Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ТОВАРІВ ЗАСОБАМИ ТЕХНОЛОГІЇ BLAZOR

DEVELOPMENT OF A WEB APPLICATION FOR AN ONLINE STORE OF ECOLOGICAL GOODS USING BLAZOR TECHNOLOGY

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Грищук Анастасія Олександрівна

Керівник:
Суринович Олена Миколаївна

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Грищук Анастасії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «*Розробка вебзастосунку для інтернет-магазину екологічних товарів засобами технології Blazor»*
Керівник роботи: *к.т.н., доцент Суринович О. М.*
затверджені наказом закладу вищої освіти від *«31» грудня 2025 р.* *№ 541/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«29» травня 2026 р.*
3. Вихідні дані до роботи: *предметна область електронної комерції та продажу екологічних товарів, технології розробки програмного забезпечення (C#, .NET, Blazor), засоби роботи з базами даних (SQLServer, Entity Framework Core), вимоги до інформаційних систем та забезпечення інформаційної безпеки.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз сучасного стану розвитку електронної комерції та автоматизації процесів реалізації екологічних товарів, постановка завдання, формування вимог до розроблюваного вебзастосунку, обґрунтування вибору технологій та засобів розробки, практична реалізація об'єкта проєктування, розробка структури бази даних, обґрунтування принципів захисту інформаційної системи, тестування та налагодження програмного забезпечення.*
5. Перелік графічного матеріалу: *18 рисунків, 4 лістинга*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Суринович О. М.*		
Специфікація вимог до розробленої системи	*Суринович О. М.*		
Розробка об’єкта проектування	*Суринович О. М.*		
Нормоконтроль	*Суринович О. М.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Суринович О. М.*		

7. Дата видачі завдання *«3» лютого 2026 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об’єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об’єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об’єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об’єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 29.05.2026 р.	

Здобувач вищої освіти _____ Анастасія ГРИЩУК

Керівник кваліфікаційної роботи _____ Олена СУРИНОВИЧ

АНОТАЦІЯ

Грищук А. О. Розробка вебзастосунку для інтернет-магазину екологічних товарів засобами технології Blazor. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі проведено аналіз сфери інтернет-торгівлі екологічними товарами. Розглянуто вже існуючі програмні рішення для онлайн-магазинів, визначено їхні сильні та слабкі сторони, а також обґрунтовано доцільність створення власної системи. У другому розділі сформовано функціональні та нефункціональні вимоги до вебзастосунку, здійснено проєктування архітектури системи з використанням UML-діаграм, а також обґрунтовано вибір сучасних інструментальних засобів і технологій розробки (зокрема платформи .NET та фреймворку Blazor). У третьому розділі описано процес практичної реалізації вебзастосунку, розроблено структуру бази даних, розглянуто питання безпеки даних та авторизації користувачів, а також наведено результати тестування готового програмного забезпечення. У висновках підсумовано отримані результати, підтверджено досягнення мети роботи та повне виконання поставлених завдань.

Ключові слова: вебзастосунок, інтернет-магазин, екологічні товари, електронна комерція, база даних, C#, .NET, Blazor, SQL Server, Entity Framework Core.

ABSTRACT

Hryshchuk A. O. Development of a Web Application for an Online Store of Ecological Goods Using Blazor Technology. Manuscript. Qualification work for bachelor's degree in Software Engineering, specialty “Software Engineering”. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references.

The first chapter provides an analysis of the e-commerce sector for eco-friendly products. Existing software solutions for online stores are reviewed, their strengths and weaknesses are identified, and the feasibility of developing a custom system is justified. In the second chapter, the functional and non-functional requirements for the web application are defined, the system architecture is designed using UML diagrams, and the selection of modern development tools and technologies (specifically, the .NET platform and the Blazor framework) is justified. The third chapter describes the practical implementation process of the web application, details the development of the database structure, addresses data security and user authorization issues, and presents the results of testing the developed software. The conclusions summarize the obtained results, confirming the achievement of the work's objective and the complete fulfillment of the assigned tasks.

Keywords: web application, online store, eco-friendly products, e-commerce, database, C#, .NET, Blazor, SQL Server, Entity Framework Core.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі електронна комерція (e-commerce) є невід'ємною частиною глобальної економіки. Водночас спостерігається стрімке зростання екологічної свідомості суспільства. Споживачі все частіше прагнуть робити відповідальний вибір, купуючи органічні, безпечні для здоров'я та довкілля товари. Тому створення зручних, надійних та швидких онлайн-майданчиків для реалізації такої продукції є комерційно перспективним та соціально значущим напрямком [1].

На ринку існує велика кількість готових платформ (CMS) для створення інтернет-магазинів. Однак вони часто є занадто громіздкими, важко піддаються глибокій кастомізації або вимагають компромісів у продуктивності. Крім того, традиційна веброзробка зазвичай вимагає використання різних мов для серверної та клієнтської частин (наприклад, C# на бекенді та JavaScript на фронтенді). Це збільшує час розробки, ускладнює підтримку коду та вимагає дублювання бізнес-логіки.

Окремим викликом є забезпечення високого рівня безпеки даних користувачів і швидкої роботи інтерфейсу. Використання сучасної технології Blazor від платформи .NET дозволяє елегантно вирішити ці проблеми. Blazor дає змогу створювати інтерактивні клієнтські вебзастосунки типу Single Page Application (SPA), використовуючи виключно мову C#. Це гарантує строгу типізацію, можливість спільного використання моделей даних між клієнтом і сервером, а також високу надійність архітектури.

Отже, тема кваліфікаційної роботи, присвячена розробці спеціалізованого вебзастосунку для інтернет-магазину екологічних товарів із застосуванням технології Blazor та надійних методів захисту даних, є актуальною, оскільки відповідає сучасним вимогам IT-індустрії до розробки програмного забезпечення.

Метою кваліфікаційної роботи є розробка сучасного, продуктивного та безпечного вебзастосунку для інтернет-магазину екологічних товарів із застосуванням мови програмування C# та технології Blazor.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– здійснити аналіз предметної області та сучасних тенденцій на ринку електронної комерції екологічних товарів;

– сформувати функціональні та нефункціональні вимоги до майбутнього вебзастосунку;

– обґрунтувати вибір стека технологій для реалізації клієнтської, серверної частин та бази даних;

– спроєктувати загальну архітектуру системи та логічну структуру реляційної бази даних;

– практично реалізувати програмний продукт (зокрема REST API, інтерфейс користувача та бізнес-логіку);

– впровадити механізми інформаційної безпеки, зокрема автентифікацію та авторизацію користувачів;

– провести тестування і налагодження розробленого програмного забезпечення.

Об’єктом дослідження є процеси проєктування, розробки та функціонування інформаційних систем у сфері електронної комерції.

Предметом дослідження є методи та засоби розробки інформаційної системи з використанням технологій платформи .NET.

Практичне значення роботи полягає у створенні готового до розгортання програмного продукту (інтернет-магазину «EcoMarket»), який забезпечує зручний пошук та купівлю екотоварів для користувачів, а також надає захищені інструменти для управління каталогом і замовленнями з боку адміністратора системи.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах глобальної цифрової трансформації електронна комерція (e-commerce) є одним із найдинамічніших секторів світової та вітчизняної економіки. З кожним роком обсяги онлайн-продажів невпинно зростають, що зумовлено зручністю, доступністю цілодобово та можливістю здійснювати покупки без географічних обмежень [2].

Паралельно з цим технологічним трендом спостерігається суттєва зміна споживацьких пріоритетів: суспільство демонструє стрімке зростання екологічної свідомості. Сучасні покупці все більше уваги приділяють здоровому способу життя, відповідальному споживанню та збереженню довкілля, віддаючи перевагу органічним, безпечним і екологічно чистим товарам, що підтверджується стрімким розвитком економіки сталого споживання [3].

Зростаючий попит на екопродукцію залучає до онлайн-покупок максимально широку аудиторію покупців з різним рівнем цифрової грамотності. Саме тому ключовою вимогою до сучасного інтернет-магазину стає простий, інтуїтивно зрозумілий та адаптивний дизайн (UI/UX). Перевантажені графікою чи зайвими функціями інтерфейси відштовхують клієнтів. Натомість зручна навігація, чітка і зрозуміла структура категорій каталогу та миттєвий доступ до структурованої інформації про наявність товарів на складі роблять процес вибору та покупки максимально комфортним.

Водночас попри візуальну простоту та мінімалізм клієнтської частини сайту, невід’ємною складовою будь-якого, навіть найпростішого комерційного вебзастосунку, є надійна архітектура інформаційної безпеки. Захист персональних даних користувачів, безпечна авторизація, збереження конфіденційності історії замовлень – це критично важливі аспекти, якими не можна нехтувати у цифровому середовищі. Використання застарілих або

вразливих методів зберігання даних (наприклад, відкритих токенів доступу) може призвести до їх витоку, що миттєво руйнує довіру до платформи.

Хорошим прикладом того, що сервіс не відповідає сучасним вимогам, є онлайн-представництво торгівельної мережі «ЕКО маркет». Аналіз показує, що їхня цифрова інфраструктура побудована фрагментарно: електронна комерція реалізована через два окремі сайти, і кожен із них має суттєві недоліки. Зокрема, офіційний корпоративний сайт мережі «eko.com.ua» виконує здебільшого функцію корпоративного порталу, а не інтегрованого інтернет-магазину, що вже на початковому етапі створює значні бар'єри для користувачів. Інтерфейс сайту перенасичений інформацією про діяльність компанії, рекламними матеріалами та відомостями для бізнес-партнерів, що ускладнює доступ до зручного каталогу товарів. Відсутність можливості безпосередньо здійснити покупку через вебресурс негативно впливає на цілісність клієнтського досвіду, примушуючи користувача звертатися до сторонніх платформ [4].

Аналіз другого ресурсу «eko.zakaz.ua», який виконує функцію інтернет-магазину, виявляє низку суттєвих проблем у сфері UI/UX дизайну та забезпечення безпеки. Хоча технічно платформа надає можливість здійснювати онлайн-покупки, її інтерфейс, замість очікуваного екологічного мінімалізму, перенасичений елементами, що створюють візуальний хаос. Велика кількість дрібних категорій, нав'язливі акційні пропозиції та таймери формують надмірний інформаційний шум, який ускладнює процес навігації та концентрування уваги покупця [5].

Окрім недоліків дизайну, важливо зазначити ризики в архітектурі вебзастосунку з точки зору захисту даних. Такі сайти, як правило, працюють за принципом односторінкових застосунків (SPA) зі значною залежністю від клієнтських скриптів. При цьому застосування застарілих методів авторизації створює небезпеку для конфіденційності даних користувачів. Зокрема, практика зберігання JWT-токенів або сесійних ідентифікаторів у відкритих сховищах, таких як LocalStorage, замість більш захищених HttpOnly-cookie, значно підвищує ймовірність XSS-атак. У результаті токени можуть бути перехоплені,

що дозволяє кіберзлочинцям отримати несанкціонований доступ до профілів користувачів, історії їхніх замовлень та особистих даних, включаючи адреси доставлення.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка продуктивного та безпечного вебзастосунку для інтернет-магазину екологічних товарів із застосуванням мови програмування C# та технології Blazor.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– здійснити аналіз предметної області та сучасних тенденцій на ринку електронної комерції екологічних товарів, дослідивши специфіку продажу екопродукції та потреби цільової аудиторії;

– сформувати функціональні та нефункціональні вимоги до майбутнього вебзастосунку, визначивши ключові сценарії використання (робота з каталогом, оформлення замовлень) та критерії продуктивності, масштабованості і зручності інтерфейсу;

– обґрунтувати вибір стека технологій (зокрема платформи .NET, фреймворків для UI та СУБД) для ефективної та надійної реалізації клієнтської, серверної частин та бази даних;

– спроєктувати загальну архітектуру системи та детальну логічну структуру реляційної бази даних, визначивши основні сутності, зв’язки між ними та принципи збереження інформації;

– практично реалізувати програмний продукт, що включає розробку серверної бізнес-логіки, налаштування взаємодії через REST API та створення адаптивного, інтуїтивно зрозумілого інтерфейсу користувача;

– впровадити механізми інформаційної безпеки, зокрема сучасні протоколи автентифікації та авторизації користувачів (наприклад, на базі JWT-токенів), для захисту персональних даних та розмежування прав доступу;

– провести комплексне тестування і налагодження розробленого програмного забезпечення для виявлення та усунення помилок, перевірки коректності роботи API і підтвердження стабільності системи.

