МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії

програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОННИХ ПРИСТРОЇВ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Даніїл КУЛИК**

«__» _____ 20__ р.

Керівник роботи

PhD,

доцентка (б.в.з) _____ **Катерина АНТІПОВА**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Кулик Даніїл

1. Тема кваліфікаційної роботи «Вебзастосунок інтернет-магазину електронних пристроїв» затверджена наказом ректора ЧНУ ім. Петра Могили № _349_ від «26» грудня 2025 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Очікуваним результатом є вебзастосунок інтернет-магазину електронних пристроїв, що забезпечує зручний процес перегляду каталогу товарів, їх порівняння, додавання до кошика та оформлення замовлень як для зареєстрованих користувачів, так і для гостей, а також надає адміністратору інструменти для керування асортиментом, замовленнями та користувачами системи.

4. Перелік питань, що підлягають розробці:

− дослідити предметну область електронної комерції електронних пристроїв та провести аналіз існуючих аналогів вебзастосунків;

− провести порівняльний аналіз існуючих інтернет-магазинів електроніки для визначення функціональних та технічних особливостей;

− сформувати специфікацію вимог до програмного забезпечення на основі проведеного аналізу;

− спроєктувати архітектуру вебзастосунку та розробити структуру бази даних для зберігання інформації про товари, користувачів, замовлення та кошики;

− розробити серверну частину вебзастосунку з реалізацією бізнес-логіки управління каталогом, кошиком та обробки замовлень;

− розробити клієнтську частину вебзастосунку з адаптивним інтерфейсом користувача;

− провести тестування клієнтської та серверної частин системи та перевірити її коректність і стабільність роботи.

5. Перелік графічних матеріалів: Презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Вебзастосунок інтернет-магазину електронних пристроїв**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	26.12.2025	18.02.2026	**виконано**
2.	Огляд літератури за темою роботи	01.01.2026	05.03.2026	**виконано**
3.	Складання календарного плану КБР	16.02.2026	18.02.2026	**виконано**
4.	Аналіз предметної області	20.02.2026	10.03.2026	**виконано**
5.	Розробка проєктних рішень	05.03.2026	20.03.2026	**виконано**
6.	Моделювання та конструювання ПЗ	15.02.2026	30.02.2026	**виконано**
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.03.2026	20.05.2026	**виконано**
8.	Відгук керівника КБР	22.05.2026	27.05.2026	**виконано**
9.	Оформлення КБР та презентації	13.05.2026	25.05.2026	**виконано**
10.	Попередній захист	26.06.2026	26.06.2026	**виконано**
11.	Рецензування	02.06.2026	08.06.2026	**виконано**
12.	Завершення оформлення КБР та презентації	08.06.2026	11.06.2026	**виконано**
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Даніїл КУЛИК**

«__» _____ 20__ р.

Керівник роботи

PhD,

доцентка (б.в.з) _____ **Катерина АНТІПОВА**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Вебзастосунок інтернет-магазину електронних пристроїв

Здобувач 408 гр.: Кулик Даніїл

Керівник: PhD, доцентка (б.в.з) Антіпова Катерина

Стрімкий розвиток електронної комерції та зростання попиту на онлайн-покупки зумовлюють необхідність створення зручних і функціональних вебзастосунків для продажу електронних пристроїв. Аналіз існуючих рішень (ROZETKA, COMFY, FOXTROT, TELEMART, CITRUS) виявив низку недоліків: перевантаженість інтерфейсу, складність навігації, недостатню оптимізацію мобільних версій та обмежені можливості управління замовленнями. Це обумовлює актуальність розробки сучасного вебзастосунку, який забезпечить ефективний процес вибору, порівняння та придбання товарів.

Об'єктом роботи є процеси електронної комерції.

Предметом роботи є програмні засоби та технології розробки вебзастосунку з підтримкою управління каталогом товарів, обробки замовлень та забезпечення взаємодії з користувачами.

Метою роботи є розробка інтернет-магазину електронних пристроїв для підвищення зручності процесів вибору, порівняння та придбання товарів користувачами.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність розробки вебзастосунку інтернет-магазину електронних пристроїв, визначено мету, завдання, об'єкт та предмет дослідження.

У першому розділі проведено аналіз предметної області електронної комерції та існуючих аналогів інтернет-магазинів електроніки, сформовано специфікацію вимог до програмного забезпечення на основі виявлених переваг та недоліків існуючих рішень.

Другий розділ присвячено моделюванню структури вебзастосунку, визначенню функціональних та нефункціональних вимог, а також опису ролей користувачів системи (клієнт, адміністратор, гість).

