтернет-магазини часто не враховують специфіку екологічних продуктів, таких як органічна їжа, біорозкладний одяг чи екокосметика. Відсутність спеціалізованих фільтрів, інформації про сертифікацію та екорейтинги ускладнюють процес вибору для клієнтів. Крім того, не всі платформи надають можливість оцінювати товари та залишати відгуки, що є важливим для формування довіри до екобрендів.

Необхідність розробки інтернет-магазину «BioGild» обумовлена потребою у створенні інтернет-магазину, який:

– централізує доступ до екологічних продуктів;

– забезпечує зручний пошук за категоріями, ціною, екорейтингом;

– надає прозору інформацію про сертифікати та склад продуктів;

– дозволяє покупцям залишати оцінки та відгуки;

Метою даної кваліфікаційної роботи є розробка програмного забезпечення інтернет-магазину екологічних продуктів «BioGild» з наступною функціональністю:

– реєстрація, пошук товарів, кошик, оплата, відстеження замовлень;

– функціональність для адміністраторів: управління товарами, обробка замовлень, модерація відгуків, формування звітів;

– екологічні інструменти: фільтрація за сертифікатами, екорейтингом.

Щоб досягти мети, слід виконати наступні завдання:

1) Провести аналіз практичної задачі інженерії програмного забезпечення автоматизації продажу товарів у мережі інтернет.

2) Провести аналіз існуючих підходів до продажу товарів у мережі інтернет – проаналізувати існуючі рішення та їх недоліки.

3) Виконати формалізацію вимог – визначення функціональних та нефункціональних вимог.

4) Виконати проєктування архітектури – вибрати технологій, спроєктувати базу даних.

5) Виконати реалізацію – розробити інтерфейси та бекенд.

6) Виконати тестування – перевірити працездатність, юзабіліті, безпеку.

Об’єкт дослідження: процес розробки інтернет-магазину для екологічних продуктів «BioGild».

Актуальність роботи полягає у поєднанні екологічної складової з сучасними рішеннями для електронної комерції. Програмне забезпечення для інтернет-магазину екологічних продуктів «BioGild» не лише спростить можливість здійснювати покупки клієнтами, але й сприятиме популяризації екологічно зваженого споживання.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BIOGILD»

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Сучасний ринок екологічних продуктів стрімко розвивається через зростання обізнаності споживачів про екологію. Незважаючи на значний інтерес до цієї теми, український сегмент електронної комерції екопродуктів досі залишається недостатньо розвиненим. Це створює значне підґрунтя для запуску спеціалізованого інтернет-магазину «BioGild», який поєднуватиме традиційні підходи електронної торгівлі з особливостями продажу екопродуктів.

Електронна комерція екопродуктів має свої особливості. Споживачі часто стикаються з труднощами у пошуку якісних екопродуктів, перевірці їх сертифікації та оцінці їхнього впливу на довкілля. Існуючі рішення часто не враховують цієї специфіки, пропонуючи лише базову функціональність електронної торгівлі без спеціалізованих інструментів для екопродуктів.

Розробка програмного забезпечення для інтернет-магазину екологічних продуктів «BioGild» передбачає вирішення низки комплексних інженерних завдань. По-перше, необхідно створити зручний інтерфейс із розширеними фільтрами, щоб користувачі могли легко знаходити товари за сертифікацією, екорейтингом чи іншими екологічними параметрами. По-друге, важливо реалізувати засоби верифікації товарів, щоб гарантувати споживачам достовірність інформації про їхні екологічні властивості.

Не менш важливим є впровадження системи рейтингів та відгуків, оскільки це сприяє довірі до екопродуктів. На відміну від звичайних інтернет-магазинів, де відгуки часто носять суб'єктивний характер, «BioGild» має забезпечувати

об'єктивну оцінку екологічних якостей товарів. Цього можна досягти за рахунок комбінації експертних оцінок, офіційних сертифікацій та відгуків споживачів.

Розробка буде проводитися з використанням LAMP-стеку [1], який включає такі компоненти:

– Linux – операційна система, яка керуватиме роботою всіх компонентів.

– Apache HTTP Server – це програмне забезпечення веб-сервера, що забезпечує доставку як статичних, так і динамічних веб-сторінок користувачам.

– MySQL – це система керування реляційними базами даних, яка використовується для створення веб-баз даних, зберігання та управління динамічним контентом.

– PHP – це мови програмування для розробки веб-застоунків.

Схематично LAMP стек можна представити наступним чином, що показано на рисунку 1.1.

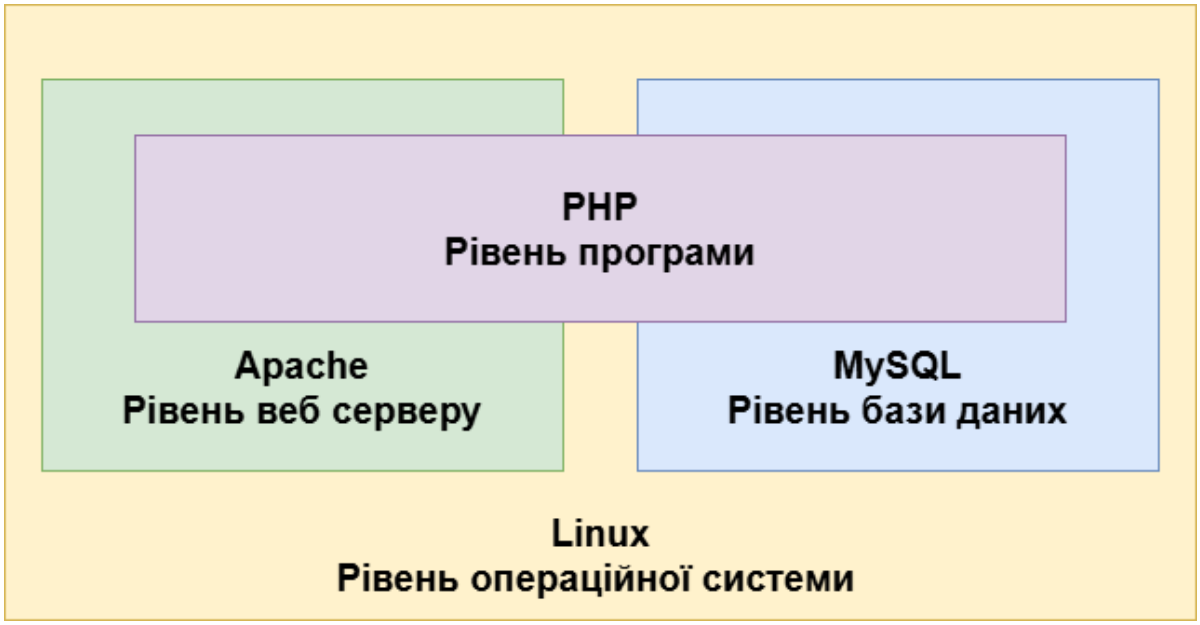


Рисунок 1.1 – Схематичне зображення LAMP стеку

Таким чином, розробка програмного забезпечення для інтернет-магазину екологічних продуктів «BioGild» є комплексним практичним завдання інженерії програмного забезпечення, яке поєднує традиційні підходи електронної комерції з

більш нестандартними рішеннями для онлайн продажу екотоварів. Вирішення цього завдання потребує розуміння як технічних аспектів створення інтернет-магазинів, так і специфіки ринку екопродуктів, що робить проєкт цікавим з точки зору інженерії програмного забезпечення.

1.2 Аналіз існуючих підходів до створенні інтернет магазинів

Платформа Shopify

Shopify [2] – це платформа для електронної комерції, яка дозволяє будь-кому легко запускати онлайн-бізнес, надаючи набір інструментів для керування, створення та розвитку інтернет-магазинів. У 2024 році Shopify стала найбільшою платформою електронної комерції, яка також дозволяє продавати товари та приймати платежі як в цифровому, так і у фізичному форматі.

Shopify – це SaaS (програмне забезпечення як послуга) платформа для продажів, яка пропонує вбудовані інструменти для успішного розвитку в галузі електронної комерції. Ця платформа пропонує всі рішення в одному місці як для невеликого бізнесу, так й для великих компаній. Вона надає повний пакет інструментів для управління бізнесом, включаючи керування магазином і запасами, мультивалютні платежі, доставку, обслуговування клієнтів, маркетинг та аналітику.

Shopify відома своєю гнучкістю та простотою: вона дозволяє легко створювати магазини, налаштовувати теми та шаблони. Платформа дає змогу бізнесам продавати товари, керувати доставкою, обробляти платежі та аналізувати дані. Адміністратор має доступ до моніторингу всього процесу продажів – від оформлення замовлення до доставки.

Створення інтернет-магазину на платформі Shopify ґрунтується на послідовному виконанні низки технологічних операцій. Процес починається з реєстрації облікового запису, де користувач обирає тарифний план та вказує базові дані магазину. Після автентифікації система надає доступ до панелі управління, де

відбувається формування товарного асортименту через спеціалізований інтерфейс із полями для опису продукції, мультимедійного контенту та ціноутворення.

На етапі кастомізації застосовуються інструменти візуального редагування, що дозволяють адаптувати інтерфейс магазину до корпоративного стилю через систему шаблонів із регульованими параметрами типографіки, кольорової схеми та компонування блоків. Технічна інтеграція передбачає прив'язку доменного імені з можливістю використання власного DNS-запису або придбання нового через маркетплейс платформи.

Фінансовий модуль налаштовується шляхом підключення платіжних шлюзів із підтримкою різних валют і методів транзакцій, включаючи вбудоване рішення Shopify Payments. Логістична конфігурація реалізується через визначення географічних зон доставки, розрахунок тарифів на основі параметрів відправлень та синхронізацію з системами провідних перевізників.

Фінальним етапом є активація торговельної площадки з автоматичним розгортанням функціоналу для прийому замовлень, обробки платежів та аналітики продажів. Система забезпечує безперебійний цикл від оформлення покупки до фулфілменту, що дозволяє власникам бізнесу концентруватися на стратегічному розвитку без необхідності технічного адміністрування інфраструктури.

Shopify пропонує різні плани для бізнесів будь-якого рівня, перелік яких наведено у таблиці 1.1.

Таблиця 1.1 – Перелік доступних планів платформи Shopify

План	Для кого?	Ціна на місяць
Basic	Новий бізнес або обмежений бюджет	$29
Shopify	Бізнес на етапі розвитку	$79
Advanced	Великі бізнеси зі складними потребами	$299
Retail	Інтеграція онлайн та офлайн-продажів	$89
Plus	Корпоративні клієнти з високими вимогами	$2300
Enterprise	Великі підприємства (індивідуальний розрахунок)	За запитом

На рисунку 1.2 наведено приклад адмін-частини платформи Shopify

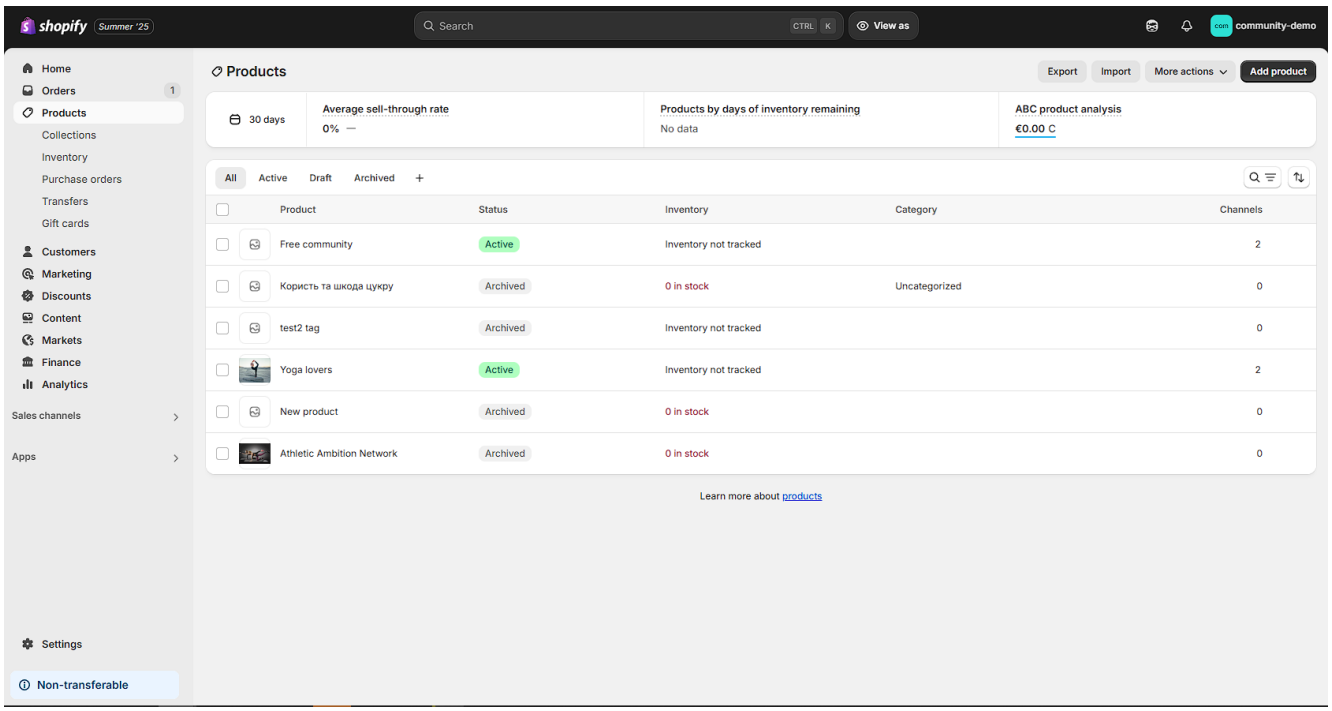


Рисунок 1.2 – Адмін панель магазину Shopify

Shopify – це універсальна платформа для різних типів бізнесу, від невеликих стартапів до великих корпорацій. Вона підходить для:

– Стартапів та малого бізнесу – можливість швидко запустити онлайн-магазин без технічних навичок.

– Ремісників та творчих осіб – продаж авторських виробів, мистецтва, одягу, прикрас.

– Дропшиперів – робота без власного складу, з автоматичним замовленням товарів у постачальників.

– Традиційних магазинів – розширення офлайн-бізнесу за рахунок онлайн-продажів.

– Виробників та дистриб'юторів – керування асортиментом, оптовими замовленнями та логістикою.

– Авторів цифрового контенту – продаж електронних книг, курсів, шаблонів, програмного забезпечення.

– Сервісних компаній – онлайн-запис на послуги, продаж абонементів, проведення подій.

– Великих брендів – масштабування бізнесу з підтримкою багатомовних та мультивалютних продажів.

– Благодійних організацій – збір коштів, продаж мерчу для підтримки соціальних ініціатив.

Платформа підтримує різні формати товарів і послуг:

– Фізичні товари (одяг, взуття, електроніка, косметика, товари для дому).

– Цифрові продукти (електронні книги, музика, фото, відео, програмне забезпечення).

– Послуги та події (консультації, онлайн-курси, квитки на вебінари, майстер-класи).

– Передплатні сервіси (бокси з товарами, регулярні поставки, членські програми).

– Товари за запитом (Print-on-Demand) – індивідуальний друк на футболках, кружках, плакатах.

– Дропшипінг – продаж без власного складу з автоматичною відправкою від виробника.

Переваги Shopify

– Все в одному – платформа надає всі необхідні інструменти для запуску та управління інтернет-магазином: хостинг, домен, платіжні рішення, маркетинг і аналітику.

– Без необхідності знань програмування – інтуїтивно зрозумілий інтерфейс дозволяє створити магазин без технічних навичок за допомогою візуального редактора.

– Безпечний та стабільний хостинг – shopify гарантує високу швидкість завантаження, захист від ddos-атак та автоматичні оновлення без простоїв

– Гнучкі налаштування дизайну – доступні сотні шаблонів (безкоштовних і платних) з можливістю кастомізації під бренд.

– Великий вибір додатків – маркетплейс app store пропонує тисячі додатків для seo, маркетингу, логістики, аналітики та інших задач.

– Якісна підтримка – доступна цілодобова допомога через чат, email та телефон, а також база знань і форуми

Недоліки Shopify

– Комісія за зовнішні платіжні системи – якщо ви не використовуєте shopify payments, з кожної транзакції стягується додаткова комісія (до 2%).

– Обмежений контроль над сторінкою оформлення замовлення – дизайн і функціонал checkout можна редагувати лише в обмеженому діапазоні (без глибокого кастомного кодування).

– Вищі тарифи порівняно з конкурентами – наприклад, woocommerce або opencart можуть бути дешевшими, але вимагають самостійного налаштування хостингу.

– Необхідність платити за додаткові функції – багато корисних інструментів (наприклад, для seo чи автоматизації маркетингу) доступні лише через платні додатки.

Shopify – гарний вибір для тих, хто хоче швидко запустити магазин без технічних складностей. Однак для великих бізнесів з індивідуальними потребами можуть знадобитися додаткові інвестиції в додатки та інтеграції.

WooCommerce

Плагін WooCommerce [3] став основним вибором для багатьох підприємців та власників бізнесу, які мають сайти на платформі WordPress [4]. Він має понад 8 мільйонів активних встановлень у репозиторії цієї CMS [5].

Навіть без досвіду в програмуванні чи веброзробці за допомогою WooCommerce можна створити повнофункціональний інтернет-магазин за короткий час. Недарма його використовують понад 30% усіх електронних комерційних проєктів у світі.

Ця платформа особливо приваблива бізнесу, адже дає можливість створювати функціональні та професійні магазини без значних фінансових витрат.

На рисунку 1.3 наведено приклад адмін-частини плагіну WooCommerce

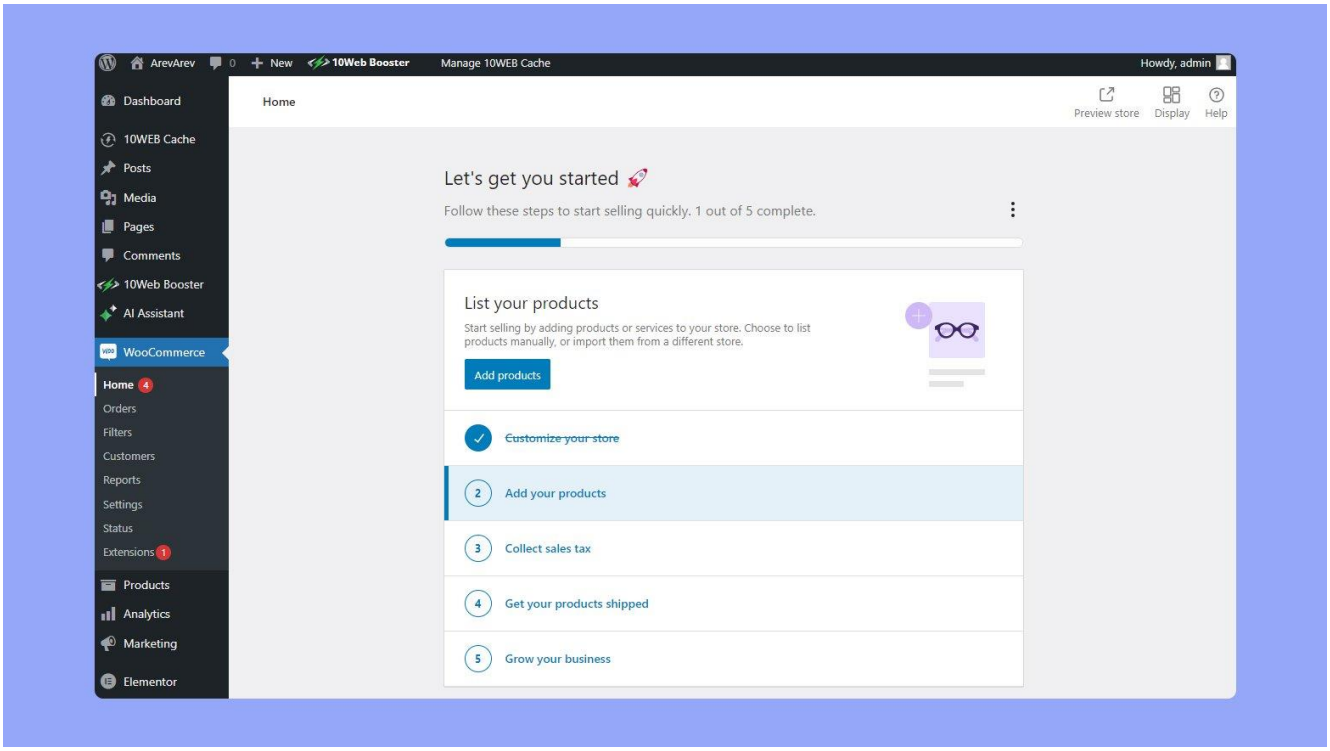


Рисунок 1.3 – Адмін панель застосунку з використанням плагіну WooCommerce

WooCommerce – це потужний плагін для електронної комерції у WordPress, який дозволяє створювати інтернет-магазини будь-якого формату та складності без необхідності вивчати PHP, HTML чи CSS. Він надає всі необхідні інструменти для управління товарами, обробки замовлень, прийому платежів тощо.

Важливо зауважити, що WooCommerce – це не окремий конструктор сайтів (як Wix чи Shopify), а розширення(плагін) WordPress. Він вимагає окремої реєстрації домену, придбання домену й SSL-сертифікату.

Переваги WooCommerce

– Безкоштовність – плагін WooCommerce є повністю безплатним – його можна вільно завантажувати, встановлювати та використовувати. Базового

функціоналу достатньо для створення дизайну, наповнення асортименту та запуску магазину. Відкритий код дозволяє модифікувати систему (при наявності навичок CSS/HTML) або розширювати її можливості.

– Надійність – завдяки відкритому коду та великій спільноті розробників, плагін постійно вдосконалюється. Уразливості оперативно виправляються, а користувачі можуть отримувати допомогу на форумах.

– Контроль даних – на відміну від закритих платформ (на кшталт Shopify), WooCommerce дає повний контроль над даними – вони зберігаються на вашому хостингу, що дозволяє гнучко керувати контентом.

– Масштабованість – система підтримує необмежену кількість товарів і може розширюватися за рахунок додаткових модулів (доступно понад 400 плагінів)

– Підтримка багатьох мов – завдяки стороннім плагінам для WordPress (наприклад, WPML чи Polylang) можна створювати багатомовні магазини та охоплювати більшу аудиторію клієнтів.

– Інтеграція через API – WooCommerce надає доступ до REST API, що дозволяє інтегрувати магазин з іншими сервісами та автоматизувати процеси.

– Простота використання – інтуїтивний інтерфейс та детальна документація дозволяють швидко освоїти роботу з плагіном, навіть без попереднього досвіду роботи з WordPress.

Недоліки WooCommerce

– Відсутність офіційної підтримки – у разі проблем доведеться шукати відповіді на форумах або в розділі faq на офіційному сайті. Хоча спільнота користувачів досить активна, вирішення деяких питань може зайняти час.

– Висока вартість преміум-додатків – базова функціональність безкоштовна, але платні теми та розширення часто коштують дорого, що може бути проблемою для малого бізнесу, особливо під час його запуску.

– Орієнтація на західний ринок – багато шаблонів і модулів розроблені для західних користувачів, тому їх адаптація для української аудиторії може вимагати додаткових ресурсів.

– Залежність від WordPress – WooCommerce працює лише у зв'язку з WordPress, тому якщо немає досвіду роботи з цією CMS, можуть виникнути складнощі.

WooCommerce пропонує потужний набір інструментів, який дозволяє створювати повноцінні інтернет-магазини без глибоких технічних знань. Однією з ключових переваг є гнучкість у дизайні – система підтримує безліч тем, від мінімалістичних до складних преміум-шаблонів. Хоча стандартна тема Storefront дозволяє швидко запустити магазин, для професійного вигляду варто розглянути спеціалізовані комерційні шаблони. Вбудований редактор дозволяє легко змінювати оформлення, а при необхідності можна глибше налаштувати дизайн через CSS.

Важливою складовою WooCommerce є спеціалізовані застосунки, які розширюють базову функціональність. На відміну від звичайних плагінів WordPress, ці розширення орієнтовані саме на електронну комерцію – від систем знижок до складського обліку. Вони дозволяють адаптувати магазин під конкретні потреби бізнесу без необхідності розробки індивідуальних рішень.

Окремо варто відзначити платіжні системи – WooCommerce підтримує всі популярні способи оплати. Хоча базово доступні лише PayPal і банківські перекази, через додаткові плагіни легко підключити будь-які платіжні шлюзи, включаючи Stripe для прийому карткових платежів. Це робить платформу універсальним рішенням для роботи з клієнтами з різних країн.

Останнім часом WooCommerce активно розвиває інструменти аналітики, що дозволяє відстежувати ефективність магазину без сторонніх сервісів. Вбудовані звіти дають змогу аналізувати продажі, перегляди товарів і конверсію, що особливо важливо для оптимізації бізнес-процесів. Разом із можливістю масштабування це робить WooCommerce гарним вибором як для невеликих магазинів, так і для великих торгових майданчиків.

Shopware

Shopware [6] – це сучасна платформа для електронної комерції, яка дозволяє бізнесам легко створювати та керувати інтернет-магазинами. Спочатку представлена в Німеччині, Shopware набула глобальної популярності завдяки своїм комплексним функціям, зручному інтерфейсу та потужному бекенду.

Одна з ключових причин успіху Shopware – її гнучкість. Побудована на модульній архітектурі, платформа дозволяє бізнесам обирати необхідні функції, що спрощує масштабування. Крім того, Shopware інтегрує різні бізнес-процеси, підтримуючи як B2C, так і B2B-операції. Функціональність Shopware для B2B включає індивідуальні ціни для клієнтів, персоналізовані каталоги продуктів та спеціальні умови покупок, що робить її ідеальним вибором для компаній з різноманітною клієнтською базою.

Ще один чинник популярності Shopware – це розвинута екосистема плагінів і розширень. Магазин додатків Shopware пропонує безліч рішень, розроблених спільнотою та сторонніми розробниками, які розширюють функціональність платформи – від маркетингових інструментів до аналітики. Ці плагіни дозволяють налаштувати магазин під конкретні потреби.

На рисунку 1.4 наведено приклад адмін-частини платформи Shopware

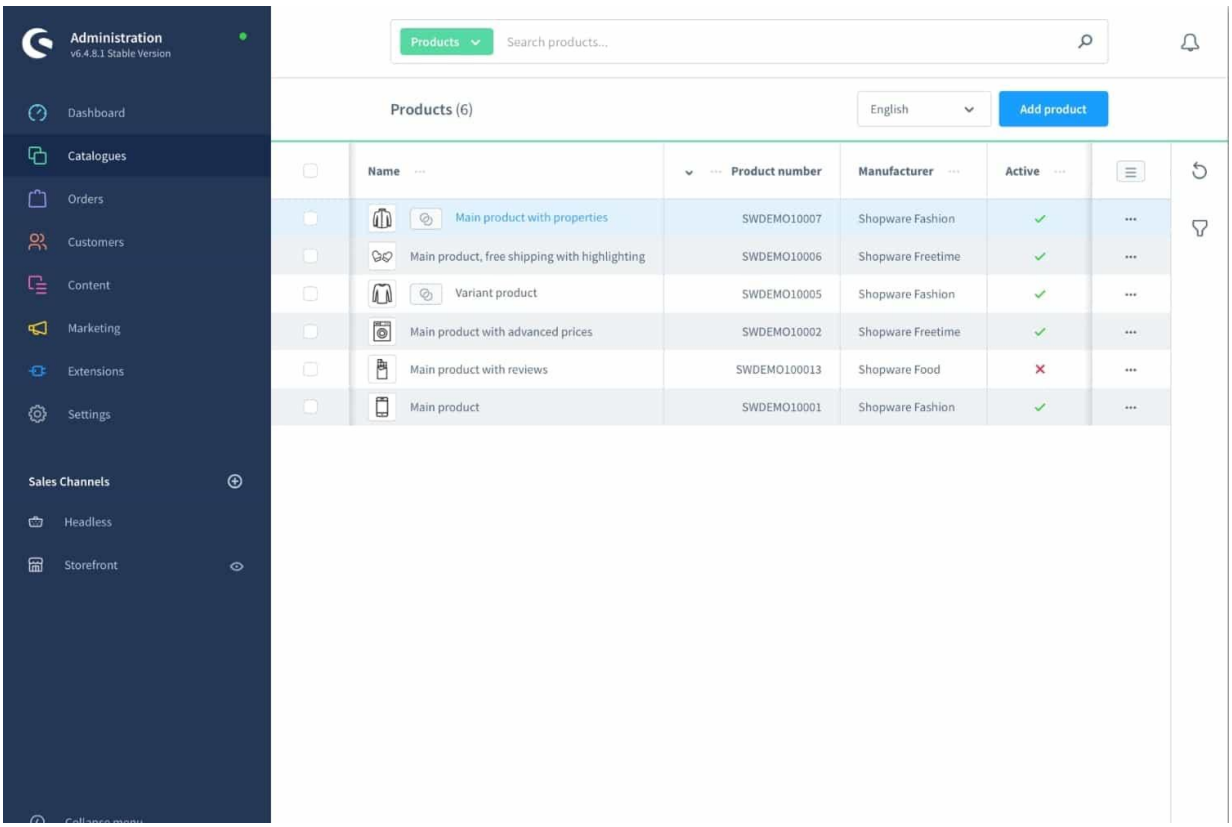


Рисунок 1.4 – Адмін панель застосунку на Shopware

Shopware побудована на сучасних технологіях, які забезпечують її потужні можливості. Основу платформи становлять PHP та фреймворк Symfony, що забезпечує масштабованість і легкість розширення. Для створення інтерактивного інтерфейсу використовуються JavaScript-бібліотеки Vue.js та ExtJS, які забезпечують зручний користувацький досвід як для покупців, так і для адміністраторів. Також Shopware інтегрує Elasticsearch для швидкого та точного пошуку товарів.

Однією з ключових переваг Shopware є її розширюваність за рахунок застосунків (apps) і плагінів, які дозволяють гнучко адаптувати платформу під конкретні бізнес-потреби. Плагіни є спеціальними модулями, що інтегруються безпосередньо в систему, додаючи новий функціонал або модифікуючи існуючий. Вони відрізняються простотою встановлення та великим вибором рішень у магазині додатків Shopware – від платіжних систем до маркетингових інструментів. Однак робота з плагінами може вимагати технічної підготовки, особливо при

конфліктах версій або несумісності з іншими розширеннями після оновлень платформи.

Застосунки, на відміну від плагінів, є самостійними програмами, які часто виконують складніші завдання, такі як інтеграція з ERP-системами чи CRM. Вони пропонують більш широкі можливості для автоматизації бізнес-процесів, але їхня реалізація може бути менш прозорою для користувача. Наприклад, деякі застосункки потребують окремої підписки або налаштування API, що ускладнює процес їх підключення порівняно зі звичайними плагінами.

Таким чином, вибір між плагінами та застосунками залежить від конкретних завдань: перші краще підходять для швидкого розширення функціоналу магазину, тоді як другі – для комплексної автоматизації бізнесу з підключенням зовнішніх сервісів. Оптимальним рішенням часто є комбінація обох варіантів з урахуванням їхніх сильних сторін.

Індивідуальний магазин на Shopware пропонує низку переваг:

– Покращений користувацький досвід – унікальний дизайн під бренд підвищує конверсію.

– Оптимізація бізнес-процесів – автоматизація управління запасами, замовленнями тощо.

– Масштабованість – модульна архітектура дозволяє адаптувати магазин до зростання.

– Конкурентні переваги – персоналізація допомагає виділитися на ринку.

– B2B-функції – підтримка складних схем роботи з оптовими клієнтами.

Серед недоліків Shopware варто врахувати:

– Складність освоєння – новачкам може знадобитися час на опанування платформи.

– Витрати – безкоштовна базова версія потребує додаткових інвестицій у плагіни, хостинг тощо.

– Сумісність – оновлення або комбінації плагінів іноді викликають конфлікти.

Ключові переваги Shopware при порівнянні з іншими платформами:

– Гнучкість – модульна структура для точного налаштування.

– B2B-функціонал – спеціальні інструменти для роботи з бізнес-клієнтами.

– Великий набір плагінів – майже безліч готових рішень.

– Продуктивність – оптимізація для великих магазинів.

Shopware посідає високу позицію серед платформ для електронної комерції, пропонуючи бізнесам інструменти для успіху в конкурентному середовищі. Її архітектура, спеціалізовані B2B-рішення та можливості масштабування роблять її привабливим вибором. Однак важливо враховувати потенційні складнощі, такі як витрати або технічні аспекти.