Висновки до розділу 1

У першому розділі було проаналізовано сучасний стан і тенденції розвитку електронної комерції, акцентуючи увагу на ринку екологічно чистих товарів. Виявлено, що зростання екологічної свідомості суспільства та підвищений попит на екопродукцію формують нові вимоги до онлайн-магазинів. Зокрема, вони мають забезпечувати гармонійне поєднання мінімалістичного й інтуїтивно зрозумілого користувацького інтерфейсу з надійною системою інформаційної безпеки.

Практичний аналіз наявних платформ на ринку показав, що багато з них не відповідають цим сучасним стандартам. Було виявлено низку проблем, серед яких фрагментарний підхід до організації продажів, перевантаженість візуальних елементів інтерфейсу, а також використання недостатньо захищених методів авторизації, таких як зберігання JWT-токенів у відкритих сховищах. Ці недоліки негативно впливають на користувацький досвід і підвищують ризик витоку персональних даних клієнтів.

На основі проведеного аналізу було обґрунтовано необхідність створення новітнього програмного рішення. Сформульовано мету та конкретні завдання кваліфікаційної роботи, які передбачають розробку продуктивного вебзастосунку для інтернет-магазину екологічних товарів із залученням сучасного стека технологій, таких як C# і Blazor. Розробка такої платформи дозволить створити безпечну архітектуру для серверної та клієнтської частин, впровадити ефективні механізми захисту даних і авторизації, забезпечуючи при цьому зручний та безпечний процес онлайн-покупок для користувачів.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Перехід від теоретичного аналізу предметної області до практичної реалізації вебзастосунку потребує створення чіткої архітектури, яка об’єднує всі функціональні та безпекові вимоги. Перед початком безпосереднього написання програмного коду критично важливим етапом є формування моделі системи, що дозволяє візуалізувати внутрішню логіку процесів, взаємодію компонентів та сценарії поведінки користувачів.

Основним інструментом для виконання цього завдання було обрано уніфіковану мову моделювання UML (Unified Modeling Language), яка на сьогодні є стандартом у розробці об’єктно-орієнтованого програмного забезпечення.

На початковому етапі розробки було створено діаграму класів, оскільки вона дозволяє отримати чітке уявлення про внутрішню структуру системи, її сутності та взаємозв’язки між ними. Особливу увагу було приділено визначенню ключових об’єктів предметної області, таких як User, Order, Product, Category, EcoTag, OrderItem та Role, а також детальному опису їхніх атрибутів і методів.

Для дотримання промислових стандартів безпеки, збереження облікових даних та автентифікації, підсистема користувачів була спроєктована на основі інтеграції з фреймворком ASP.NET Core Identity. Це забезпечило можливість гнучкого розширення базових безпекових компонентів відповідно до потреб екологічного маркету.

Діаграма класів слугувала основою для подальшого формування структури бази даних, забезпечуючи узгодженість між шаром зберігання даних і бізнес-логікою додатка. Загальну діаграму класів сутностей зображено на рисунку 2.1.

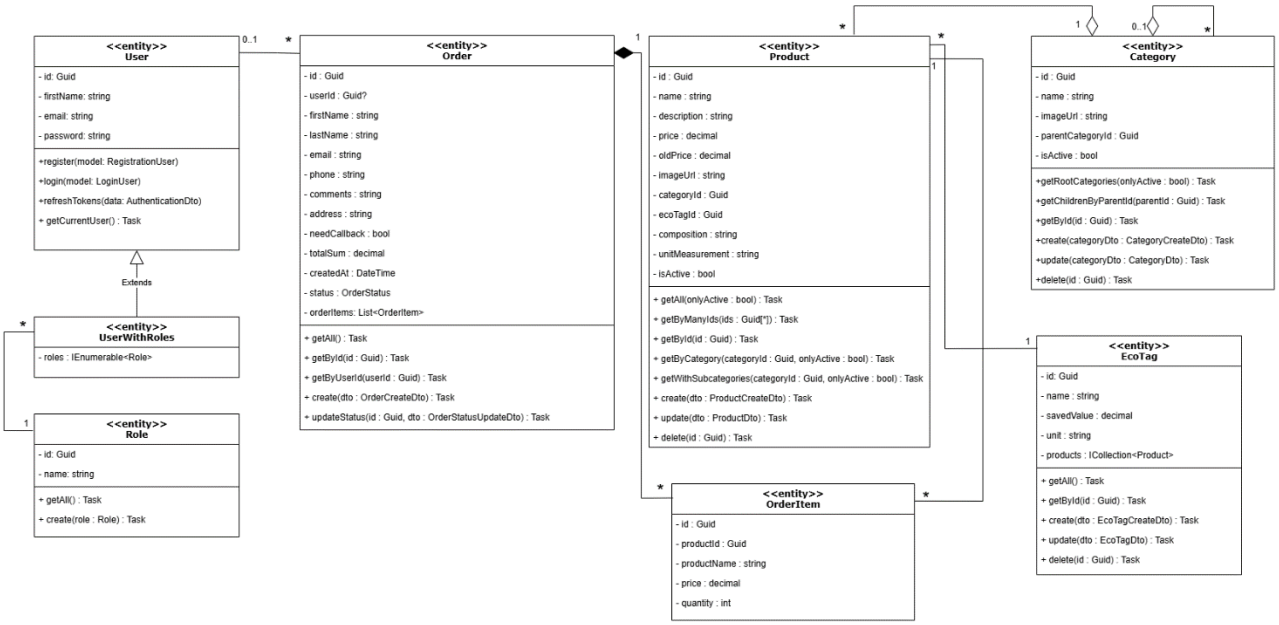

Рисунок 2.1 – Загальна діаграма класів сутностей системи EcoMarket

Розроблена діаграма містить у собі взаємопов’язані класи, які складають ядро доменної моделі інтернет-магазину.

Клас користувача (User) містить у собі такі дані як: унікальний ідентифікатор (id), персональне ім’я (firstName), електронну пошту (email) та зашифрований пароль (password). Ці атрибути відіграють ключову роль у забезпеченні безпечної автентифікації користувачів на вебсайті. Клас користувача має наступні методи (register), (login), (refreshTokens) та (getCurrentUser).

Для реалізації розмежування прав доступу (Role-Based Access Control) [6] базову модель розширено за допомогою класу користувача з ролями (UserWithRoles). Цей клас переймає всі властивості сутності (User) і додатково містить поле (roles) для збереження призначеного привілею. Цей клас взаємодіє із додатковим класом системних ролей (Role), який містить атрибути унікального номера (id) та назви права доступу (name), наприклад: «Admin» чи «Manager».

Зв’язок між (UserWithRoles) та (Role) має кратність 1 до * (один-до-багатьох), що відповідає суворому бізнес-правилу системи EcoMarket. Це правило встановлює, що кожен обліковий запис співробітника може мати лише

одну системну роль одночасно, водночас одна роль може бути закріплена за багатьма різними користувачами додатку. Такий підхід забезпечує можливість оперативно змінювати відображення елементів керування (наприклад, приховувати або відображати кнопку доступу до адмінпанелі) залежно від ролі користувача.

Клас категорії товарів (Category) призначений для побудови структурованого каталогу інтернет-магазину. Він включає такі атрибути: унікальний ідентифікатор (id), назву категорії (name), URL медіаіконки (imageUrl), індикатор відображення на сайті (isActive) та зовнішній ключ для зв’язку з батьківською категорією (parentCategoryId). Ключове значення має поле (parentCategoryId), оскільки воно реалізує рекурсивний зв’язок (само посилання), дозволяючи будувати ієрархічну структуру категорій із необмеженою кількістю рівнів вкладеності (наприклад, для створення підкатегорій). Щоб зручно працювати з цим деревом, клас має методи для отримання головних категорій (getRootCategories), пошуку дочірніх елементів (getChildrenByParentId), а також базові операції для додавання (create), редагування (update) та видалення (delete) категорій адміністратором.

Клас товару (Product) є центральним об’єктом вебсайту та зберігає детальні характеристики кожної доступної для продажу позиції. До його атрибутів належать: унікальний ідентифікатор (id), назва (name), детальний текстовий опис (description), поточна вартість (price) та попередня ціна для відображення акційних знижок (oldPrice). Модель також включає посилання на зображення (imageUrl), опис складу або характеристики матеріалів (composition), визначення одиниці виміру продукту (unitMeasurement), статус активності на сайті (isActive) і зовнішні ключі для прив’язки до відповідної категорії каталогу (categoryId) та екологічного маркування (ecoTagId). Методи які включає в себе цей клас забезпечують повний життєвий цикл роботи з асортиментом на рівні API. Вони охоплюють операції отримання повного списку товарів (getAll), вибірки продуктів для конкретної категорії (getByCategory), а також завантаження товарів з автоматичним урахуванням усіх її підкатегорій (getWithSubcategories).

Для ідентифікації використовується запит на отримання детальної інформації про один конкретний продукт за його ID (getById) та масової вибірки товарів за масивом їхніх ідентифікаторів (getByManyIds). Крім того, реалізовано базові операції для управління товарними картками адміністратором, що включають створення нового продукту (create), оновлення його характеристик (update) та повне видалення позиції із системи (delete).

Клас замовлення (Order) відповідає за процес купівлі товарів клієнтом та фіксацію фінансових транзакцій. Він зберігає всю інформацію про замовлення: унікальний номер (id), необов'язкове посилання на профіль клієнта (userId), дані покупця (firstName, lastName, email, phone), адресу доставлення (address), коментарі (comments), прапорець зворотного дзвінка (needCallback), суму замовлення (totalSum), дату створення (createdAt) та статус обробки (status). Завдяки полю userId система може підтримувати як швидкі гостьові покупки без авторизації, так і повну історію замовлень постійних клієнтів. Логіка класу реалізується методами отримання всіх замовлень (getAll), пошуку за ідентифікатором (getById), вибірки покупок конкретного користувача (getByUserId), створення (create) та оновлення статусу (updateStatus).

Клас позиції замовлення (OrderItem) деталізує вміст кошика: він містить ідентифікатор рядка (id), зовнішній ключ товару (productId), його назву (productName), ціну на момент покупки (price) та кількість одиниць (quantity). Цей клас не має власної бізнес-логіки, адже перебуває у строгій UML-композиції з головним класом (Order). Його головна роль – зафіксувати точний стан товарів і фінансів у момент оплати, що гарантує цілісність даних навіть у випадку зміни ціни чи назви продукту в каталозі (Product).

Унікальну екологічну складову концепції інтернет-магазину реалізує клас екологічного тегу (EcoTag). Він зберігає інформацію про маркування через такі атрибути як: унікальний номер (id), назва еко-маркера (name), чисельне значення збережених природних ресурсів (savedValue), одиниці виміру та зворотну колекцію пов'язаних товарів (products). Зв'язок між (EcoTag) та (Product) має кратність 1 до *, що дозволяє одному створеному тегу повторно

використовуватися для безлічі різних товарів. Ці дані потрібні для того, щоб наочно демонструвати покупцям екологічну цінність продуктів. Для адміністрування еко-тегів у класі реалізовано методи виведення всіх маркувань (getAll), пошуку за ідентифікатором (getById), додавання нових еко-тегів (create), оновлення значень (update) та їхнього видалення (delete).

Варто наголосити, що для практичної реалізації принципів чистої архітектури (Clean Architecture) та забезпечення надійного розділення обов'язків між шарами системи, вищеописані доменні сутності не взаємодіють із зовнішнім клієнтським інтерфейсом Blazor напряму. Замість цього на рівні контролерів API застосовуються об'єкти передачі даних (DTO – Data Transfer Objects) [7], такі як OrderCreateDto, ProductCreateDto, RegistrationUser чи EcoTagDto. акий підхід дозволяє повністю ізолювати бізнес-логіку та структуру бази даних від представлення користувачеві, мінімізуючи ризики надлишкової передачі конфіденційних даних по мережі. Крім того, використання DTO спрощує валідацію вхідних даних безпосередньо перед їх обробкою на серверній стороні.

Для аналізу функціональних можливостей вебзастосунку, визначення меж системи та моделювання поведінкових сценаріїв користувачів на етапі проєктування була створена діаграма прецедентів (Use Case Diagram). Цей інструмент надає можливість оцінити систему з перспективи кінцевого користувача і чітко окреслити, які задачі може вирішувати користувач через інтерфейс взаємодії інтернет-магазину. Завдяки візуалізації зв'язків між сутностями, така діаграма виступає надійною сполучною ланкою між концептуальним баченням проєкту, вимогами замовника та безпосереднім етапом написання програмного коду.