У третьому розділі описано архітектуру вебзастосунку, наведено UML-діаграми (діаграма класів, діаграма розгортання, діаграми використання), описано структуру бази даних ElectronicStoreDB на основі SQL Server 2022 та Entity Framework Core.

У четвертому розділі описано процес програмної реалізації серверної частини на ASP.NET Core та клієнтської частини з використанням HTML, CSS, Bootstrap і JavaScript, а також представлено результати тестування функціональності системи.

У висновках узагальнено результати виконаної роботи, підтверджено відповідність розробленого вебзастосунку поставленим вимогам та окреслено можливі напрями подальшого розвитку системи.

Кваліфікаційна робота викладена на 65 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 20 найменувань. Праця містить 2 таблиць та 35 рисунків.

Ключові слова: вебзастосунок, електронна комерція, електронні пристрої, інтернет-магазин, ASP.NET Core, Entity Framework Core, SQL Server.

ABSTRACT

to the qualifying bachelor's thesis

Web application for an online store of electronic devices

Student of 408 group: Kulyk Daniil

Supervisor: PhD, Associate Professor Kateryna Antipova

The rapid development of e-commerce and the growing demand for online shopping necessitate the creation of convenient and functional web applications for selling electronic devices. An analysis of existing solutions (ROZETKA, COMFY, FOXTROT, TELEMART, CITRUS) revealed a number of shortcomings: overloaded interfaces, complex navigation, insufficient optimization of mobile versions and limited order management capabilities. This determines the relevance of developing a modern web application that will ensure an efficient process of selecting, comparing and purchasing goods.

The object of the work is the processes of e-commerce.

The subject of the work is the software tools and technologies for developing a web application with support for product catalog management, order processing and user interaction.

The purpose of the work is to develop an online store of electronic devices to improve the convenience of the processes of selecting, comparing and purchasing products by users.

The qualification work consists of an introduction, 4 chapters, conclusions and a list of references.

The introduction substantiates the relevance of developing a web application for an online store of electronic devices, and defines the purpose, objectives, object and subject of research.

The first chapter analyzes the subject area of e-commerce and existing analogues of electronics online stores, and forms a software requirements specification based on the identified advantages and disadvantages of existing solutions.

The second chapter is devoted to modeling the structure of the web application, defining functional and non-functional requirements, as well as describing the roles of system users (client, administrator, guest).

The third chapter describes the architecture of the web application, provides UML diagrams (class diagram, deployment diagram, use case diagrams), and describes the structure of the ElectronicStoreDB database based on SQL Server 2022 and Entity Framework Core.

The fourth chapter describes the process of software implementation of the server-side using ASP.NET Core and the client-side using HTML, CSS, Bootstrap and JavaScript, and presents the results of functionality testing.

The conclusions summarize the results of the completed work, confirm the compliance of the developed web application with the set requirements and outline possible directions for further development of the system.

The qualification work is presented on 65 pages of typewritten text, consists of an introduction, 4 chapters, general conclusions, a list of references with 20 titles. The work contains 2 tables and 35 figures.

Keywords: ASP.NET Core, e-commerce, electronic devices, Entity Framework Core, online store, SQL Server, web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3
ВСТУП 4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ 6
1.1 Аналіз предметної області електронної комерції 6
1.2 Структурні та функціональні особливості інтернет-магазинів електроніки 8
1.3 Огляд та аналіз існуючих програмних рішень 9
1.4 Постановка завдань розробки 14
Висновки до розділу 1 16
2 МОДЕЛЮВАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ЗАСТОСУНКУ 17
2.1 Аналіз інструментарію та методів розробки 17
2.1.1 Серверна частина (Backend) 17
2.1.2 Клієнтська частина (Frontend) 19
2.2 Моделі та методи реалізації застосунку 19
2.3 Специфікація вимог до програмного забезпечення 21
Висновки до розділу 2 29
3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 30
3.1 UML-моделювання системи 30
3.2 Проєктування бази даних 37
3.3 Архітектура програмного забезпечення 40
3.4 Опис інтерфейсів застосунку 42
Висновки до розділу 3 46
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ 47
4.1 Структура проєкту 47
4.2 Реалізація серверної частини 49
4.3 Керівництво користувача 52
4.4 Тестування вебзастосунку 59
Висновки до розділу 4 61
ВИСНОВКИ 62
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 64