1.3 Обґрунтування доцільності розробки програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Аналіз існуючих платформ для електронної комерції (Shopify, WooCommerce, Shopware) виявляє ряд суттєвих обмежень, які роблять їх недостатньо ефективними для спеціалізованого інтернет-магазину екологічних продуктів. Ці обмеження підкреслюють необхідність розробки оригінального веб-рішення, спеціально адаптованого під специфіку ринку екопродуктів. Головним недоліком готових рішень є відсутність спеціалізованої функціональності для роботи з екологічними характеристиками товарів, зокрема жодна з аналізованих платформ не пропонує вбудованих інструментів для рейтингування товарів за екологічними критеріями. Таким чином, аналіз існуючих рішень демонструє, що розробка програмного забезпечення інтернет-магазину екологічних продуктів «BioGild» є доцільною.

1.4 Постановка задачі на розробку програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Аналіз предметної галузі дозволяє поставити задачу на розробку програмного забезпечення інтернет-магазину екологічних продуктів «BioGild», який не матиме

вищенаведених недоліків. Розроблений веб-застосунок дозволить організувати онлайн торгівлю екологічними продуктами більш ефективно.

Застосунок матиме наступні функціональні характеристики:

– Реєстрацію, авторизацію, особистий кабінет з історією замовлень та налаштуваннями профілю, а також реалізувати розподіл ролей (покупці та адміністратори).

– Додавати та редагувати товари через адмін-панель, фільтрувати їх за категоріями, ціною, еко-рейтингом чи сертифікатами, шукати за назвою, а також відображати детальні картки товарів з фотографіями та відгуками.

– Додавати товари до кошика, бачити підсумкову вартість з урахуванням знижок, обирати спосіб оплати.

– Відстежувати статус замовлення (у обробці, доставка, отримано), надсилати сповіщення на email, а також обробляти повернення чи скасування замовлень.

– Залишати оцінки (1–5 зірок) та коментарі, які проходять модерацію, а система формує рейтинг «Найкращі екотовари» на основі відгуків.

– Підтримувати промокоди, а також email-розсилки про новинки та спецпропозиції.

– Реалізувати збір статистики щодо продажів (популярні товари, сезонність).

– Створювати звіти по продажам.

Вимоги для вхідних та вихідних даних

Вхідними даними для товару назва, опис, категорія, ціна, ціна зі знижкою, екологічний рейтинг товару, походження, сертифікат, оцінка товару (1-5 зірок, залишається користувачами), період зберігання, одиниці виміру, кількість одиниць та зображення у форматі .png, .jpeg (jpg), .webp. Текстова інформація (назва, опис, ціна, екологічний рейтинг товару, сертифікат) заповнюватиметься з клавіатури. Ціна є додатнім дійсним числом більше одиниці. Оцінка товару є натуральним числом від 1 до 5, оцінка ставиться при натисканні на бажану кількість зірок на

картці товару. Оцінити товар може лише користувач, який придбав його. Зображення додаються з комп’ютеру серед доступних у пам’яті.

Вихідними даними про товар є запис у базі даних системи. Користувачі та адміністратори веб-застосунку бачитимуть товар на сторінках веб-застосунку у вигляді картки товару. Адміністратори в адмін-частині матимуть можливість створювати нові товари та вносити зміни до існуючих товару. Звичайні користувачі можуть переглядати товари лише в частині веб-застосунку, що призначена для користувачів.

Система дозволяє створювати облікові записи користувачів. Вхідними даними для користувача є ПІБ, email, телефон, роль (клієнт / адмін). Вихідними даними про користувача є відповідний запис в бд системи

Користувачі можуть переглядати товари та шукати їх за допомогою відповідного поля на сторінці веб-застосунку, поле пошуку виконує текстовий пошук по назві товару. Користувачі сайту можуть сортувати товари за обраним критерієм (ціною, еко-рейтингом, оцінкою, сертифікатом)

Для полегшення оформлення замовлень веб-застосунок надає можливість додавати товари до кошика, в якому вони будуть зберігатись до тих пір, поки користувач не видалить їх, або не оформить замовлення з цими товарами. Кошик користувача містить перелік товарів з загальною вартістю.

Вхідними даними для формування замовлення є дані про товари, що знаходяться у кошику на момент оформлення замовлення та даних про користувача, який оформлює замовлення, промокод (за наявності). При оформленні замовлення товари видаляються з кошику. Після оформлення замовлення зареєстровані користувачі можуть перейти в особистий кабінет та переглянути перелік своїх замовлень.

Вихідними даними про замовлення є запис у базі даних. При перегляді замовлення користувач бачить перелік товарів, які до нього входять, вартість замовлення, його статус (в обробці, доставка, отримано), знижку за застосованим промокодом (за наявності). Адміністратори бачать перелік усіх замовлень інтернет-

магазину в його адмін-частині в такому ж вигляді й можуть редагувати інформацію про замовлення. За потреби користувач може відмінити замовлення.

На сторінці товару користувачі можуть залишати відгуки про товари. В адмін-частині веб застосунку адміністратори можуть переглядати відгуки користувачів. Вхідними даними про відгук є дані про користувача, що залишив відгук, заголовок відгуку, текст відгуку, оцінка товару. Вихідними даними про відгук є запис у базі даних системи.

Розроблюваний веб застосунок дозволятиме застосувати промокоди. Вхідними даними для створення промокодів буде: код, тип промокоду (сума або відсоток), числове значення знижки (дійсне додатне число), період дії, мінімальна вартість замовлення для застосування, максимальне значення знижки, кількість застосувань. Вихідними даними про промокод є запис у базі даних системи. Перелік доступних промокодів користувач може побачити на сторінці профілю користувача. Адміністратори можуть переглянути всі промокоди на відповідній сторінці адмін-частини веб-застосунку.

Для отримання структурованої інформації щодо роботи інтернет-магазину адміністратор може створювати звіти по продажам за період часу. Групування продажів відбуватиметься по дням. Звіт буде містити наступну інформацію:

1) Період звітності.

2) Дохід за цей час.

3) Назву, ціну, кількість та вартість товарів

Вимоги до складу та параметрів технічних засобів

Серверна частина та її вимоги:

– ЦП – 4 ядра, 2.0 ГГц.

– Оперативна пам’ять – 4 ГБ.

– Дискова пам’ять – 128 ГБ.

– Мережа – 1 Гбіт Ethernet

Пристрій користувача має задовольняти наступні мінімальні вимоги:

– ЦП – 2 ядра, 1.5–2.0 ГГц.

– Оперативна пам’ять – 2-4 ГБ.

– Дискова пам’ять – 128 ГБ.

– Доступ до мережі інтернет

Вимоги до інформаційної та програмної сумісності

Розробку ПЗ слід використовувати з сучасним технологічним стеком: останню версію Visual Studio Code як середовище розробки, мову програмування PHP (версія 8.2) у поєднанні з фреймворком Laravel 12 для створення надійного бекенду, реляційну базу даних на основі MySQL (8.0) для ефективного зберігання та управління даними, а також веб-сервер Apache (Apache HTTP Server 2.4.57) для стабільної роботи готового рішення. Така комбінація інструментів забезпечує оптимальний баланс між продуктивністю, стабільністю та сучасними підходами до веб-розробки.

2 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BIOGILD»

2.1 Аналіз вимог до програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

2.1.1 Розробка моделі варіантів використання програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Діаграма варіантів використання – це інструмент для візуалізації системи та її користувачів. Вона показує взаємодії між компонентами системи у вигляді графічного зображення. Ця діаграма фіксує події в системі та їх взаємозв’язки, але не розкриває деталі реалізації цих подій.

Варіант використання – це елемент системного аналізу, що служить для визначення, уточнення та структурування вимог до системи. Під «системою» тут розуміється будь-який розроблюваний або експлуатований об'єкт, наприклад, веб-сайт для онлайн-продажів. Такі діаграми є частиною стандарту UML (Unified Modeling Language), який використовується для моделювання об’єктів і систем. У порівнянні з іншими діаграмами, як-от блок-схемами, варіанти використання мають ряд переваг [7].

Діаграми варіантів використанням застосовуються для:

– Відображення цілей системи та користувачів.

– Визначення контексту, у якому слід розглядати систему.

– Вказівка вимог до системи.

– Надання моделі для послідовності подій під час взаємодії з користувачами.

– Зовнішнє представлення системи.

– Відображення зовнішніх та внутрішніх впливів на систему.

Наприклад, у середовищі продажу товарів варіанти використання можуть включати замовлення товарів, оновлення каталогу, обробку платежів та взаємодію з клієнтами.

Ця діаграма має наступні елементи:

– Межа системи – визначає область інтересу системи відносно навколишнього світу.

– Актори – зазвичай це користувачі, які взаємодіють із системою відповідно до їхніх ролей.

– Варіант використання (прецедент) – конкретні дії, які виконують актори всередині системи або навколо неї.

– Зв’язки елементів діаграми.

Діаграма варіантів використання схожа на блок-схему, де інтуїтивно зрозумілі символи відображають елементи системи. Діаграма прецедентів показано на рисунку 2.1.

Рисунок 2.1 – Use-case діаграма програмного забезпечення інтернет-магазину «BioGild»

Для створеної use-case діаграми виконаємо детальну специфікацію сценаріїв.

2.1.2 Розробка специфікацій прецедентів програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Специфікація варіанта використання «Створити обліковий запис»

Найменування варіанта використання: «Створити обліковий запис».

Зв'язок з іншими варіантами використання: відсутній.

Передумова: запуск програмного забезпечення.

Короткий опис: сценарій використовується для додавання нового користувача до системи шляхом реєстрації.

Потік подій:

1) Користувач ініціює процедуру реєстрації.

2) Система відкриває форму для введення реєстраційних даних.

3) Користувач заповнює усі обов'язкові поля та підтверджує створення акаунта.

4) Система зберігає інформацію про нового користувача в базі даних, автоматично автентифікує його та переадресовує на основний інтерфейс.

Альтернативний потік: При спробі реєстрації з некоректними або неповними даними система блокує створення акаунта та виводить повідомлення про помилку.

Спеціальні вимоги: відсутні.

Постумови: в системі з'являється новий зареєстрований користувач.

Додаткові зауваження: немає.

Специфікація варіанта використання «Залогінитись»

Найменування варіанта використання: «Залогінитись».

Зв’язок з іншими варіантами використання: включається варіантом використання «Переглянути історію замовлень».

Передумова: запуск програмного забезпечення.

Короткий опис: сценарій використовується для ідентифікації користувача та надання доступу до системи.

Потік подій:

1) Користувач обирає функцію входу в систему.

2) Система відображає вікно для введення облікових даних.

3) Користувач вказує свій логін (телефон або email) та пароль, після чого підтверджує вхід.

4) Система перевіряє надані дані, підтверджує особу та надає доступ, перенаправляючи на основну сторінку інтерфейсу.

Альтернативний потік: При введенні невірних облікових даних система відмовляє у доступі та інформує про невдалу спробу автентифікації.

Спеціальні вимоги: не передбачено.

Постумови: відсутні.

Додаткові зауваження: немає.

Специфікація варіанта використання «Ввести дані користувача»

Найменування варіанта використання: «Ввести дані користувача».

Зв'ззок з іншими варіантами використання: Включає варіант використання «Залогінитись».

Передумова: запуск програмного забезпечення.

Короткий опис: сценарій дозволяє клієнтам оновлювати свої персональні дані у системі.

Потік подій:

1) Користувач проходить процедуру входу в систему.

2) Система підтверджує автентичність користувача.

3) Користувач переходить до розділу редагування особистих даних.

4) Система відкриває інтерфейс для внесення змін до персональної інформації.

5) Користувач вносить необхідні корективи та підтверджує оновлення.

6) Система зберігає оновлені дані у базі даних клієнтів.

Альтернативний потік: При спробі зберегти некоректні або неповні дані система блокує оновлення та надає інструкції щодо виправлення помилок.

Спеціальні вимоги: не передбачено.

Постумови: оновлення інформації про клієнта в базі даних системи.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Переглянути продукти»

Найменування варіанта використання: «Переглянути продукти».

Зв'язок з іншими варіантами використання: відсутній.

Передумова: запуск програмного забезпечення, присутність продуктів.

Короткий опис: сценарій дозволяє користувачам переглянути продукти у системі.

Потік подій:

1) Користувач ініціює перегляд продуктів.

2) Система відображає перелік доступних продуктів.

Альтернативний потік: У разі відсутності товарів система не відображає жодних елементів.

Спеціальні вимоги: не передбачено.

Постумови: відсутні

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Шукати продукти»

Найменування варіанта використання: «Шукати продукти ».

Зв ' язок з іншими варіантами використання: відсутній

Передумова: запуск програмного забезпечення, присутність продуктів.

Короткий опис: сценарій призначений для пошуку продуктів за текстовим запитом.

Потік подій:

1) Користувач вводить назву продукту у спеціальне поле пошуку.
2) Система аналізує запит та відображає відповідні продукти.

Альтернативний потік: Якщо продукти відсутні у системі, пошук не виконується.

Якщо не знайдено продуктів за вказаним критерієм, система показує відповідне повідомлення.

Спеціальні вимоги: не передбачено.

Постумови: відсутні.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Помістити продукт у кошик»

Найменування варіанта використання: «Помістити продукт у кошик».

Зв'язок з іншими варіантами використання: включається варіантом використання «Купити продукт».

Передумова: запуск програмного забезпечення, наявність продуктів.

Короткий опис: сценарій призначений для додавання продуктів до кошика для подальшого оформлення замовлення.

Потік подій:

1) Користувач обирає продукт та ініціює його додавання до кошика.
2) Система додає обраний продукт до кошика користувача.

Спеціальні вимоги: не передбачено.

Постумови: відсутні.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Купити продукт»

Найменування варіанта використання: «Купити продукт».

Зв'язок з іншими варіантами використання: включає в себе варіант використання «Помістити продукт у кошик».

Передумова: запуск програмного забезпечення, наявність продуктів.

Короткий опис: варіант використання призначений для процедури купівлі продуктів.

Потік подій:

1) Клієнт знаходить потрібний йому продукт.

2) Клієнт додає продукт до кошика.

3) Система додає обраний продукт до кошика.

4) Клієнт ініціює покупку продукту.

5) Система відображає форму для заповнення персональних даних.

6) Клієнт заповнює особисту інформацію (для авторизованих користувачів використовуються попередньо збережені дані).

7) Клієнт підтверджує покупку продукту.

Спеціальні вимоги: не передбачено.

Постумови: створення запису про покупку в базі даних системи.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Управляти продуктами»

Найменування варіанта використання: «Управляти продуктами ».

Зв'язок з іншими варіантами використання: включає варіанти використання «Додати продукт», «Редагувати продукт», «Видалити продукт».

Передумова: запуск програмного забезпечення, авторизація в адмін панель.

Короткий опис: варіант використання призначений для керування продуктами адміністратором.

Потік подій:

1) Користувач авторизується як адмін.

2) Система підтверджує авторизацію.

3) Клієнт обирає необхідну операцію з управління продуктами.

4) Система надає відповідний інтерфейс для виконання обраної операції.

Спеціальні вимоги: не передбачено.

Постумови: відсутні.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Створити продукт»

Найменування варіанта використання: «Створити продукт».

Зв’язок з іншими варіантами використання: включається варіантом використання «Управляти продуктами».

Передумова: запуск програмного забезпечення, авторизація в адмін панель.

Короткий опис: варіант використання призначений для створення продуктів.

Потік подій:

1) Клієнт авторизується як адмін.

2) Система підтверджує авторизацію.

3) Клієнт ініціює створення продукту.

4) Система відображає сторінку створення продуктів.

5) Клієнт вказує назву, опис, ціну та зображення продукту та ініціює збереження.

6) Система створює запис про новий продукт у базі даних.

Альтернативний потік: Створення продукту неможливе через невалідні дані. У цьому випадку система відображатиме повідомлення про помилку з вказівкою полів, які потребують виправлення.

Спеціальні вимоги: не передбачено.

Постумови: створення запису про продукт у базі даних системи.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Редагувати продукт»

Найменування варіанта використання: «Редагувати продукт»

Зв'язок з іншими варіантами використання: включається варіантом використання «Управляти продуктами»

Передумова: запуск програмного забезпечення, авторизація в адмін панель,

наявність продуктів

Короткий опис: варіант використання призначений для внесення змін до інформації про певний продукт

Потік подій:

1) Адмін авторизується в системі.
2) Система підтверджує авторизацію.
3) Адмін ініціює перегляд усіх продуктів в адмін-панелі.
4) Система відображає перелік усіх продуктів з бази даних.
5) Адмін обирає потрібний продукт та ініціює його редагування.
6) Система зберігає внесені зміни до обраного продукту.

Альтернативний потік: Редагування продукту неможливе через невалідні дані.

Система відображає повідомлення про помилку з вказівкою полів, які потребують виправлення.

Спеціальні вимоги: не передбачено.

Постумови: оновлення інформації про продукт у базі даних системи .

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Видалити продукт»

Найменування варіанта використання: «Видалити продукт»

Зв'язок з іншими варіантами використання: відсутній

Передумова: запуск програмного забезпечення, авторизація в адмін панель, наявність продуктів.

Короткий опис: варіант використання призначений для видалення продуктів з системи.

Потік подій:

1) Адмін авторизується в системі.
2) Система підтверджує авторизацію.
3) Адмін ініціює перегляд усіх продуктів в адмін-панелі.

4) Система відображає перелік усіх продуктів.

5) Адмін обирає потрібний продукт та ініціює його видалення.

6) Система видаляє обраний продукт та всі пов'язані з ним дані.

Спеціальні вимоги: не передбачено.

Постумови: повне видалення запису про продукт та всіх пов'язаних даних з бази даних

Додаткові зауваження: відсутні

Специфікація варіанта використання «Переглянути історію замовлень»

Найменування варіанта використання: «Переглянути історію замовлень»

Зв'язок з іншими варіантами використання: відсутній

Передумова: запуск програмного забезпечення, авторизація, наявність замовлень.

Короткий опис: варіант призначений для перегляду замовлень у системі

Потік подій:

1) Користувач авторизується в системі.

2) Система підтверджує авторизацію.

3) Користувач ініціює перегляд замовлень.

4) Система відображає список замовлень.

Спеціальні вимоги: не передбачено.

Постумови: оновлення статусу замовлення у базі даних

Додаткові зауваження: відсутні

Специфікація варіанта використання «Опрацювати замовлення»

Найменування варіанта використання: «Опрацювати замовлення»

Зв'язок з іншими варіантами використання: відсутній

Передумова: запуск програмного забезпечення, авторизація в адмін панель, наявність замовлень.

Короткий опис: варіант призначений для обробки замовлень клієнтів

Потік подій:

1) Адмін авторизується в системі

2) Система підтверджує авторизацію

3) Адмін ініціює перегляд нових замовлень в адмін-панелі

4) Система відображає список доступних нових замовлень

5) Адмін обирає конкретне замовлення, аналізує його та підтверджує обробку

6) Система оновлює статус замовлення на "Опрацьовано"

Спеціальні вимоги: не передбачено

Постумови: оновлення статусу замовлення у базі даних

Додаткові зауваження: відсутні

Специфікація варіанта використання «Застосувати промокод»

Найменування варіанта використання: «Застосувати промокод».

Зв'язок з іншими варіантами використання: розширює варіант використання «Купити продукт».

Передумова: запуск програмного забезпечення, наявність продуктів, наявність промокодів.

Короткий опис: варіант використання призначений для застосування промокоду клієнтом при оформленні замовлення.

Потік подій:

1) Клієнт ініціює оформлення замовлення.

2) Система показує форму оформлення замовлення.

3) Клієнт застосовую до замовлення промокод.

4) Система враховує промокод при розрахунку вартості замовлення.

5) Клієнт підтверджує замовлення.

6) Система створює замовлення з урахуванням промокоду.

Альтернативний потік: Застосування промокоду неможливе через відсутність продуктів.

Спеціальні вимоги: не передбачено.

Постумови: створення запису про замовлення у базі даних системи з урахуванням промокоду.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Створити промокод»

Найменування варіанта використання: «Створити промокод».

Зв’язок з іншими варіантами використання: відсутній.

Передумова: запуск програмного забезпечення, авторизація в адмін панель.

Короткий опис: варіант використання призначений для створення промокодів.

Потік подій:

1) Клієнт авторизується як адмін.

2) Система підтверджує авторизацію.

3) Клієнт ініціює створення промокоду.

4) Система відображає сторінку створення промокодів.

5) Клієнт вказує назву, опис, тип, значення, період дії та ініціює збереження.

6) Система створює запис про новий промокод у базі даних.

Альтернативний потік: Створення промокоду неможливе через невалідні дані. У цьому випадку система відображатиме повідомлення про помилку з вказівкою полів, які потребують виправлення.

Спеціальні вимоги: не передбачено.

Постумови: створення запису про промокод у базі даних системи.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Видалити промокод»

Найменування варіанта використання: «Видалити промокод»

Зв'язок з іншими варіантами використання: відсутній

Передумова: запуск програмного забезпечення, авторизація в адмін панель, наявність промокодів.

Короткий опис: варіант використання призначений для видалення промокодів з системи.

Потік подій:

1) Адмін авторизується в системі.

2) Система підтверджує авторизацію.

3) Адмін ініціює перегляд усіх промокодів в адмін-панелі.

4) Система відображає перелік усіх промокодів.

5) Адмін обирає потрібний промокод та ініціює його видалення.

6) Система видаляє обраний промокод.

Спеціальні вимоги: не передбачено

Постумови: повне видалення запису про промокод.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Переглянути промокоди»

Найменування варіанта використання: «Переглянути промокоди».

Зв'язок з іншими варіантами використання: відсутній.

Передумова: запуск програмного забезпечення, присутність промкодів.

Короткий опис: сценарій дозволяє користувачам переглянути промокоди.

Потік подій:

1) Клієнт авторизується.

2) Система підтверджує авторизацію.

3) Клієнт відкриває сторінку перегляду промокодів.

4) Система відображає перелік доступних промокодів.

Альтернативний потік: У разі відсутності промокодів система не відображає жодних елементів.

Спеціальні вимоги: не передбачено.

Постумови: відсутні

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Створити відгук»

Найменування варіанта використання: «Створити відгук».

Зв'язок з іншими варіантами використання: відсутній.

Передумова: запуск програмного забезпечення, авторизація.

Короткий опис: сценарій дозволяє клієнтам залишити відгук про придбаний продукт.

Потік подій:

1) Клієнт авторизується.

2) Система підтверджує авторизацію.

3) Клієнт відкриває сторінку продукту.

4) Система відображає сторінку продукту.

5) Клієнт заповнює форму відгуку, вказуючи оцінку, заголовок, текст відгуку й ініціює його створення.

6) Система створює запис про новий відгук у базі даних системи.

Альтернативний потік: У разі відсутності продуктів створити відгук неможливо.

Спеціальні вимоги: не передбачено.

Постумови: відсутні

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Опрацювати відгук»

Найменування варіанта використання: «Опрацювати відгук».

Зв'язок з іншими варіантами використання: розширюється варіантами використання «Видалити відгук», «Підтвердити відгук».

Передумова: запуск програмного забезпечення, авторизація в адмін панель.

Короткий опис: сценарій дозволяє адмінам опрацювати відгук – залишити його або видалити.

Потік подій:

1) Адмін авторизується в системі.

2) Система підтверджує авторизацію.

3) Адмін ініціює перегляд усіх відгуків.

4) Система відображає перелік усіх відгуків.

5) Адмін обирає потрібний відгук та ініціює його опрацювання.

6) Система виконує обрану операцію.

Альтернативний потік: У разі відсутності відгуків їх опрацювання неможливе.

Спеціальні вимоги: не передбачено.

Постумови: відсутні

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Видалити відгук»

Найменування варіанта використання: «Видалити відгук»

Зв'язок з іншими варіантами використання: розширює варіант використанням «Опрацювати відгук»

Передумова: запуск програмного забезпечення, авторизація в адмін панель, наявність відгуків.

Короткий опис: варіант використання призначений для видалення відгуків з системи.

Потік подій:

1) Адмін авторизується в системі.

2) Система підтверджує авторизацію.

3) Адмін ініціює перегляд усіх відгуків в адмін-панелі.

4) Система відображає перелік усіх відгуків.

5) Адмін обирає потрібний відгук та ініціює його видалення.

6) Система видаляє обраний відгук.

Спеціальні вимоги: не передбачено

Постумови: повне видалення запису про відгук.

Додаткові зауваження: відсутні.

Специфікація варіанта використання «Підтвердити відгук»

Найменування варіанта використання: «Підтвердити відгук»

Зв'язок з іншими варіантами використання: розширює варіант використанням «Опрацювати відгук»

Передумова: запуск програмного забезпечення, авторизація в адмін панель, наявність відгуків.

Короткий опис: варіант використання призначений для підтвердження відгуків з клієнтів.

Потік подій:

1) Адмін авторизується в системі.

2) Система підтверджує авторизацію.

3) Адмін ініціює перегляд усіх відгуків в адмін-панелі.

4) Система відображає перелік усіх відгуків.

5) Адмін обирає потрібний відгук та ініціює його відтвердження.

6) Система підтверджує обраний відгук.

Спеціальні вимоги: не передбачено

Постумови: повне видалення запису про відгук.

Додаткові зауваження: відсутні.

2.2 Розробка архітектури програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Model-View-Controller (MVC) – це архітектурний шаблон проєктування, який розділяє логіку програми на три складові з різними завданнями. Ці шари взаємодіють між собою, щоб забезпечити узгоджену та послідовну роботу програми.

Підхід MVC охоплює всю програму – від інтерфейсу користувача (UI) до базової моделі даних. MVC, який виник у 1970-х роках для побудови графічних

інтерфейсів, сьогодні широко використовується у веб-розробці та об’єктно-орієнтованому програмуванні (ООП).

Переваги MVC для розробників

– Паралельна розробка – різні програмісти можуть працювати над окремими компонентами одночасно.

– Повторне використання коду – компоненти можна використовувати в інших частинах програми.

– Незалежне розгортання та підтримка – оновлення одного шару не впливає на інші.

– Масштабованість – спрощує розробку великих та складних програм.

Складові цього шаблону:

– Model (Модель) – реалізує зберігання даних та отримує дані з бази даних, може включати валідацію. Забезпечує цілісність і доступність даних.

– View (Представлення) – Відповідає за інтерфейс користувача (UI). Відображає дані (наприклад, кнопки, таблиці, форми тощо). Не містить бізнес-логіки – лише показує інформацію, отриману від Controller або Model.

– Controller (Контролер) – Керує взаємодією між View та Model. Обробляє запити користувачів (натискання кнопок, введення даних тощо). Оновлює Model та передає дані до View для відображення. Часто називають «мозком» програми, оскільки він координує всі процеси.

Приклад роботи цього шаблону можна представити як перелік наступних кроків:

1) Користувач взаємодіє з View (наприклад, натискає кнопку).

2) Controller обробляє цю дію та визначає, які зміни потрібно внести.

3) Controller оновлює Model (змінює дані в бд).

4) Model може передати оновлені дані назад до View через Controller.

5) View відображає нові дані користувачеві.

На рисунку 2.2 графічно продемонстровано вищезазначений процес

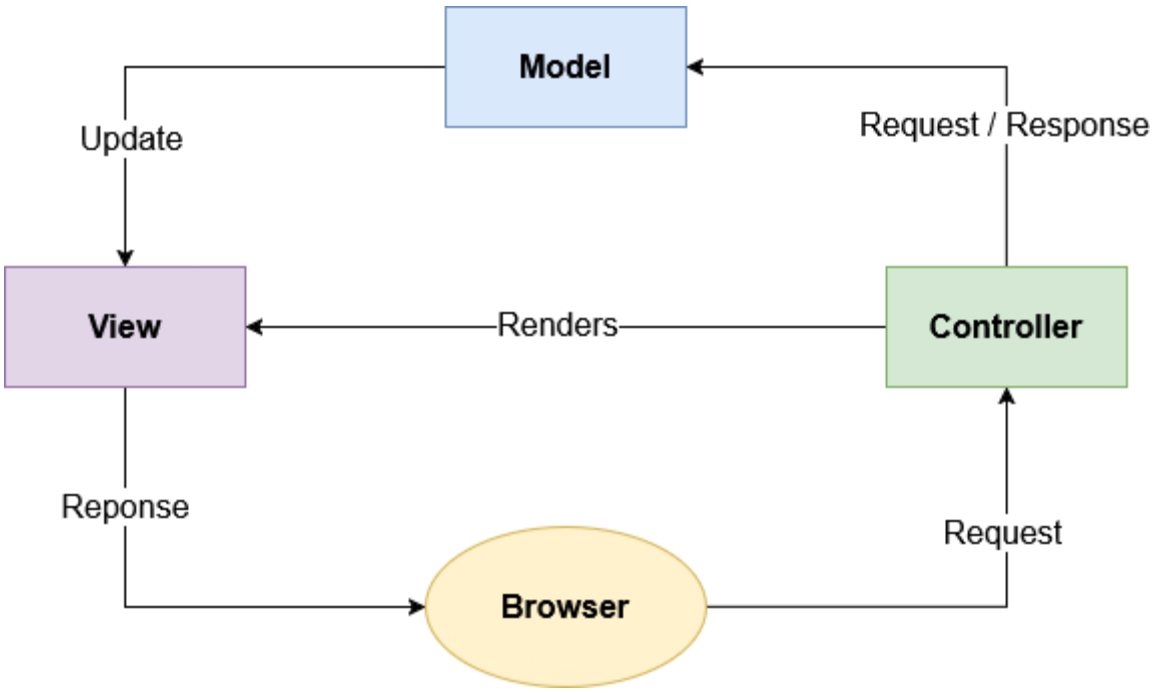


Рисунок 2.2 – Приклад застосування шаблону MVC

Таким чином в якості архітектурного патерну обрано MVC.

Розробка діаграми компонентів програмного забезпечення інтернет магазину «BioGild»

Діаграма компонентів використовується для відображення взаємозв'язків між різними компонентами системи. У контексті UML 2.0, термін «компонент» означає модуль класів, який представляє незалежні системи або підсистеми, здатні взаємодіяти з рештою системи. Існує цілий підхід до розробки, заснований на компонентах – Component-Based Development (CBD). У цьому підході діаграми компонентів допомагають планувальникам визначити різні компоненти, щоб забезпечити коректну роботу всієї системи [8].

У об'єктно-орієнтованому програмуванні (ООП) діаграма компонентів дозволяє старшому розробнику групувати класи за спільним призначенням, що спрощує високорівневий аналіз проєкту.

Хоча спочатку діаграми компонентів можуть здатися складними, вони дуже корисні для побудови системи. Вони допомагають:

– Візуалізувати фізичну структуру системи.

– Аналізувати компоненти та їх взаємозв'язки.

– Акцентувати увагу на поведінці сервісів через інтерфейси.

Діаграма компонентів у UML дає загальний огляд програмної системи. Розуміння того, які саме функції надає кожен компонент, робить розробку більш ефективною.

Основні елементи діаграми компонентів:

– Компоненти – модулі, які інкапсулюють певну функціональність.

– Інтерфейси – точки взаємодії між компонентами.

– Зв'язки – лінії, що показують залежності (assembly connectors) або делегування (delegation connectors).

Діаграми компонентів можуть описувати системи, реалізовані на будь-якій мові програмування або у будь-якому стилі.

Діаграму компонентів програмного забезпечення інтернет магазину «BioGild» показано на рисунку 2.3.