Основним актором (Actor) у цьому випадку є (Клієнт), який відіграє ключову роль у запуску основних бізнес-процесів платформи, таких як керування кошиком чи оформлення замовлень. Розроблену діаграму прецедентів для актора (Клієнт) інформаційної системи EcoMarket детально зображено на рисунку 2.2.

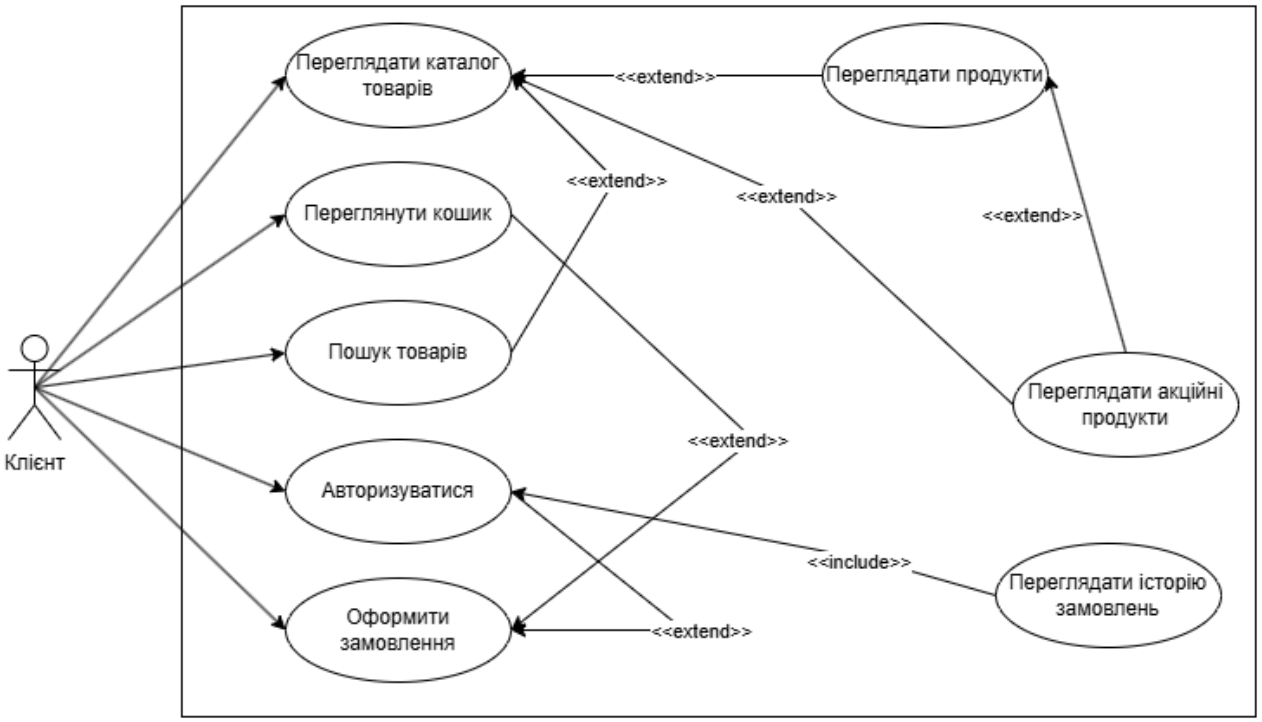

Рисунок 2.2 – Діаграма прецедентів системи з позиції актора Клієнт

Діаграма ілюструє логіку взаємодії користувача з вебзастосунком, а також демонструє взаємозв’язки між окремими функціональними модулями сайту через стандартні UML-відношення включення «include» та розширення «extend». Актор (Клієнт) має безпосередній асоціативний зв’язок із п’ятьма базовими прецедентами платформи, які повністю описують його взаємодію із вебзастосунком.

Первинною точкою входу користувача виступає прецедент (Переглядати каталог) товарів, який відкриває доступ до навігації по всьому асортименту інтернет-магазину. Подальша логіка взаємодії передбачає перехід до розділу (Переглянути кошик), який слугує інструментом для управління обраними продуктами, регулювання їх кількості та перевірки поточної вартості замовлення. Для зручності покупця інтегровано (Пошук товарів), інструмент швидкої фільтрації та знаходження продукції за ключовими словами, який пов’язаний зв’язком «extend» із переглядом каталогу, оскільки є додатковим сценарієм під час роботи з вітриною. Однією з ключових складових безпеки є прецедент (Авторизуватися), який забезпечує процес автентифікації клієнта та

відкриває доступ до персоналізованих функцій системи. Завершальним етапом бізнес-процесу в цьому ланцюжку виступає прецедент (Оформити замовлення). Ця функція дозволяє користувачеві внести свої персональні дані, обрати параметри для зворотного дзвінка через тогл і успішно здійснити оплату.

Для глибокого аналізу динамічної поведінки системи та детального покрокового опису дій покупця на етапі проєктування було розроблено діаграму діяльності (Activity Diagram). Цей інструмент дозволяє візуалізувати алгоритми переходу між сторінками сайту, послідовність виконання бізнес-процесів з боку інтерфейсу та логіку умовних розгалужень у програмі. Повний життєвий цикл взаємодії клієнта з платформою від моменту першого входу до завершення сесії зображено на рисунку 2.3.

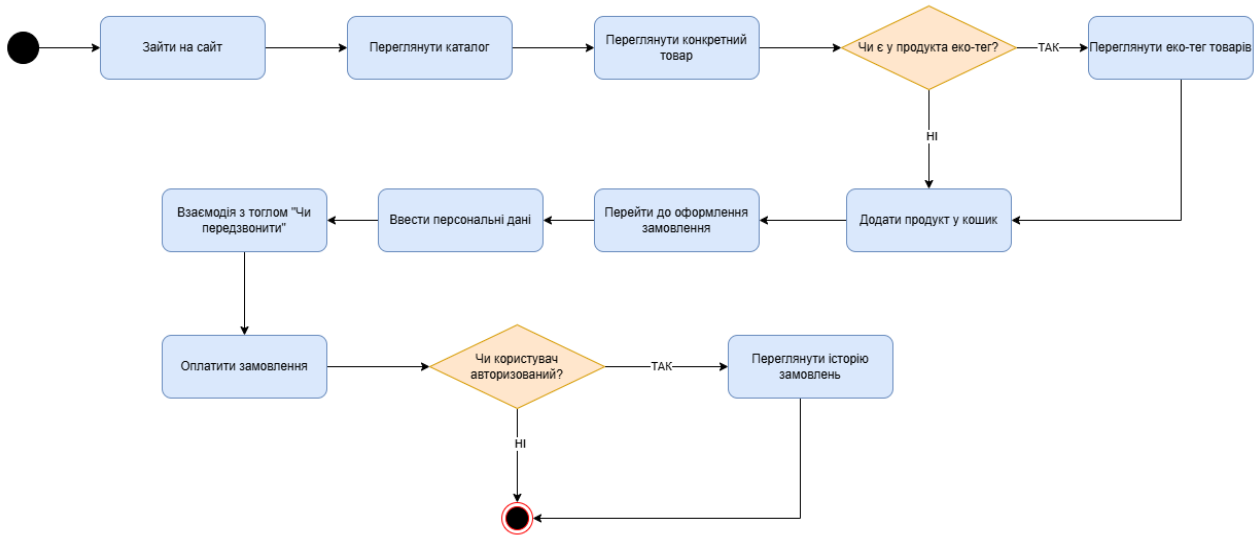

Рисунок 2.3 – Діаграма діяльності процесу взаємодії клієнта з вебзастосунком EcoMarket

Потік управління, представлений на діаграмі, починається з початкового вузла, позначеного жирною чорною точкою. Від нього стрілка вказує на першу активність, яка передбачає перехід користувача на вебсайт інтернет-магазину. Наступним кроком є ознайомлення клієнта з каталогом товарів, в якому реалізована функція навігації за категоріями. На цьому етапі користувач обирає певний товар для детального ознайомлення з його характеристиками,

переходячи до відповідної картки продукту. Подальший алгоритм передбачає перший вузол прийняття рішень, відображений у вигляді ромба. У ньому перевіряється наявність у вибраного товару спеціального екологічного маркування. У разі, якщо таке маркування присутнє, робочий потік слідує умовній гілці «Так», що надає користувачу можливість докладніше ознайомитися з інформацією про екологічні переваги товару, зокрема про заощадження природних ресурсів. Після цього система пропонує додати вибраний продукт у кошик. Якщо ж екологічне маркування відсутнє, алгоритм переходить альтернативною гілкою «Ні», оминаючи перегляд додаткової інформації. У такому разі клієнт потрапляє безпосередньо до етапу наповнення кошика товаром.

Наступний логічний блок діаграми процесу відповідає за завершення оформлення покупки та фіксацію даних замовлення в інформаційній базі системи. Після успішного додавання товару до кошика користувач переходить до форми оформлення замовлення. У цьому кроці система надає інтерфейс для введення персональної інформації, зокрема імені, номера телефону та адреси доставлення. Важливою особливістю цього етапу є інтерактивність у вигляді перемикача «Чи передзвонити», який дає змогу клієнту самостійно визначати необхідність зворотного зв’язку від представника магазину. Інформацію про вибраний варіант автоматично зберігають у параметрах замовлення. Завершальним кроком цього процесу є здійснення платежу, що підтверджує остаточне завершення фінансової транзакції.

Завершальна частина діаграми діяльності зосереджена на перевірці безпеки поточної сесії та наданні додаткових функцій особистого кабінету. Після завершення операції оплати система переходить до вузла прийняття рішень, щоб визначити, чи авторизований користувач. У разі успішної авторизації залучається гілка «Так», що дозволяє покупцеві перейти на захищену сторінку для перегляду історії замовлень. Якщо ж транзакція виконувалася в гостьовому режимі, без авторизації, система за умовою «Ні» спрямовує робочий процес в

обхід особистого кабінету прямо до кінцевого вузла діяльності. Це забезпечує успішне завершення всього сценарію взаємодії з платформою EcoMarket.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для успішної реалізації екологічного інтернет-магазину EcoMarket необхідно ретельно обґрунтувати вибір технологічного стека, здатного забезпечити ефективну розробку, надійний захист даних та легку масштабованість системи. З огляду на актуальні тенденції у сфері веброзробки, особливу увагу слід приділити створенню інтерактивних застосунків із чітко розмежованою логікою. На основі таких вимог для реалізації проєкту була обрана клієнт-серверна архітектура, побудована на платформі .NET, яка є оптимальним рішенням для досягнення поставлених цілей.

У процесі аналізу існуючих рішень для розробки клієнт-серверних вебзастосунків розглядалися кілька альтернативних підходів. Зокрема, популярною альтернативою є використання екосистеми JavaScript/TypeScript (наприклад, Node.js для бекенду та фреймворків React або Vue.js для фронтенду). Хоча ці інструменти мають велику спільноту та гнучкість, їх недоліком у контексті даного проєкту є необхідність підтримки різних середовищ виконання та дублювання моделей даних. Також розглядалися альтернативні СУБД, такі як PostgreSQL та MySQL. Проте, зважаючи на глибоку інтеграцію технологій від Microsoft, перевагу було надано .NET та SQL Server, що забезпечує максимальну сумісність інструментів, надійність транзакцій та знижує загальні витрати часу на налаштування інфраструктури.

Основною мовою програмування для розробки було обрано C# [8], а базовим серверним фреймворком ASP.NET Core [9]. Таке рішення пояснюється тим, що C# є об’єктно-орієнтованою мовою високого рівня, яка дозволяє ефективно структурувати код і створювати застосунки з високим рівнем масштабованості [10]. У межах розробки вебзастосунку EcoMarket цей вибір

особливо виправданий завдяки універсальності мови та платформі .NET Core, що забезпечує кросплатформність. Це означає, що серверний застосунок з легкістю може працювати на Windows, macOS і Linux [10]. Додатково, строгий механізм статичної типізації в C# забезпечує зменшення кількості помилок під час виконання програми, адже типи перевіряються ще на етапі компіляції [10]. Також для створення сучасних веб-API важливим є забезпечення нативної підтримки асинхронного програмування з використанням операторів async/await. Такий підхід дає змогу системі ефективно обробляти значну кількість паралельних клієнтських запитів, зокрема під час виконання операцій вводу/виводу, пов’язаних із доступом до бази даних, без блокування основних потоків виконання. Це, у свою чергу, сприяє досягненню високої продуктивності програмного забезпечення [10].