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
ПЗ	–	програмне забезпечення
СКБД	–	система керування базами даних
API	–	Application Programming Interface
ASP.NET	–	Active Server Pages .NET
BCrypt	–	алгоритм хешування паролів
CORS	–	Cross-Origin Resource Sharing
CRUD	–	Create, Read, Update, Delete
CSS	–	Cascading Style Sheets
EF Core	–	Entity Framework Core
HTML	–	HyperText Markup Language
HTTPS	–	HyperText Transfer Protocol Secure
JSON	–	JavaScript Object Notation
JWT	–	JSON Web Token
.NET	–	платформа розробки програмного забезпечення Microsoft
REST	–	Representational State Transfer
SPA	–	Single Page Application
SQL	–	Structured Query Language
SSMS	–	SQL Server Management Studio
UI	–	User Interface
URL	–	Uniform Resource Locator
UX	–	User Experience
XSS	–	Cross-Site Scripting

ВСТУП

Стрімкий розвиток цифрових технологій та зростання попиту на онлайн-покупки призвели до змінення традиційних моделей роздрібної торгівлі. Електронна комерція стала невід'ємною частиною сучасного споживчого ринку, надаючи користувачам можливість здійснювати покупки в будь-який час та з будь-якого місця. Ринок електронних пристроїв та побутової техніки займає одну з провідних позицій серед онлайн-продажів, що зумовлено високою вартістю товарів, необхідністю детального ознайомлення з характеристиками перед покупкою та постійним оновленням асортименту новими моделями.

Сучасні споживачі електронних пристроїв потребують зручних інструментів для порівняння технічних характеристик, перегляду відгуків інших покупців, відстеження наявності товарів та оперативного оформлення замовлень. Водночас, підприємства роздрібної торгівлі прагнуть автоматизувати процеси управління асортиментом, обробки замовлень та взаємодії з клієнтами для підвищення ефективності бізнес-процесів.

Аналіз існуючих вебзастосунків інтернет-магазинів електроніки (ROZETKA, COMFY, FOXTROT, TELEMART, CITRUS) виявив низку проблем: перевантаженість інтерфейсу рекламними елементами, складність навігації через надмірну кількість категорій, повільну роботу системи фільтрації, недостатню оптимізацію мобільних версій, а також обмежені можливості персоналізації пропозицій для користувачів. Ці недоліки негативно впливають на користувацький досвід та знижують конверсію відвідувачів у покупців.

Актуальність цієї роботи полягає у необхідності створення сучасного вебзастосунку інтернет-магазину електронних пристроїв, який забезпечить швидкий та зручний доступ до товарів, ефективну систему пошуку та фільтрації, оптимізований користувацький інтерфейс для різних типів пристроїв, а також надійну обробку замовлень. Розробка такого застосунку дозволить підвищити задоволеність користувачів, спростити процес вибору та придбання електронних

пристроїв, а також забезпечить ефективне управління комерційними операціями для адміністраторів системи.

Метою роботи є розробка інтернет-магазину електронних пристроїв для підвищення зручності процесів вибору, порівняння та придбання товарів користувачами.

Для досягнення визначеної мети необхідно вирішити такі завдання:

– провести аналіз предметної області електронної комерції та існуючих рішень інтернет-магазинів електроніки;

– сформувати специфікацію вимог до програмного забезпечення на основі аналізу функціональних потреб користувачів та адміністраторів;

– спроєктувати архітектуру вебзастосунку з використанням сучасних патернів проєктування;

– розробити структуру бази даних для зберігання інформації про товари, користувачів, замовлення та кошики;

– розробити серверну частину вебзастосунку з реалізацією бізнес-логіки обробки замовлень та управління каталогом;

– розробити клієнтську частину вебзастосунку з адаптивним користувацьким інтерфейсом;

– виконати тестування функціональності розробленої системи та перевірити її стабільність роботи.

Об'єктом роботи є процеси електронної комерції.

Предметом роботи є програмні засоби та технології розробки вебзастосунку з підтримкою управління каталогом товарів, обробки замовлень та забезпечення взаємодії з користувачами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

У цьому розділі досліджено предметну область електронної комерції та проаналізовано наявні програмні рішення у сфері інтернет-торгівлі електронними пристроями. Аналіз спрямовано на виявлення структурних і функціональних особливостей об'єкта дослідження, а також на оцінку актуального стану ринку програмного забезпечення. Отримані результати слугують підґрунтям для формулювання вимог до вебзастосунку, що відповідає реаліям сучасного цифрового ринку.