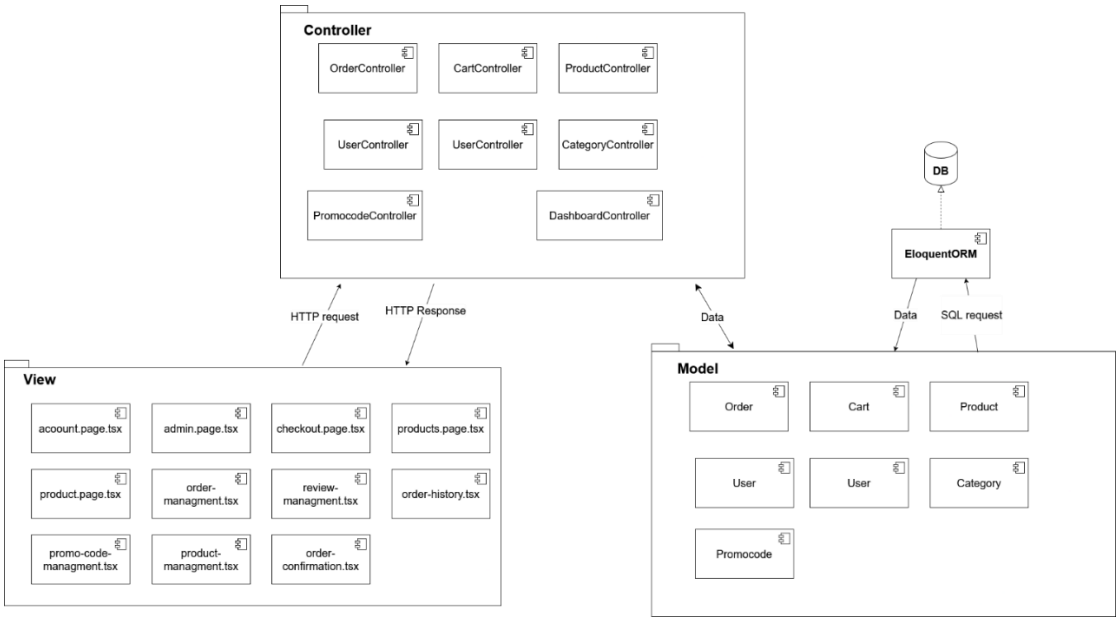


Рисунок 2.3 – Діаграма компонентів програмного забезпечення інтернет магазину «BioGild»

Діаграма компонентів демонструє основні сутності патерну MVC та їх взаємодію.

Розробка діаграми розгортання програмного забезпечення інтернет магазину «BioGild»

Діаграма розгортання UML належить до структурних діаграм і відображає фізичний розподіл компонентів програмного забезпечення на апаратній інфраструктурі. Вона показує, як артефакти (результати роботи ПЗ) розміщуються на фізичних пристроях системи [9]. Термін «артефакт» у цьому контексті має специфічне значення і відрізняється від його трактування в інших методологіях моделювання, зокрема BPMN.

Діаграми розгортання складаються з кількох стандартних UML-елементів:

– Вузли (Nodes) – тривимірні прямокутники, які представляють базові (софт/хард) компоненти системи.

– Зв'язки – лінії між вузлами, що показують взаємозв'язки.

– Артефакти – менші фігури всередині вузлів, які позначають розгорнуті програмні компоненти.

Діаграми розгортання мають кілька важливих застосувань. Їх можна використовувати для того, щоб:

– Вказати, які програмні елементи розгортаються на яких апаратних компонентах.

– Візуалізувати процеси виконання (runtime) для апаратного забезпечення.

– Показати топологію апаратної системи.

Діаграму розгортання програмного забезпечення інтернет магазину «BioGild» показано на рисунку 2.4.

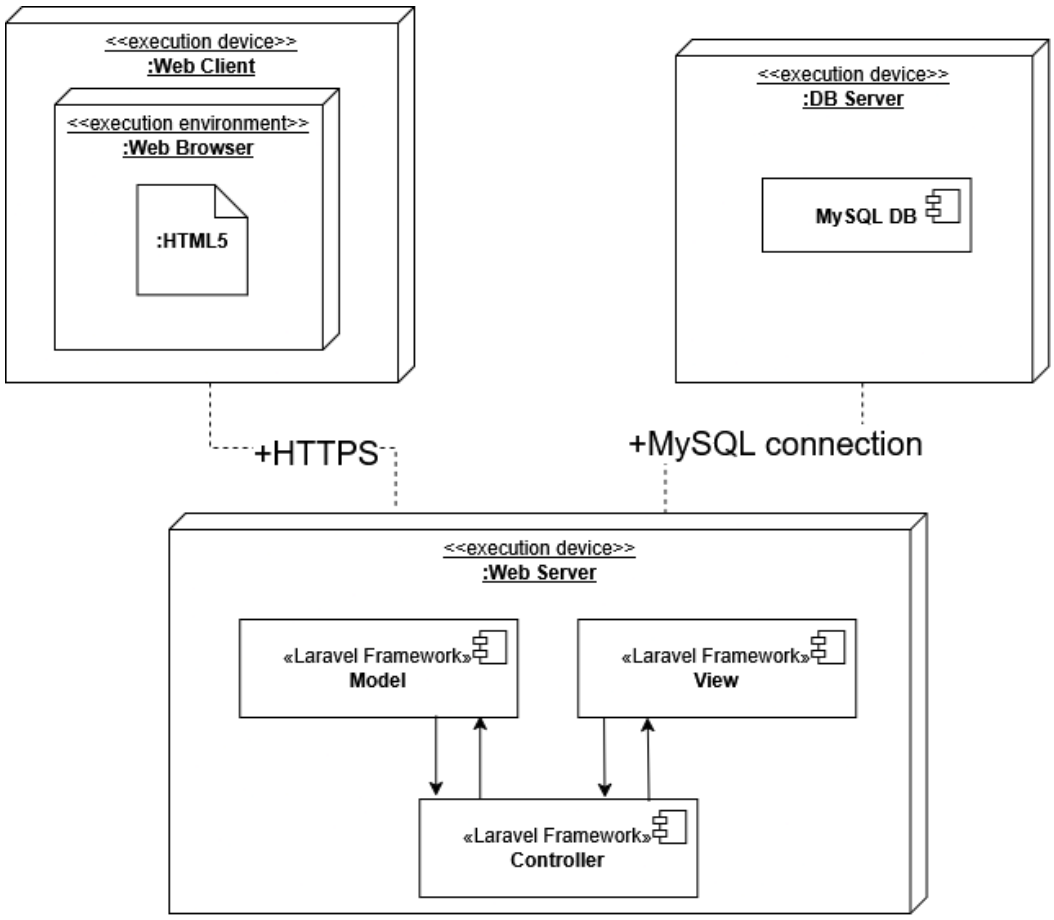


Рисунок 2.4 – Діаграма розгортання програмного забезпечення інтернет магазину «BioGild»

Розроблена діаграма розгортання показує фізичний розподіл компонентів програмного забезпечення інтернет магазину «BioGild».

2.3 Проєкт програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

2.3.1 Ескізний проєкт програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Моделювання сценаріїв варіантів використання програмного забезпечення інтернет магазину «BioGild» з використанням діаграм діяльності

У Unified Modeling Language (UML) діаграми поділяються на підмножини: структурні, діаграми взаємодії та діаграми поведінки. Діаграми активності, разом із діаграмами варіантів використання та діаграмами станів, належать до діаграм поведінки, оскільки вони описують процеси, які мають відбуватися в системі.

Для зацікавлених сторін важливо чітко та стисло комунікувати. Діаграми активності допомагають бізнесу та розробникам погодити спільне розуміння процесів і поведінки системи. Для їх побудови використовується набір спеціальних символів (початок, завершення, об'єднання тощо) [8].

Діаграми активності пропонують користувачам низку переваг. Їх варто використовувати, щоб:

– Продемонструвати логіку алгоритму.

– Описати кроки у варіанті використання UML.

– Проілюструвати бізнес-процес або робочий потік між користувачами та системою.

– Спростити та покращити процес за рахунок роз'яснення складних варіантів використання.

– Змоделювати елементи архітектури ПЗ (методи, функції, операції).

До основних компонентів належать:

– Дія (Action) – крок у активності, де користувачі або система виконують певне завдання. Позначається прямокутником із закругленими кутами.

– Вузол рішення (Decision Node) – умовне розгалуження у потоці, яке позначається ромбом. Має один вхід і два або більше виходів.

– Потоки управління (Control Flows) – сполучники, що показують послідовність кроків.

– Початковий вузол (Start Node) – позначає початок активності. Виглядає як чорне коло.

– Кінцевий вузол (End Node) – позначає завершення активності. Позначається чорним кільцем.

На рисунку 2.5 показано діаграму діяльності для прецеденту «Ввести дані користувача».

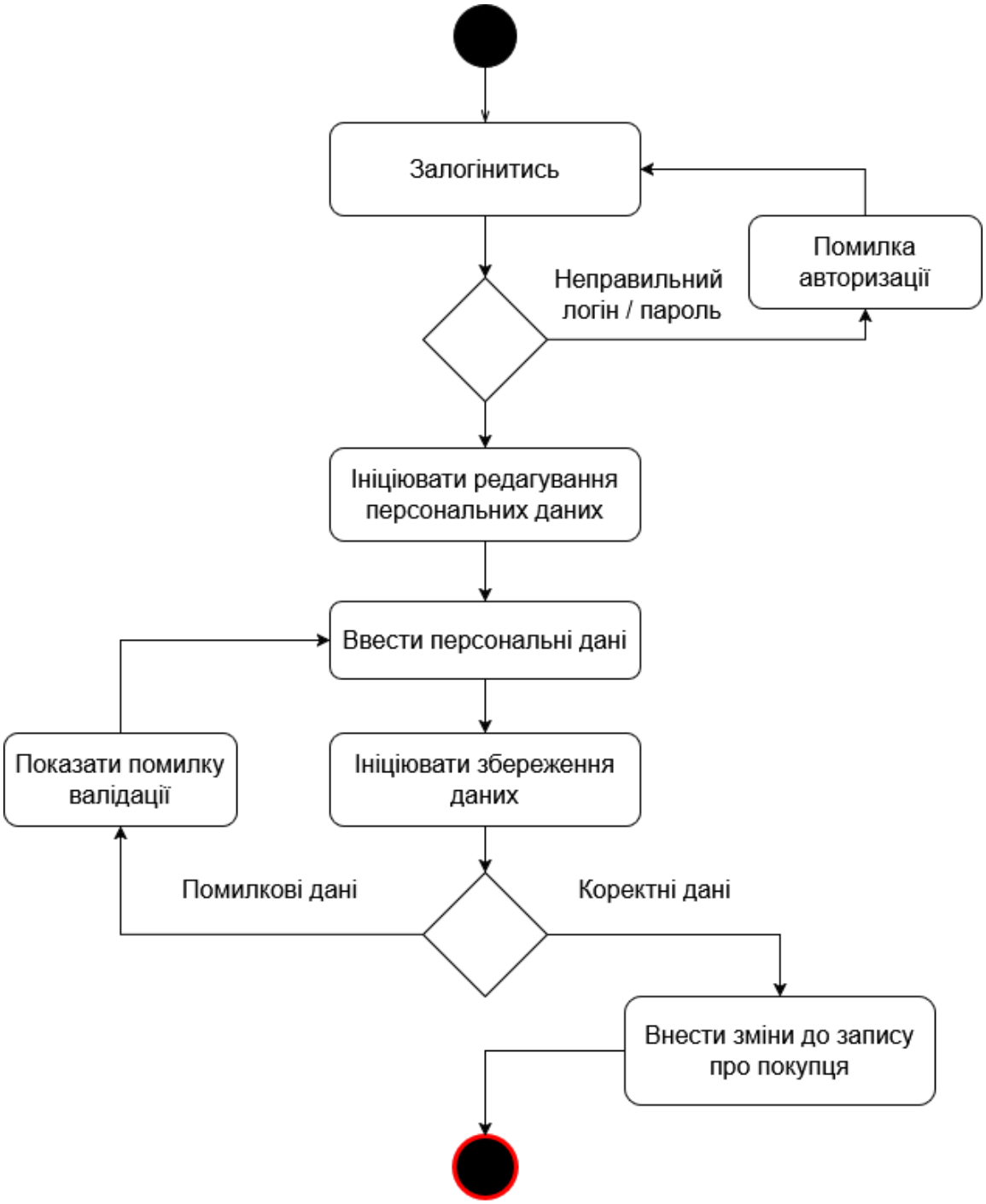


Рисунок 2.5 – Діаграма діяльності прецеденту «Ввести дані користувача»

На рисунку 2.6 показано діаграму діяльності для прецеденту «Створити продукт».

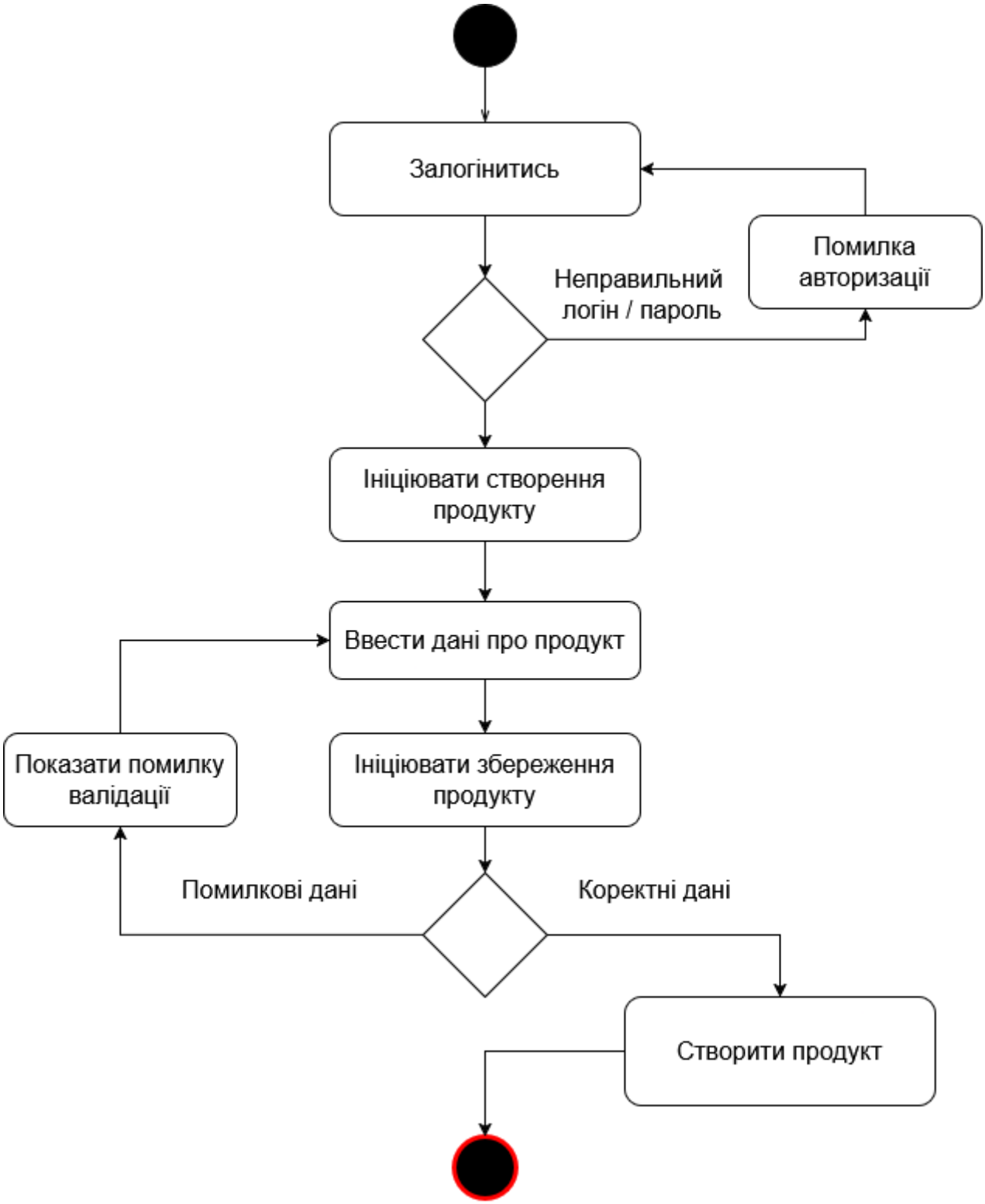


Рисунок 2.6 – Діаграма діяльності прецеденту «Створити продукт»

Розроблені дозволяють концептуально представити логіку алгоритму виконання операцій «Створити продукт» та «Ввести дані користувача».

Концептуальна (інформаційна) модель даних програмного забезпечення інтернет магазину «BioGild»

Концептуальна (інформаційна) модель даних (КМД) працює на високому рівні, забезпечує загальний погляд на потреби організації в даних. Вона визначає широкий і спрощений вигляд даних, які бізнес використовує або планує використовувати у своїй щоденній діяльності. Концептуальне моделювання даних має на меті створити спільне розуміння бізнесу шляхом визначення ключових понять бізнес-процесу. Ці ключові поняття зазвичай відображаються у діаграмі «сутність-зв’язок» (ERD) та супутніх визначеннях сутностей [9].

Розробка концептуальної моделі даних допомагає команді та заінтересованим сторонам зрозуміти основні моменти та загальну картину – з якими даними працювати та як різні сутності даних пов'язані між собою. Крім того, вона створює спільне розуміння бізнес-процесу та уніфіковану мову для всіх учасників, як технічних, так і нетехнічних.

ER-діаграма (діаграма «сутність-зв'язок») – це різновид блок-схеми, що відображає взаємозв'язки між сутностями (наприклад, людьми, об'єктами чи поняттями) у межах певної системи. Вона широко використовується при розробці та оптимізації реляційних баз даних у програмній інженерії, бізнес-інформаційних системах, освіті та дослідженнях. Такі діаграми, також відомі як ERD або ER-моделі, будуються із використанням прямокутників, ромбів, овалів і ліній, які ілюструють взаємозв'язки між сутностями, зв'язками та їх атрибутами. Умовно, сутності відповідають іменникам, а зв'язки – дієсловам.

ER-діаграми схожі на діаграми структури даних (DSD), однак останні фокусуються на внутрішніх зв'язках елементів у межах сутності, а не між сутностями. Також ER-діаграми часто застосовуються разом із діаграмами потоків даних (DFD), які ілюструють обмін інформацією між процесами чи системами.

Основні напрями використання ER-діаграм:

– Проєктування баз даних – ER-діаграми дозволяють моделювати як логічну, так і фізичну структуру реляційної бази даних. Вони зазвичай є першим кроком при визначенні вимог до інформаційної системи, а згодом - основою побудови конкретної бази.

– Аналіз та виправлення помилок – використовуються для виявлення та усунення логічних або структурних проблем у вже створених базах даних.

– Бізнес-аналіз – допомагають у моделюванні інформаційних систем, які підтримують бізнес-процеси, засновані на роботі з структурованими даними.

ER-діаграма складається з трьох основних компонентів:

– Сутність – об'єкт, про який зберігаються дані (наприклад, студент, товар), зображується прямокутником.

– Зв'язок – логічна взаємодія між сутностями (наприклад, студент *записується* на курс), позначається ромбом або підписом на лінії.

– Атрибут – характеристика сутності, представлена овалом.

Крім того, ER-діаграми фіксують кардинальність – кількісні характеристики взаємозв'язків. Вона може бути:

– один-до-одного (один студент - одна адреса),

– один-до-багатьох (один студент - кілька курсів),

– багато-до-багатьох (студенти - викладачі).

Діаграму «Сутність-Зв’язок» для зображення концептуальної моделі даних програмного забезпечення інтернет магазину «BioGild» показано на рисунку 2.7.

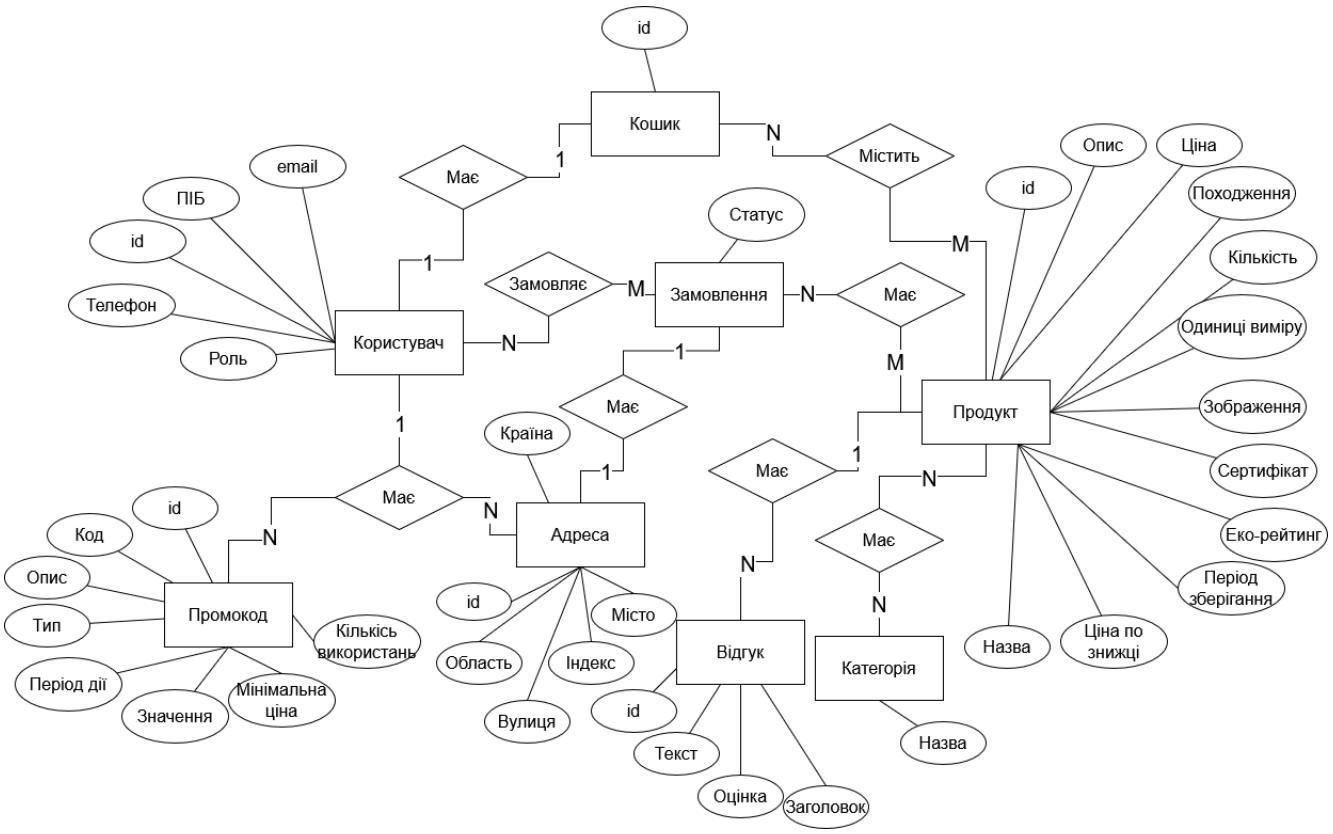


Рисунок 2.7 – Діаграма «Сутність-Зв’язок» програмного забезпечення інтернет магазину «BioGild»

Розрблема концептуальна модель даних буде застосована при переході до логічної моделі даних.

Проєкт інтерфейсу користувача програмного забезпечення інтернет магазину «BioGild»

UI прототип – це макет рішення, яке необхідно створити. Він дозволяє показати всі функції, які заплановані для реалізації у програмного застосунку, перевірити свою ідею та загальну UX-стратегію. Мета прототипу може змінюватися залежно від ваших потреб та етапу проєкту.

Прототипи створюються за допомогою різних інструментів: від простих графічних редакторів (як Sketch) до комплексних рішень для перетворення дизайну

в код. Деякі з них мають функції спільної роботи для збору відгуків, інші включають інструменти для опрацювання початкових вайрфреймів.

Можна виділити 4 ключові характеристики прототипів

– Подання – рорма прототипу: паперовий макет, мобільний інтерфейс, HTML-верстка тощо.

– Точність – рівень деталізації: від чернетки до реалістичного відображення.

– Інтерактивність – доступні функції: повний набір функцій, обмежена взаємодія або лише перегляд.

– Еволюція – життєвий цикл: швидке створення/видалення («швидке прототипування») або поступове вдосконалення до фінального продукту.

За типом прототипи діляться на

– Низькоточний (low-fidelity) прототип – швидкий та дешевий (часто паперовий), дає попередній перегляд продукту, але не підходить для повноцінного тестування.

– Високоточний (high-fidelity) прототип – максимально наближений до реального продукту, інтерактивний, корисний для демонстрації інвесторам або стейкхолдерам.

Головна перевага прототипування – воно допомагає розробникам краще зрозуміти потреби користувачів. Без цього навіть найкращий дизайн може призвести до проблем:

Основні переваги від застосування прототипів інтерфейсу:

– Економія часу та грошей.

– Можливість тестування ідеї на цільовій аудиторії.

– Наочний орієнтир для розробників.

– Документація проєкту.

– Спільна робота команди над конкретним результатом.

Рисунок 2.8 показує головний екран інтернет-магазину.

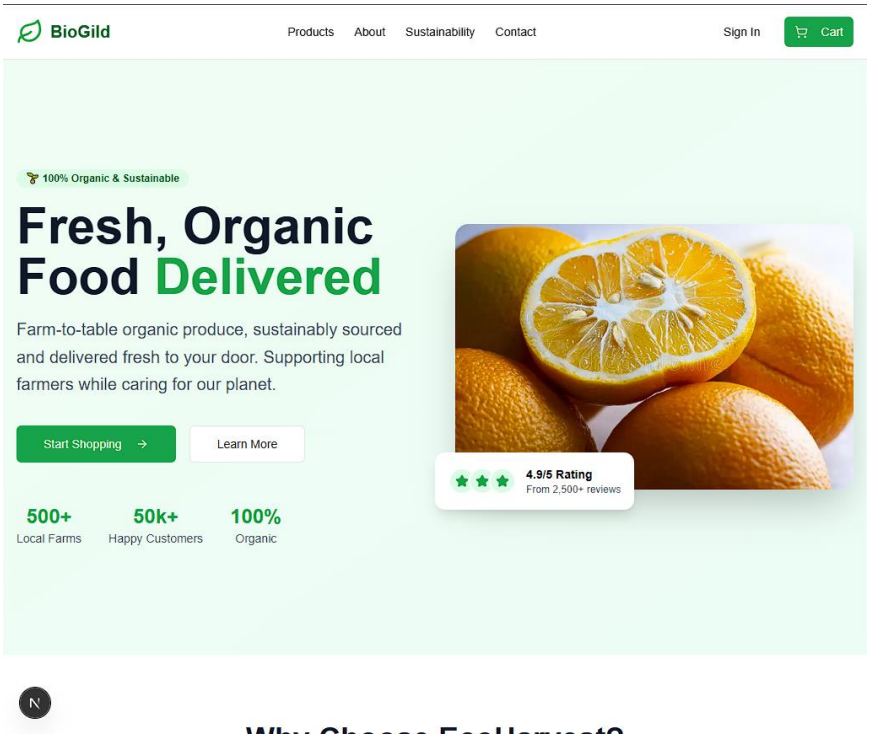


Рисунок 2.8 – Головний екран інтернет-магазину

Рисунок 2.9 показує сторінку продуктів інтернет-магазину.

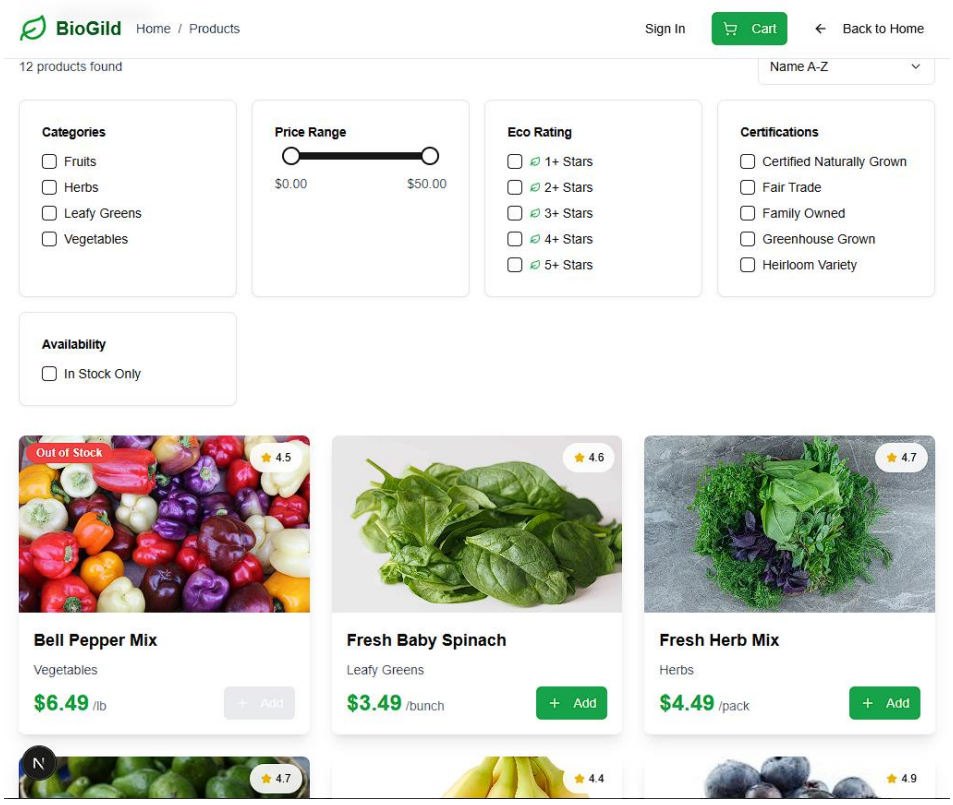


Рисунок 2.9 – Сторінка продуктів інтернет-магазину

Рисунок 2.10 показує сторінку обраного продутку інтернет-магазину.

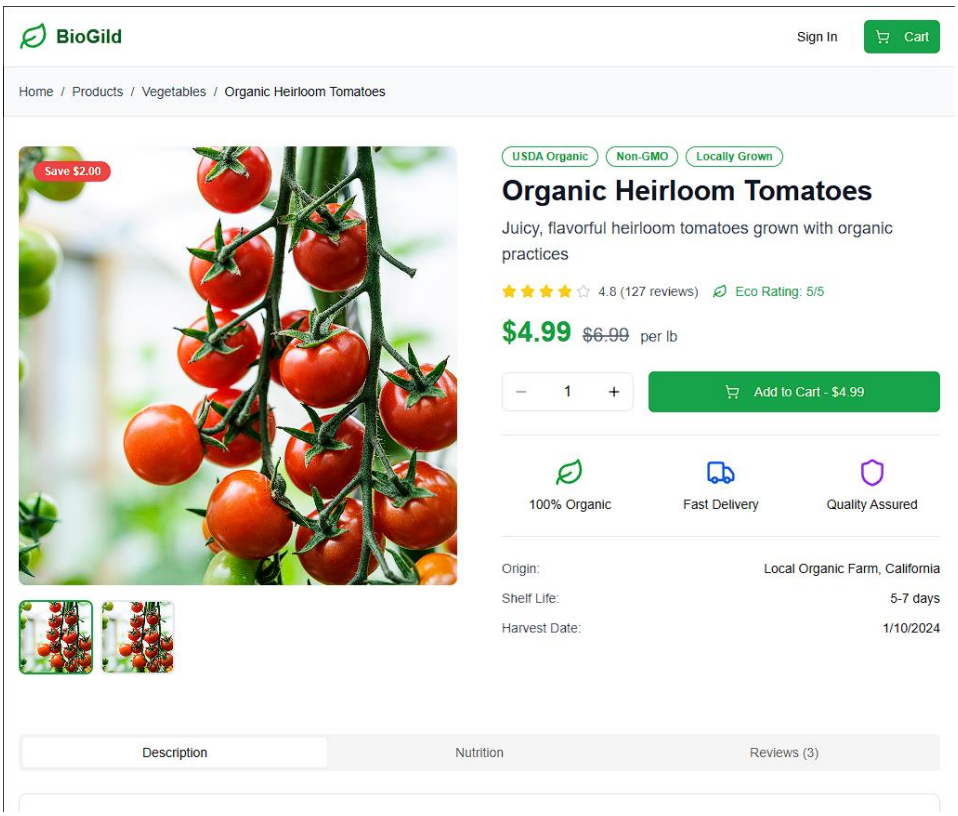


Рисунок 2.10 – Сторінка продукту інтернет-магазину

Приклади інтерфейсу дозволяють отримати краще уявлення про роботу веб-застосунку.

2.3.2 Технічний проєкт програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Статична модель програмного забезпечення інтернет-магазину «BioGild»

Діаграма класів UML є одним з найпоширеніших інструментів для опису архітектури програмного забезпечення. Як структурна діаграма, вона відображає

ключові елементи системи та їх взаємозв'язки [8]. Основні компоненти діаграми класів включають:

– Класи, що підлягають реалізації у коді.

– Основні об'єкти системи.

– Способи взаємодії між класами та об'єктами .

Переваги застосування діаграм класів

– Візуалізація моделі даних для інформаційних систем будь-якої складності

– Отримання чітке уявлення про загальну структуру програми

– Наочне представлення специфічних вимог системи та поширення цієї інформації в бізнесі.

– Створення детальних схеми, які вказують конкретний код, необхідний для реалізації описаної структури .