Для побудови користувацького інтерфейсу обрано технологію Blazor, яка дозволяє створювати інтерактивні вебдодатки за допомогою C# та HTML, мінімізуючи потребу в JavaScript [11]. Головна перевага цього підходу – використання єдиної мови та спільних моделей даних на клієнті й сервері, а компонентна архітектура Blazor ідеально підходить для інкапсуляції логіки таких елементів інтернет-магазину, як каталоги товарів, кошик та модальні вікна.

Разом із цим, для стилізації інтерфейсу та забезпечення адаптивності під різні розміри екранів було обрано сучасні можливості CSS та частково JavaScript для специфічних взаємодій. Для автоматизації рутинних завдань фронтенду, таких як мініфікація стилів чи компіляція ресурсів, було обрано збирач Gulp, що пришвидшує підготовку статичних файлів [12].

Як систему керування базами даних обрано Microsoft SQL Server [13] у поєднанні з об’єктно-реляційним відображенням (ORM) Entity Framework Core [14]. SQL Server є надійним реляційним сховищем, що ідеально підходить для електронної комерції, де важлива цілісність транзакцій і збереження фінансових даних. Своєю чергою, EF Core дозволяє взаємодіяти з базою даних через класичні об’єкти C# із використанням LINQ-запитів, що абстрагує розробника

від написання «сирого» SQL-коду та значно спрощує підтримку схеми БД завдяки механізму міграцій.

Для забезпечення безпеки, автентифікації та авторизації користувачів було обрано ASP.NET Core Identity [15]. Цей модуль надає надійні механізми для безпечного зберігання облікових даних, управління ролями та генерації JWT-токенів доступу, що є стандартом для сучасних REST API.

Для безпосереднього написання коду та управління проєктом як інтегроване середовище розробки (IDE) використовується кросплатформний інструмент JetBrains Rider [16]. Вибір цього середовища зумовлений його високою продуктивністю, глибокою інтеграцією з екосистемою .NET та наявністю потужних засобів статичного аналізу коду і рефакторингу, що суттєво пришвидшує процес розробки застосунку.

Підсумовуючи, обраний технологічний стек (C#, ASP.NET Core, Blazor, Entity Framework Core, SQL Server, ASP.NET Core Identity, CSS, Gulp) повною мірою відповідає всім вимогам створення екологічного інтернет-магазину EcoMarket. Він враховує потребу в надійному бекенді та інтерактивному клієнтському інтерфейсі, забезпечуючи продуктивність, безпеку й масштабованість. Використання Blazor виділяється як ключове рішення для об’єднання логіки клієнта та сервера єдиною мовою, а ASP.NET Core у поєднанні з SQL Server створюють потужну й безпечну основу для обробки транзакцій та управління даними, що робить цей набір інструментів оптимальним для успішної реалізації проєкту.

Висновок до розділу 2

У другому розділі було проведено детальний аналіз вимог та здійснено комплексне проєктування програмного забезпечення для екологічного інтернет-магазину EcoMarket. Для візуалізації архітектури та логіки роботи системи було застосовано уніфіковану мову моделювання (UML). Розроблена діаграма класів дозволила чітко структурувати ключові сутності предметної області

(користувачі, товари, замовлення, екологічні теги) та визначити зв’язки між ними відповідно до принципів чистої архітектури (Clean Architecture). Створені діаграми прецедентів та діяльності наочно описали поведінкові сценарії клієнта, охопивши повний цикл взаємодії з платформою: від перегляду каталогу до оформлення замовлення та авторизації.

На основі сформованих архітектурних вимог було проведено аналіз існуючих інструментів та здійснено обґрунтований вибір технологічного стека. Як оптимальне рішення для побудови надійної клієнт-серверної системи було обрано платформу .NET. Використання мови програмування C#, серверного фреймворку ASP.NET Core та клієнтської технології Blazor забезпечує створення масштабованого та високопродуктивного вебзастосунку з єдиною кодовою базою. Надійне зберігання та обробка транзакційних даних реалізовані за допомогою Microsoft SQL Server та Entity Framework Core, а захист інформації гарантується модулем ASP.NET Core Identity.

Таким чином, розроблена проєктна документація, побудовані UML-моделі та обраний набір сучасних програмних засобів повністю відповідають поставленим цілям. Проведений у розділі аналіз формує надійний концептуальний та технологічний фундамент, необхідний для переходу до наступного етапу – безпосередньої програмної реалізації інформаційної системи та її подальшого тестування.

РОЗДІЛ 3
РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ ЕКОЛОГІЧНИХ ТОВАРІВ

3.1 Практична реалізація об’єкта проєктування

Практична реалізація інформаційної системи EcoMarket розпочалася з побудови надійного та масштабованого архітектурного фундаменту. Для забезпечення високої якості програмного коду, зручності його подальшої підтримки та можливості незалежного тестування окремих модулів, розробку було побудовано на принципах чистої архітектури (Clean Architecture) та багаторівневого розподілу обов’язків.

Такий підхід є сучасним стандартом промислової розробки, оскільки він дозволяє досягти слабкої зв’язності (loose coupling) між компонентами системи. Це означає, що зміни в інтерфейсі користувача чи зміна бази даних жодним чином не вплинуть на ключову бізнес-логіку застосунку. Відповідно до цих принципів, загальне рішення (Solution) у середовищі розробки було розділено на п’ять окремих, логічно ізольованих проєктів (рис. 3.1), кожен з яких має чітко визначену зону відповідальності.

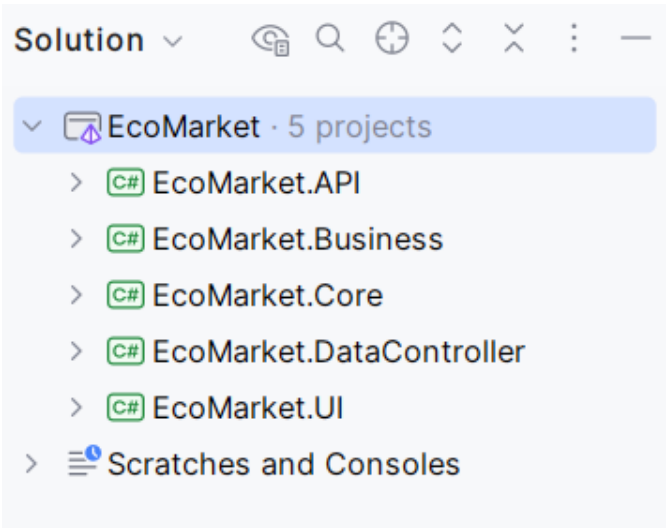

Рисунок 3.1 – Архітектурна структура загального рішення EcoMarket

EcoMarket.Core (Доменний рівень) – це фундаментальне ядро системи, яке не має жодних залежностей від інших рівнів чи зовнішніх фреймворків. Тут розміщені інтерфейси репозиторіїв, об’єкти передачі даних (DTO) та базові структури даних. На рисунку 3.2 зображено структуру проєкту EcoMarket.Core.

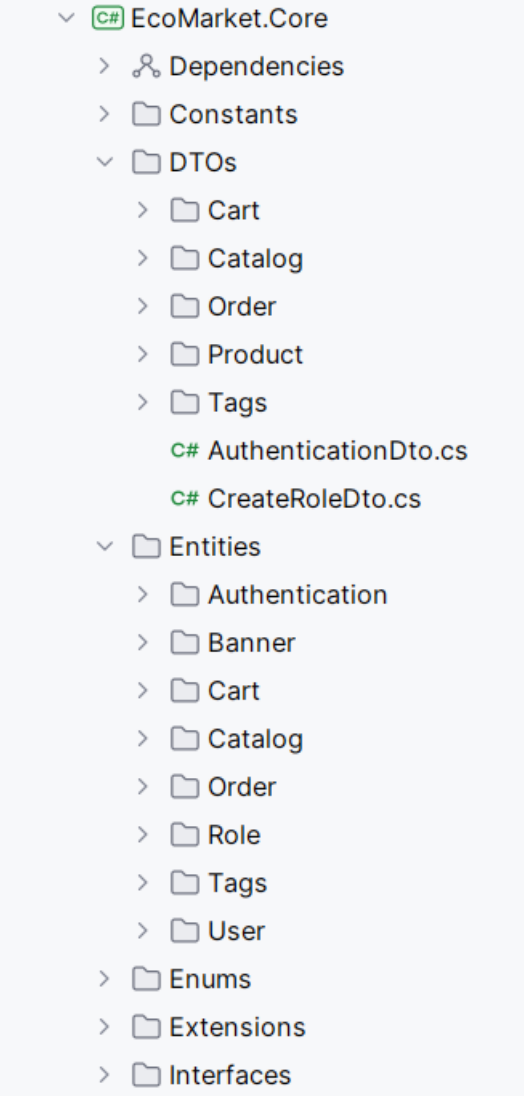

Рисунок 3.2 – Архітектурна структура проєкту EcoMarket.Core

EcoMarket.Business (Рівень бізнес-логіки) – це проєкт, який відповідає за реалізацію ключових бізнес-правил інтернет-магазину. Він містить сервіси, які інкапсулюють у собі логіку обробки даних та взаємодію між різними сутностями. Цей шар використовує абстракції з рівня Core, залишаючись незалежним від конкретних реалізацій доступу до даних. На рисунку 3.3 зображено структуру проєкту EcoMarket.Business.

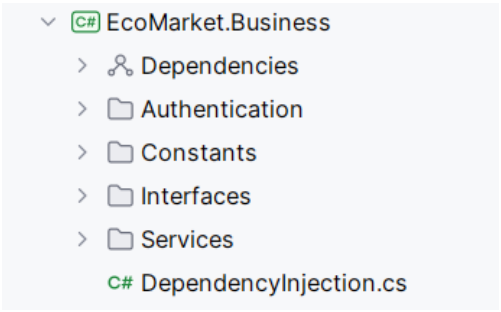

Рисунок 3.3 – Архітектурна структура проєкту EcoMarket.Business

EcoMarket.DataController (Рівень доступу до даних) – це інфраструктурний шар, який відповідає за фізичну взаємодію з реляційною базою даних. Як видно зі структури проєкту, тут налаштовано контекст бази даних (AppDbContext.cs), зберігаються файли міграцій (Migrations), моделі сутностей бази даних (Models, такі як ApplicationUser, ApplicationProduct) та класи-розширення для мапінгу даних (Extensions). На рисунку 3.4 зображено структуру проєкту EcoMarket. DataController.

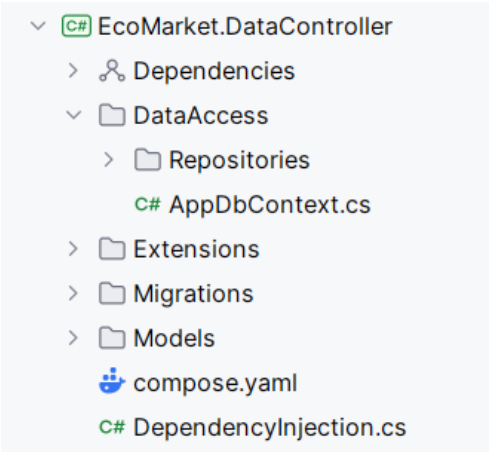

Рисунок 3.4 – Архітектурна структура проєкту EcoMarket.DataController

EcoMarket.API (Рівень представлення бекенду) – це головна точка входу серверної частини. Проєкт містить набір REST-контролерів (Controllers), розділених за функціоналом (наприклад, AuthController, ProductController, OrderController), які обробляють HTTP-запити від клієнтів. Також тут налаштовано маршрутизацію та конфігурацію впровадження залежностей у файлі Program.cs. На рисунку 3.5 зображено структуру проєкту EcoMarket. API.