1.1 Аналіз предметної області електронної комерції

Електронна комерція сьогодні є одним із найбільш динамічних секторів цифрової економіки. Під цим поняттям об'єднано сукупність комерційних процесів, реалізованих засобами інтернет-технологій – купівлю та продаж товарів і послуг, інформаційний обмін між суб'єктами ринку, обробку платежів і управління логістикою. Ринок електронних пристроїв займає в онлайн-торгівлі особливе місце. Це пояснюється кількома чинниками: значна вартість товарів, потреба споживача у детальному ознайомленні з технічними характеристиками перед ухваленням рішення про купівлю, а також постійне оновлення асортименту, яке практично унеможливлює вичерпне офлайн-представлення.

З архітектурної точки зору, інтернет-магазин електроніки є складною багаторівневою системою. Її каркас формують кілька взаємопов'язаних підсистем:

– каталог товарів забезпечує структуровану організацію асортименту за категоріями, брендами та технічними характеристиками;

– підсистема пошуку та фільтрації дозволяє користувачам ефективно знаходити потрібні позиції серед тисяч найменувань;

– кошик і модуль оформлення замовлень реалізує повний цикл купівлі від вибору товару до підтвердження транзакції;

– управління користувачами охоплює реєстрацію, автентифікацію та функціональність особистого кабінету;

– адміністративний модуль надає інструменти управління каталогом, замовленнями та комунікацією з клієнтами.

Функціональні особливості магазину електроніки принципово відрізняють його від інших товарних категорій. На відміну від торгівлі одягом чи продовольством, тут критично важливі детальні технічні специфікації, механізм порівняння моделей, достовірні відгуки покупців та актуальні дані про наявність товару. Ці вимоги суттєво впливають на проєктування як клієнтської, так і серверної частини застосунку.

Система обслуговує дві принципово різні категорії користувачів. Покупці потребують зручних інструментів для пошуку, порівняння, відстеження замовлень та зворотного зв'язку зі службою підтримки. Адміністратори натомість орієнтовані на ефективне управління каталогом, обробку замовлень і формування звітності. Розробка має повноцінно задовольнити обидві групи.

Ринок онлайн-продажів електроніки в Україні демонструє стійке зростання. Частка інтернет-продажів у загальному обсязі роздрібної торгівлі електронними пристроями помітно збільшилась упродовж останніх років. Споживачі дедалі частіше обирають онлайн-формат через можливість швидко порівняти ціни, ознайомитись з досвідом інших покупців і здійснити покупку без відвідування магазину. Ці тенденції підтверджують доцільність розробки сучасного вебзастосунку для торгівлі електронікою.

Серед ключових напрямів розвитку галузі виокремлюються такі:

– персоналізація формування пропозицій на основі аналізу поведінки користувача та його попередніх покупок;

– мобільна комерція адаптація інтерфейсу для зручного використання на смартфонах і планшетах;

– омніканальність інтеграція онлайн і офлайн каналів продажів для забезпечення єдиного користувацького досвіду;

– розширення платіжних можливостей підтримка різних методів оплати, зокрема оплати частинами та відстрочки платежу.

Кожен із цих напрямів має бути врахований на етапі проєктування вебзастосунку.

1.2 Структурні та функціональні особливості інтернет-магазинів електроніки

Інтернет-магазин електронних пристроїв реалізується за триланковою архітектурою (3-tier architecture), що забезпечує чіткий поділ між рівнями представлення, бізнес-логіки та зберігання даних. Клієнтська частина (фронтенд), серверна частина (бекенд) і база даних утворюють взаємозалежну, але розділену структуру – кожен рівень має власну зону відповідальності.

На фронтенді домінують компонентні JavaScript-фреймворки: React та Vue.js. Вони забезпечують динамічний рендеринг і прийнятну швидкодію за умови великого каталогу. Адаптивний дизайн тут не опція – він є базовою вимогою, оскільки мобільний трафік у сегменті електроніки стабільно переважає десктопний. Вибір технологічного стека бекенду визначається масштабом і навантаженням системи. Для зберігання даних застосовуються переважно реляційні СКБД – PostgreSQL або MySQL, що гарантують цілісність структурованих даних про товари, замовлення та користувачів.

Логічне ядро системи охоплює п'ять бізнес-процесів:

– управління каталогом додавання та редагування товарів, організація ієрархії категорій, управління зображеннями й технічними характеристиками;

– пошук і фільтрація повнотекстовий пошук за назвою, брендом і моделлю; фільтрація за ціновим діапазоном, характеристиками та наявністю на складі;

– кошик і замовлення формування кошика, оформлення замовлення, вибір способу доставки й оплати;

– управління користувачами реєстрація, автентифікація, ведення профілю та перегляд історії замовлень;

– адміністрування управління асортиментом, обробка замовлень, формування звітності.