– Можливість надати незалежні від реалізації описи типів даних, які передаються між компонентами системи

Діаграму класів програмного забезпечення інтернет-магазину «BioGild» показано на рисунку 2.11.

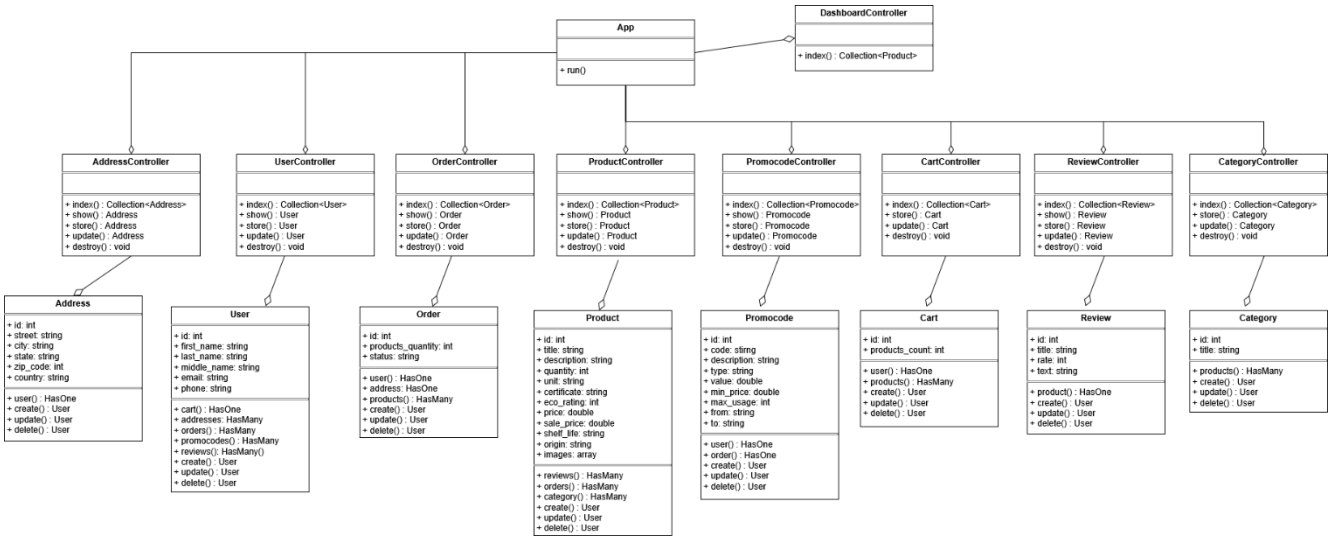


Рисунок 2.11 – Архітектурне рішення у вигляді діаграми класів програмного забезпечення інтернет-магазину «BioGild»

Розроблена діаграма класі будет основою для архітектурного рішення та подальшої специфікації класів.

Специфікації класів програмного забезпечення інтернет-магазину «BioGild»

Мета специфікації класів полягає у документуванні очікуваної поведінки класів (що повинні робити його методи), а не їх реалізації (як вони реалізовані).

1) Клас App – базовий клас програми, який відповідає за ініціалізацію та запуск веб-застосунку при отриманні HTTP-запиту. Виконує роль центрального компонента, що координує роботу всієї системи.

2) Клас DashboardController – контролер адміністративної панелі, який відповідає за відображення та управління інтерфейсом адміністратора системи.

3) Клас UserController – контролер користувачів, що реалізує всі операції CRUD (створення, читання, оновлення, видалення) для роботи з обліковими записами користувачів системи.

4) Клас ProductController – контролер продуктів, який забезпечує повну функціональність для управління продуктами

5) Клас OrderController – контролер замовлень, що реалізує всі бізнес-процеси, пов'язані з життєвим циклом замовлення – від створення нових замовлень до їх підтвердження, скасування або повернення коштів.

6) Клас CartController – контролер кошика, який надає функціональність для додавання товарів до кошика, їх перегляду, оновлення кількості та видалення з кошика перед оформленням замовлення.

7) Клас ReviewController – контролер відгуків, що відповідає за повний цикл роботи з відгуками про продукти: публікацію нових відгуків, їх перегляд, модерацію та видалення при необхідності.

8) Клас PromocodeController – контролер промокодів, який реалізує механізми створення, перегляду та управління промокодами для знижок і спеціальних пропозицій.

9) Клас AddressController – контролер адрес, який реалізує механізми створення, перегляду та управління адресами користувачів.

10) Клас CategoryController – контролер категорій продуктів, що забезпечує повний функціонал для класифікації товарів: створення нових категорій, їх редагування, перегляд та видалення.

11) Клас User – модель користувача, яка відображає всі атрибути та властивості облікового запису користувача в системі.

12) Клас Cart – модель кошика, що представляє тимчасове сховище товарів, які користувач вибрав для майбутнього замовлення.

13) Клас Product – модель продукту, яка містить всі необхідні властивості та характеристики товару в системі.

14) Клас Order – модель замовлення, що відображає всі дані про оформлені покупки користувачів.

15) Клас Review – модель відгуку, яка зберігає інформацію про оцінки та коментарі користувачів до продуктів.

16) Клас Promocode – модель промокоду, що містить дані про знижки та спеціальні пропозиції в системі.

17) Клас Address – модель адреси, що містить дані про адресу користувача для оформлення замовлення.

18) Клас Category – модель бази даних, представляє категорію.

Динамічна модель програмного забезпечення інтернет-магазину «BioGild» у вигляді діаграм послідовності

Діаграма послідовності належить до діаграм взаємодії, оскільки ілюструє, як і в якій послідовності об’єкти обмінюються повідомленнями між собою. Вона

використовується як розробниками, так і бізнес-аналітиками для моделювання вимог до нової системи або документування вже існуючих процесів. Такі діаграми також відомі як діаграми подій або сценаріїв подій.

Існують два основні типи діаграм послідовності: UML-діаграми та код-орієнтовані діаграми [8].Діаграми послідовності можуть бути корисними довідковими матеріалами для бізнесів та інших організацій, дозволяючи:

– Відобразити деталі варіанта використання UML.

– Змоделювати логіку складних процедур, функцій або операцій.

– Побачити взаємодію об'єктів та компонентів процесу.

– Запланувати та зрозуміти детальну функціональність існуючого чи майбутнього сценарію.

Наступні сценарії ідеально підходять для використання діаграм послідовності:

– Сценарій використання – це діаграма того, як ваша система потенційно може використовуватися. Вона допомагає перевірити логіку кожного сценарію використання системи.

– Логіка методу – так само, як можна використовувати діаграму послідовності UML для дослідження логіки варіанта використання, також можна застосовувати її для аналізу логіки будь-якої функції, процедури чи складного процесу.

– Логіка сервісу – якщо розглядати сервіс як високорівневий метод, який використовують різні клієнти, то діаграма послідовності є ідеальним способом для його візуалізації.

На рисунку 2.12 показано діаграму послідовності прецеденту «Створити продукт».

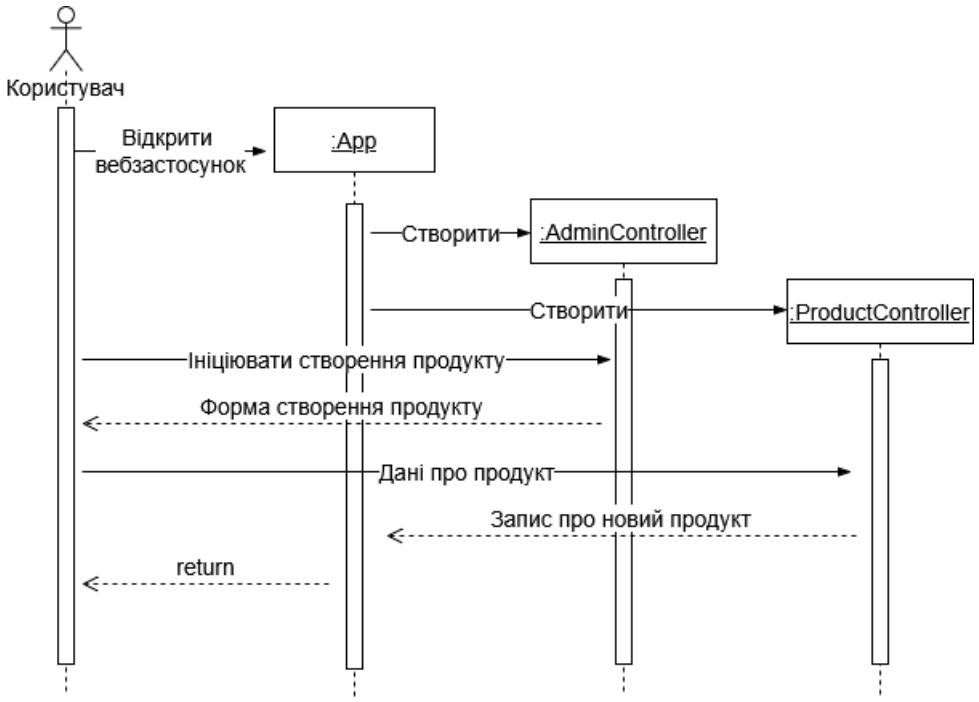


Рисунок 2.12 – Діаграма послідовності прецеденту «Створити продукт»

На рисунку 2.13 показано діаграму послідовності прецеденту «Ввести дані користувача».

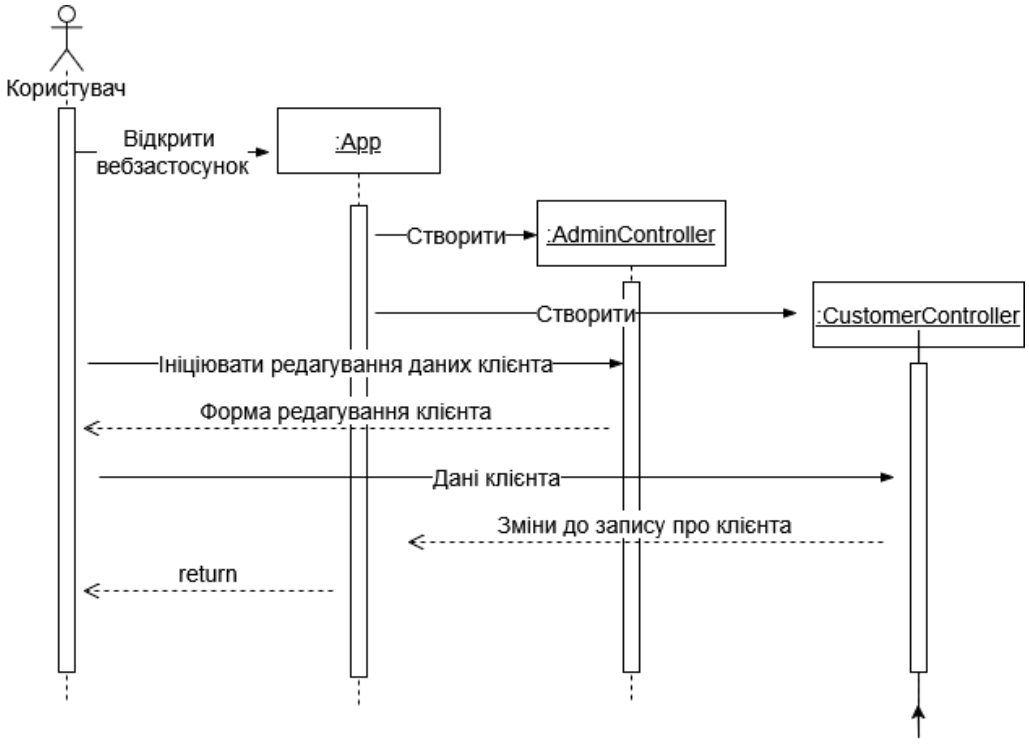


Рисунок 2.13 – Діаграма послідовності прецеденту «Ввести дані користувача»

Застосуємо розроблені діаграми послідовностей при кодувані алгоритмів.

Логічна модель даних програмного забезпечення інтернет-магазину «BioGild»

Логічною моделлю даних називають вдосконалену версію концептуальної моделі. У ній схематично представлені обмеження даних, імена сутностей і зв'язки, які потрібно реалізувати незалежно від платформи. За допомогою логічної моделі даних бізнес-аналітики та архітектори даних можуть візуалізувати операційні або транзакційні процеси на діаграмі взаємозв'язків між сутностями. Логічні моделі даних визначають, як працюють і взаємодіють об'єкти даних, у зрозумілий для зацікавлених представників бізнесу спосіб. Саме тому вони розробляються незалежно від реальної бази даних, у якій згодом будуть розгорнуті дані [9].

Для побудови логічної моделі даних потрібно виконати наступне:

1) знайти всі необхідні об'єкти та визначити їх атрибути;

2) обрати відповідні первинні ключі (PK) як унікальні ідентифікатори для груп атрибутів;

3) нормалізувати та денормалізувати модель даних відповідно до експлуатаційних вимог; визначити зв'язки між бізнес-об'єктами в моделі даних;

4) перевірити, чи точно об'єкти даних та їх взаємозв'язки відображають потрібну бізнес-логіку.

Потрібно визначити зв'язки між окремими об'єктами. Деякі з них пов'язані між собою безпосередньо, а інші – через проміжну сутність. Зазвичай цей процес відбувається під час консультацій із зацікавленими сторонами, щоб їхня точка зору була правильно врахована при співставленні сутностей даних із бізнес-вимогами. Крім того, деякі об'єкти можна дублювати, а інші стратегічно обмежити строго одним екземпляром, щоб підвищити ефективність запитів і знизити вимоги до простору для зберігання.

На рисунку 2.14 показано логічну модель даних програмного забезпечення інтернет-магазину «BioGild».

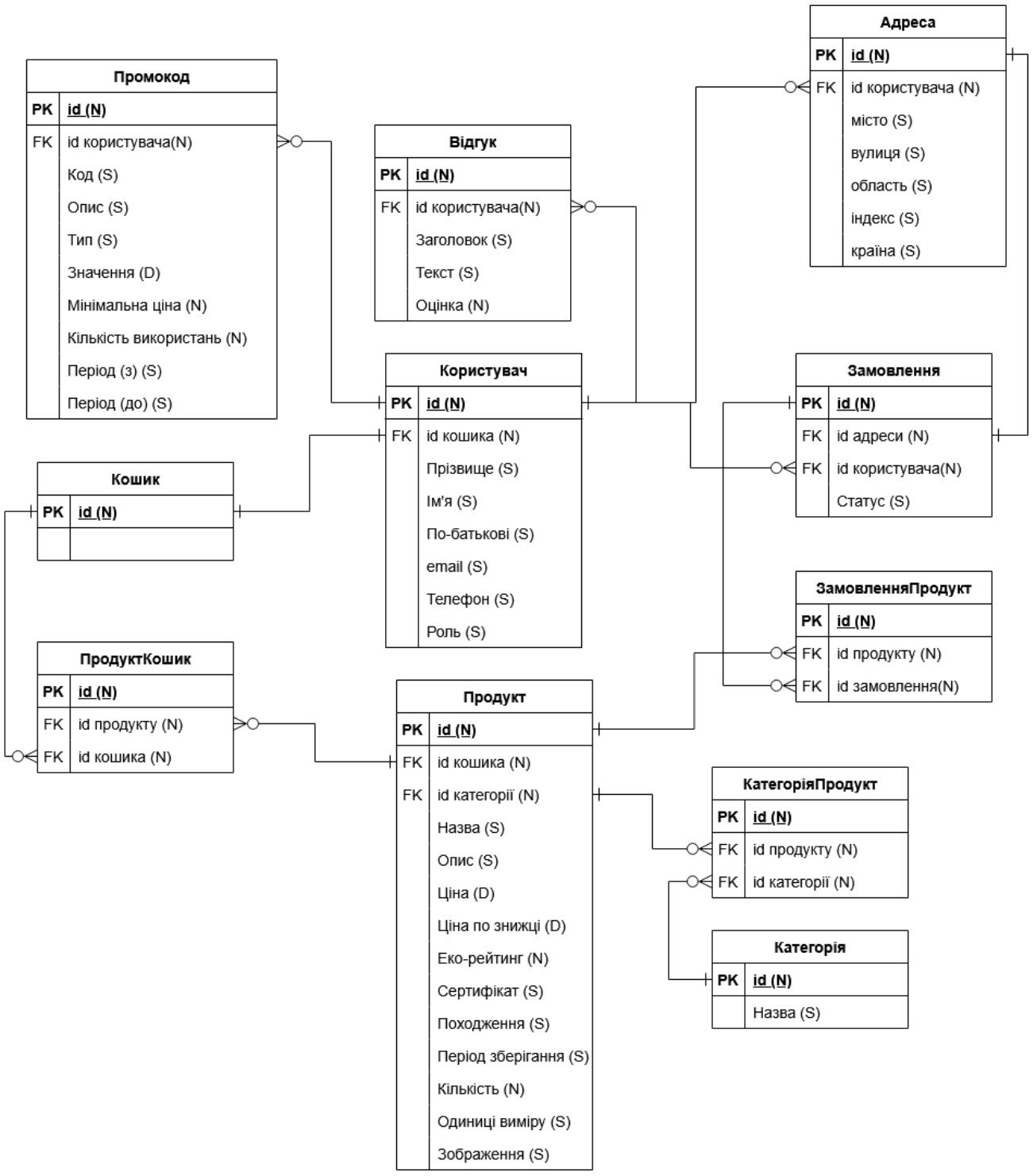


Рисунок 2.14 – Логічна модель даних програмного забезпечення інтернет-магазину «BioGild»

Розрблема логічна модель даних буде застосована при переході до фізичної моделі даних.

2.3.3 Робочий проєкт програмного забезпечення інтернет-магазину екологічних продуктів «BioGild»

Вибір та обґрунтування засобів реалізації програмного забезпечення інтернет-магазину «BioGild»

Laravel – це безкоштовний та потужний PHP-фреймворк з відкритим кодом, створений для веб-розробників. Він призначений для створення веб-додатків з підтримкою баз даних, дотримуючись архітектурного шаблону Model-View-Controller (MVC), що дозволяє розробникам швидко будувати надійні програми [10]. Laravel надає:

– Велику екосистему – включає перевірену бібліотеку основних компонентів та пакетів.

– Плагіни для IDE – розширюють можливості середовища розробки для покращення досвіду кодування.

– Просте налаштування середовища – з різними варіантами команди php artisan serve для запуску додатків на PHP-сервері.

– Корисні інструменти – дозволяють писати менше коду та зосередитись на логіці програми.

Laravel відрізняється великою спільнотою розробників, які постійно створюють нові пакети для покращення фреймворку, забезпечуючи його розвиток. Водночас він пропонує вбудовані механізми безпеки (наприклад, хешування паролів за допомогою Bcrypt), що робить його ідеальним вибором для захищених додатків.

Фреймворк повністю підтримує архітектуру MVC, спрощуючи розробку, а також інтегрується з Vue.js та Nuxt.js для створення сучасних односторінкових додатків (SPA).

Про зручність: Laravel поєднує чітку документацію, модульну структуру та підтримку різних СУБД, що дозволяє розробникам працювати ефективно – від міграцій баз даних до створення складних інтерфейсів.

LAMP – популярний стек відкритих технологій, який переважно використовується у веб-розробці. Склад LAMP Акронім LAMP утворено з перших літер чотирьох компонентів, необхідних для створення повноцінного середовища розробки [11]:

– Linux – операційна система, яка керуватиме роботою всіх компонентів.

– Apache HTTP Server – це програмне забезпечення веб-сервера, що забезпечує доставку як статичних, так і динамічних веб-сторінок користувачам.

– MySQL – це система керування РБД, яка застосовується при створенні веб бд, зберігання та управління динамічним контентом.

– PHP – мови програмування, що дозволяє розроблювати веб-застоунків.

Стек LAMP часто застосовують в розробці веб-застосунків, а саме:

– Системи управління контентом (CMS) – платформи як от WordPress або Joomla базуються на PHP, а MySQL забезпечує надійну роботу з даними.

– Інтернет-магазини – РБД ефективно обробляють транзакції, великі каталоги товарів та користувацькі дані.

– Динамічні вебсайти – PHP дозволяє адаптувати контент у реальному часі на основі взаємодії з користувачем.

– Data-oriented застосунки – соціальні мережі, інтерактивні карти та форуми використовують LAMP для обробки складних структур даних.

LAMP стек надає наступний перелік переваг від його застосування:

– Відкритість – всі компоненти можна вільно модифікувати.

– Гнучкість – будь-який елемент стека можна замінити аналогом.

– Підтримку – великі спільноти та численні ресурси для розробників.

– Стабільність – технології перевірені роками.

– Простота – швидке розгортання та зручне управління.

LAMP стек має наступний перелік недоліків:

– Залежність від Linux – хоча компоненти крос-платформенні, стек асоціюється з Linux.

– Обмеження MySQL – менш ефективний для неструктурованих даних порівняно з NoSQL.

– Навантаження на Apache – можливі проблеми з продуктивністю при високому трафіку.

– Складність робочого процесу – перемикання між PHP та JavaScript може ускладнювати розробку.

Сьогодні MySQL посідає друге місце серед РСКБД у світі за даними DB Engines. Цією системою користуються численні вебсайти та застосунки як от Spotify, Netflix, Facebook та Booking.com. MySQL підходить для невеликих компаній або організацій без потужних команд з обробки даних завдяки низькій вартості, простоті впровадження та підтримці великої спільноти. Втім, остаточний вибір залежить від конкретних потреб проєкту. MySQL може бути ідеальним рішенням для стандартних веб-застосунків [12].

Для MySQL притаманні наступні переваги від використання:

– Відкритий код MySQL – позиціонує себе як ПЗ з вільним доступом за рахунок відкритого коду, що робить його безкоштовним для використання, модифікації та розповсюдження. Це дозволяє користуватися СКБД як приватним розробникам, так і великим компаніям без значних витрат на ліцензії.

– Масштабованість – MySQL пропонує гнучкі можливості масштабування, дозволяючи обробляти зростаючі обсяги даних та навантаження. Завдяки шардингу, реплікації та кластеризації він ефективно масштабується: вертикально (додавання ресурсів до одного сервера) та горизонтально (розподіл даних між кількома серверами).

– Висока продуктивність MySQL – відомий швидкодією, особливо в системах із інтенсивним читанням даних. MySQL використовує оптимізацію запитів, індексацію та кешування для швидкої обробки навіть великих масивів інформації.

– Надійність – MySQL підтримує ACID (атомарність, узгодженість, ізоляція, стійкість), транзакціям та механізмам відновлення після збоїв, що робить його придатним для критично важливих систем, даючи цілісність даних.

– Підтримка спільноти – як одна з найпопулярніших реляційних СКБД, MySQL має велику активну спільноту, яка надає документацію, форуми та інші ресурси для розробників.

Для MySQL притаманні наступні недоліки при використанні:

– Обмежена функціональність – порівняно з іншими корпоративними базами даних, MySQL може не підтримувати деякі складні типи даних, повнотекстовий пошук або аналітичні функції без додаткових плагінів.

– Складність масштабування – хоча MySQL підтримує масштабування, управління розподіленими системами може бути складним і вимагати досвіду в адмініструванні. Горизонтальне масштабування іноді призводить до проблем узгодженості даних.

– Обмеження рушіїв зберігання – архітектура MySQL залежить від рушіїв зберігання (InnoDB, MyISAM тощо). Деякі з них мають обмеження у швидкодії, функціях або надійності, що може вплинути на роботу додатків.

– Одно-майстерна реплікація – стандартна реплікація MySQL працює за принципом одного головного сервера (master), що обробляє записи, та підлеглих (slaves) для читання. Це може створювати вузькі місця та ризик відмови в системах з високою доступністю.

– Ліцензування та підтримка – хоча MySQL є відкритим ПО, деякі додаткові інструменти та підтримка від Oracle (наприклад, Enterprise Edition) вимагають оплати, що збільшує витрати для бізнесу.

Обраний стек технологій (Laravel + LAMP з MySQL) оптимально відповідає вимогам розробки веб-додатку, поєднуючи перевірену надійність із сучасними

можливостями. Laravel забезпечує швидку реалізацію функціональності завдяки зручній архітектурі та потужній екосистемі, а LAMP-стек гарантує стабільну роботу при помірних навантаженнях. MySQL як СКБД була обрана через простоту впровадження та гарну сумісність із обраними технологіями. Разом ці інструменти дозволяють створити продуктивне та масштабоване рішення з оптимальними витратами ресурсів на розробку та підтримку.

Фізична модель даних програмного забезпечення інтернет-магазину «BioGild»

Фізична модель даних ще глибше уточнює логічну модель з урахуванням особливостей реалізації на конкретній технології баз даних. Фізичні моделі даних надають детальну інформацію, на основі якої адміністратори та розробники баз даних відтворюють бізнес-логіку у фізичній базі даних. Ці моделі містять додаткові атрибути, яких не було в логічній моделі даних, наприклад тригери, збережені процедури та типи даних. Оскільки фізичні моделі даних фактично представляють собою модель реальної БД, вони повинні враховувати специфічні обмеження платформи, зокрема угоди про іменування та зарезервовані слова [9].

Розробка фізичної моделі передбачає наступне:

1) перетворити локальну модель даних відповідно до платформи обраного постачальника баз даних;

2) співставити об'єкти даних з таблицями;

3) визначити та створити первинні (PK) та зовнішні (FK) ключі в таблицях бази даних за необхідності;

4) переконатися, що структура бази даних правильно нормалізована, щоб уникнути надмірного дублювання даних та покращити їх цілісність;

5) додати необхідні обмеження, правила, розділи та програмні функції бази даних, які спростять розробку додатків;

6) порівняти фізичну модель даних із логічною моделлю даних і переконатися, що всі бізнес-вимоги правильно перетворено.

У деяких випадках одна сутність розділяється на кілька таблиць. Кожна таблиця містить кілька стовпців, у яких зберігається інформація, визначена атрибутами попередніх моделей даних з урахуванням реального типу зберігання такого атрибуту у бд (цілочисельні, логічні, varchar тощо).

На рисунку 2.15 показано фізичну модель даних програмного забезпечення інтернет-магазину «BioGild».

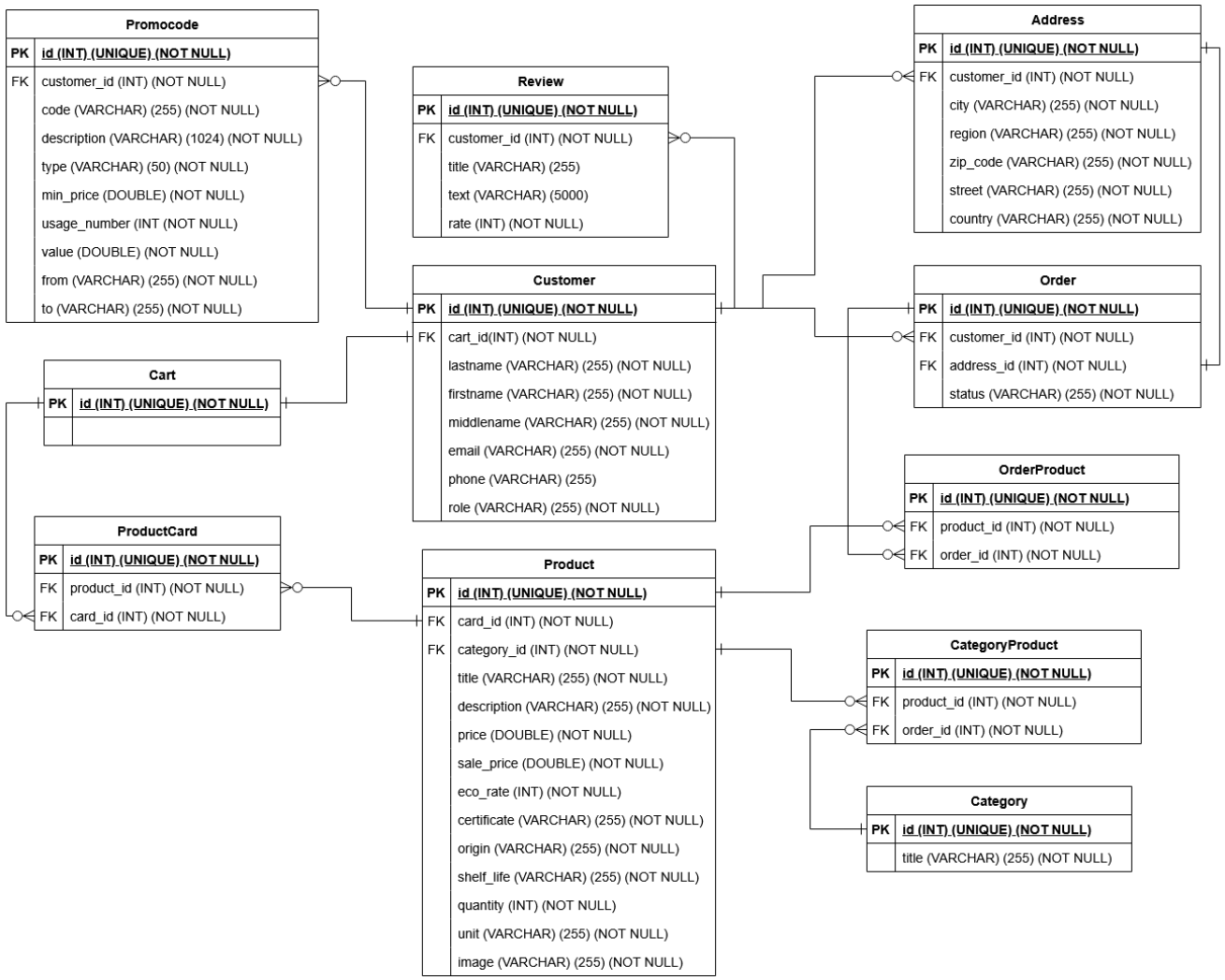


Рисунок 2.15 – Фізична модель даних програмного забезпечення інтернет-магазину «BioGild»

За розробленою фізичною моделлю даних буде сконфігуровано базу даних програмного забезпечення інтернет-магазину «BioGild».

Реалізація основних компонентів програмного забезпечення інтернет-магазину «BioGild»

Наведемо вихідний текст реалізації основних класів програмного забезпечення інтернет-магазину «BioGild».