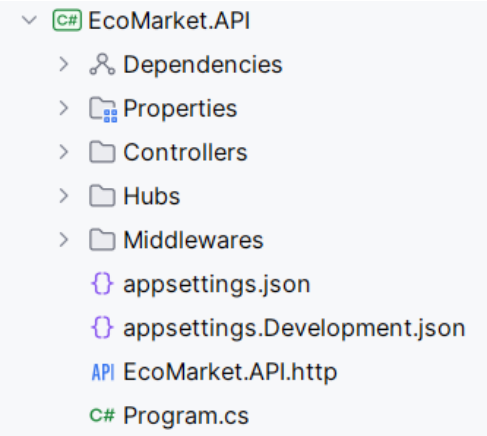

Рисунок 3.5 – Архітектурна структура проєкту EcoMarket.API

EcoMarket.UI (Клієнтський рівень) – проєкт користувацького інтерфейсу, розроблений на базі технології Blazor Server. Він має Components (де в свою чергу є реалізовані компоненти, сторінки та окремі елементи), Services для взаємодії з API, а також конфігураційні файли збирача Gulp (gulpfile.js) для автоматизації роботи зі статичними ресурсами (wwwroot). На рисунку 3.6 зображено структуру проєкту EcoMarket. UI.

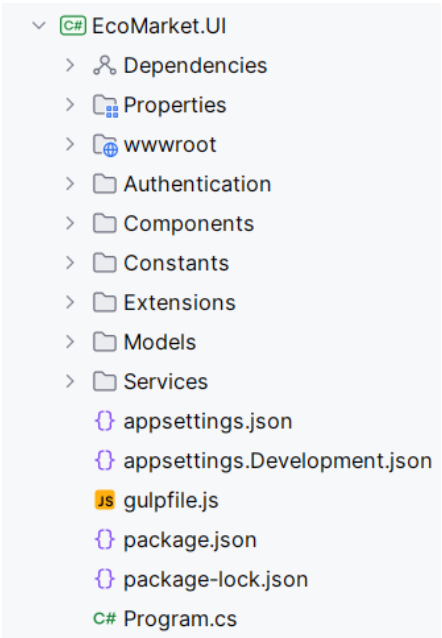

Рисунок 3.6 – Архітектурна структура проєкту EcoMarket.UI

Важливою архітектурною особливістю серверної частини (EcoMarket.API) є реалізація механізму глобального перехоплення винятків. Замість того, щоб

обробляти помилки в кожному окремому контролері (через блоки try-catch), у теці Middlewares створено спеціальний проміжний шар (Middleware).

Цей компонент перехоплює всі необроблені винятки, що виникають під час виконання HTTP-запиту, логує їх для подальшого аналізу, а також формує стандартизовану JSON-відповідь для клієнта (наприклад, повідомлення про помилку валідації чи відсутність доступу). Такий підхід гарантує, що сервер не припинить свою роботу через критичну помилку, а користувацький інтерфейс (EcoMarket.UI) завжди отримає коректний статус-код і зможе показати клієнту зрозуміле повідомлення замість технічного збою системи.

Ключовим етапом практичної реалізації бізнес-процесів стала розробка безпечної підсистеми ідентифікації користувачів. Архітектура взаємодії між клієнтською (UI) та серверною (API) частинами побудована з урахуванням найвищих стандартів веббезпеки.

На стороні сервера створено AuthController, який через сервіс IAuthService перевіряє облікові дані користувача за допомогою механізмів ASP.NET Core Identity. У разі успішної перевірки API генерує та повертає пару криптографічних ключів: короткостроковий токен доступу (Access Token) та довгостроковий токен оновлення (Refresh Token).

Для захисту системи від атак типу міжсайтового скриптингу (XSS) було прийнято архітектурне рішення не зберігати чутливі JWT-токени у відкритому локальному сховищі браузера (LocalStorage). Замість цього, у клієнтському проєкті EcoMarket.UI налаштовано власні проміжні маршрути обробки (наприклад, /auth/login). Логіка взаємодії працює наступним чином: коли користувач відправляє форму входу, клієнтський сервер Blazor звертається до віддаленого API, отримує згенеровані токени і на їх основі створює зашифрований авторизаційний Cookie-файл. Цей Cookie налаштовано з критично важливими прапорцями безпеки: HttpOnly (заборона доступу з боку JavaScript), SecurePolicy.Always (передача лише через HTTPS) та SameSiteMode.Strict (захист від підробки міжсайтових запитів CSRF).

Під час подальшої роботи з інтернет-магазином, налаштований HTTP-клієнт (MainApiHttpClient) за допомогою обробника JwtAuthorizationHandler автоматично витягує Access Token із захищеного Cookie та додає його до заголовка авторизації при зверненні до захищених ресурсів API.

Щоб користувачеві не доводилося постійно проходити повторну авторизацію після закінчення терміну дії короткострокового Access Token, у системі реалізовано механізм «тихого оновлення» (Silent Refresh). За допомогою викликів JavaScript (JS Interop), ініційованих у головному компоненті застосунку, система автоматично відстежує час життя токена. За дві хвилини до його закінчення браузер у фоновому режимі звертається до ендпоінту (/auth/refreshTokens). Сервер перевіряє чинність Refresh Token і непомітно оновлює авторизаційний Cookie, забезпечуючи безперервний та безпечний користувацький досвід.

Візуальним відображенням цієї складної внутрішньої взаємодії є Razor-компоненти інтерфейсу. Для доступу до існуючого облікового запису розроблено сторінку авторизації (Login), яка містить захищену форму для введення електронної пошти та пароля, а також опцію збереження активної сесії «Запам’ятати мене». Інтерфейс сторінки авторизації зображено на рисунку 3.7.

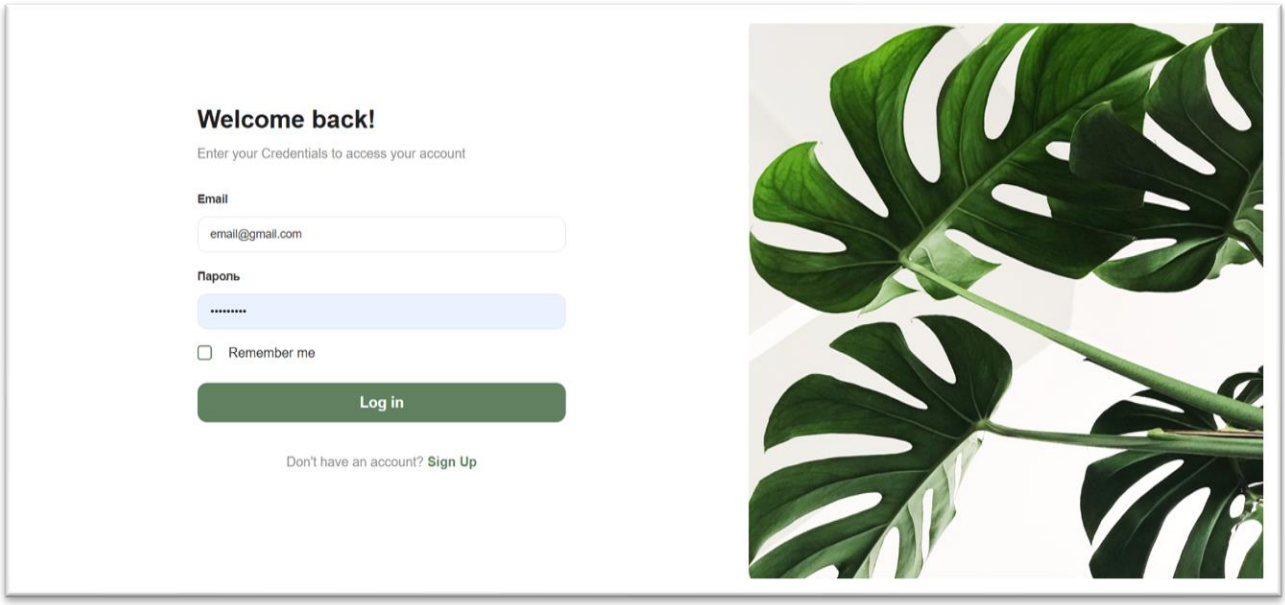

Рисунок 3.7 – Інтерфейс сторінки авторизації користувача

У випадку, якщо відвідувач ще не має власного профілю в системі EcoMarket, він може скористатися функцією створення нового облікового запису. Для цього розроблено сторінку реєстрації (Register), яка містить розширену форму введення даних. Окрім стандартних полів електронної пошти та пароля, форма вимагає обов’язкового підтвердження пароля, що мінімізує ризик друкарських помилок з боку клієнта. Зовнішній вигляд сторінки реєстрації представлено на рисунку 3.8.

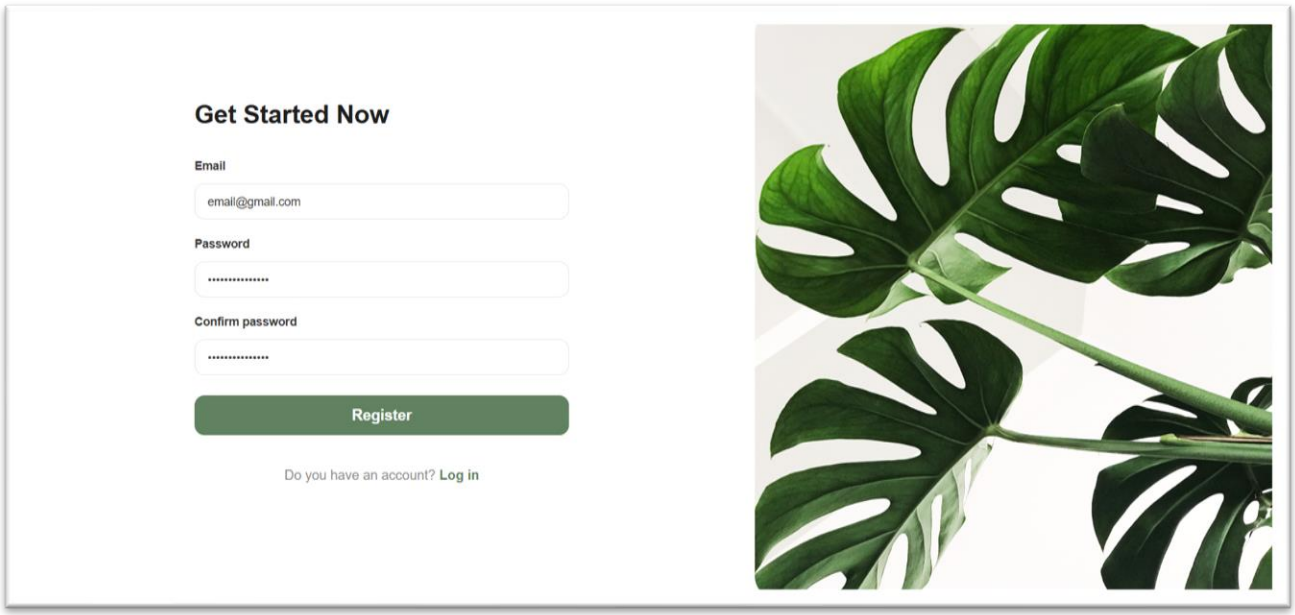

Рисунок 3.8 – Інтерфейс сторінки реєстрації нового користувача

Після успішного проходження валідації та відправки форми, система автоматично створює нового користувача у базі даних і здійснює його первинну авторизацію, одразу надаючи доступ до функцій інтернет-магазину.

Наступним важливим етапом стала розробка ядра електронної комерції – каталогу товарів та категорій. Для ефективної взаємодії з базою даних на рівні інфраструктурного проєкту EcoMarket.DataController було застосовано архітектурний патерн «Репозиторій» (Repository Pattern). Це дало змогу зручно організувати та приховати складні запити до бази даних за допомогою інструментів Entity Framework Core та LINQ.

Зокрема, для побудови ієрархічного дерева каталогу реалізовано алгоритм завантаження головних категорій разом із їхніми підкатегоріями. Щоб уникнути проблеми «N+1 запитів» та оптимізувати навантаження на СУБД, у методі застосовано техніку «жадібного завантаження» (Eager Loading) через оператор Include. Фрагмент коду класу CategoryRepository наведено в лістингу 3.1.

Лістинг 3.1 – Метод отримання головних категорій каталогу

```
public async Task<IEnumerable<Category>> GetRootCategoriesAsync(bool onlyActive = false)
{
   var query = context.Categories
      .Include(c => c.ChildCategories)
      .Include(c => c.Products)
      .Where(c => c.ParentCategoryId == null);

   if (onlyActive)
   {
      query = query.Where(c => c.IsActive);
   }

   var categories = await query.ToListAsync();
   return categories.Select(e => e.MapToCoreCategory());
}
```

кінець лістингу 3.1

Візуальне представлення каталогу та акційних пропозицій побудовано за допомогою інтерактивних Razor-компонентів. Завдяки можливості повторного використання компонентів забезпечується модульність кодової бази та одноманітність дизайну на всіх сторінках застосунку. Крім того, архітектура Blazor дозволяє динамічно оновлювати елементи інтерфейсу без повного перезавантаження сторінки, що значно підвищує швидкість відгуку системи.