Специфіка електроніки як товарної групи породжує окремий технічний виклик. Для одного смартфона кількість параметрів може перевищувати 30: процесор, обсяг оперативної та внутрішньої пам'яті, характеристики камери, ємність акумулятора, підтримувані стандарти зв'язку – і це неповний перелік. Для вирішення цього завдання в базі даних реалізується механізм динамічних атрибутів або застосовуються підходи до зберігання напівструктурованих даних. Система зобов'язана надавати зручний інтерфейс для перегляду й порівняння цих параметрів.

Окремої уваги потребує управління залишками. Система має відображати актуальні дані про наявність товарів у реальному часі, оновлювати статус після кожного підтвердженого замовлення й автоматично сигналізувати адміністраторам про критичне зниження залишків. Це накладає жорсткі вимоги на проєктування логіки роботи з даними та забезпечення транзакційної цілісності в базі даних.

1.3 Огляд та аналіз існуючих програмних рішень

Для оцінки актуального стану ринку проведено інженерний аудит п'яти провідних платформ онлайн-торгівлі електронікою в Україні: ROZETKA, TELEMART, COMFY, FOXTROT і CITRUS. Вибір зумовлений їхнім домінуючим становищем у сегменті онлайн-продажів електронних пристроїв та побутової техніки.

Платформа ROZETKA [1] розгорнута як триланковий вебзастосунок із мікросервісною архітектурою; фронтенд побудовано на React або Vue.js (загальний вигляд інтерфейсу наведено на рис. 1.1). Паралельно функціонують мобільні застосунки для Android і iOS. Перевага платформи – динамічний адаптивний інтерфейс і розширені можливості фільтрації. Водночас рекламні елементи та popup-вікна перевантажують UI, навігація ускладнена надлишковою кількістю категорій, а під час акційних розпродажів система помітно втрачає в продуктивності.

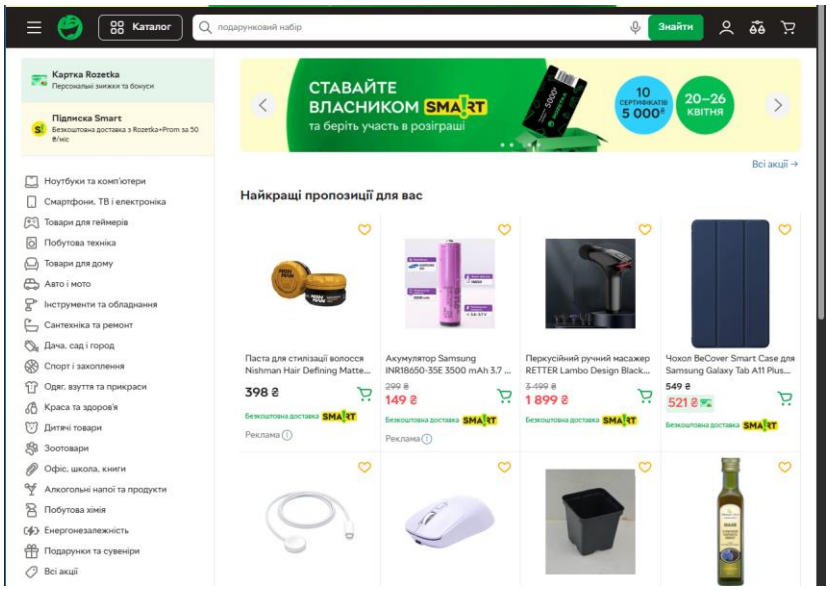

Рисунок 1.1 – Інтерфейс вебзастосунку ROZETKA

У рішенні від TELEMART [2] акцент зміщено на комп'ютерну техніку й комплектуючі (рис. 1.2). Ключова особливість – інтерактивний конфігуратор персональних комп'ютерів із перевіркою сумісності компонентів і 3D-візуалізацією зібраної системи. Модульна архітектура охоплює окремі модулі для каталогу, кошика, замовлень і конфігуратора. Деталізовані картки товарів і зручна система порівняння – беззаперечні сильні сторони. Проте застарілий дизайн і відсутність мобільного застосунку обмежують охоплення аудиторії.