ProductController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreProductRequest;
use App\Http\Requests\UpdateProductRequest;
use App\Http\Resources\ProductResource;
use App\Http\Resources\ProductCollection;
use App\Models\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class ProductController extends Controller
{
    public function index(Request $request)
    {
        $products = Product::query()
            ->when($request->category_id, function ($query, $categoryId) {
                $query->where('category_id', $categoryId);
            })
            ->when($request->search, function ($query, $search) {
                $query->where(function ($q) use ($search) {
                    $q->where('title', 'like', "%{$search}%")
                      ->orWhere('description', 'like', "%{$search}%");
                });
            })
            ->when($request->price_min, function ($query, $priceMin) {
                $query->where('price', '>=', $priceMin);
            })
            ->when($request->price_max, function ($query, $priceMax) {
                $query->where('price', '<=', $priceMax);
```

```
        })
        ->when($request->eco_rating, function ($query, $ecoRating) {
            $query->where('eco_rating', '>=', $ecoRating);
        })
        ->when($request->in_stock, function ($query) {
            $query->where('quantity', '>', 0);
        })
        ->when($request->sort_by, function ($query, $sortBy) use ($request) {
            $direction = $request->sort_direction ?? 'asc';
            switch ($sortBy) {
                case 'price':
                    $query->orderBy('price', $direction);
                    break;
                case 'rating':
                    $query->withAvg('reviews', 'rate')
                          ->orderBy('reviews_avg_rate', $direction);
                    break;
                case 'eco_rating':
                    $query->orderBy('eco_rating', $direction);
                    break;
                case 'newest':
                    $query->orderBy('created_at', 'desc');
                    break;
                default:
                    $query->orderBy('title', $direction);
            }
        })
        ->with(['category', 'reviews'])
        ->paginate($request->per_page ?? 15);

    return new ProductCollection($products);
}

public function store(StoreProductRequest $request)
{
    $data = $request->validated();

    // Handle image uploads
    if ($request->hasFile('images')) {
        $images = [];
        foreach ($request->file('images') as $image) {
            $path = $image->store('products', 'public');
            $images[] = $path;
        }
        $data['images'] = $images;
    }

    $product = Product::create($data);
    return new ProductResource($product->load(['category', 'reviews']));
```

```
}

public function show(Product $product)
{
    return new ProductResource($product->load([
        'category',
        'reviews.user',
        'orders' => function ($query) {
            $query->latest()->limit(5);
        }
    ]));
}

public function update(UpdateProductRequest $request, Product $product)
{
    $data = $request->validated();

    // Handle image uploads
    if ($request->hasFile('images')) {
        // Delete old images
        if ($product->images) {
            foreach ($product->images as $oldImage) {
                Storage::disk('public')->delete($oldImage);
            }
        }

        $images = [];
        foreach ($request->file('images') as $image) {
            $path = $image->store('products', 'public');
            $images[] = $path;
        }
        $data['images'] = $images;
    }

    $product->update($data);
    return new ProductResource($product->load(['category', 'reviews']));
}

public function destroy(Product $product)
{
    // Delete images
    if ($product->images) {
        foreach ($product->images as $image) {
            Storage::disk('public')->delete($image);
        }
    }

    $product->delete();
    return response()->json(['message' => 'Product deleted successfully']);
```

```
    }

    public function featured(Request $request)
    {
        $products = Product::query()
            ->where('eco_rating', '>=', 4)
            ->withAvg('reviews', 'rate')
            ->having('reviews_avg_rate', '>=', 4)
            ->with(['category', 'reviews'])
            ->limit($request->limit ?? 8)
            ->get();