Для зручної навігації головна сторінка відображає плитку з ключовими (батьківськими) категоріями магазину, а для просування спеціальних пропозицій використовується динамічний компонент банерів (ecoMarketBanner). Такий підхід дозволяє уникнути перевантаження інтерфейсу та формує зрозумілий шлях користувача. Інтерфейс головної сторінки зображено на рисунку 3.9.

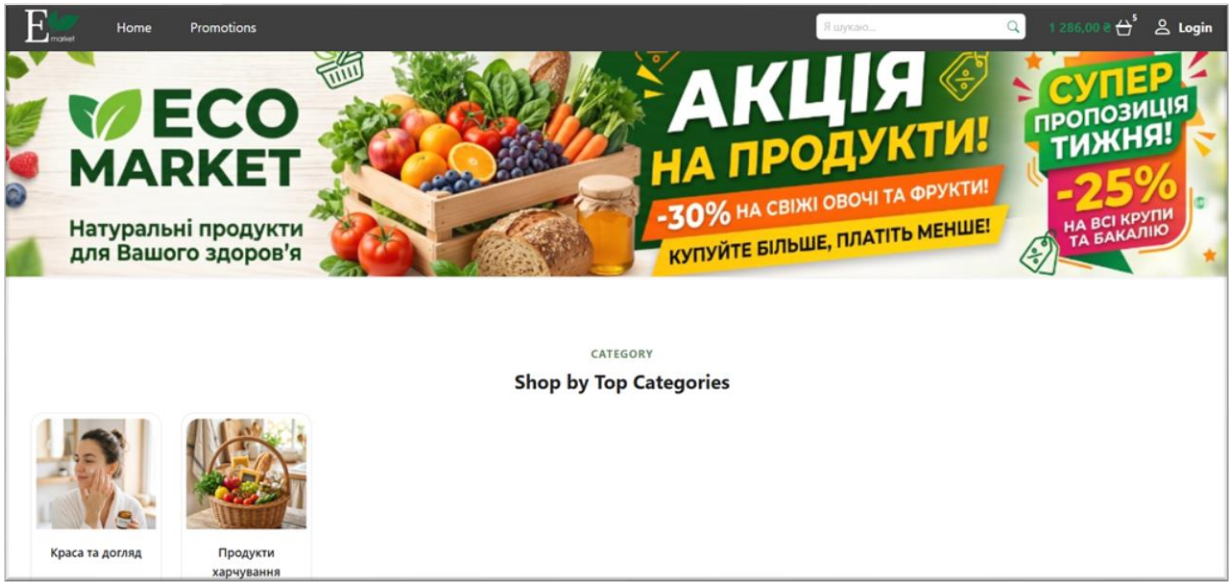

Рисунок 3.9 – Інтерфейс головної сторінки та категорій каталогу

Навігація каталогом побудована просто та інтуїтивно. Після вибору головної категорії користувач переходить на сторінку зі списком товарів. Тут відображаються доступні підкатегорії та загальний перелік усіх продуктів, що входять до обраного розділу (рис. 3.10).

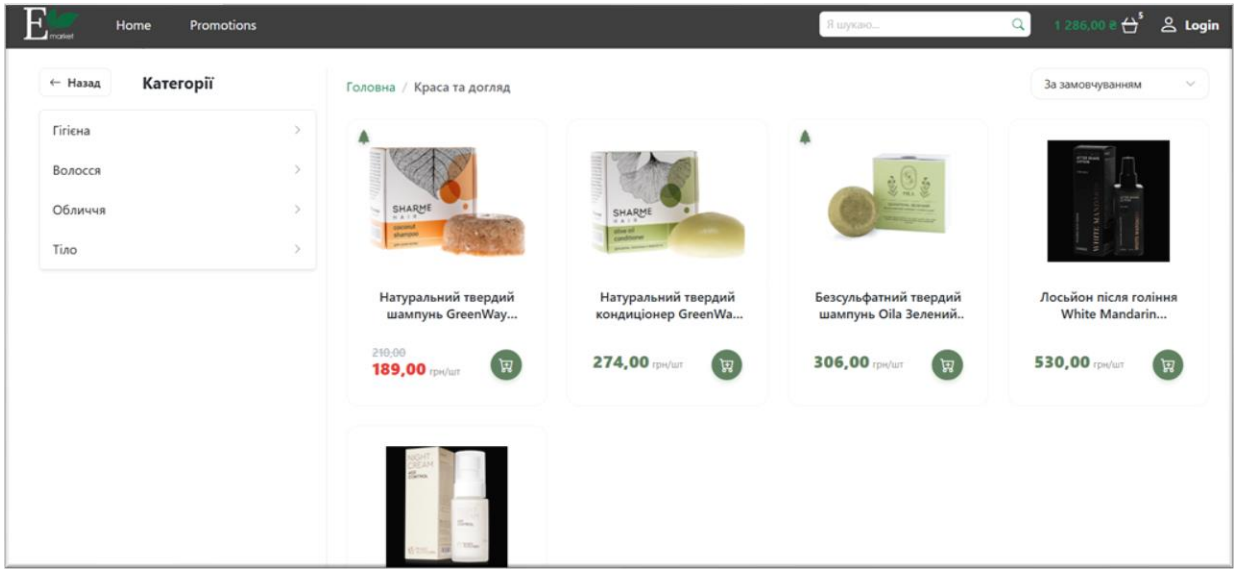

Рисунок 3.10 – Інтерфейс сторінки з усіма продуктами та підкатегоріями

Клієнт бачить картки товарів із зазначеними цінами та спеціальним екологічним маркуванням (еко-тегами). Прямо з цієї сторінки товар можна

швидко додати до кошика. Якщо ж користувачу потрібно дізнатися більше деталей, наприклад, переглянути склад, матеріали або інші характеристики, достатньо натиснути на картку товару. Після цього відкриється окрема сторінка з повною інформацією про конкретний продукт (рис. 3.11).

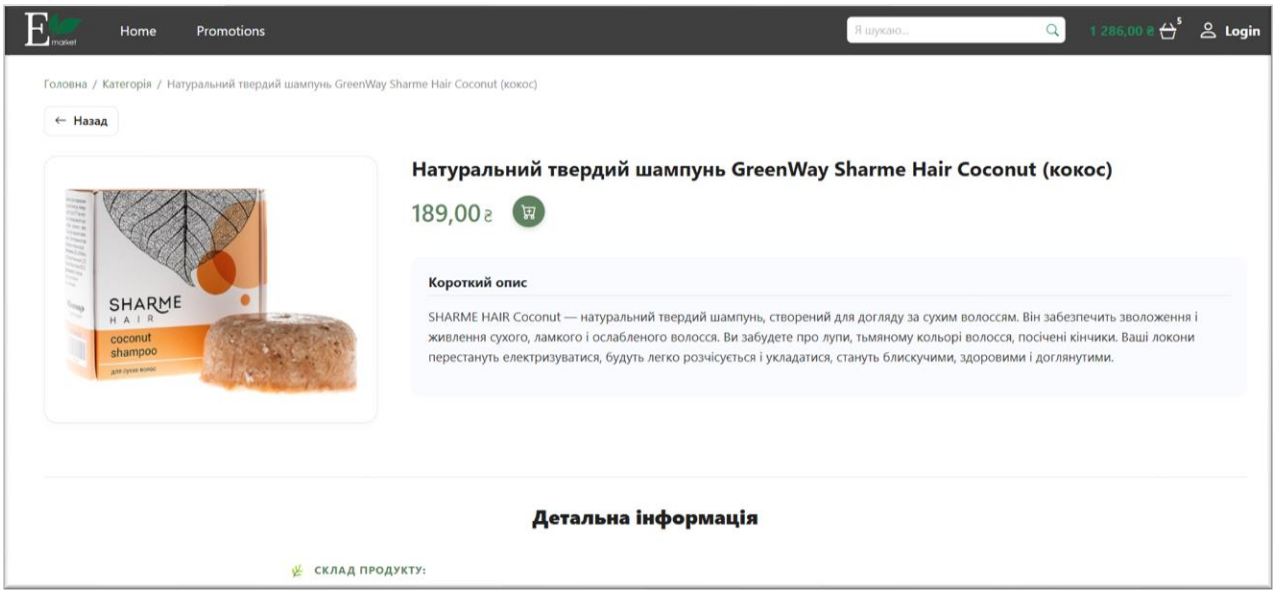

Рисунок 3.11 – Інтерфейс сторінки з інформацією про продукт

Визначившись із вибором, користувач може додати необхідні товари до кошика та перейти до оформлення покупки. Для максимальної зручності цей процес розділено на два логічні етапи, кожен з яких реалізовано у вигляді окремого інтерактивного модального вікна. Такий покроковий підхід набагато зручніший для користувача, адже йому не потрібно заповнювати велику та заплутану форму за один раз. Перехід між вікнами відбувається миттєво без перезавантаження сторінки, що робить процес оформлення швидким та комфортним.

Спочатку на екрані з’являється форма оформлення замовлення, де клієнт вводить свої контактні дані, коментарі та необхідну інформацію для доставлення (рис. 3.12).

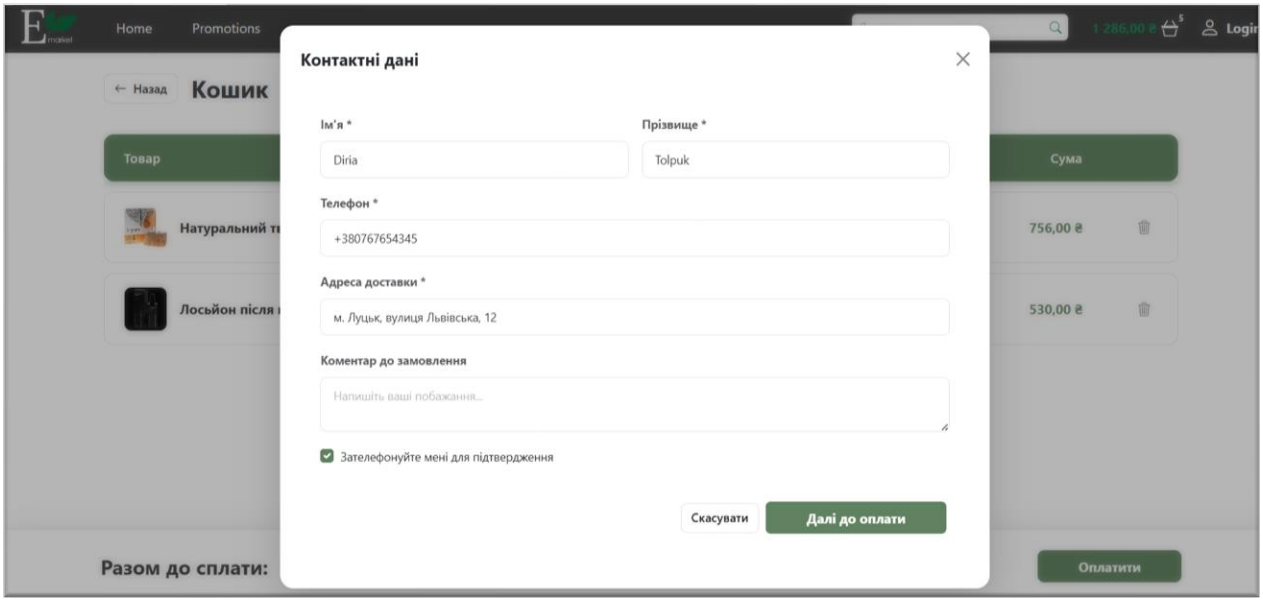

Рисунок 3.12 – Модальне вікно оформлення замовлення та даних доставлення

Після підтвердження цих даних користувач переходить до наступного кроку – вікна безпечної оплати. У цій формі клієнт заповнює реквізити платіжної картки для здійснення транзакції. Таке розділення на два окремих модальних вікна дозволяє не перевантажувати інтерфейс зайвими полями та утримує покупця у зрозумілому контексті кроків оплати (рис. 3.13).