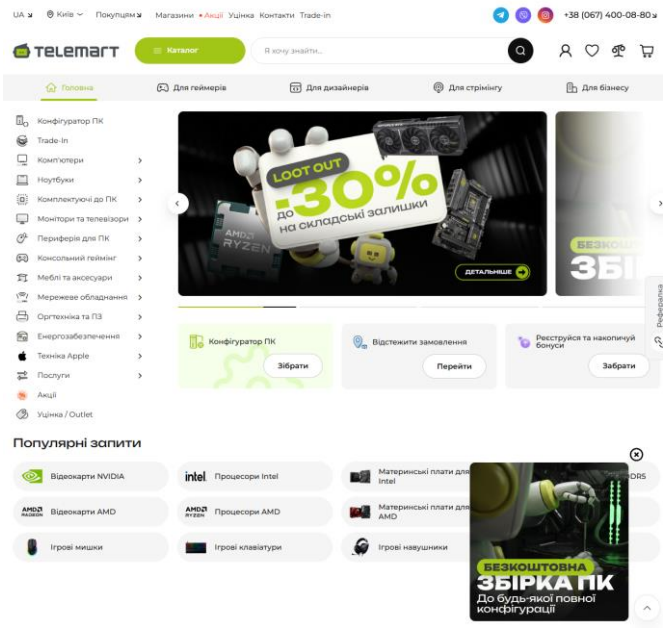

Рисунок 1.2 – Інтерфейс вебзастосунку TELEMART

Стратегія COMFY [3] базується на омніканальності: платформа інтегрує онлайн і офлайн канали продажів, надаючи можливість перевірити наявність товару в конкретному фізичному магазині, забронювати його, переглянути в AR або у відеоогляді (рис. 1.3). Синхронізація кошика між вебверсією та мобільним застосунком забезпечує безперервний досвід при переключенні між пристроями. Серед недоліків – повільна робота мобільного застосунку та надмірно ускладнений процес реєстрації.