        return ProductResource::collection($products);
    }
}
```

Тестування основних класів програмного забезпечення інтернет-магазину «BioGild»

Еквівалентне розбиття (Equivalence Partitioning) або розбиття на класи еквівалентності – це техніка тестування програмного забезпечення типу «чорного ящика», яка покликана зменшити кількість тестових випадків до оптимального числа, зберігаючи при цьому адекватне покриття тестування. Простіше кажучи, основний підхід для виконання еквівалентного розбиття полягає в тому, що вхідні дані будь-якого програмного застосунку діляться на розділи або класи таким чином, що всі вхідні дані, які належать до одного розділу або класу, обробляються системою однаково. Якщо один набір вхідних даних поводиться правильно, всі інші набори повинні поводитися аналогічно. Це дозволяє тестувальникам вибрати лише один набір вхідних даних певного класу для тестування.

Тестувальник може визначити тестові випадки, які можуть виявити серію помилок при мінімальній кількості класів. Вхідні дані кожного класу еквівалентності дозволяють тестувальникам ефективно перевіряти коректні та некоректні входи, гарантуючи, що різні сценарії обробляються програмою правильно. Еквівалентне розбиття також дуже корисне там, де кількість вхідних даних досить велика, що повне тестування неможливе. У цьому відношенні метод оптимізує процес тестування, дозволяючи отримати високе покриття тестування з

меншою кількістю тестових випадків, забезпечуючи хороший контроль якості без витрат зайвого часу та ресурсів.

Можна виділити наступні переваги еквівалентного розбиття:

– Зменшує кількість тестів – групує вхідні дані у класи, тестуючи лише представників.

– Підвищує ефективність – скорочує обсяг тестів для ручної/автоматизованої перевірки.

– Забезпечує покриття – охоплює всі ключові сценарії, знижуючи ризик пропуску дефектів.

– Виявляє межі – допомагає знаходити граничні умови, де ймовірність помилок вища.

– Спрощує тест-дизайн – структурований підхід полегшує створення зрозумілих тестів.

– Раннє виявлення помилок – системний аналіз класів даних допомагає знаходити дефекти на ранніх етапах.

– Покращує комунікацію – спрощує обговорення тестової стратегії між командою та стейкхолдерами.

Можна виділити наступні недоліки еквівалентного розбиття:

– Може пропускати специфічні дефекти – якщо помилка виникає лише для певних значень у класі.

– Залежить від правильності розбиття – неточне визначення класів призводить до прогалин у тестуванні.

– Обмежений «чорною скринькою» – не враховує внутрішню логіку коду.

– Не завжди ефективний для складної логіки – може бути недостатнім для ПЗ з великою кількістю умов.

– Ризик пропуску граничних значень – потребує додаткового аналізу.

За розробленою у розділі 2.1.2 специфікацією варіантів використання проведемо тестування.

Проведемо тестування прецеденту «Створити продукт», таблиця 2.1 містить виявлені класи.

Таблиця 2.1 – Класи для прецеденту «Створити продукт»

Вхідні дані	Правильні класи	Неправильні класи
Назва, опис, ціна, ціна зі знижкою, походження, час зберігання, кількість, одиниці виміру, сертифікат, еко-рейтинг, категорія, зображення у форматі .jpeg (.jpg)/.webp/.png	Валідні дані щодо назви, опису, ціни, ціни зі знижкою, походження, часу зберігання, кількість, одиниці виміру, сертифікат, еко-рейтинг, категорія, зображення (1).	Введено назву більше 255 символів (2). Назва відсутня (3). Введено опис більше 1000 символів (4). Опис відсутній (5). Введено не число замість значення ціни (6). Введено від'ємне число замість ціни (7). Ціна відсутня (8). Введено не число замість значення ціни зі знижкою (9). Введено від'ємне число замість ціни зі знижкою (10). Ціна зі знижкою відсутня (11). Введено походження більше 255 символів (12). Походження відсутнє (13). Введено не число замість кількості (14). Введено від'ємне число замість кількості (15). Кількість відсутня (16). Введено текст одиниці виміру більше 50 символів (17). Одиниці виміру відсутні (18). Введено текст сертифікату більше 255 символів (19). Сертифікат відсутній (20). Введено не число замість еко-рейтингу (21). Введено від'ємне число замість еко-рейтингу (22). Еко-рейтинг відсутній (23). Додано зображення неправильного формату (24). Додано зображення більше 20 МБ (25) Зображення відсутнє (26).

Таблиця 2.2 показує результат тестування правильних класів

Таблиця 2.2 – Тестування з правильними класами

Тест №	Перелік класів	Input	Output	Результат
1	1	Fresh Baby Spinach Tender baby spinach leaves, perfect for salads and cooking 3,49$ Sustainable Farm Oregon 3-5 days 50 bunch USDA Organic, Certified Naturally Grown 4 Leafy Greens .jpeg image	Новий товар у базі даних системи	Тест пройдено

Таблиця 2.3 показує результат тестування неправильних класів

Таблиця 2.3 – Тестування з неправильними класами

Тест №	Перелік класів	Input	Output	Результат
2	2, 4, 12, 17, 19	<Text with more than 1000 characters>	Помилка на екрані користувача	Тест пройдено
3	3, 5, 8, 11, 13, 16,18, 20, 23, 26	-	Помилка на екрані користувача	Тест пройдено
4	6,9,14,21	This is NAN	Помилка на екрані користувача	Тест пройдено
5	7, 10, 15, 22	-123456789	Помилка на екрані користувача	Тест пройдено
6	24	.gif image	Помилка на екрані користувача	Тест пройдено
7	25	Зображення > 20Mb	Повідомлення про помилку	Тест пройдено

Випробування програмного забезпечення інтернет-магазину «BioGild»

Випробування ПЗ є комплексом заходів, спрямованих на перевірку відповідності програмного продукту встановленим вимогам, технічній документації та стандартам. Випробування включають тестування функціональності, надійності, продуктивності та інших характеристик системи для виявлення можливих дефектів та підтвердження готовності ПЗ до експлуатації.

Процес випробувань проводиться за заздалегідь затвердженими методиками та програмами, які визначають порядок, умови та критерії оцінки якості програмного забезпечення. Результати випробувань фіксуються у звітній документації та є підставою для прийняття рішення про впровадження ПЗ або необхідність його доопрацювання.

За підготовленою та наведеною у додатку Б програмою та методикою випробувань, проведемо випробування програмного забезпечення інтернет-магазину «BioGild».

Після відкриття програмного ПЗ користувач бачить головну сторінку інтернет-магазину, яка містить коротку інформацію про магазин та короткий перелік товарів, що показано на рисунках 2.16-2.17.

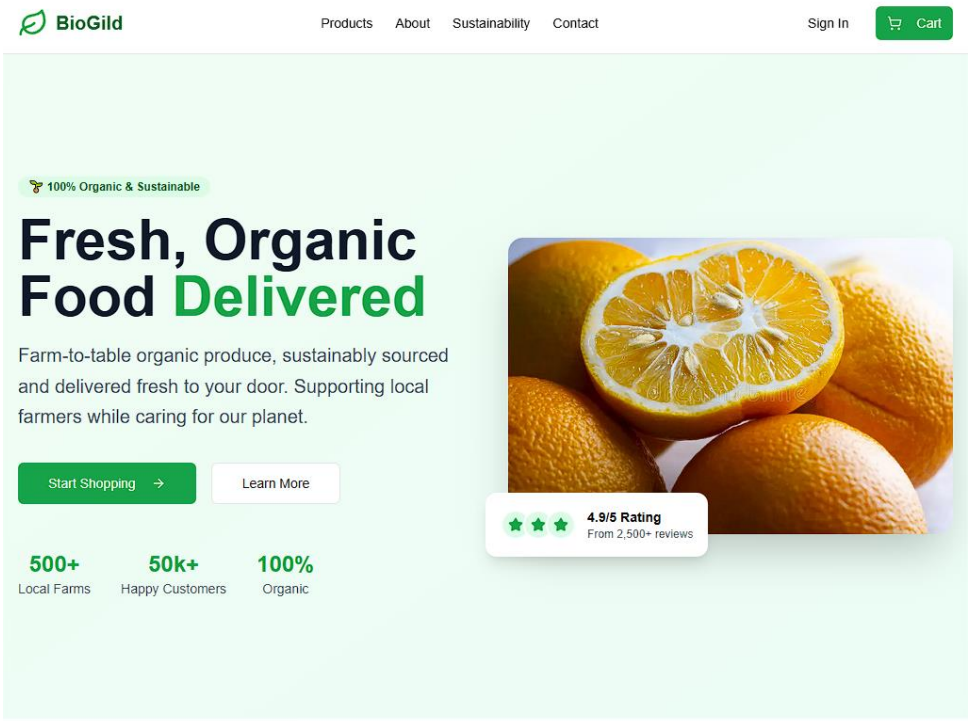


Рисунок 2.16 – Головна сторінка інтернет-магазину

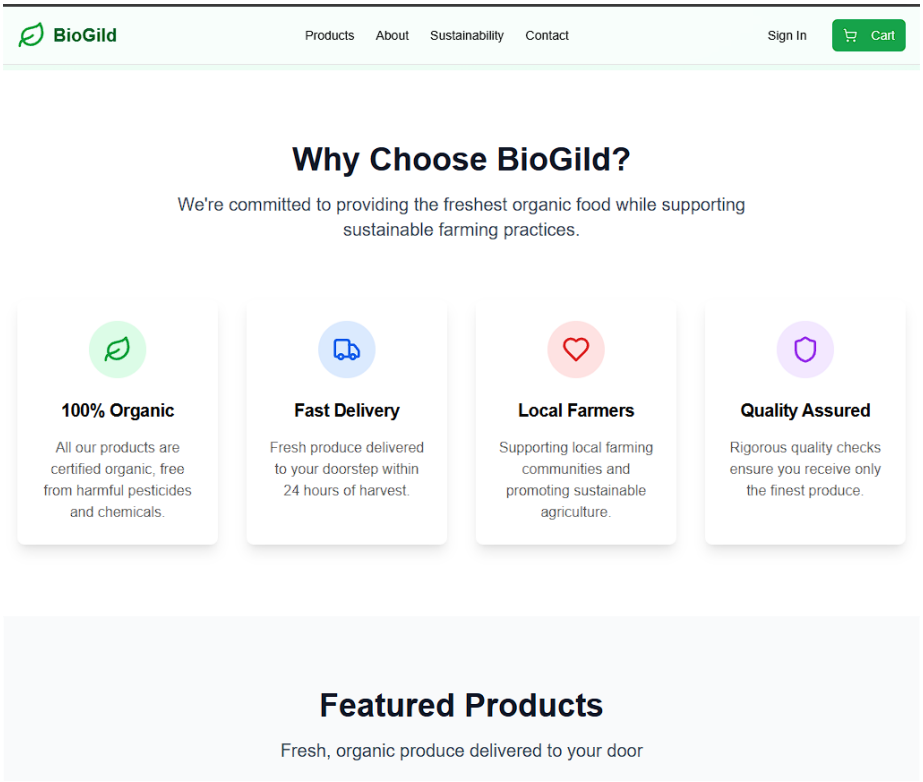


Рисунок 2.17– Головна сторінка інтернет-магазину

Щоб переглянути усі продукти, слід скористатись навігацією у хедері сайту та перейти на сторінку всіх продуктів, яку показано на рисунках 2.18-2.29.

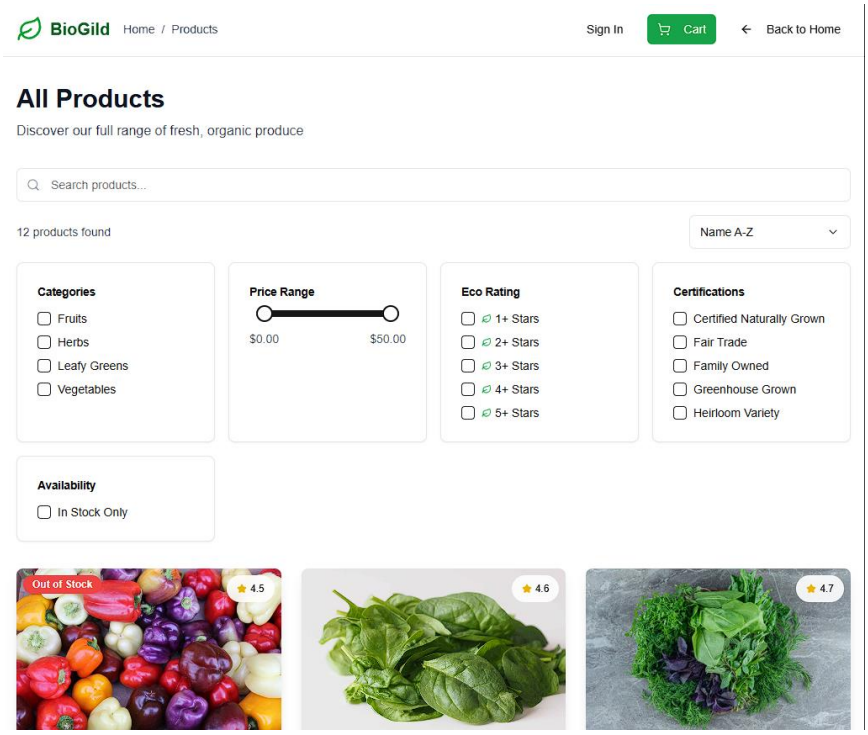


Рисунок 2.18 – Сторінка перегляду продуктів інтернет-магазину

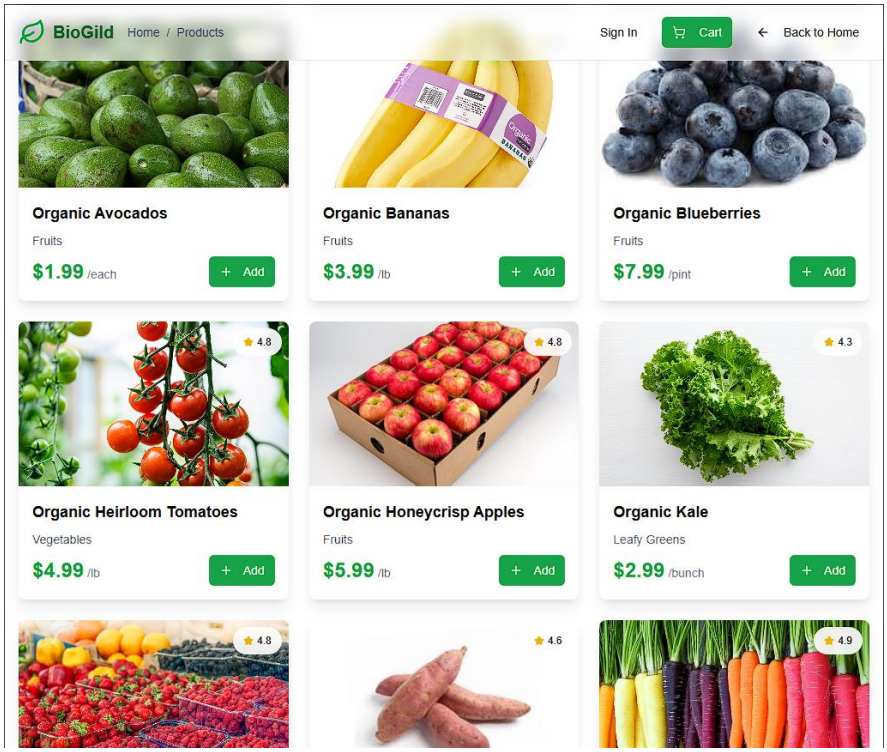


Рисунок 2.19 – Сторінка перегляду продуктів інтернет-магазину

Натискання на картку продукту перенаправляє користувача на сторінку продукту, де показано повну інформацію про продукт. Це можна побачити на рисунку 2.20.

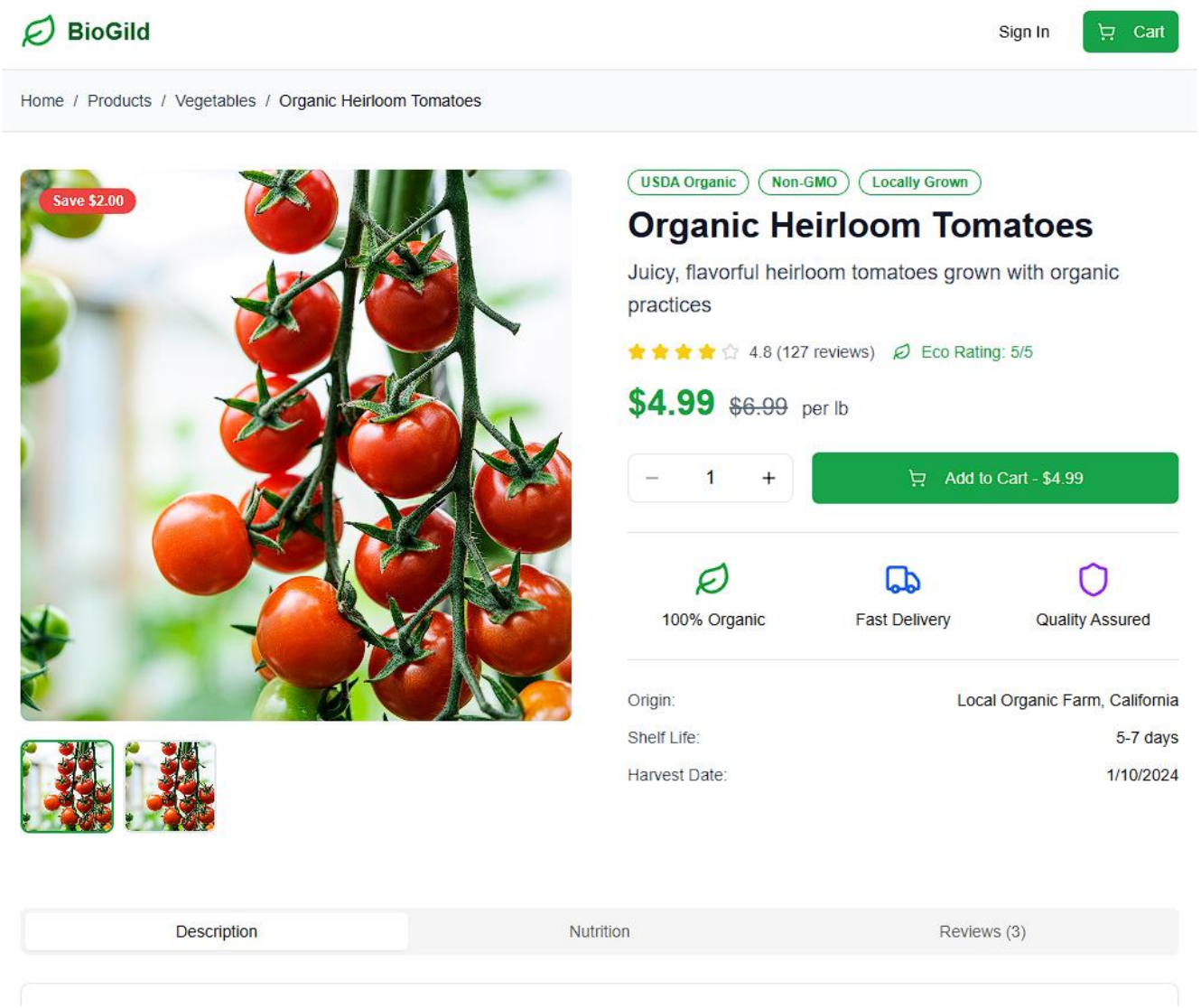


Рисунок 2.20 – Сторінка перегляду продукту інтернет-магазину

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BioGild»

У процесі реалізації проєкту ПЗ створено повнофункціональний веб-застосунок, призначений для ефективного управління онлайн-продажами екологічних продуктів. Реалізована система відповідає поставленим вимогам і забезпечує стабільну роботу всіх основних бізнес-процесів: від взаємодії з клієнтом до аналітики та адміністрування.

Функціональність інтернет-магазину включає можливість реєстрації користувачів, збереження історії замовлень, оформлення покупок із використанням кошика та промокодів, а також оцінювання товарів. Інтуїтивно зрозумівлий інтерфейс дозволяє користувачу здійснювати пошук за категоріями, цінами або екологічними параметрами, що спрощує процес вибору продукції.

Окрему увагу приділено модулю адміністрування, який дає змогу керувати асортиментом товарів, модерувати відгуки користувачів, а також формувати звіти про динаміку продажів. Реалізовано механізми контролю якості даних – усі внесення та редагування супроводжуються валідацією, що гарантує коректність і узгодженість інформації в базі даних.

Програмне забезпечення спроєктоване відповідно до сучасних принципів веб-розробки з використанням PHP-фреймворку Laravel, що забезпечує надійність, розширюваність і захист від типових вразливостей. Бекенд-сервер взаємодіє з реляційною СКБД MySQL, у якій зберігається структурована інформація про товари, користувачів, замовлення та інші сутності.

Фронтенд-частина створена із застосуванням адаптивної верстки, що гарантує коректне відображення на різних пристроях – від смартфонів до десктопів. Інтерфейс реалізовано з урахуванням рекомендацій щодо UX/UI-дизайну, що сприяє зручності використання системи для різних категорій користувачів.

Фінальне тестування веб-застосунку продемонструвало стабільну роботу усіх функціональних компонентів. Проведено модульне та інтеграційне тестування основних сценаріїв взаємодії з системою, зокрема: реєстрації, авторизації, фільтрації товарів, оформлення замовлення, адміністрування контенту. Жодних критичних помилок у процесі тестування не виявлено, що свідчить про високу якість реалізованого програмного продукту.

Розроблений інтернет-магазин «BioGild» не лише задовольняє базові вимоги до електронної комерції, але й включає спеціалізовані інструменти для просування культури екоспоживання, забезпечуючи зручність як для покупців, так і для адміністраторів.

4 ОХОРОНА ПРАЦІ

Охорона праці є ключовим елементом сучасної системи управління на підприємствах будь-якої форми власності. Вона охоплює широкий спектр заходів і дій, що мають на меті забезпечити безпечні та нешкідливі умови праці, зберегти життя, здоров'я та працездатність працівників. У контексті розвитку виробничих процесів, технічного прогресу та підвищення вимог до безпеки, охорона праці набуває особливої актуальності.

Поняття охорони праці включає не лише безпосереднє усунення небезпек на робочому місці, а й створення цілісної системи, яка передбачає аналіз, попередження та управління професійними ризиками. До складу цієї системи входять правові, технічні, організаційні та медико-профілактичні механізми, спрямовані на профілактику нещасних випадків, зниження рівня професійних захворювань та загального виробничого травматизму.

У практичній площині охорона праці реалізується через:

– розробку нормативної бази, що встановлює вимоги до безпеки умов праці;

– регулярне проведення інструктажів, навчання та атестації працівників з питань безпеки;

– оснащення робочих місць необхідними засобами захисту – як колективного, так і індивідуального;

– постійний моніторинг виробничого середовища для виявлення потенційно небезпечних факторів;

– дотримання санітарно-гігієнічних норм.

Особлива увага в системі охорони праці приділяється соціально вразливим категоріям працівників - жінкам, молоді, особам із інвалідністю, а також тим, хто працює в умовах підвищеної небезпеки. Для цих груп передбачаються додаткові гарантії, спеціальні умови праці, обмеження щодо часу роботи та забезпечення лікувально-профілактичними заходами.

Законодавче регулювання питань охорони праці в Україні здійснюється на основі Закону України "Про охорону праці", який визначає основні принципи державної політики у цій сфері. Закон встановлює відповідальність роботодавця за створення безпечних умов праці, передбачає права працівників на відмову від виконання робіт у разі виникнення загрози для їхнього життя чи здоров'я, а також закріплює механізми державного нагляду та контролю.

Інвестування в охорону праці є не лише юридичним обов'язком, а й стратегічною необхідністю для роботодавця. Безпечне виробниче середовище сприяє зниженню витрат на лікування травмованих працівників, скороченню кількості простоїв, підвищенню мотивації персоналу та якості продукції. В результаті, ефективна система охорони праці позитивно впливає як на соціальні, так і на економічні показники підприємства.

4.1 Оцінка несприятливих та ризикових чинників у офісному середовищі

Під час роботи персонал відділу стикається з низкою небезпечних та шкідливих факторів, зокрема:

1) недостатнє освітлення робочих місць;

2) підвищений рівень шуму та вібрації від роботи офісної техніки (вентилятори, принтери);

3) підвищена пожежонебезпека через велику кількість паперових носіїв;

4) ризик ураження електричним струмом через можливі замикання в мережі;

5) вплив електромагнітного випромінювання, електростатичних та магнітних полів від комп’ютерної техніки;

6) невідповідність параметрів мікроклімату (температура, вологість, рух повітря).

Освітлення робочого місця

Робота співробітників вимагає напруженого зорового сприйняття, тому правильне освітлення має вирішальне значення. На підприємстві використовується три типи освітлення:

– Природне – через вікна в денний час, що забезпечує м’яке розсіяне світло, сприятливе для очей.

– Комбіноване – застосовується в умовах недостатнього денного світла (передвечірній час, похмура погода), де додатково використовуються настільні лампи.

– Штучне – ввечері та взимку, з використанням ламп розжарювання. Нормативна освітленість у приміщенні становить 750 люкс.

Шум та вібрація

Основними джерелами шуму є вентилятори та комп’ютери. Для зниження рівня шуму застосовуються звукопоглинальні матеріали (перфоровані плити, мінеральна вата), що ефективно зменшують шум у діапазоні 31,5–8000 Гц. Рівень вібрації від роботи двох ПК та принтера не перевищує допустимих норм.

Пожежна безпека

У приміщенні є легкозаймисті предмети (меблі, папір), що збільшує ризик швидкого поширення вогню у разі виникнення пожежі.

Електробезпека

Електричний струм може спричинити теплові, хімічні та біологічні ушкодження організму, що призводить до електротравм. У відділі використовуються заземлені розетки для підключення холодильника, електрочайника, кондиціонера та касового апарату.

Електромагнітне випромінювання

Робота комп’ютерів супроводжується:

– електростатичним зарядом, що притягує пил (може викликати подразнення шкіри);

– електромагнітним випромінюванням, яке знижує продуктивність працівників.

Виміряні показники:

1) рентгенівське випромінювання (на відстані 5 см від екрану) – $7{,}74\times10^{-12}$ А/кг;

2) вміст озону – 0,1 мг/м³;

3) окиси азоту – 5 мг/м³;

4) пил – 4 мг/м³.

Мікроклімат

Температура, вологість та рух повітря впливають як на самопочуття працівників, так і на стабільність роботи техніки. У приміщенні працюють система опалення та кондиціювання, що підтримують оптимальні параметри відповідно до норм.

4.2 Профілактичні рішення для оптимізації умов праці за комп’ютером

Для покращення умов праці та зниження ризиків у робочому приміщенні необхідно впровадити такі заходи:

Оптимізація освітлення

1) Використовувати світильники типу УПМ, кожен з яких обладнаний двома лампами.

2) Розмістити світильники у два паралельні ряди (по три в кожному) та один додатковий – перпендикулярно до них у центрі приміщення.

3) Це забезпечить рівномірне освітлення робочих місць та знизить навантаження на зір

Зниження рівня шуму

1) Оптимізувати розташування джерел шуму (вентиляторів, принтерів) у приміщенні.

2) Використовувати сучасне малошумне обладнання.

3) Застосовувати додаткову звукоізоляцію (звукопоглинальні плити, мінеральну вату).

4) Впровадити раціональний режим праці та відпочинку для зменшення втоми від шуму.

Пожежна безпека

1) Скоротити обсяг паперового документообігу через повну або часткову автоматизацію облікових процесів.

2) Зберігати документи в електронному вигляді, що зменшить кількість легкозаймистих матеріалів.

Електробезпека

1) Підвищити технічну грамотність працівників щодо правил експлуатації електроприладів.

2) Чітко дотримуватися Правил технічної експлуатації та техніки безпеки.

3) Проводити регулярні інструктажі та навчання для запобігання помилкам.

Захист від електромагнітного випромінювання

1) Встановити захисні екрани на монітори для зменшення впливу випромінювання.

2) Вибирати екрани з оптимальними конструкціями та матеріалами для максимальної ефективності.

Покращення якості повітря

1) Обладнати приміщення системою кондиціювання для регулювання температури, вологості та очищення повітря.

2) Це допоможе знизити концентрацію пилу, озону та інших шкідливих речовин.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи вирішено практичну задачу інженерії програмного забезпечення, що полягала у розробці програмного забезпечення інтернет-магазину екологічних продуктів «BioGild», яке відповідає поставленим вимогам і реалізує основні функції електронної комерції з урахуванням специфіки екотоварів.

В рамках роботи виконано аналіз предметної галузі, досліджено сучасні платформи для електронної торгівлі (Shopify, WooCommerce, Shopware) та обґрунтовано доцільність створення окремого веб-застосунку, орієнтованого на екологічно свідомого споживача. Визначено функціональні та нефункціональні вимоги до програмного продукту, сформовано технічне завдання.

Проведено проєктування архітектури програмного забезпечення, включаючи створення бази даних, реалізацію моделей, контролерів і шаблонів користувацького інтерфейсу. Впроваджено механізми реєстрації, автентифікації, пошуку та фільтрації товарів, керування кошиком, оформлення замовлень, додавання відгуків, застосування промокодів, а також адміністративний функціонал для керування контентом і звітністю.

Реалізацію програмного забезпечення виконано із застосуванням стеку технологій PHP 8.2 / Laravel 12, MySQL 8.0 та веб-сервера Apache, що забезпечило належний рівень продуктивності, стабільності та безпеки системи.

Після завершення етапу розробки проведено тестування веб-застосунку, що підтвердило коректність реалізованих функцій. Усі модулі функціонують згідно з технічним завданням, помилок критичного рівня не виявлено. Інтерфейс відповідає вимогам до зручності використання.

Таким чином, поставлені завдання виконано в повному обсязі, а розроблене програмне забезпечення буде використано як основа для подальшого впровадження інтернет-магазину екологічних продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a LAMP Stack? - LAMP Stack Explained URL: https://aws.amazon.com/what-is/lamp-stack/ (дата звернення: 17.03.2025).

2. Shopify. URL: https://www.shopify.com/ (дата звернення: 17.03.2025).

3. WooCommerce. URL: https://woocommerce.com/ (дата звернення: 17.03.2025).

4. What is WordPress? Explained for beginners. URL: https://kinsta.com/knowledgebase/what-is-wordpress/ (дата звернення: 17.03.2025).

5. WooCommerce. URL: https://wordpress.org/plugins/woocommerce/ (дата звернення: 17.03.2025).

6. The ecommerce platform to power your online business. Shopware. URL: https://www.shopware.com/en/ (дата звернення: 17.03.2025).

7. Jacobson, I., Booch, G., & Rumbaugh, J. The unified software development process. Addison-Wesley, 1999. 463p.

8. Booch, G., Rumbaugh, J., & Jacobson, I. (2017). Unified Modeling Language User Guide, the (2nd Edition). Addison-Wesley Professional, 2005. 475p.

9. Data Modeling: Conceptual vs Logical vs Physical Data Model. URL: https://online.visual-paradigm.com/knowledge/visual-modeling/conceptual-vs-logical-vs-physical-data-model#:~:text=Conceptual%20ERD%20models%20the%20business,entities%20exist%2C%20NOT%20which%20tables (дата звернення: 17.03.2025).

10. Laravel – the PHP framework for web artisans. URL: https://laravel.com/. (дата звернення: 17.03.2025).

11. Kaplarevic, V. What is LAMP? URL: https://phoenixnap.com/kb/what-is-a-lamp-stack (дата звернення: 17.03.2025).

12. Why use MySQL for database management? URL: https://www.ropstam.com/why-use-mysql-advantages/ (дата звернення: 17.03.2025).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BioGild»

Вступ

Найменування програми: «BioGild».

1 Підстави для розробки

Підставою для розробки є наказ на затвердження тем кваліфікаційних робіт бакалаврів.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням програми є надання користувачу можливість купувати екологічні товари в мережі інтернет.

2.2 Експлуатаційне призначення

Програмне забезпечення буде працювати на персональних комп’ютерах (ноутбуках) та мобільних пристроях, дозволяючи користувачам створювати, переглядати та купувати екологічні продукти в інтернет-магазине екологічних продуктів «BioGild».

3 Вимоги до програми або програмного продукту

3.1 Вимоги для функціональних характеристик

3.1.1 Вимоги до переліку виконуваних функцій

Програмне забезпечення виконуватиме наступні функції:

– Реєстрацію, авторизацію, особистий кабінет з історією замовлень та налаштуваннями профілю, а також реалізувати розподіл ролей (покупці та адміністратори).

– Додавати та редагувати товари через адмін-панель, фільтрувати їх за категоріями, ціною, еко-рейтингом чи сертифікатами, шукати за назвою, а також відображати детальні картки товарів з фотографіями та відгуками.

– Додавати товари до кошика, бачити підсумкову вартість з урахуванням знижок, обирати спосіб оплати.

– Відстежувати статус замовлення (у обробці, доставка, отримано), надсилати сповіщення на email, а також обробляти повернення чи скасування замовлень.

– Залишати оцінки (1–5 зірок) та коментарі, які проходять модерацію, а система формує рейтинг «Найкращі екотовари» на основі відгуків.

– Підтримувати промокоди, а також email-розсилки про новинки та спецпропозиції.

– Реалізувати збір статистики щодо продажів (популярні товари, сезонність).

– Створювати звіти по продажам.

3.1.2 Вимоги для організації вхідних та вихідних даних

Вхідними даними для товару назва, опис, категорія, ціна, ціна зі знижкою, екологічний рейтинг товару, походження, сертифікат, оцінка товару (1-5 зірок, залишається користувачами), період зберігання, одиниці виміру, кількість одиниць та зображення у форматі .png, .jpeg (jpg), .webp. Текстова інформація (назва, опис, ціна, екологічний рейтинг товару, сертифікат) заповнюватиметься з клавіатури. Ціна є додатнім дійсним числом більше одиниці. Оцінка товару є натуральним числом від 1 до 5, оцінка ставиться при натисканні на бажану кількість зірок на картці товару. Оцінити товар може лише користувач, який придбав його. Зображення додаються з комп’ютеру серед доступних у пам’яті.

Вихідними даними про товар є запис у базі даних системи. Користувачі та адміністратори веб-застосунку бачитимуть товар на сторінках веб-застосунку у вигляді картки товару. Адміністратори в адмін-частині матимуть можливість створювати нові товари та вносити зміни до існуючих товару. Звичайні користувачі можуть переглядати товари лише в частині веб-застосунку, що призначена для користувачів.

Користувачі можуть переглядати товари та шукати їх за допомогою відповідного поля на сторінці веб-застосунку, поле пошуку виконує текстовий пошук по назві товару. Користувачі сайту можуть сортувати товари за обраним критерієм (ціною, еко-рейтингом, оцінкою, сертифікатом)

Для полегшення оформлення замовлень веб-застосунок надає можливість додавати товари до кошика, в якому вони будуть зберігатись до тих пір, поки користувач не видалить їх, або не оформить замовлення з цими товарами. Кошик користувача містить перелік товарів з загальною вартістю.

Вхідними даними для формування замовлення є дані про товари, що знаходяться у кошику на момент оформлення замовлення та даних про користувача, який оформлює замовлення, промокод (за наявності). При оформленні замовлення товари видаляються з кошику. Після оформлення замовлення зареєстровані користувачі можуть перейти в особистий кабінет та переглянути перелік своїх замовлень.

Вихідними даними про замовлення є запис у базі даних. При перегляді замовлення користувач бачить перелік товарів, які до нього входять, вартість замовлення, його статус (в обробці, доставка, отримано), знижку за застосованим промокодом (за наявності). Адміністратори бачать перелік усіх замовлень інтернет-магазину в його адмін-частині в такому ж вигляді й можуть редагувати інформацію про замовлення. За потреби користувач може відмінити замовлення.

На сторінці товару користувачі можуть залишати відгуки про товари. В адмін-частині веб застосунку адміністратори можуть переглядати відгуки користувачів. Вхідними даними про відгук є дані про користувача, що залишив відгук, заголовок

відгуку, текст відгуку, оцінка товару. Вихідними даними про відгук є запис у базі даних системи.

Розроблюваний веб застосунок дозволятиме застосувати промокоди. Вхідними даними для створення промокодів буде: код, тип промокоду (сума або відсоток), числове значення знижки (дійсне додатне число), період дії, мінімальна вартість замовлення для застосування, максимальне значення знижки, кількість застосувань. Вихідними даними про промокод є запис у базі даних системи. Перелік доступних промокодів користувач може побачити на сторінці профілю користувача. Адміністратори можуть переглянути всі промокоди на відповідній сторінці адмін-частини веб-застосунку.

Для отримання структурованої інформації щодо роботи інтернет-магазину адміністратор може створювати звіти по продажам за період часу. Групування продажів відбуватиметься по дням. Звіт буде містити наступну інформацію:

1) Період звітності.

2) Дохід за цей час.

3) Назву, ціну, кількість та вартість товарів

3.2 Вимоги до надійності

Надійна (стабільна) робота програмного забезпечення має бути забезпечена комплексом організаційно-технічних заходів, включаючи валідацію вхідних даних відповідно до вимог, а також виведення повідомлень про виявлення помилкових даних із підказкою щодо введення коректних.

3.3 Умови експлуатації

Кліматичні умови експлуатації, за яких мають забезпечуватися задані характеристики, повинні відповідати вимогам, встановленим для технічних засобів щодо умов їх використання.

3.4 Вимоги до технічної та програмної сумісності

Серверна частина та її вимоги:

– ЦП – 4 ядра, 2.0 ГГц.

– Оперативна пам’ять – 4 ГБ.

– Дискова пам’ять – 128 ГБ.

– Мережа – 1 Гбіт Ethernet

Пристрій користувача має задовольняти наступні мінімальні вимоги:

– ЦП – 2 ядра, 1.5–2.0 ГГц.

– Оперативна пам’ять – 2-4 ГБ.

– Дискова пам’ять – 128 ГБ.

– Доступ до мережі інтернет

Розробку ПЗ слід використовувати з сучасним технологічним стеком: останню версію Visual Studio Code як середовище розробки, мову програмування PHP (версія 8.2) у поєднанні з фреймворком Laravel 12 для створення надійного бекенду, реляційну базу даних на основі MySQL (8.0) для ефективного зберігання та управління даними, а також веб-сервер Apache (Apache HTTP Server 2.4.57) для стабільної роботи готового рішення. Така комбінація інструментів забезпечує оптимальний баланс між продуктивністю, стабільністю та сучасними підходами до веб-розробки.

3.5 Вимоги до захисту інформації та програм

Не висуваються

3.6 Вимоги до маркування та упаковки

Не висуваються.

3.7 Вимоги до транспортування та зберігання

Не висуваються.

4 Вимоги до програмної документації

До складу програмної документації входить:

1) Технічне завдання.

2) Програму та методику випробувань.

3) Інструкцію користувача.

4) Опис програмного забезпечення.

5) Текст програми.

5 Техніко-економічні показники

Не розраховуються

6 Стадії та етапи розробки програмного забезпечення

Таблиця А.1. містить стадії виконання розробки.

Таблиця А.1 – Стадії та етапи розробки

Стадії розробки	Назва етапу	Термін виконання
Технічне завдання	Розробка технічного завдання	01.02.2025
	Затвердження технічного завдання	14.02.2025
Ескізний проєкт	Розробка ескізного проєкту	15.02.2025
	Затвердження ескізного проєкту	02.03.2025
Технічний проєкт	Розробка технічного проєкту	03.03.2025
	Затвердження технічного проєкту	28.03.2025
Робочий проєкт	Розробка програми	29.03.2025
	Розробка програмної документації	14.05.2025
	Випробування програми	02.06.2025

7 Порядок контролю та прийомки

Прийомо-здавальні роботи мають виконуватись під контролем Замовника або його уповноваженого представника.

Випробування при прийманні-здачі проводяться відповідно до документа «Програма та методика випробувань», який затверджується обома сторонами – розробником та замовником.

Процес проведення прийомо-здавальних робіт фіксується Замовником і Виконавцем у «Протоколі проведення випробувань».

На підставі зазначеного Протоколу Виконавець та Замовник підписують «Акт прийому-здачі програми в експлуатацію».

ДОДАТОК Б – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BioGild»

1 Об’єкт випробувань

В якості об’єкту дослідження виступає ПЗ інтернет-магазину екологічних продуктів «BioGild».

2 Мета випробувань

Випробування проводяться для перевірки функціонування ПЗ та з'ясування, чи відповідають його реальні параметри вимогам технічного завдання.

3 Вимоги до застосунка

Під час тестування здійснюється перевірка функціональних можливостей системи на відповідність вимогам, зазначеним у технічному завданні.

4 Вимоги до програмної документації

4.1 Склад програмної документації, що пропонується на випробування

До складу програмної документації входять:

1) технічне завдання;

2) програма та методика випробувань;

3) інструкція користувача;

4) опис програмного забезпечення.

4.2 Спеціальні вимоги

До програмної документації не висунуто спеціальних вимог.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувались під час випробувань:

– Lenovo IdeaPad 3 15IAU7.

– Intel Core i3-1215U, 1.2 ГГц (до 4.4 ГГц, 6 ядер / 8 потоків).

– Оперативна пам’ять: 8 ГБ DDR4.

– Накопичувач: SSD 512 ГБ.

5.2 Програмні засоби, що використовувались під час випробувань

Операційна система: Windows 10 Pro 22H2 (19045.4529).

5.3 Порядок проведення випробувань

Процедура випробувань передбачає: перевірку відповідності функціональних можливостей веб-додатку встановленим вимогам та контроль якості програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

У разі відсутності чітких вимог щодо оформлення та подання програмної документації, її перевірка здійснюється представником замовника візуально. Уповноважена особа звіряє наявність і відповідність розроблених документів переліку, зазначеному в розділі «Склад програмної документації, що пропонується на випробування».

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Функціональне тестування програми проводиться відповідно до розділу «Вимоги до функціональних характеристик» технічного завдання. Тестування вважається успішним, якщо фактичні можливості програми повністю відповідають

вимогам ТЗ. Метою випробувань є перевірка роботи ПЗ інтернет-магазину екологічних продуктів «BioGild». Результати тестування наведено в таблиці Б.1

Таблиця Б.1 – Методика випробувань

№	Дія	Очікуваний результат
1	2	3
1	**Створити обліковий запис**	Програма створює запис про нового користувача у базі даних.
2	**Ввести дані користувача** (включено в «Створити обліковий запис»)	Програма зберігає введені дані (ім'я, email, пароль тощо) під час реєстрації.
3	**Залогінитись**	Програма виконує авторизацію раніше створеного користувача..
4	**Помістити продукт у кошик**	Програма додає обраний продукт до кошика користувача.
5	**Купити продукт**	Програма оформляє замовлення, створює запис у базі даних.
6	**Переглянути промокоди** (розширення «Купити продукт»)	Програма відображає активні промокоди для застосування під час покупки.
7	**Застосувати промокод** (розширення «Купити продукт»)	Програма дозволяє застосувати промокод для отримання знижки при оформленні замовлення
8	**Опрацювати відгук**	Програма дозволяє опрацювати відгук користувача, підтвердивши його, або видаливши.
9	**Видалити відгук (**розширює «Опрацювати відгук»**)**	Програма видаляє відгук користувача про продукт у базі даних.
10	**Підтвердити відгук (**розширює «Опрацювати відгук»**)**	Програма зберігає відгук користувача про продукт у базі даних.
11	**Створити відгук**	Програма дозволяє залишити відгук на придбаний продукт.
12	**Створити промокод** (для адміністратора)	Програма додає новий промокод (з обмеженнями) до бази даних.
13	**Видалити промокод** (для адміністратора)	Програма видаляє обраний промокод з системи.
14	**Переглянути історію замовлень**	Програма показує список усіх замовлень користувача з деталями.
15	**Шукати продукти**	Програма фільтрує продукти за назвою.
16	**Переглянути продукти**	Програма відображає каталог продуктів (з можливістю сортування).
17	**Управляти продуктами** (для адміністратора)	Програма надає інтерфейс для додавання, редагування або видалення продуктів.
18	**Редагувати продукт** (включено в «Управляти продуктами»)	Програма оновлює інформацію (ціна, опис тощо) про продукт.

Кінец таблиці Б.1

1	2	3
19	**Видалити продукт** (включено в «Управляти продуктами»)	Програма видаляє продукт з системи.
20	**Додати продукт** (включено в «Управляти продуктами»)	Програма додає новий продукт до бази даних.

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BioGild»

Інструкція користувача для програмного забезпечення є невід'ємною частиною успішної експлуатації ПЗ, оскільки:

1) Спрощує роботу – надає чіткі покрокові вказівки з використання функцій, що дозволяє користувачам швидко освоїти програмний продукт.

2) Зменшує кількість помилок – пояснює правильні методи роботи, запобігаючи неправильним діям, які можуть призвести до збоїв або втрати даних.

3) Економить час підтримки – знижує навантаження на службу підтримки, оскільки користувачі можуть самостійно знаходити відповіді на типові запитання.

4) Забезпечує безпеку – містить інформацію про потенційні ризики та рекомендації щодо безпечного використання ПЗ.

5) Є частиною документації – виконує нормативні вимоги щодо наявності технічної документації на програмний продукт.

Після відкриття програмного ПЗ користувач бачить головну сторінку інтернет-магазину, яка містить коротку інформацію про магазин та короткий перелік товарів, що показано на рисунках В.1-В.3

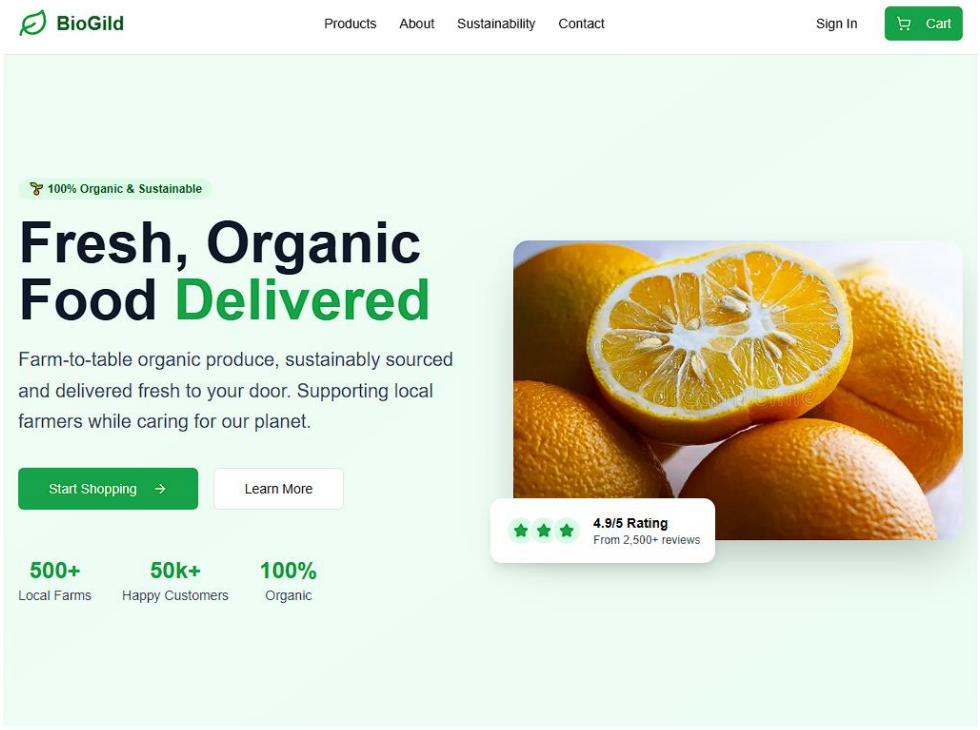


Рисунок В.1 – Головна сторінка інтернет-магазину

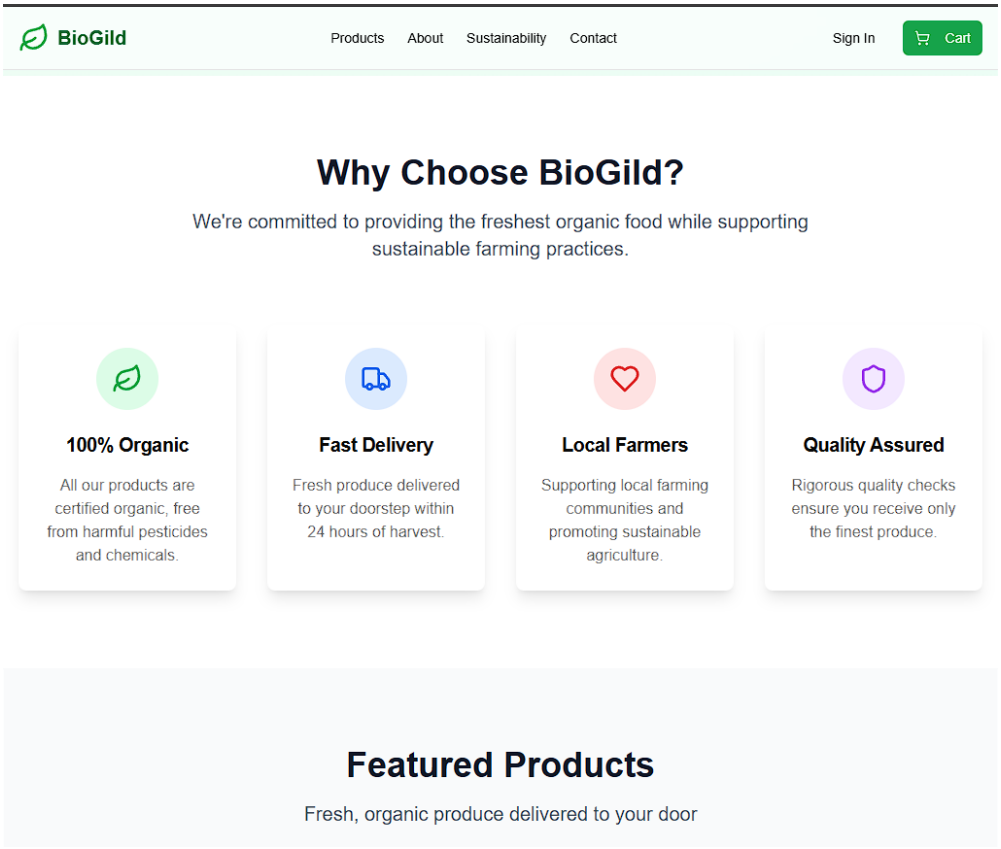


Рисунок В.2 – Головна сторінка інтернет-магазину

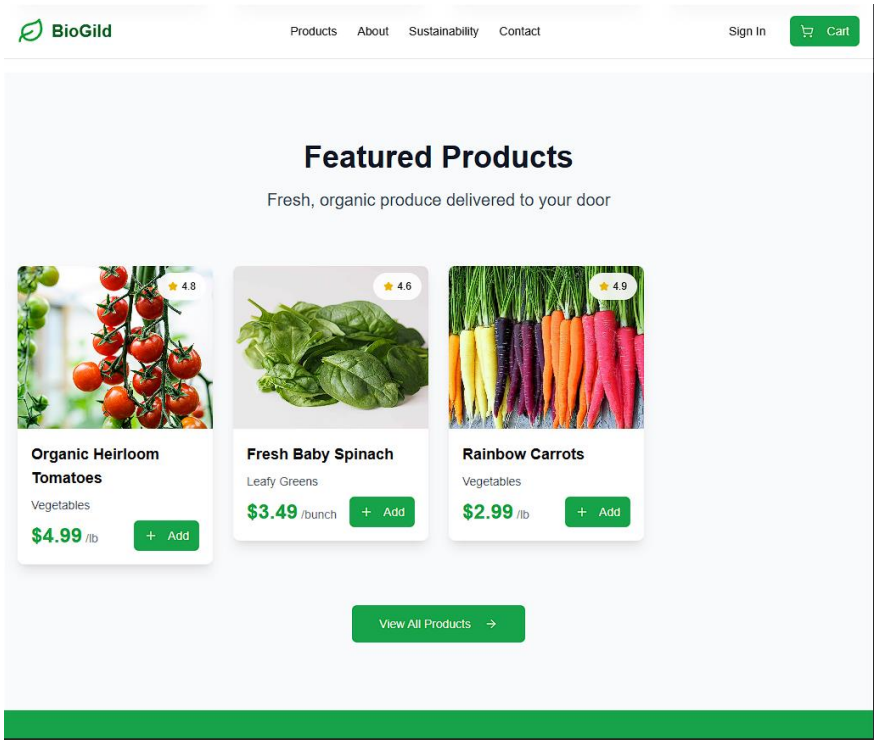


Рисунок В.3 – Головна сторінка інтернет-магазину

Щоб переглянути усі продукти, слід скористатись навігацією у хедері сайту та перейти на сторінку всіх продуктів, яку показано на рисунках В.4-В.5.

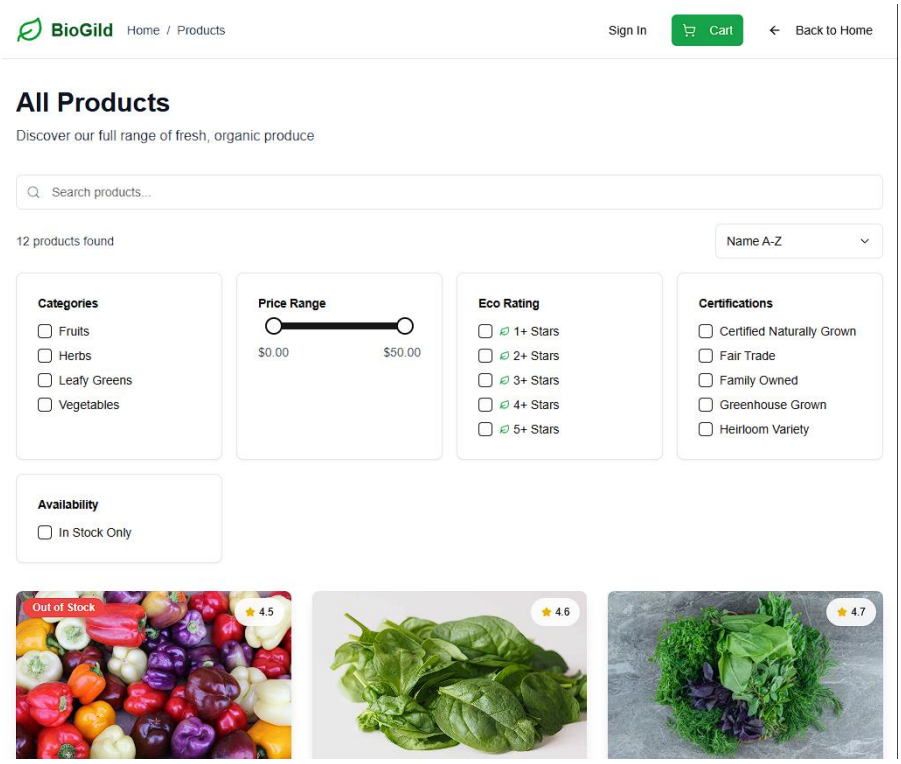


Рисунок В.4 – Сторінка перегляду продуктів інтернет-магазину

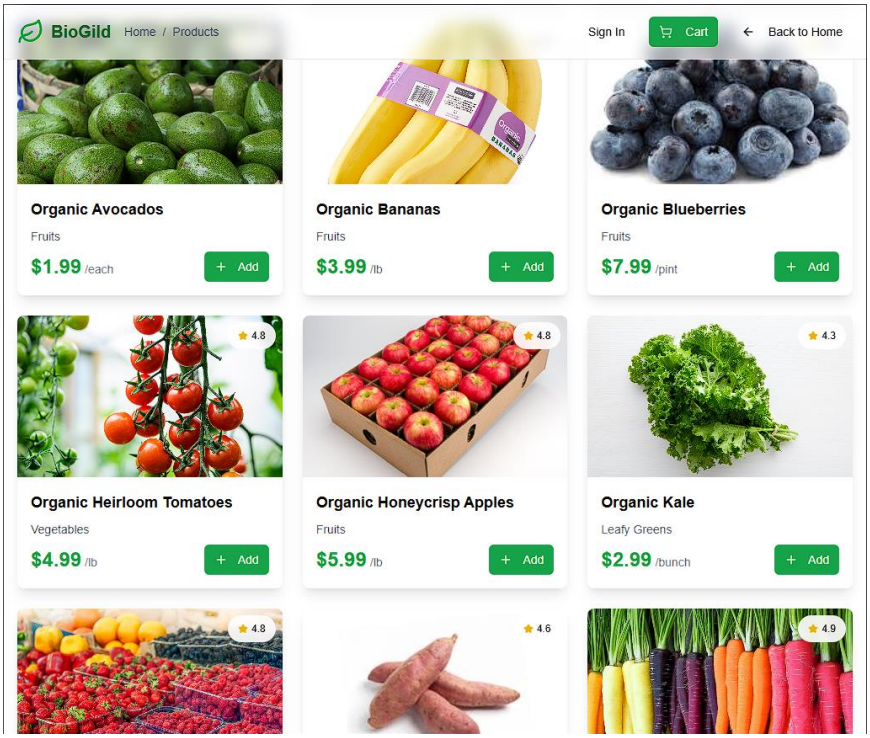


Рисунок В.5 – Сторінка перегляду продуктів інтернет-магазину

На сторінці всіх продуктів користувач може застосовувати фільтри щоб знайти лише потрібні продукти, приклад цього показано на рисунку В.6.

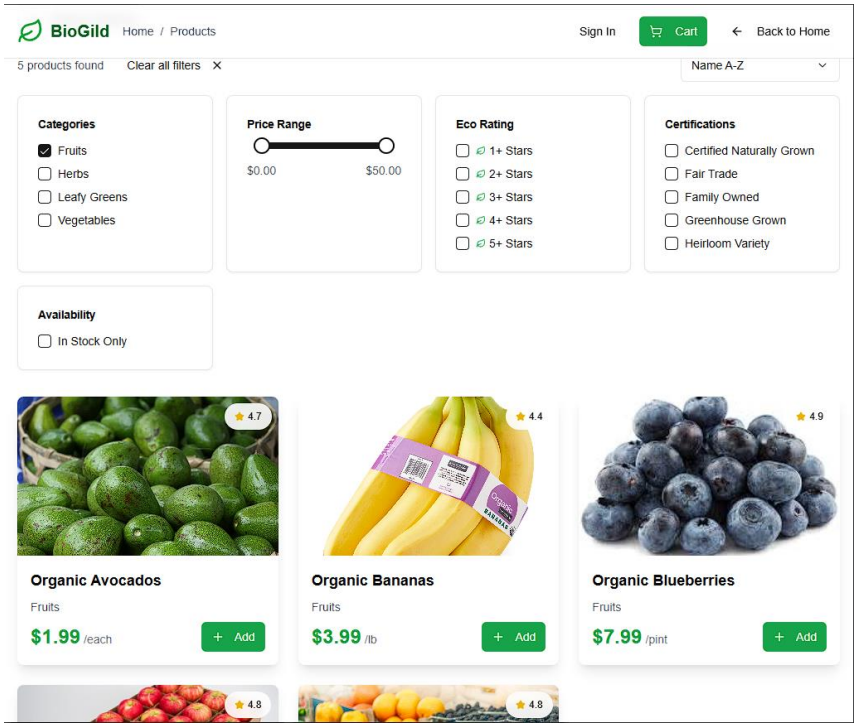


Рисунок В.6 – Фільтрація продуктів за критерієм

Також користувач може застосувати пошук продуктів по назві, що показано на рисунку В.7.

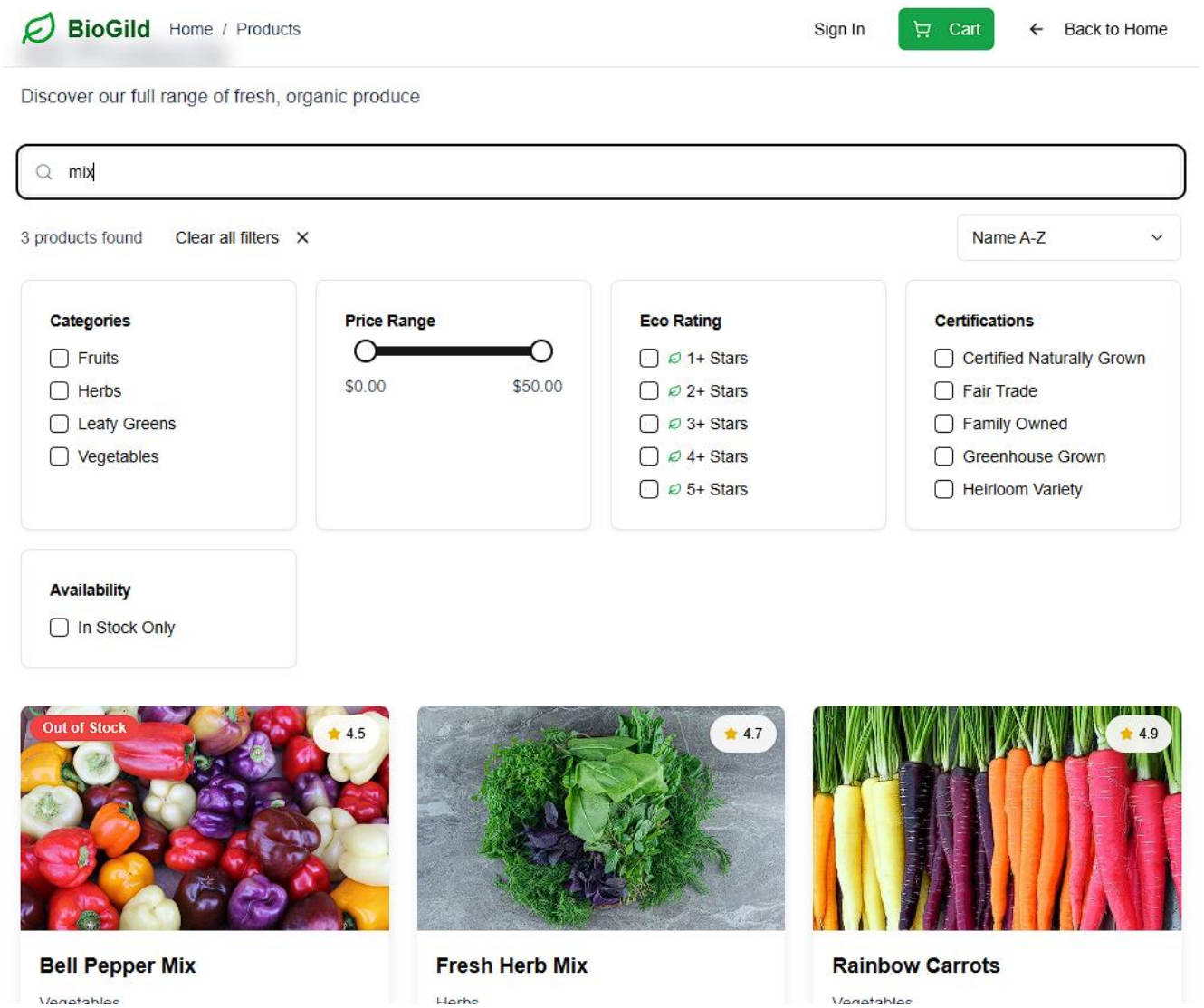


Рисунок В.7 – Текстовий пошук продуктів

Натискання на картку продукту перенаправляє користувача на сторінку продукту, до показано повну інформацію про продукт. Це можна побачити на рисунку В.8.

Рисунок В.8 – Сторінка перегляду продукту інтернет-магазину

Щоб замовити продукт, його необхідно помістити у кошик й продовжити оформлення замовлення. При оформленні замовлення необхідно вказати персональну інформацію (адресу, платіжні дані) та підтвердити замовлення. Процес оформлення замовлення показано на рисунках В.9-В.13.

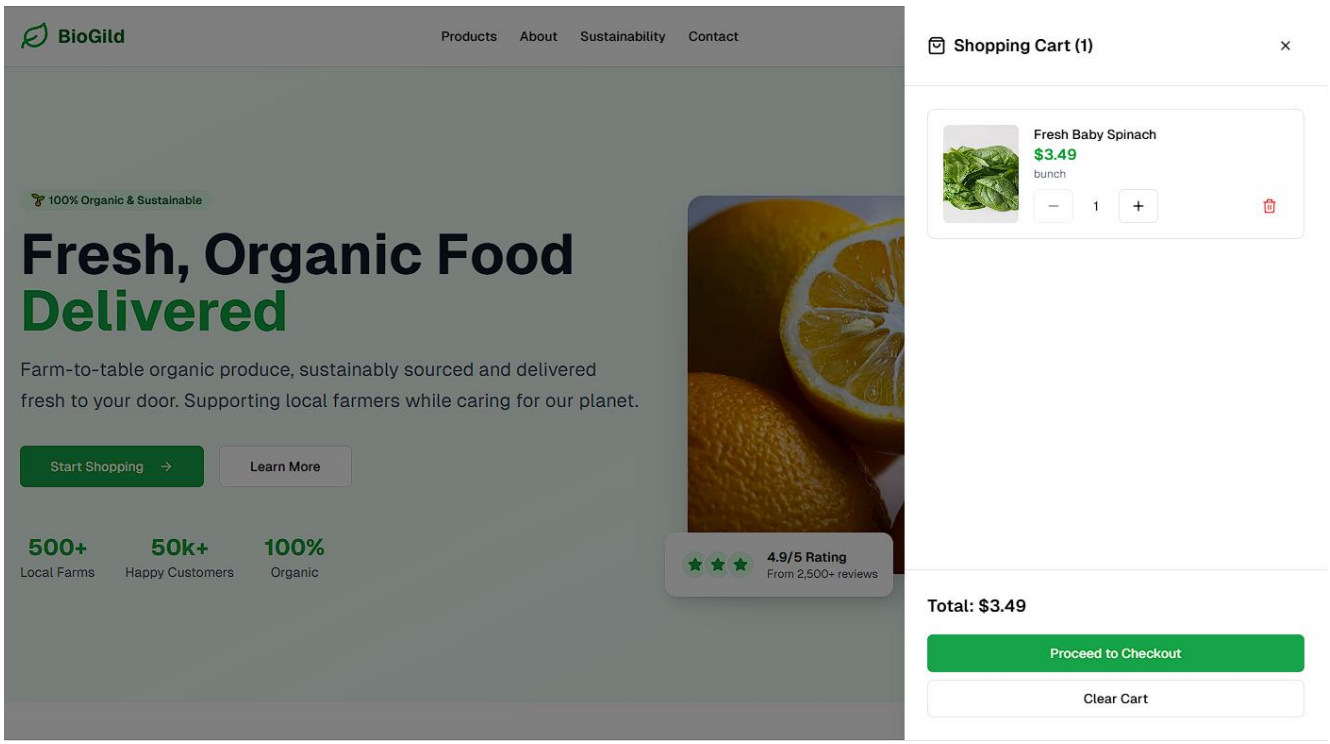


Рисунок В.9 – Процес оформлення замовлення

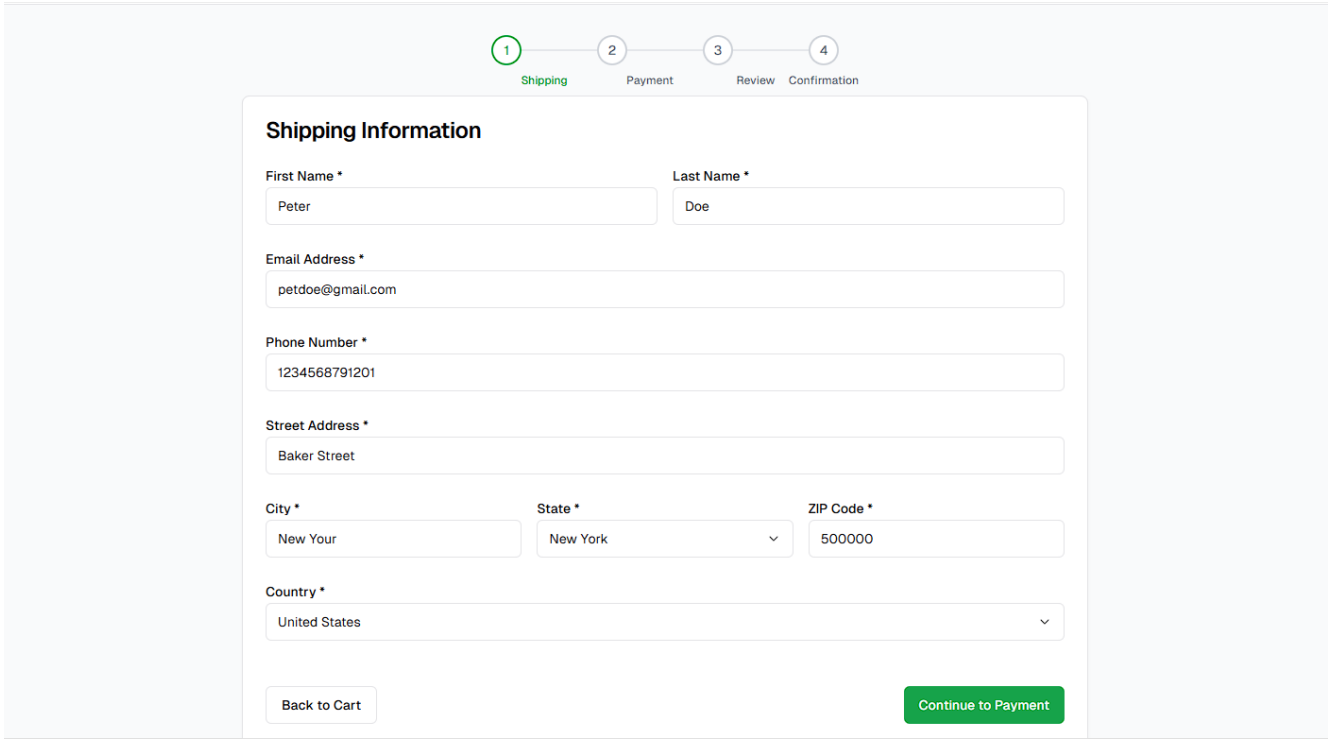


Рисунок В.10 – Процес оформлення замовлення

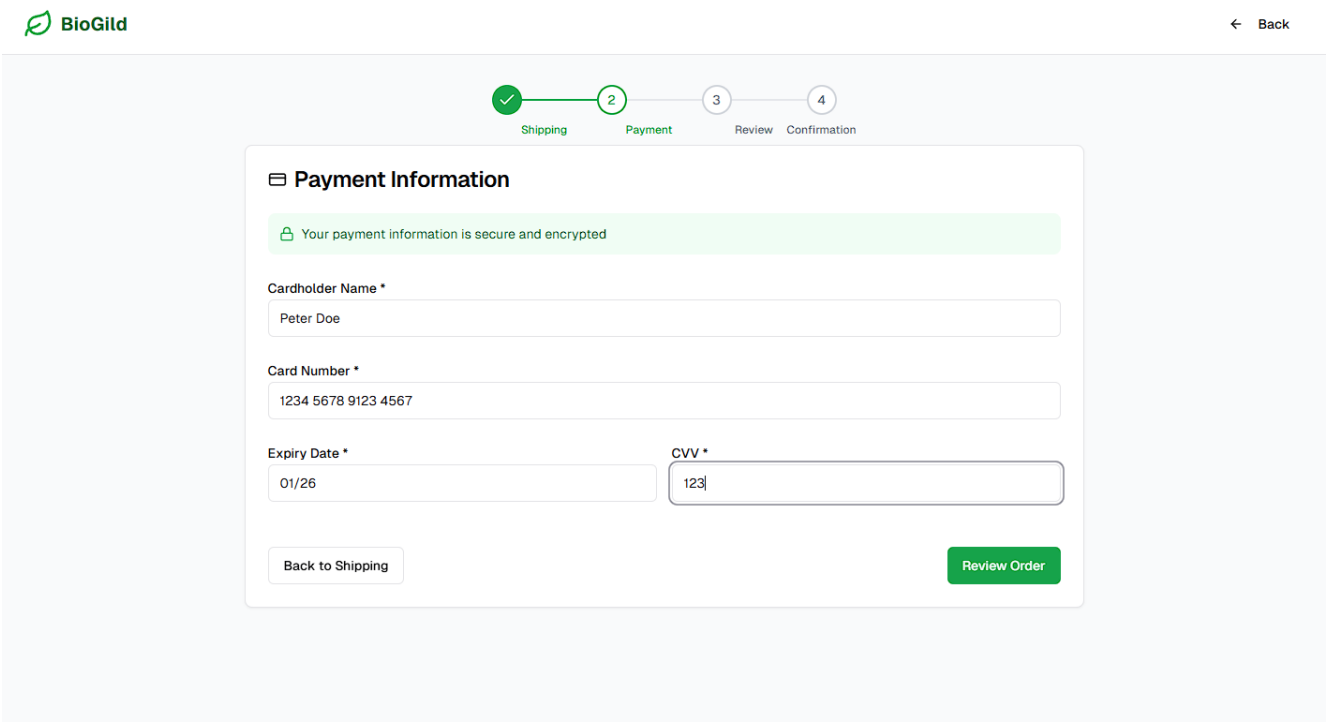


Рисунок В.11 – Процес оформлення замовлення

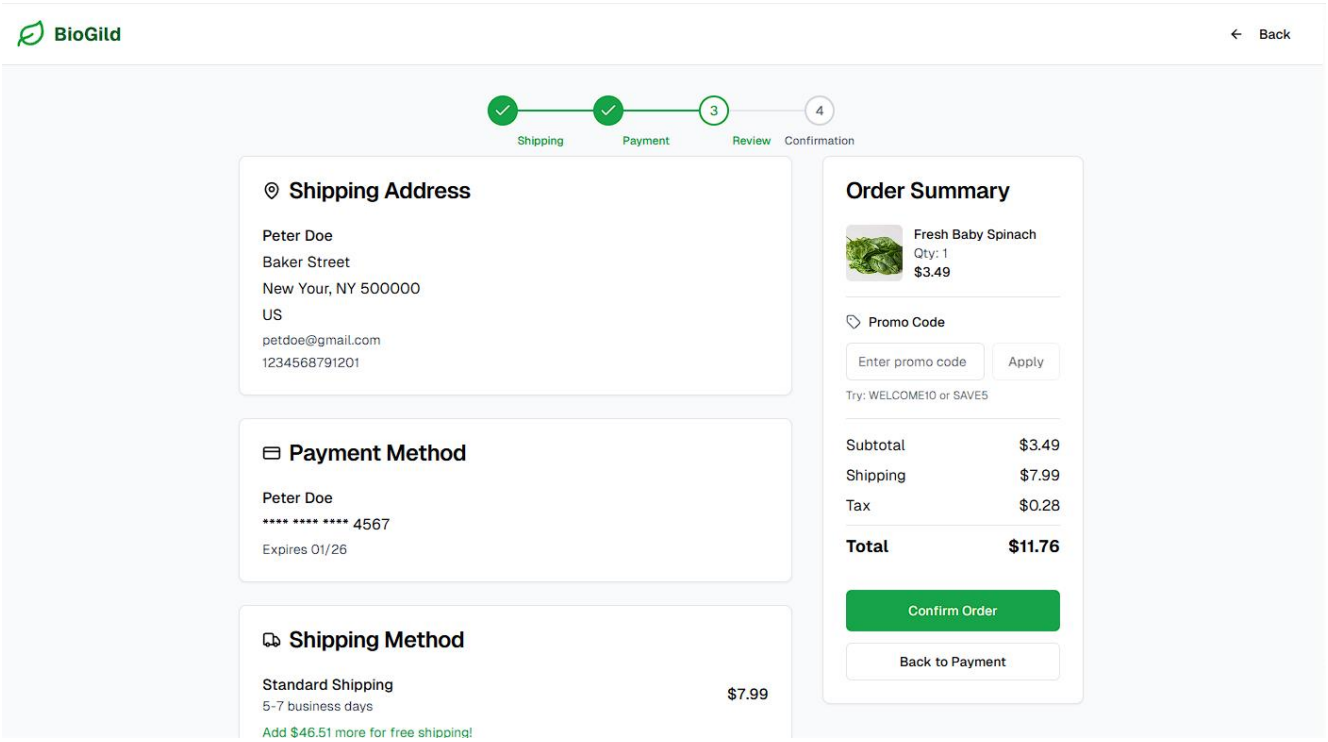


Рисунок В.12 – Процес оформлення замовлення

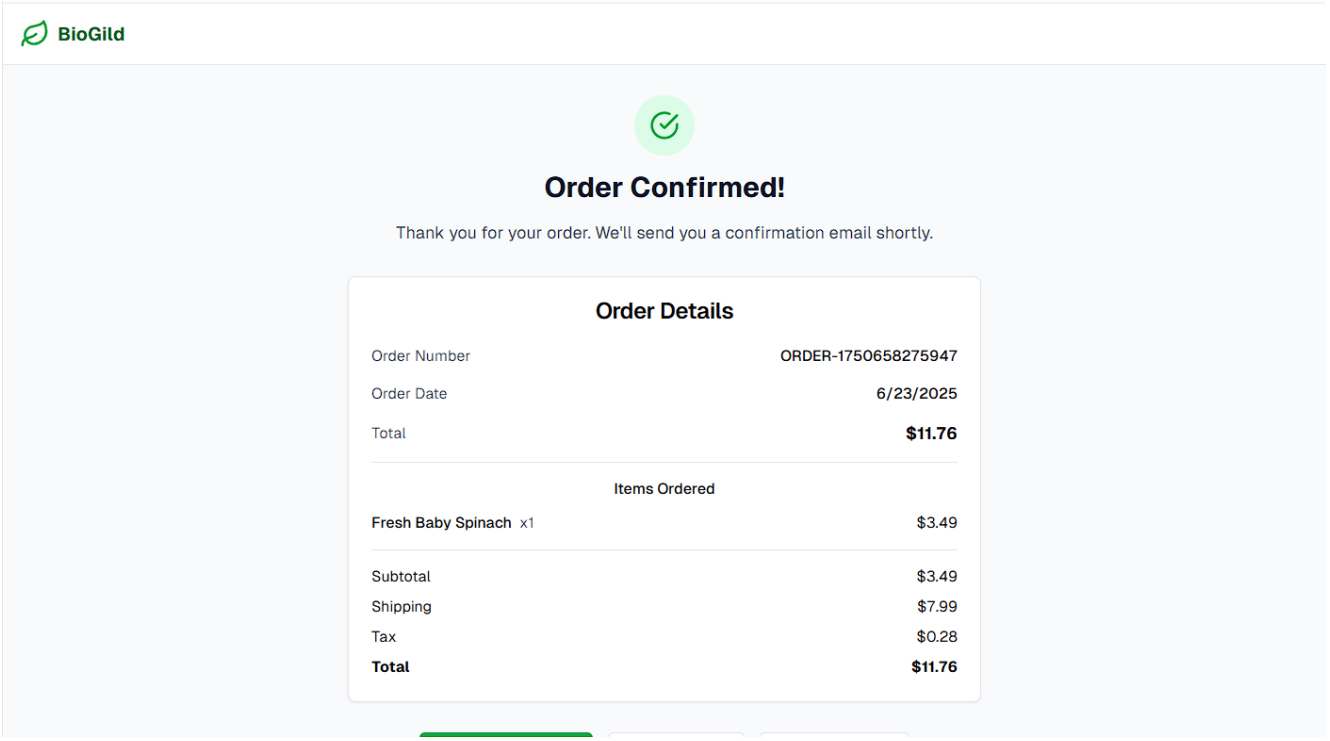


Рисунок В.13 – Процес оформлення замовлення

При авторизації клієнт інтернет-магазину побачить сторінку свого облікового запису, де буде відображено історію замовлень, що можна побачити на рисунках В.14-В.15

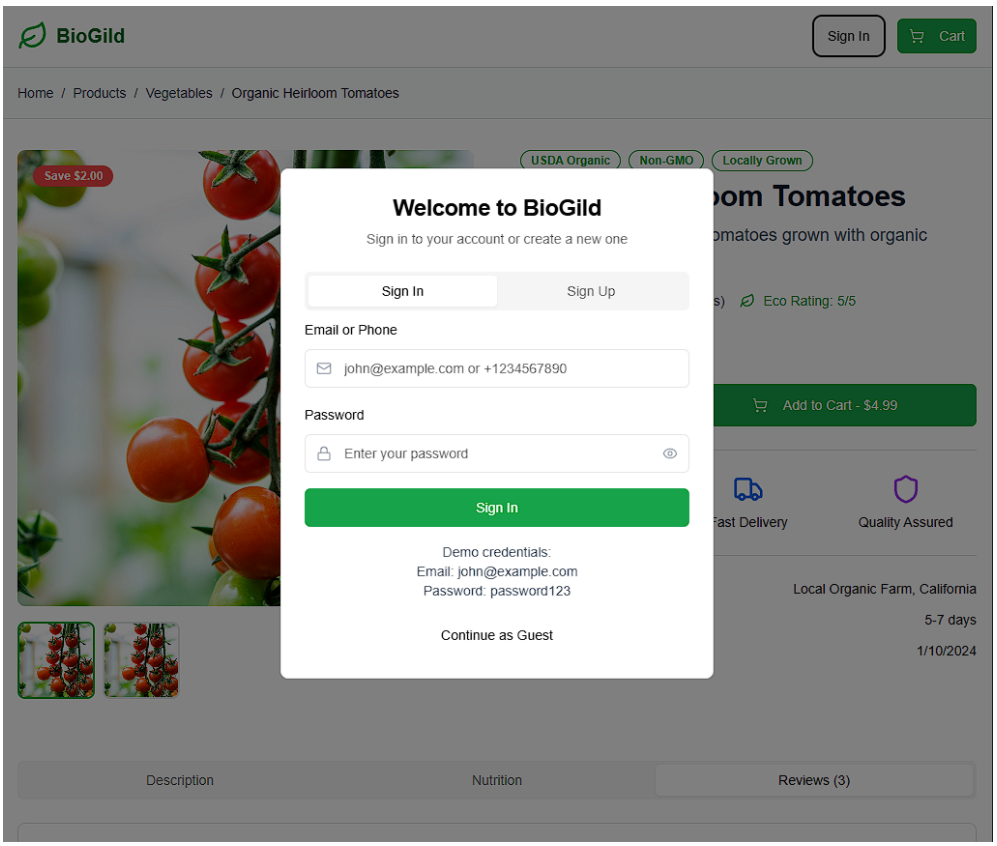


Рисунок В.14 – Авторизація користувача4

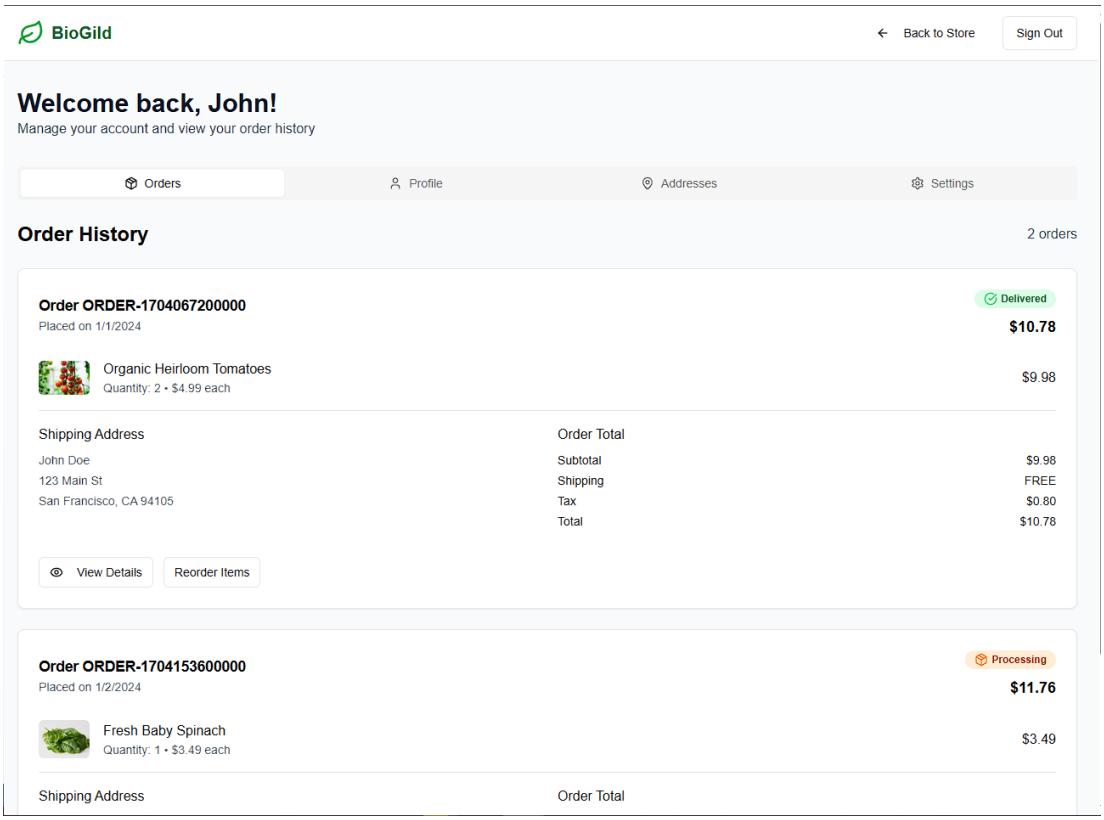


Рисунок В.15 – Перегляд замовлень у профілі користувача

ДОДАТОК Г – ТЕКСТ ПРОГРАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BioGild»

account.page.tsx