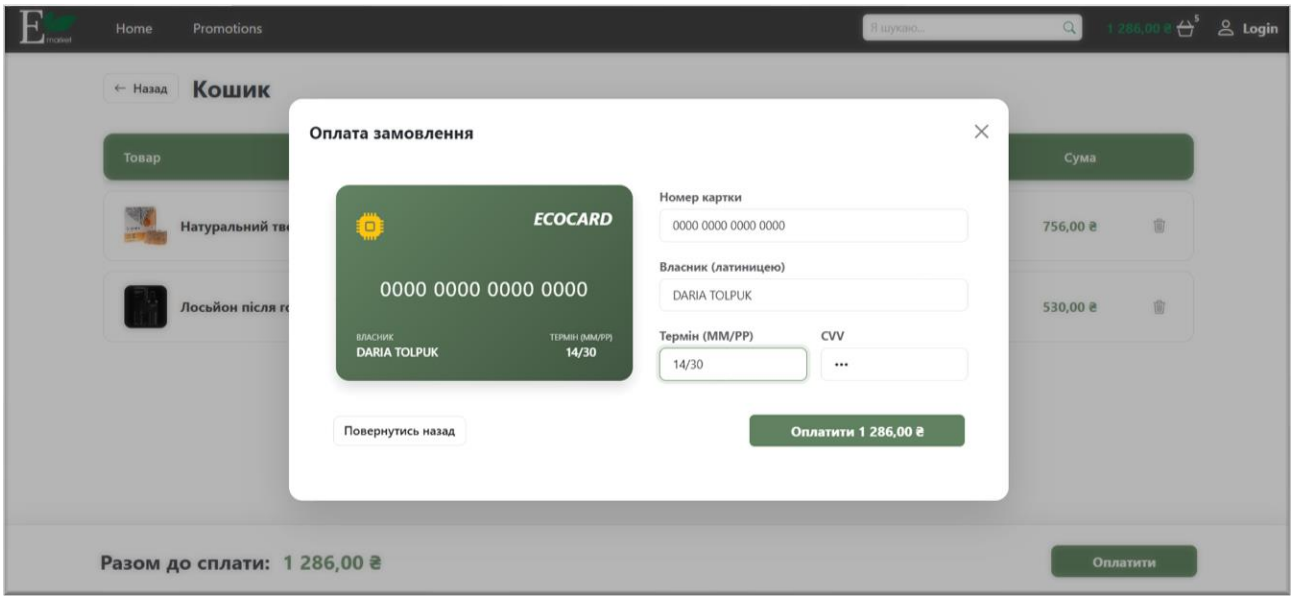

Рисунок 3.13 – Модальне вікно безпечної оплати замовлення платіжною карткою

Для авторизованих клієнтів у системі передбачено додатковий функціонал – доступ до закритого розділу з історією замовлень. У цій частині особистого кабінету користувач може переглядати повний перелік своїх покупок і деталізовану інформацію по кожній з них (склад кошика, загальну суму, дату створення). Важливою перевагою цього розділу є можливість відстежувати актуальний статус обробки свого замовлення (наприклад, «Нове», «Підтверджено», «Доставляється» чи «Виконано»). Це забезпечує повну прозорість взаємодії з магазином та дозволяє клієнту завжди бути в курсі стану своїх покупок (рис. 3.14).

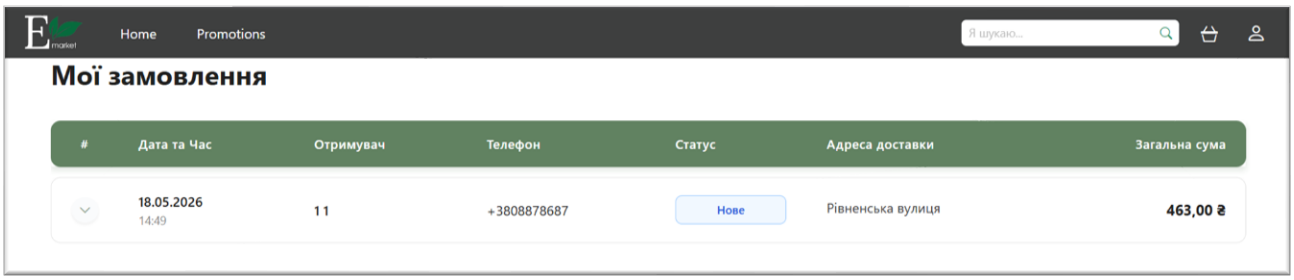

Рисунок 3.14 – Інтерфейс перегляду історії замовлень клієнта

Окрім функціоналу для покупців, вебзастосунок містить закриту панель управління для адміністраторів та менеджерів магазину. Цей розділ захищений системою авторизації на базі ролей і забезпечує повний контроль над контентом та налаштуваннями платформи. Звичайні користувачі не мають доступу до цієї панелі, що гарантує безпеку та захист внутрішніх даних магазину. Через цей інтерфейс адміністратори можуть легко керувати замовленнями, додавати нові екотовари, змінювати ціни та оновлювати категорії.

Для просування товарів та управління маркетинговими активностями адміністратор має змогу завантажувати, редагувати та налаштовувати динамічні банери, які згодом відображатимуться на головній сторінці магазину (рис. 3.15).

Він може додавати або оновлювати зображення, ставити послідовність показу банерів, а також приховувати його, щоб користувачі не бачили цей банер.

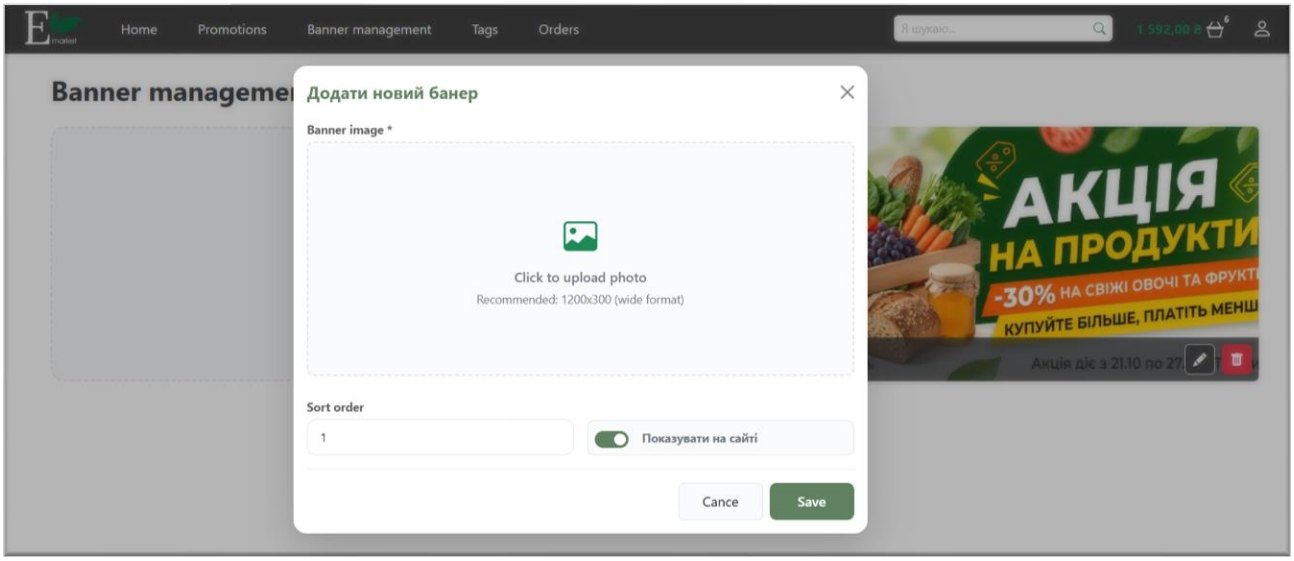

Рисунок 3.15 – Сторінка управління рекламними банерами

Важливою частиною адміністрування є керування концептуальною складовою магазину – екологічним маркуванням. Адміністратор може додавати екологічний тег, який надалі може додаватись при створенні продукту. Також адміністративний доступ надає інструменти для повного управління каталогом. Уповноважені співробітники можуть не лише створювати нові категорії та підкатегорії, а й безпосередньо редагувати товарний асортимент. Це включає додавання нових продуктів, оновлення їхніх характеристик, зміну цін чи фотографій, а також зняття з публікації позицій, які тимчасово недоступні.

3.2 Розробка бази даних

Проєктування й розробка бази даних для системи EcoMarket реалізується за методологією «Code-First», яка підтримується об’єктно-реляційним мапером Entity Framework Core. Цей підхід дозволяє розробникам визначати структуру даних, використовуючи мову програмування C#, шляхом створення класів моделей (сутностей), таких як ApplicationUser, ApplicationProduct і ApplicationOrder.

Ключовим елементом взаємодії з базою даних є контекстний клас AppDbContext, що наслідує функціонал IdentityDbContext для інтеграції функцій автентифікації. У цьому класі визначені набори даних (DbSet) для кожної таблиці, зокрема категорій, продуктів, банерів і замовлень. Крім того, у методі OnModelCreating активно застосовується Fluent API [17] для налаштування зв'язків між таблицями (наприклад, типу «один-до-багатьох») та для визначення правил каскадного видалення, забезпечуючи гнучкість і керованість у конфігурації бази даних. Фрагмент реалізації контексту бази даних наведено в лістингу 3.2.

Лістинг 3.2 – Конфігурація контексту бази даних AppDbContext

```
public class AppDbContext(DbContextOptions<AppDbContext> options)
    : IdentityDbContext<ApplicationUser, ApplicationRole,
Guid>(options)
{
    public DbSet<ApplicationCategory> Categories { get; set; }
    public DbSet<ApplicationProduct> Products { get; set; }
    public DbSet<ApplicationBanner> Banners { get; set; }
    public DbSet<ApplicationEcoTag> EcoTags { get; set; }
    public DbSet<ApplicationCartItem> CartItems { get; set; }
    public DbSet<ApplicationOrder> Orders { get; set; }
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
builder.ApplyConfigurationsFromAssembly(typeof(AppDbContext).Assem
bly);
        builder.Entity<ApplicationCategory>(entity =>
        {
            entity.HasOne(c => c.ParentCategory)
                .WithMany(c => c.ChildCategories)
                .HasForeignKey(c => c.ParentCategoryId)
                .OnDelete(DeleteBehavior.Restrict);
        });
        builder.Entity<ApplicationProduct>(entity =>
        {
            entity.HasOne(p => p.Category)
                .WithMany(c => c.Products)
                .HasForeignKey(p => p.CategoryId)
                .OnDelete(DeleteBehavior.Cascade);
        });
    }
}
```

кінець лістингу 3.2

Під час виконання програми взаємодія з даними реалізується через застосування патерну «Репозиторій». Замість ручного написання «сирих» SQL-запитів, вибірка даних виконується за допомогою мови інтегрованих запитів (LINQ), що забезпечує типобезпеку ще на етапі компіляції.

Наприклад, у класі ProductRepository реалізовано метод, який забезпечує отримання списку товарів відповідно до ідентифікатора категорії (ліст. 3.3). У межах цього методу реалізовано механізм «жадібного завантаження» за допомогою функції Include, яка дозволяє приєднати пов’язані екологічні теги. Додатково передбачається фільтрація результатів за критерієм активності товарів, що гарантує подальше уточнення вибірки.

Лістинг 3.3 – Метод вибірки товарів з бази даних за допомогою LINQ

```
public async Task<IEnumerable<Product>>
GetProductsByCategoryIdAsync(Guid categoryId, bool onlyActive =
false)
{
    var query = context.Products
        .Include(p => p.EcoTag)
        .Where(p => p.CategoryId == categoryId);

    if (onlyActive)
    {
        query = query.Where(p => p.IsActive);
    }

    var products = await query.ToListAsync();
    return products.Select(p => p.MapToCoreProduct());
}
```

кінець лістингу 3.3

Під час виклику методу ToListAsync() у фреймворку Entity Framework Core відповідний C#-код автоматично перетворюється на оптимізований SQL-запит із параметрами. Цей запит включає основні операції, такі як SELECT, FROM, INNER JOIN та WHERE, і надсилається до сервера Microsoft SQL Server. Такий підхід не лише сприяє ефективнішій розробці програмного забезпечення, але й забезпечує високий рівень захисту системи від вразливостей типу SQL-ін’єкцій.

Це досягається завдяки передачі змінних (наприклад, значення змінної categoryId) як параметрів запиту, що виключає потребу у формуванні запиту шляхом конкатенації рядків, тим самим мінімізуючи потенційні загрози для безпеки.