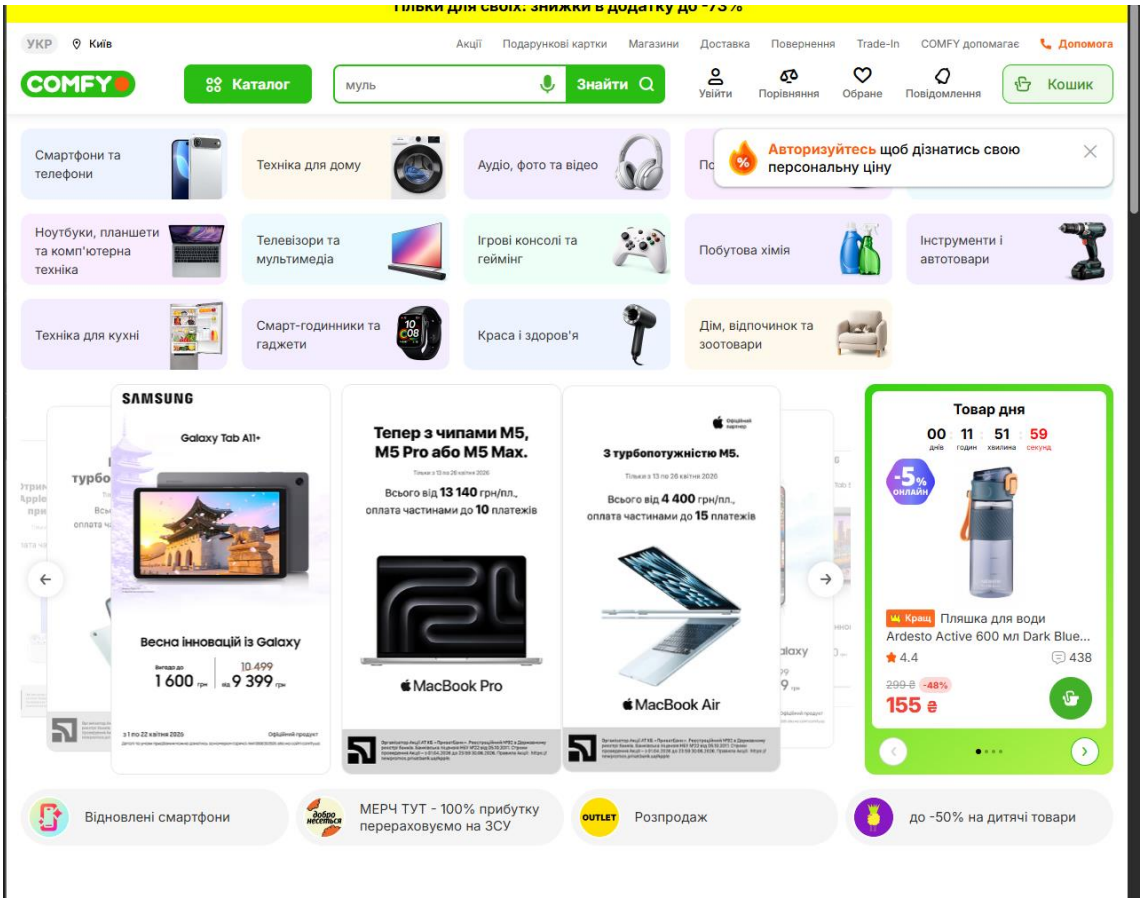

Рисунок 1.3 – Інтерфейс вебзастосунку COMFY

FOXTROT [4] реалізує концепцію єдиного кошика для онлайн і офлайн покупок: замовлення можна розпочати в браузері та завершити в найближчому магазині (рис. 1.4). Платформа підтримує пошук товарів на інтерактивній карті, QR-коди для переходу на товарні сторінки й онлайн-чат із консультантами. Застарілий дизайн і повільний пошук негативно позначаються на загальному користувацькому досвіді.

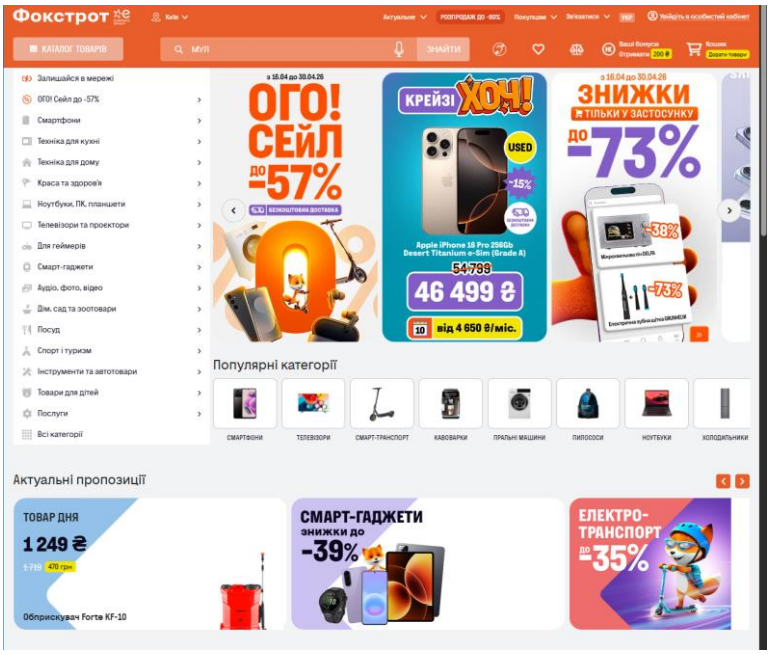

Рисунок 1.4 – Інтерфейс вебзастосунку FOXTROT

CITRUS [5] спроєктовано за принципом mobile-first із фокусом на смартфонах, аксесуарах і гаджетах (рис. 1.5). Мінімалістичний дизайн, висока швидкість мобільної версії та нетипові для конкурентів функції – фільтрація за кольором, система cashback та інтеграція з Instagram – виокремлюють платформу серед аналогів. Разом із тим обмежений функціонал застосунку, слабкі фільтри й відсутність функції порівняння товарів є відчутними прогалинами.

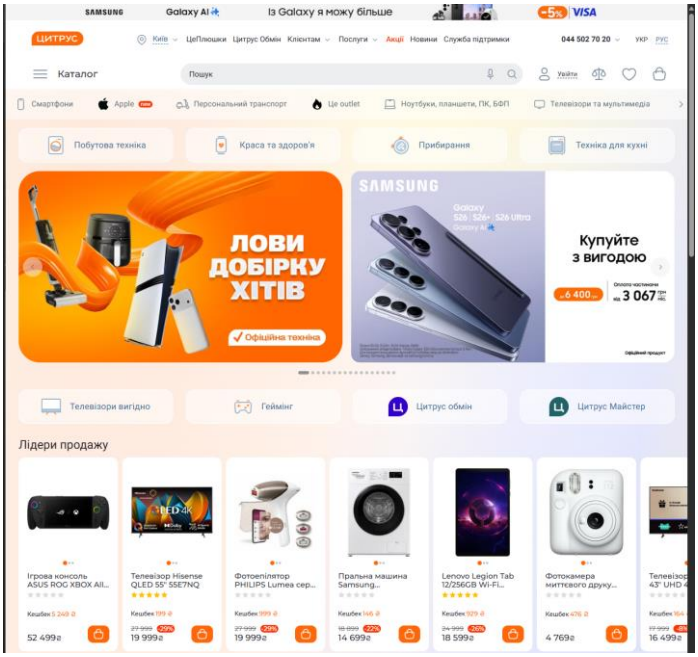

Рисунок 1.5 – Інтерфейс вебзастосунку CITRUS

Для систематизації результатів аудиту складено порівняльну таблицю основних характеристик розглянутих платформ (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика існуючих програмних рішень