```
"use client"

import { useEffect } from "react"
import { useRouter } from "next/navigation"
import Link from "next/link"
import { ArrowLeft, Leaf, Package, User, Settings, MapPin } from "lucide-react"

import { Button } from "@/components/ui/button"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"
import { AuthProvider, useAuth } from "../../context/auth-context"
import { OrderHistory } from "../../components/order-history"
import { UserProfile } from "../../components/user-profile"
import { AddressBook } from "../../components/address-book"

function AccountContent() {
  const { state, logout } = useAuth()
  const router = useRouter()

  useEffect(() => {
    if (!state.isLoading && !state.isAuthenticated) {
      router.push("/")
    }
  }, [state.isAuthenticated, state.isLoading, router])

  if (state.isLoading) {
    return (
      <div className="min-h-screen bg-gray-50 flex items-center justify-center">
        <div className="text-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-
green-600 mx-auto mb-4"></div>
          <p>Loading your account...</p>
        </div>
      </div>
    )
  }

  if (!state.isAuthenticated || !state.user) {
    return null
  }

  const handleLogout = () => {
    logout()
    router.push("/")
```

```
  }

  return (
    <div className="min-h-screen bg-gray-50">
      {/* Header */}
      <header className="bg-white border-b">
        <div className="container flex h-16 items-center justify-between px-4">
          <Link href="/" className="flex items-center gap-2">
            <Leaf className="h-8 w-8 text-green-600" />
            <span className="text-xl font-bold text-green-800">EcoHarvest</span>
          </Link>
          <div className="flex items-center gap-4">
            <Link href="/">
              <Button variant="ghost">
                <ArrowLeft className="h-4 w-4 mr-2" />
                Back to Store
              </Button>
            </Link>
            <Button variant="outline" onClick={handleLogout}>
              Sign Out
            </Button>
          </div>
        </div>
      </header>

      <div className="container px-4 py-8">
        <div className="mb-8">
          <h1 className="text-3xl font-bold text-gray-900">Welcome back,
{state.user.firstName}!</h1>
          <p className="text-gray-600">Manage your account and view your order
history</p>
        </div>

        <Tabs defaultValue="orders" className="w-full">
          <TabsList className="grid w-full grid-cols-4">
            <TabsTrigger value="orders" className="flex items-center gap-2">
              <Package className="h-4 w-4" />
              Orders
            </TabsTrigger>
            <TabsTrigger value="profile" className="flex items-center gap-2">
              <User className="h-4 w-4" />
              Profile
            </TabsTrigger>
            <TabsTrigger value="addresses" className="flex items-center gap-2">
              <MapPin className="h-4 w-4" />
              Addresses
            </TabsTrigger>
            <TabsTrigger value="settings" className="flex items-center gap-2">
              <Settings className="h-4 w-4" />
              Settings
            </TabsTrigger>
          </TabsList>
```

```
            <TabsContent value="orders" className="mt-6">
              <OrderHistory />
            </TabsContent>

            <TabsContent value="profile" className="mt-6">
              <UserProfile />
            </TabsContent>

            <TabsContent value="addresses" className="mt-6">
              <AddressBook />
            </TabsContent>

            <TabsContent value="settings" className="mt-6">
              <Card>
                <CardHeader>
                  <CardTitle>Account Settings</CardTitle>
                </CardHeader>
                <CardContent>
                  <p className="text-gray-600">Settings panel coming soon...</p>
                </CardContent>
              </Card>
            </TabsContent>
          </Tabs>
        </div>
      </div>
    )
}

export default function AccountPage() {
  return (
    <AuthProvider>
      <AccountContent />
    </AuthProvider>
  )
}
```

admin.page.tsx

```
"use client"

import { useEffect } from "react"
import { useRouter } from "next/navigation"
import Link from "next/link"
import {
  LayoutDashboard,
  Package,
  ShoppingCart,
  Tag,
  Users,
  TrendingUp,
  DollarSign,
```

```
  Leaf,
  LogOut,
  Star,
} from "lucide-react"

import { Button } from "@/components/ui/button"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"
import { AdminProvider, useAdmin } from "../../context/admin-context"
import { AdminLogin } from "../../components/admin-login"
import { ProductManagement } from "../../components/product-management"
import { OrderManagement } from "../../components/order-management"
import { PromoCodeManagement } from "../../components/promo-code-management"
import { ReviewManagement } from "../../components/review-management"

function AdminDashboardContent() {
  const { state, adminLogout, getStats } = useAdmin()
  const router = useRouter()

  useEffect(() => {
    if (state.isAuthenticated) {
      getStats()
    }
  }, [state.isAuthenticated, getStats])

  if (state.isLoading) {
    return (
      <div className="min-h-screen bg-gray-50 flex items-center justify-center">
        <div className="text-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-
green-600 mx-auto mb-4"></div>
          <p>Loading admin dashboard...</p>
        </div>
      </div>
    )
  }

  if (!state.isAuthenticated) {
    return <AdminLogin />
  }

  const handleLogout = () => {
    adminLogout()
    router.push("/")
  }

  return (
    <div className="min-h-screen bg-gray-50">
      {/* Header */}
      <header className="bg-white border-b">
        <div className="container flex h-16 items-center justify-between px-4">
          <div className="flex items-center gap-4">
```

```
            <Link href="/" className="flex items-center gap-2">
              <Leaf className="h-8 w-8 text-green-600" />
              <span className="text-xl font-bold text-green-800">BioGild Admin</span>
            </Link>
          </div>
          <div className="flex items-center gap-4">
            <span className="text-sm text-gray-600">Welcome,
{state.adminUser?.name}</span>
            <Link href="/">
              <Button variant="ghost" size="sm">
                View Store
              </Button>
            </Link>
            <Button variant="outline" size="sm" onClick={handleLogout}>
              <LogOut className="h-4 w-4 mr-2" />
              Sign Out
            </Button>
          </div>
        </div>
      </header>

      <div className="container px-4 py-8">
        <div className="mb-8">
          <h1 className="text-3xl font-bold text-gray-900">Admin Dashboard</h1>
          <p className="text-gray-600">Manage your store, products, and orders</p>
        </div>

        <Tabs defaultValue="overview" className="w-full">
          <TabsList className="grid w-full grid-cols-5">
            <TabsTrigger value="overview" className="flex items-center gap-2">
              <LayoutDashboard className="h-4 w-4" />
              Overview
            </TabsTrigger>
            <TabsTrigger value="products" className="flex items-center gap-2">
              <Package className="h-4 w-4" />
              Products
            </TabsTrigger>
            <TabsTrigger value="orders" className="flex items-center gap-2">
              <ShoppingCart className="h-4 w-4" />
              Orders
            </TabsTrigger>
            <TabsTrigger value="reviews" className="flex items-center gap-2">
              <Star className="h-4 w-4" />
              Reviews
            </TabsTrigger>
            <TabsTrigger value="promos" className="flex items-center gap-2">
              <Tag className="h-4 w-4" />
              Promo Codes
            </TabsTrigger>
          </TabsList>

          <TabsContent value="overview" className="mt-6">
```

```
            <div className="grid md:grid-cols-2 lg:grid-cols-4 gap-6 mb-8">
              <Card>
                <CardHeader className="flex flex-row items-center justify-between
space-y-0 pb-2">
                  <CardTitle className="text-sm font-medium">Total
Revenue</CardTitle>
                  <DollarSign className="h-4 w-4 text-muted-foreground" />
                </CardHeader>
                <CardContent>
                  <div className="text-2xl font-
bold">${state.stats?.totalRevenue.toFixed(2) || "0.00"}</div>
                  <p className="text-xs text-muted-foreground">+20.1% from last
month</p>
                </CardContent>
              </Card>
              <Card>
                <CardHeader className="flex flex-row items-center justify-between
space-y-0 pb-2">
                  <CardTitle className="text-sm font-medium">Total Orders</CardTitle>
                  <ShoppingCart className="h-4 w-4 text-muted-foreground" />
                </CardHeader>
                <CardContent>
                  <div className="text-2xl font-bold">{state.stats?.totalOrders ||
0}</div>
                  <p className="text-xs text-muted-foreground">+12% from last
month</p>
                </CardContent>
              </Card>
              <Card>
                <CardHeader className="flex flex-row items-center justify-between
space-y-0 pb-2">
                  <CardTitle className="text-sm font-medium">Total
Products</CardTitle>
                  <Package className="h-4 w-4 text-muted-foreground" />
                </CardHeader>
                <CardContent>
                  <div className="text-2xl font-bold">{state.stats?.totalProducts ||
0}</div>
                  <p className="text-xs text-muted-foreground">+3 new this week</p>
                </CardContent>
              </Card>
              <Card>
                <CardHeader className="flex flex-row items-center justify-between
space-y-0 pb-2">
                  <CardTitle className="text-sm font-medium">Total
Customers</CardTitle>
                  <Users className="h-4 w-4 text-muted-foreground" />
                </CardHeader>
                <CardContent>
                  <div className="text-2xl font-bold">{state.stats?.totalCustomers ||
0}</div>
```

```
                <p className="text-xs text-muted-foreground">+8% from last
month</p>
              </CardContent>
            </Card>
          </div>

          <div className="grid md:grid-cols-2 gap-6">
            <Card>
              <CardHeader>
                <CardTitle>Recent Activity</CardTitle>
              </CardHeader>
              <CardContent>
                <div className="space-y-4">
                  <div className="flex items-center gap-3">
                    <div className="w-2 h-2 bg-green-500 rounded-full"></div>
                    <span className="text-sm">New order #ORDER-1704153600000</span>
                  </div>
                  <div className="flex items-center gap-3">
                    <div className="w-2 h-2 bg-blue-500 rounded-full"></div>
                    <span className="text-sm">Product "Organic Kale" updated</span>
                  </div>
                  <div className="flex items-center gap-3">
                    <div className="w-2 h-2 bg-purple-500 rounded-full"></div>
                    <span className="text-sm">New promo code "SAVE5" created</span>
                  </div>
                </div>
              </CardContent>
            </Card>

            <Card>
              <CardHeader>
                <CardTitle>Quick Actions</CardTitle>
              </CardHeader>
              <CardContent className="space-y-3">
                <Button className="w-full justify-start" variant="outline">
                  <Package className="h-4 w-4 mr-2" />
                  Add New Product
                </Button>
                <Button className="w-full justify-start" variant="outline">
                  <Tag className="h-4 w-4 mr-2" />
                  Create Promo Code
                </Button>
                <Button className="w-full justify-start" variant="outline">
                  <TrendingUp className="h-4 w-4 mr-2" />
                  View Analytics
                </Button>
              </CardContent>
            </Card>
          </div>
        </TabsContent>

        <TabsContent value="products" className="mt-6">
```

```
            <ProductManagement />
          </TabsContent>

          <TabsContent value="orders" className="mt-6">
            <OrderManagement />
          </TabsContent>

          <TabsContent value="reviews" className="mt-6">
            <ReviewManagement />
          </TabsContent>