3.3 Захист інформаційно-комп'ютерної системи

Враховуючи специфіку електронної комерції та обробку персональних даних клієнтів, у розроблюваній системі EcoMarket реалізовано комплексний підхід до безпеки. Найважливішим завданням є надійне збереження облікових даних.

У системі паролі користувачів ніколи не зберігаються у базі даних у відкритому текстовому вигляді. Під час реєстрації нового облікового запису підсистема ASP.NET Core Identity застосовує механізм хешування з додаванням криптографічної «солі».

Наступним етапом гарантування інформаційної безпеки системи виступає захист сесій після успішного процесу авторизації. Після того як користувач вводить свої облікові дані, сервер не просто зберігає інформацію про сеанс, але також генерує спеціальний цифрову перепустку, відому як JSON Web Token (JWT). JWT містить важливу інформацію про користувача, зокрема його унікальний ідентифікатор та роль у системі.

Щоб запобігти створенню підроблених токенів, сервер накладає на кожен токен криптографічний підпис, сформований за допомогою секретного ключа, який належить виключно серверній стороні (у цьому випадку EcoMarket.API). У моменти, коли клієнт здійснює запит до захищених розділів вебсайту, таких як перегляд історії замовлень, сервер проводить перевірку цього криптографічного підпису. У разі спроби зловмисника модифікувати дані токена, наприклад, змінити роль із клієнта на адміністратора, криптографічний підпис буде анульовано. Це спричиняє автоматичне блокування запиту та унеможливлення доступу до захищених ресурсів.

Для гарантування максимальної безпеки та запобігання викраденню JWT-токенів у системі впроваджено механізм використання захищених cookies. Згенерований токен не зберігається у відкритих локальних сховищах браузера. Натомість клієнтський додаток (EcoMarket.UI) упаковує його в спеціальний cookie-файл із прапорцем HttpOnly. Цей прапорець технічно унеможливлює доступ JavaScript-скриптів до вмісту cookie-файлів, що ефективно нейтралізує ризики викрадення токенів через атаки типу міжсайтового скриптингу (XSS). Додатковий прапорець Secure забезпечує передачу конфіденційних даних виключно за допомогою шифрованого з’єднання (HTTPS), що істотно підвищує надійність системи.

3.4 Тестування та налагодження інформаційно-комп’ютерної системи

Налаштування серверної частини (API) та клієнтського інтерфейсу (UI) застосунку здійснювалося за допомогою вбудованих інструментів інтегрованого середовища розробки JetBrains Rider. Використання точок зупину (Breakpoints) дало змогу виконувати код покроково, аналізувати стек викликів (Call Stack) і відстежувати стан змінних у реальному часі, що стало вкрай важливим для розробки складних алгоритмів кошика та підрахунку загальної вартості замовлень.

Для моніторингу роботи системи на серверному рівні було впроваджено стандартний механізм логування ILogger<T>. Наприклад, у контролері автентифікації (AuthController) всі критичні винятки, які виникають під час спроб входу або реєстрації, перехоплюються і записуються у системний журнал (ліст. 3.4). Це забезпечує можливість розробникам швидко реагувати на несподівані збої.

Лістинг 3.4 – Фрагмент коду з логуванням ILogger<T>

```
catch (Exception ex)
{
    logger.LogError(ex, ex.Message);
```

```
    return StatusCode(StatusCodes.Status500InternalServerError);
}
```

кінець лістингу 3.4

Крім того, як зазначалося у попередніх розділах, для уникнення «падіння» сервера через необроблені помилки, було реалізовано глобальний Middleware обробки винятків, який гарантує повернення клієнту стандартизованої відповіді зі статусом помилки.

Оскільки архітектура проєкту передбачає чітке розділення на бекенд та фронтенд, тестування REST API здійснювалося ізольовано від клієнтського інтерфейсу. Для цього було використано специфікацію OpenAPI та інтерактивний інструмент Swagger UI.

Swagger автоматично генерує візуальну документацію для всіх доступних ендпоінтів контролерів (наприклад, ProductController, OrderController). За допомогою цього інструменту проводилося мануальне тестування HTTP-запитів: перевірялася коректність вибірки даних із бази даних, створення та працездатність механізмів. Особлива увага приділялася тестуванню системи безпеки – перевірці доступу до захищених маршрутів без валідного JWT-токена (отримання статусу 401 Unauthorized).

На рівні клієнтського застосунку (EcoMarket.UI) проводилося комплексне функціональне тестування (Functional Testing) та тестування інтерфейсу користувача (UI Testing). У рамках перевірки навігації та адаптивності за допомогою інструментів Chrome DevTools досліджувалася коректність відображення каталогу, модальних вікон та рекламних банерів на екранах із різною роздільною здатністю.

Окрему увагу було приділено управлінню станом (State Management), зокрема тестуванню логіки кошика: перевірялася коректність зміни кількості товарів, динамічний перерахунок загальної вартості замовлення та надійність збереження поточного стану. Важливим етапом стала перевірка підсистеми безпеки та авторизації, що охоплювала весь життєвий цикл сесії користувача. Було протестовано процеси успішної автентифікації, безпечне збереження

токенів доступу у захищених HttpOnly файлах cookie, безперебійну роботу механізму «тихого оновлення» (Silent Refresh) та гарантоване очищення локальних даних під час виходу з облікового запису (Logout). Крім того, здійснювалася ретельна перевірка обробки введення, що включала клієнтську валідацію форм оформлення замовлення та реєстрації. Система тестувалася на коректну реакцію щодо порожніх полів, перевірку формату електронної пошти та збігу паролів безпосередньо в браузері, ще до моменту відправки мережевого запиту на сервер.

Висновок до розділу 3

У третьому розділі проведено повну практичну реалізацію інформаційної системи екологічного інтернет-магазину EcoMarket. Основою розробки стали принципи чистої архітектури (Clean Architecture), завдяки яким загальне рішення було структуровано на п’ять незалежних рівнів. Такий підхід забезпечив високу масштабованість, мінімальну взаємозалежність компонентів (серверного API та клієнтського UI) і зручність в обслуговуванні системи.

Процес проєктування та розгортання бази даних здійснювався за методологією «Code-First» з використанням Entity Framework Core. Інтеграція патерну «Репозиторій» у поєднанні з оптимізованими LINQ-запитами забезпечила швидкий і безпечний доступ до даних, виключивши ризики, пов’язані із SQL-ін’єкціями. Окрім цього, особливу увагу приділено кібербезпеці: реалізовано надійне хешування паролів із застосуванням криптографічної «сіллі» та впроваджено сучасну систему автентифікації, базовану на JWT-токенах, які додатково захищені через HttpOnly та Secure cookie.

Клієнтську частину застосунку розроблено за допомогою технології Blazor, що дозволило створити інтерактивний і зручний інтерфейс. У межах цього вирішено завдання навігації каталогом, динамічного управління кошиком, безпечного оформлення замовлень за допомогою модальних вікон, а також

розроблено повноцінну панель адміністрування для керування контентом і еко-тегами.

На завершальних етапах проєкту виконано ретельне налагодження та тестування. Ізольована перевірка REST API у середовищі Swagger UI, а також комплексне функціональне тестування клієнтського інтерфейсу підтвердили коректність реалізації всіх бізнес-процесів. Як результат, створено надійний, стабільний і стійкий до навантажень програмний продукт, який відповідає всім вимогам і готовий до впровадження.

ВИСНОВКИ

У кваліфікаційній роботі було досягнуто поставлену мету, яка полягала у проєктуванні та розробці інформаційної системи екологічного інтернет-магазину EcoMarket з використанням сучасного технологічного стека .NET та принципів чистої архітектури (Clean Architecture).

На початковому етапі було здійснено глибокий аналіз предметної області електронної комерції та сучасних тенденцій на ринку екологічних товарів. У ході цього аналізу вдалося виявити ключові проблеми наявних рішень та визначити актуальну потребу ринку у створенні спеціалізованої платформи, яка не лише автоматизує процеси продажу, а й популяризує свідоме споживання.

На основі проведеного дослідження сформовано детальні функціональні та нефункціональні вимоги до майбутнього вебзастосунку. Це дозволило закласти чітке технічне бачення продукту, визначити межі системи, описати логіку взаємодії користувачів із платформою та встановити суворі критерії щодо її продуктивності, масштабованості й адаптивності інтерфейсу.

У роботі детально обґрунтовано вибір сучасного стека технологій для реалізації клієнтської, серверної частин та бази даних. Як базову платформу було обрано екосистему .NET (мову C#), для побудови серверної частини та REST API – ASP.NET Core, для розробки інтерактивного користувацького інтерфейсу – фреймворк Blazor, а взаємодію з базою даних Microsoft SQL Server реалізовано через ORM-фреймворк Entity Framework Core.

На етапі проєктування виконано комплексне моделювання загальної архітектури системи та логічної структури реляційної бази даних за допомогою уніфікованої мови UML. Створені діаграми прецедентів, діяльності та класів дозволили візуалізувати алгоритми обробки даних, а за методологією «Code-First» розроблено нормалізовану базу даних, що забезпечує цілісність збереження інформації про користувачів, товари, категорії та замовлення.

У процесі практичної реалізації створено повноцінний клієнт-серверний вебзастосунок, який поєднує REST API, інтерфейс користувача та бізнес-логіку.

На серверному рівні розроблено логіку обробки замовлень, управління каталогом та кошиком, а на клієнтському рівні побудовано сучасний, простий та інтуїтивно зрозумілий інтерфейс користувача з динамічним відображенням даних без перезавантаження сторінок.

Окрему увагу приділено впровадженню надійних механізмів інформаційної безпеки, зокрема процесам автентифікації та авторизації користувачів. Безпеку реалізовано на базі JWT-токенів із захистом від XSS-атак за допомогою механізму HttpOnly файлів cookie, а також застосовано криптографічне хешування паролів для захисту персональних даних від несанкціонованого доступу.

Завершальним етапом роботи стало проведення комплексного тестування і налагодження розробленого програмного забезпечення. Результати як ізольованого тестування серверних ендпоінтів через інструментарій Swagger, так і ручного UI-тестування користувацького інтерфейсу підтвердили високу якість написаного коду, коректність обробки бізнес-логіки та стійкість системи до помилок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ecocorn. Environmental awareness: how it changes through generations. URL: https://surl.lu/sbhvqk (дата звернення: 16.02.2026).

2. Interkassa. E-commerce. URL: https://interkassa.com/blog/shho-take-elektronna-komerciya-e-commerce-dlya-pochatkivciv (дата звернення: 18.02.2026).

3. Mas Agency. The Future of E-commerce. URL: https://surl.li/aqbsua (дата звернення: 21.02.2026).

4. Екомаркет. Екомаркет. URL: https://www.eko.com.ua/ (дата звернення: 22.02.2026).

5. ZakazUa. Eko zakaz. URL: https://eko.zakaz.ua/uk/ (дата звернення: 24.02.2026).

6. Auth0. Контроль доступу на основі ролей. URL: https://auth0.com/docs/manage-users/access-control/rbac (дата звернення: 15.04.2026).

7. Medium. Understanding Data Transfer Objects (DTOs) in C# .NET. URL: https://surl.li/cfqvak (дата звернення: 16.04.2026).

8. Microsoft. C# language documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 18.04.2026).

9. Microsoft. .NET documentation. URL: https://learn.microsoft.com/en-us/dotnet/ (дата звернення: 23.04.2026).

10. Cool code company. Плюси та мінуси C#. URL: https://surl.li/lpipyn (дата звернення: 25.04.2026).

11. Microsoft. ASP.NET Core Blazor. URL: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-10.0 (дата звернення: 01.05.2026).

12. Gulp js. Gulp js documentation URL: https://gulpjs.com/ (дата звернення: 02.05.2026).

13. Microsoft. SQL Server technical documentation. URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17 (дата звернення: 04.05.2026).

14. Microsoft. Entity Framework Core. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 05.05.2026).

15. Microsoft. Introduction to Identity on ASP.NET Core. URL: https://surl.li/wbmyez (дата звернення: 08.05.2026).

16. JetBrains Rider. JetBrains Rider documentation URL: https://www.jetbrains.com/help/rider/Introduction.html (дата звернення: 11.05.2026).

17. Microsoft. API Fluent. URL: https://learn.microsoft.com/ru-ru/ef/ef6/modeling/code-first/fluent/types-and-properties (дата звернення: 17.05.2026).