          <TabsContent value="promos" className="mt-6">
            <PromoCodeManagement />
          </TabsContent>
        </Tabs>
      </div>
    </div>
  )
}

export default function AdminDashboard() {
  return (
    <AdminProvider>
      <AdminDashboardContent />
    </AdminProvider>
  )
}
```

checkout.page.tsx

```
"use client"

import { useEffect } from "react"
import { useRouter } from "next/navigation"
import Link from "next/link"
import { ArrowLeft, Leaf } from "lucide-react"

import { Button } from "@/components/ui/button"
import { CheckoutProgress } from "../../components/checkout-progress"
import { CheckoutShipping } from "../../components/checkout-shipping"
import { CheckoutPayment } from "../../components/checkout-payment"
import { CheckoutReview } from "../../components/checkout-review"
import { OrderConfirmation } from "../../components/order-confirmation"
import { CartProvider, useCart } from "../../context/cart-context"
import { CheckoutProvider, useCheckout } from "../../context/checkout-context"

const steps = [
  { id: 1, name: "Shipping" },
  { id: 2, name: "Payment" },
  { id: 3, name: "Review" },
  { id: 4, name: "Confirmation" },
]
```

```
function CheckoutContent() {
  const { items, clearCart } = useCart()
  const { state, setCurrentStep, processOrder, clearCheckout } = useCheckout()
  const router = useRouter()

  const handleNext = () => {
    setCurrentStep(state.currentStep + 1)
  }

  const handleBack = () => {
    if (state.currentStep === 1) {
      router.push("/")
    } else {
      setCurrentStep(state.currentStep - 1)
    }
  }

  const handleConfirmOrder = async () => {
    try {
      await processOrder(items)
      clearCart()
      setCurrentStep(4)
    } catch (error) {
      console.error("Order processing failed:", error)
    }
  }

  const handleStartNewOrder = () => {
    clearCheckout()
    router.push("/products")
  }

  if (items.length === 0 && !state.order) {
    return null // Will redirect
  }

  return (
    <div className="min-h-screen bg-gray-50">
      {/* Header */}
      <header className="bg-white border-b">
        <div className="container flex h-16 items-center justify-between px-4">
          <Link href="/" className="flex items-center gap-2">
            <Leaf className="h-8 w-8 text-green-600" />
            <span className="text-xl font-bold text-green-800">EcoHarvest</span>
          </Link>
          {state.currentStep < 4 && (
            <Button variant="ghost" onClick={handleBack}>
              <ArrowLeft className="h-4 w-4 mr-2" />
              Back
            </Button>
          )}
```

```
        </div>
      </header>

      <div className="container px-4 py-8">
        {state.currentStep < 4 && <CheckoutProgress currentStep={state.currentStep}
steps={steps} />}

        <div className="max-w-4xl mx-auto">
          {state.currentStep === 1 && <CheckoutShipping onNext={handleNext}
onBack={handleBack} />}
          {state.currentStep === 2 && <CheckoutPayment onNext={handleNext}
onBack={handleBack} />}
          {state.currentStep === 3 && <CheckoutReview onBack={handleBack}
onConfirm={handleConfirmOrder} />}
          {state.currentStep === 4 && state.order && <OrderConfirmation
order={state.order} />}
        </div>

        {state.currentStep === 4 && (
          <div className="text-center mt-8">
            <Button onClick={handleStartNewOrder} className="bg-green-600 hover:bg-
green-700">
              Start New Order
            </Button>
          </div>
        )}
      </div>
    </div>
  )
}

export default function CheckoutPage() {
  return (
    <CartProvider>
      <CheckoutProvider>
        <CheckoutContent />
      </CheckoutProvider>
    </CartProvider>
  )
}
```

products.page.tsx

```
"use client"

import { useState, useEffect } from "react"
import Link from "next/link"
import { ShoppingCart, User, Leaf } from "lucide-react"
import { Button } from "@/components/ui/button"
import { ProductCard } from "../../components/product-card"
import { SearchFilters } from "../../components/search-filters"
import { LoadingSpinner } from "../../components/loading-spinner"
```

```tsx
import { useCart } from "../../context/cart-context"
import { useAuth } from "../../context/auth-context"
import { api } from "../../services/api"
import type { Product } from "../../types/cart"
import type { SearchFilters as SearchFiltersType } from "../../types/search"

export default function ProductsPage() {
  const [products, setProducts] = useState<Product[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [filters, setFilters] = useState<SearchFiltersType>({
    search: "",
    category: "all",
    priceRange: [0, 100],
    ecoRating: 1,
    sortBy: "name",
  })

  const { getTotalItems, setIsCartOpen } = useCart()
  const { state: authState } = useAuth()

  useEffect(() => {
    fetchProducts()
  }, [filters])

  const fetchProducts = async () => {
    try {
      setLoading(true)
      setError(null)
      const data = await api.getProducts(filters)

      // Apply sorting
      const sortedData = [...data]
      switch (filters.sortBy) {
        case "price-low":
          sortedData.sort((a, b) => a.price - b.price)
          break
        case "price-high":
          sortedData.sort((a, b) => b.price - a.price)
          break
        case "rating":
          sortedData.sort((a, b) => b.rating - a.rating)
          break
        case "eco-rating":
          sortedData.sort((a, b) => (b.ecoRating || 0) - (a.ecoRating || 0))
          break
        default:
          sortedData.sort((a, b) => a.name.localeCompare(b.name))
      }

      setProducts(sortedData)
    } catch (err) {
```

```
      setError("Failed to load products. Please try again.")
      console.error("Error fetching products:", err)
    } finally {
      setLoading(false)
    }
  }

  const handleFiltersChange = (newFilters: SearchFiltersType) => {
    setFilters(newFilters)
  }

  return (
    <div className="min-h-screen bg-gray-50">
      {/* Header */}
      <header className="border-b bg-white/95 backdrop-blur supports-[backdrop-
filter]:bg-white/60 sticky top-0 z-50">
        <div className="container flex h-16 items-center justify-between px-4">
          <Link href="/" className="flex items-center gap-2">
            <Leaf className="h-8 w-8 text-green-600" />
            <span className="text-xl font-bold text-green-800">BioGild</span>
          </Link>

          <div className="flex items-center gap-4">
            {authState.isAuthenticated ? (
              <Link href="/account">
                <Button variant="ghost" size="sm" className="flex items-center gap-
2">
                  <User className="h-4 w-4" />
                  Account
                </Button>
              </Link>
            ) : (
              <Link href="/">
                <Button variant="ghost" size="sm">
                  Sign In
                </Button>
              </Link>
            )}
            <Button size="sm" className="bg-green-600 hover:bg-green-700 relative"
onClick={() => setIsCartOpen(true)}>
              <ShoppingCart className="h-4 w-4 mr-2" />
              Cart
              {getTotalItems() > 0 && (
                <span className="absolute -top-2 -right-2 bg-red-500 text-white text-
xs rounded-full h-5 w-5 flex items-center justify-center">
                  {getTotalItems()}
                </span>
              )}
            </Button>
          </div>
        </div>
      </header>
```

```
      <div className="container px-4 py-8">
        <div className="mb-8">
          <h1 className="text-3xl font-bold text-gray-900 mb-2">Our Products</h1>
          <p className="text-gray-600">Fresh, organic produce delivered to your
door</p>
        </div>

        <div className="grid lg:grid-cols-4 gap-8">
          {/* Filters Sidebar */}
          <div className="lg:col-span-1">
            <SearchFilters filters={filters} onFiltersChange={handleFiltersChange} />
          </div>

          {/* Products Grid */}
          <div className="lg:col-span-3">
            {loading ? (
              <LoadingSpinner size="lg" text="Loading products..." />
            ) : error ? (
              <div className="text-center py-12">
                <p className="text-red-600 mb-4">{error}</p>
                <Button onClick={fetchProducts} variant="outline">
                  Try Again
                </Button>
              </div>
            ) : products.length === 0 ? (
              <div className="text-center py-12">
                <p className="text-gray-600 mb-4">No products found matching your
criteria.</p>
                <Button
                  onClick={() =>
                    setFilters({
                      search: "",
                      category: "all",
                      priceRange: [0, 100],
                      ecoRating: 1,
                      sortBy: "name",
                    })
                  }
                  variant="outline"
                >
                  Clear Filters
                </Button>
              </div>
            ) : (
              <>
                <div className="flex justify-between items-center mb-6">
                  <p className="text-gray-600">
                    Showing {products.length} product{products.length !== 1 ? "s" :
""}

                  </p>
                </div>
```

```
                <div className="grid md:grid-cols-2 xl:grid-cols-3 gap-6">
                  {products.map((product) => (
                    <ProductCard key={product.id} product={product} />
                  ))}
                </div>
              </>
            )}
          </div>
        </div>
      </div>
    </div>
  )
}
```

product.page.tsx

```
"use client"

import { useState, useEffect } from "react"
import { notFound } from "next/navigation"
import { ProductDetail } from "../../../components/product-detail"
import { LoadingSpinner } from "../../../components/loading-spinner"
import { CartProvider } from "../../../context/cart-context"
import { AuthProvider } from "../../../context/auth-context"
import { api } from "../../../services/api"
import type { Product, Review } from "../../../types/cart"

interface ProductPageProps {
  params: {
    id: string
  }
}

export default function ProductPage({ params }: ProductPageProps) {
  const [product, setProduct] = useState<Product | null>(null)
  const [reviews, setReviews] = useState<Review[]>([])
  const [relatedProducts, setRelatedProducts] = useState<Product[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)

  useEffect(() => {
    const fetchProductData = async () => {
      try {
        setLoading(true)
        setError(null)

        // Fetch product details
        const productData = await api.getProduct(params.id)

        if (!productData) {
          notFound()
          return
```

```
        }

        setProduct(productData)

        // Fetch reviews and related products in parallel
        const [reviewsData, allProducts] = await
Promise.all([api.getProductReviews(params.id), api.getProducts()])

        setReviews(reviewsData)

        // Get related products from the same category
        const related = allProducts.filter((p) => p.id !== params.id && p.category
=== productData.category).slice(0, 4)
        setRelatedProducts(related)
      } catch (err) {
        setError("Failed to load product details. Please try again.")
        console.error("Error fetching product data:", err)
      } finally {
        setLoading(false)
      }
    }

    fetchProductData()
  }, [params.id])

  const handleReviewSubmitted = async () => {
    // Refresh reviews after a new review is submitted
    try {
      const updatedReviews = await api.getProductReviews(params.id)
      setReviews(updatedReviews)
    } catch (err) {
      console.error("Error refreshing reviews:", err)
    }
  }

  if (loading) {
    return (
      <AuthProvider>
        <CartProvider>
          <LoadingSpinner size="lg" text="Loading product details..." />
        </CartProvider>
      </AuthProvider>
    )
  }

  if (error) {
    return (
      <AuthProvider>
        <CartProvider>
          <div className="min-h-screen bg-white flex items-center justify-center">
            <div className="text-center">
              <div className="text-6xl mb-4">⚠</div>
```

```
              <h1 className="text-2xl font-bold text-gray-900 mb-4">Error Loading
Product</h1>
              <p className="text-gray-600 mb-6">{error}</p>
              <button
                onClick={() => window.location.reload()}
                className="bg-green-600 hover:bg-green-700 text-white px-4 py-2
rounded"
              >
                Try Again
              </button>
            </div>
          </div>
        </CartProvider>
      </AuthProvider>
    )
  }

  if (!product) {
    notFound()
  }

  return (
    <AuthProvider>
      <CartProvider>
        <ProductDetail
          product={product}
          reviews={reviews}
          relatedProducts={relatedProducts}
          onReviewSubmitted={handleReviewSubmitted}
        />
      </CartProvider>
    </AuthProvider>
  )
}


namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Resources\CartResource;
use App\Models\Product;
use Illuminate\Http\Request;

class CartController extends Controller
{
    public function index(Request $request)
    {
        $cart = $request->user()->cart()->with('products')->first();

        if (!$cart) {
            $cart = $request->user()->cart()->create(['products_count' => 0]);
```

```
        }

        return new CartResource($cart);
    }

    public function store(Request $request)
    {
        $request->validate([
            'product_id' => 'required|exists:products,id',
            'quantity' => 'integer|min:1|max:100',
        ]);

        $product = Product::findOrFail($request->product_id);
        $quantity = $request->quantity ?? 1;

        if ($product->quantity < $quantity) {
            return response()->json(['message' => 'Insufficient stock'], 400);
        }

        $cart = $request->user()->cart()->firstOrCreate(['products_count' => 0]);
        $cart->addProduct($request->product_id, $quantity);

        return new CartResource($cart->load('products'));
    }

    public function update(Request $request)
    {
        $request->validate([
            'product_id' => 'required|exists:products,id',
            'quantity' => 'required|integer|min:0|max:100',
        ]);

        $cart = $request->user()->cart()->with('products')->first();

        if (!$cart) {
            return response()->json(['message' => 'Cart not found'], 404);
        }

        $product = Product::findOrFail($request->product_id);

        if ($request->quantity > 0 && $product->quantity < $request->quantity) {
            return response()->json(['message' => 'Insufficient stock'], 400);
        }

        $cart->updateProductQuantity($request->product_id, $request->quantity);

        return new CartResource($cart->load('products'));
    }
```

```
    public function destroy(Request $request)
    {
        $request->validate(['product_id' => 'required|exists:products,id']);

        $cart = $request->user()->cart()->first();

        if (!$cart) {
            return response()->json(['message' => 'Cart not found'], 404);
        }

        $cart->removeProduct($request->product_id);

        return new CartResource($cart->load('products'));
    }

    public function clear(Request $request)
    {
        $cart = $request->user()->cart()->first();

        if ($cart) {
            $cart->clear();
        }

        return response()->json(['message' => 'Cart cleared successfully']);
    }
}
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreUserRequest;
use App\Http\Requests\UpdateUserRequest;
use App\Http\Resources\UserResource;
use App\Http\Resources\UserCollection;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index(Request $request)
    {
        $users = User::query()
            ->when($request->search, function ($query, $search) {
                $query->where(function ($q) use ($search) {
                    $q->where('first_name', 'like', "%{$search}%")
                        ->orWhere('last_name', 'like', "%{$search}%")
```

```
                    ->orWhere('email', 'like', "%{$search}%");
            });
        })
        ->when($request->has_orders, function ($query) {
            $query->has('orders');
        })
        ->with(['addresses', 'orders'])
        ->paginate($request->per_page ?? 15);

    return new UserCollection($users);
}

public function store(StoreUserRequest $request)
{
    $user = User::create([
        'first_name' => $request->first_name,
        'last_name' => $request->last_name,
        'middle_name' => $request->middle_name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'phone' => $request->phone,
    ]);

    // Create cart for new user
    $user->cart()->create(['products_count' => 0]);

    return new UserResource($user->load(['addresses', 'cart']));
}

public function show(User $user)
{
    return new UserResource($user->load([
        'addresses',
        'orders.products',
        'cart.products',
        'reviews.product',
        'promocodes'
    ]));
}

public function update(UpdateUserRequest $request, User $user)
{
    $data = $request->validated();

    if (isset($data['password'])) {
        $data['password'] = Hash::make($data['password']);
    }

    $user->update($data);
```

```
        return new UserResource($user->load(['addresses', 'cart']));
    }

    public function destroy(User $user)
    {
        $user->delete();
        return response()->json(['message' => 'User deleted successfully']);
    }

    public function profile(Request $request)
    {
        return new UserResource($request->user()->load([
            'addresses',
            'orders.products',
            'cart.products',
            'reviews'
        ]));
    }
}

namespace App\Models;

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

class Cart extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'products_count',
    ];

    protected $casts = [
        'products_count' => 'integer',
    ];

    // Relationships
    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function products()
    {
        return $this->belongsToMany(Product::class, 'cart_products')
            ->withPivot('quantity')
```

```
            ->withTimestamps();
    }

    // Accessors
    public function getTotalAmountAttribute()
    {
        return $this->products->sum(function ($product) {
            return $product->price * $product->pivot->quantity;
        });
    }

    public function getTotalItemsAttribute()
    {
        return $this->products->sum('pivot.quantity');
    }

    // Methods
    public function addProduct($productId, $quantity = 1)
    {
        $existingProduct = $this->products()->where('product_id', $productId)-
>first();

        if ($existingProduct) {
            $this->products()->updateExistingPivot($productId, [
                'quantity' => $existingProduct->pivot->quantity + $quantity
            ]);
        } else {
            $this->products()->attach($productId, ['quantity' => $quantity]);
        }

        $this->updateProductsCount();
    }

    public function removeProduct($productId)
    {
        $this->products()->detach($productId);
        $this->updateProductsCount();
    }

    public function updateProductQuantity($productId, $quantity)
    {
        if ($quantity <= 0) {
            $this->removeProduct($productId);
        } else {
            $this->products()->updateExistingPivot($productId, ['quantity' =>
$quantity]);
        }
        $this->updateProductsCount();
    }
```

```
    public function clear()
    {
        $this->products()->detach();
        $this->update(['products_count' => 0]);
    }

    private function updateProductsCount()
    {
        $this->update(['products_count' => $this->products()->count()]);
    }
}

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Order extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'address_id',
        'status',
        'total_amount',
        'promocode_id',
    ];

    protected $casts = [
        'total_amount' => 'decimal:2',
    ];

    const STATUS_PENDING = 'pending';
    const STATUS_CONFIRMED = 'confirmed';
    const STATUS_PROCESSING = 'processing';
    const STATUS_SHIPPED = 'shipped';
    const STATUS_DELIVERED = 'delivered';
    const STATUS_CANCELLED = 'cancelled';

    // Relationships
    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function address()
```

```
    {
        return $this->belongsTo(Address::class);
    }

    public function promocode()
    {
        return $this->belongsTo(Promocode::class);
    }

    public function products()
    {
        return $this->belongsToMany(Product::class, 'order_products')
            ->withPivot('quantity', 'price')
            ->withTimestamps();
    }

    // Scopes
    public function scopeByStatus($query, $status)
    {
        return $query->where('status', $status);
    }

    // Accessors
    public function getStatusLabelAttribute()
    {
        return ucfirst(str_replace('_', ' ', $this->status));
    }
}

<?php

namespace App\Models;

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

class Product extends Model
{
    use HasFactory;

    protected $fillable = [
        'title',
        'description',
        'quantity',
        'price',
        'certificate',
        'eco_rating',
        'price_double',
        'shelf_life',
```

```
    'origin',
    'images',
    'category_id',
];

protected $casts = [
    'price' => 'decimal:2',
    'price_double' => 'decimal:2',
    'eco_rating' => 'integer',
    'quantity' => 'integer',
    'images' => 'array',
];

// Relationships
public function category()
{
    return $this->belongsTo(Category::class);
}

public function reviews()
{
    return $this->hasMany(Review::class);
}

public function orders()
{
    return $this->belongsToMany(Order::class, 'order_products')
                ->withPivot('quantity', 'price')
                ->withTimestamps();
}

public function carts()
{
    return $this->belongsToMany(Cart::class, 'cart_products')
                ->withPivot('quantity')
                ->withTimestamps();
}

// Accessors
public function getAverageRatingAttribute()
{
    return $this->reviews()->avg('rate') ?? 0;
}

public function getReviewsCountAttribute()
{
    return $this->reviews()->count();
}
```

```
    public function getInStockAttribute()
    {
        return $this->quantity > 0;
    }
}
```

```
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    protected $fillable = [
        'first_name',
        'last_name',
        'middle_name',
        'email',
        'password',
        'phone',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    // Relationships
    public function addresses()
    {
        return $this->hasMany(Address::class);
    }

    public function orders()
    {
        return $this->hasMany(Order::class);
    }
```

```
    public function cart()
    {
        return $this->hasOne(Cart::class);
    }

    public function reviews()
    {
        return $this->hasMany(Review::class);
    }

    public function promocodes()
    {
        return $this->hasMany(Promocode::class);
    }

    // Accessors
    public function getFullNameAttribute()
    {
        return trim($this->first_name . ' ' . $this->middle_name . ' ' . $this-
>last_name);
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreOrderRequest;
use App\Http\Requests\UpdateOrderRequest;
use App\Http\Resources\OrderResource;
use App\Http\Resources\OrderCollection;
use App\Models\Order;
use App\Models\Product;
use App\Models\Promocode;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class OrderController extends Controller
{
    public function index(Request $request)
    {
        $orders = Order::query()
            ->when($request->user_id, function ($query, $userId) {
                $query->where('user_id', $userId);
            })
```

```
            ->when($request->user() && !$request->user_id, function ($query) use
($request) {
                $query->where('user_id', $request->user()->id);
            })
            ->when($request->status, function ($query, $status) {
                $query->where('status', $status);
            })
            ->when($request->date_from, function ($query, $dateFrom) {
                $query->whereDate('created_at', '>=', $dateFrom);
            })
            ->when($request->date_to, function ($query, $dateTo) {
                $query->whereDate('created_at', '<=', $dateTo);
            })
            ->with(['user', 'address', 'promocode', 'products'])
            ->latest()
            ->paginate($request->per_page ?? 15);

        return new OrderCollection($orders);
    }

    public function store(StoreOrderRequest $request)
    {
        return DB::transaction(function () use ($request) {
            $user = $request->user();
            $cart = $user->cart()->with('products')->first();

            if (!$cart || $cart->products->isEmpty()) {
                return response()->json(['message' => 'Cart is empty'], 400);
            }

            // Calculate total
            $totalAmount = $cart->total_amount;
            $promocode = null;

            // Apply promocode if provided
            if ($request->promocode_code) {
                $promocode = Promocode::where('code', $request->promocode_code)
                    ->available()
                    ->first();

                if (!$promocode) {
                    return response()->json(['message' => 'Invalid or expired
promocode'], 400);
                }

                if ($totalAmount < $promocode->min_price) {
                    return response()->json([
                        'message' => "Minimum order amount for this promocode is
{$promocode->min_price}"
```

```
                    ], 400);
                }

                if ($promocode->type === Promocode::TYPE_PERCENTAGE) {
                    $totalAmount -= ($totalAmount * $promocode->value / 100);
                } else {
                    $totalAmount -= $promocode->value;
                }
            }

            // Create order
            $order = Order::create([
                'user_id' => $user->id,
                'address_id' => $request->address_id,
                'status' => Order::STATUS_PENDING,
                'total_amount' => max(0, $totalAmount),
                'promocode_id' => $promocode?->id,
            ]);

            // Attach products to order
            foreach ($cart->products as $product) {
                $order->products()->attach($product->id, [
                    'quantity' => $product->pivot->quantity,
                    'price' => $product->price,
                ]);

                // Update product quantity
                $product->decrement('quantity', $product->pivot->quantity);
            }

            // Clear cart
            $cart->clear();

            return new OrderResource($order->load(['user', 'address', 'promocode',
'products']));
        });
    }

    public function show(Order $order)
    {
        $this->authorize('view', $order);
        return new OrderResource($order->load([
            'user',
            'address',
            'promocode',
            'products'
        ]));
    }
```

```
    public function update(UpdateOrderRequest $request, Order $order)
    {
        $this->authorize('update', $order);

        $data = $request->validated();

        // Only allow status updates for completed orders
        if ($order->status !== Order::STATUS_PENDING && isset($data['status'])) {
            $allowedTransitions = [
                Order::STATUS_CONFIRMED => [Order::STATUS_PROCESSING,
Order::STATUS_CANCELLED],
                Order::STATUS_PROCESSING => [Order::STATUS_SHIPPED,
Order::STATUS_CANCELLED],
                Order::STATUS_SHIPPED => [Order::STATUS_DELIVERED],
            ];

            if (!in_array($data['status'], $allowedTransitions[$order->status] ??
[])) {
                return response()->json(['message' => 'Invalid status transition'],
400);
            }
        }

        $order->update($data);
        return new OrderResource($order->load(['user', 'address', 'promocode',
'products']));
    }

    public function destroy(Order $order)
    {
        $this->authorize('delete', $order);

        if (!in_array($order->status, [Order::STATUS_PENDING,
Order::STATUS_CANCELLED])) {
            return response()->json(['message' => 'Cannot delete processed order'],
400);
        }

        // Restore product quantities
        foreach ($order->products as $product) {
            $product->increment('quantity', $product->pivot->quantity);
        }

        $order->delete();
        return response()->json(['message' => 'Order deleted successfully']);
    }

    public function cancel(Order $order)
    {
```

```
        $this->authorize('update', $order);

        if (!in_array($order->status, [Order::STATUS_PENDING,
Order::STATUS_CONFIRMED])) {
            return response()->json(['message' => 'Cannot cancel this order'], 400);
        }

        // Restore product quantities
        foreach ($order->products as $product) {
            $product->increment('quantity', $product->pivot->quantity);
        }

        $order->update(['status' => Order::STATUS_CANCELLED]);
        return new OrderResource($order->load(['user', 'address', 'promocode',
'products']));
    }
}
```

ДОДАТОК Д – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ПРОДУКТІВ «BioGild»

1 Загальні відомості

Найменування: програмне забезпечення інтернет-магазину «BioGild».

1.2 Програмне забезпечення, необхідне для функціонування програми

Для стабільної роботи програми потрібен комп’ютер або смартфон із підключенням до інтернету.

1.3 Мови програмування

Програмне забезпечення інтернет-магазину «BioGild» створене на основі PHP-фреймворку Laravel. Клієнтська частина інтерфейсу реалізована за допомогою HTML, CSS та JavaScript з використання React-фреймворку Next.js.

2 Функціональне призначення

1.1 Класи розв’язання задач та їх призначення

Функціональним призначенням розроблюваного ПЗ є продаж екологічних продуктів у мережі інтернет.

Перелік функцій ПЗ інтернет-магазину:

– Реєстрацію, авторизацію, особистий кабінет з історією замовлень та налаштуваннями профілю, а також реалізувати розподіл ролей (покупці та адміністратори).

– Додавати та редагувати товари через адмін-панель, фільтрувати їх за категоріями, ціною, еко-рейтингом чи сертифікатами, шукати за назвою, а також відображати детальні картки товарів з фотографіями та відгуками.

– Додавати товари до кошика, бачити підсумкову вартість з урахуванням знижок, обирати спосіб оплати.

– Відстежувати статус замовлення (у обробці, доставка, отримано), надсилати сповіщення на email, а також обробляти повернення чи скасування замовлень.

– Залишати оцінки (1–5 зірок) та коментарі, які проходять модерацію, а система формує рейтинг «Найкращі екотовари» на основі відгуків.

– Підтримувати промокоди, а також email-розсилки про новинки та спецпропозиції.

– Реалізувати збір статистики щодо продажів (популярні товари, сезонність).

– Створювати звіти по продажам.

2.2 Функціональні обмеження

Для роботи з програмою обов'язково потрібне підключення до інтернету.

3 Опис логічної структури

Model-View-Controller (MVC) – це архітектурний шаблон проєктування, який розділяє логіку програми на три складові з різними завданнями. Ці шари взаємодіють між собою, щоб забезпечити узгоджену та послідовну роботу програми.

Підхід MVC охоплює всю програму – від інтерфейсу користувача (UI) до базової моделі даних. MVC, який виник у 1970-х роках для побудови графічних інтерфейсів, сьогодні широко використовується у веб-розробці та об'єктно-орієнтованому програмуванні (ООП).

Складові цього шаблону:

– Model (Модель) – реалізує зберігання даних та отримує дані з бази даних, може включати валідацію. Забезпечує цілісність і доступність даних.

– View (Представлення) – Відповідає за інтерфейс користувача (UI). Відображає дані (наприклад, кнопки, таблиці, форми тощо). Не містить бізнес-логіки - лише показує інформацію, отриману від Controller або Model.

– Controller (Контролер) – Керує взаємодією між View та Model. Обробляє запити користувачів (натискання кнопок, введення даних тощо). Оновлює Model та передає дані до View для відображення. Часто називають "мозком" програми, оскільки він координує всі процеси.

Графічно структуру системи наведено на рисунку Д.1.

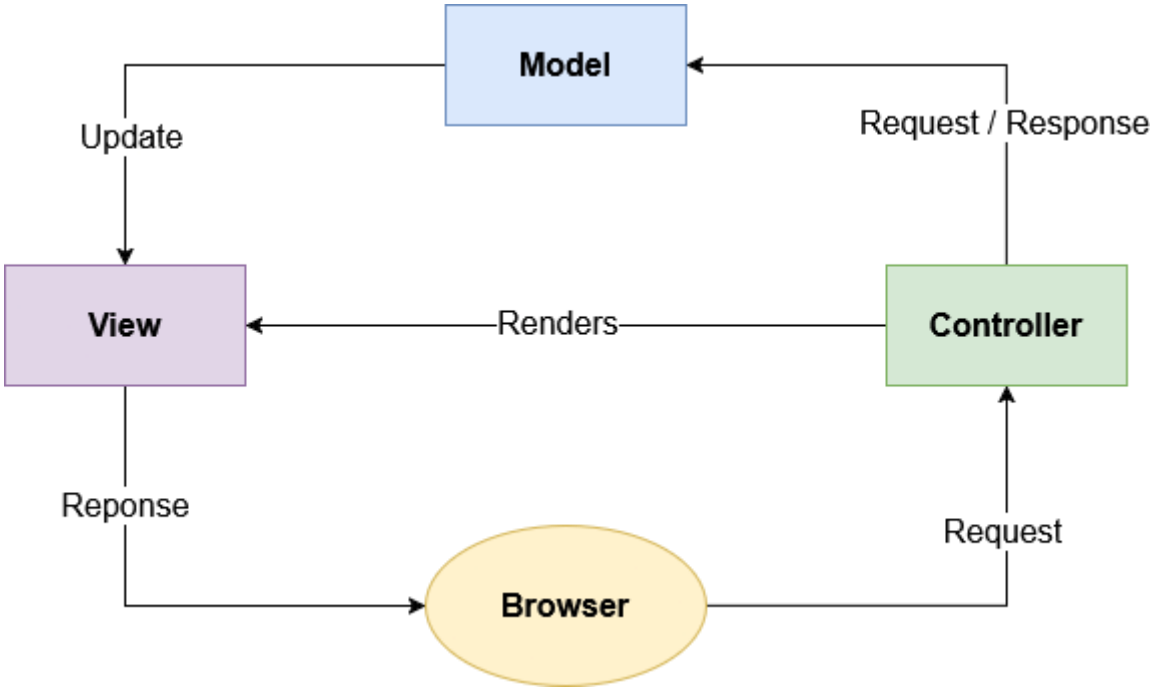


Рисунок Д.1 – Модель MVC

Логічна структура представлена діаграмою прецедентів, де кожен прецедент – виявлена функції ПЗ з актором, що її виконує.

3 Технічні засоби, що використовуються

Серверна частина та її вимоги:

– ЦП – 4 ядра, 2.0 ГГц.

– Оперативна пам’ять – 4 ГБ.

– Дискова пам’ять – 128 ГБ.

– Мережа – 1 Гбіт Ethernet

Пристрій користувача має задовольняти наступні мінімальні вимоги:

– ЦП – 2 ядра, 1.5–2.0 ГГц.

– Оперативна пам’ять – 2-4 ГБ.

– Дискова пам’ять – 128 ГБ.

– Доступ до мережі інтернет

4 Виклик та завантаження

Для запуску ПЗ необхідно відкрити будь-яку з його сторінок у браузері.

5 Вхідні дані

Вхідними даними для продукту є назва, опис, категорія, ціна, ціна зі знижкою, екологічний рейтинг товару, походження, сертифікат, оцінка товару (1-5 зірок, залишається користувачами), період зберігання, одиниці виміру, кількість одиниць та зображення у форматі .png, .jpeg (jpg), .webp.

Вхідними даними для користувача є ПІБ, email, телефон, роль (клієнт / адмін)

Вхідними даними для формування замовлення є дані про товари, що знаходяться у кошику на момент оформлення замовлення та даних про користувача, який оформлює замовлення, промокод (за наявності).

Вхідними даними про відгук є дані про користувача, що залишив відгук, заголовок відгуку, текст відгуку, оцінка товару.

Вхідними даними для створення промокодів буде: код, тип промокоду (сума або відсоток), числове значення знижки (дійсне додатне число), період дії, мінімальна вартість замовлення для застосування, максимальне значення знижки, кількість застосувань.

6 Вихідні дані

Вихідними даними про продукт є запис у базі даних системи, яку користувач буде бачити у вигляді карти продукту, яка відображає заповнену раніше інформацію.

Вихідними даними про користувача є відповідний запис в бд системи

Вихідними даними про відгук є запис у базі даних системи, разом з карткою відгуку в адмін частині застосунку.

Вихідними даними про промокод є запис у базі даних системи.

Вихідними даними про замовлення є запис у базі даних.