Назва	ROZETKA	TELEMART	COMFY	FOXTROT	CITRUS
Розробник (дистриб'ютор)	ROZETKA (Україна)	TELEMART.UA	COMFY Trade LLC	ТОВ «Фокстрот»	ТОВ «Цитрус»
Архітектура	3-tier web application; веб-інтерфейс + REST API + БД; мікросервісна архітектура; мобільні застосунки Android/iOS	3-tier web application; веб-магазин з API; модульна структура (каталог, кошик, замовлення, конфігуратор)	3-tier web application + мобільні застосунки; інтеграція з офлайн-мережею та сервісними системами	3-tier web application; інтеграція з CRM/ERP; модульна архітектура для онлайн/офлайн	3-tier web application; клієнт-серверна архітектура; веб і мобільні клієнти; програми лояльності
Мова реалізації	Frontend: HTML5, CSS3, JavaScript (React/Vue.js); Backend: публічно не розкрито; Mobile: Swift, Kotlin/Java	Frontend: HTML5, CSS3, JavaScript; Backend: публічно не розкрито; БД: MySQL/PostgreSQL	Frontend: HTML5, CSS3, JavaScript; Backend: публічно не розкрито; Mobile: iOS/Android	Frontend: HTML5, CSS3, JavaScript; Backend: публічно не розкрито; інтеграції 1С, SAP	Frontend: HTML5, CSS3, JavaScript; Backend: публічно не розкрито; БД: реляційна
Функції, характеристики	Каталог (30+ категорій), кошик, різні способи доставки та оплати, відгуки, особистий кабінет, маркетплейс, 14 днів на повернення	Каталог ПК; інтерактивний конфігуратор; перевірка сумісності; 3D-візуалізація; розрахунок БЖ; збереження конфігурацій; технічні фільтри; порівняння; кошик	Інтеграція наявності офлайн; карта магазинів; фільтр «є сьогодні»; AR-перегляд; синхронізація кошика; відеоогляди; бронювання товару;	Єдиний кошик онлайн/офлайн; пошук товару в магазинах; QR-коди; онлайн-чат; історія покупок; збереження адрес; push-сповіщення	Візуальний каталог; фільтр за кольором; wishlist з price alert; покупка в 1 клік; Instagram-інтеграція; cashback; mobile-first дизайн

Кінець таблиці 1

Переваги	Зручний інтерфейс, адаптивний дизайн, швидка робота, розширені фільтри, мобільний застосунок, зручний кабінет	Потужний конфігуратор ПК, деталізовані картки товарів, порівняння, зручний пошук	Синхронізація кошика, перевірка наявності в магазині, відеоогляди, швидке оформлення	Єдиний кошик онлайн/офлайн, карта магазинів, онлайн-чат, зручна мобільна версія	Мінімалістичний дизайн, швидка мобільна версія, фільтри для гаджетів, cashback
Недоліки	Перевантажений інтерфейс, складна навігація, popup-вікна, зниження швидкості під час акцій	Застарілий дизайн, відсутній мобільний застосунок, слабка мобільна версія	Повільна робота застосунку, складна реєстрація, обмежений AR-функціонал	Застарілий дизайн, повільний пошук, складна навігація	Обмежений функціонал застосуку, слабкі фільтри, відсутність порівняння

Результати порівняльного аудиту виявили кілька системних проблем, спільних для більшості досліджених платформ. Рекламні та промоційні елементи перевантажують інтерфейс і відволікають користувача від цільової дії – вибору й придбання товару. Система фільтрації або реалізована недостатньо гнучко, або взагалі відсутня – як у випадку CITRUS, де звузити пошук за технічними параметрами практично неможливо. Мобільні версії оптимізовано нерівномірно: TELEMART не має мобільного застосунку взагалі, тоді як у COMFY він функціонує повільно. Нарешті, персоналізація на основі поведінки користувача залишається незадіяним резервом підвищення конверсії на всіх п'яти платформах.

1.4 Постановка завдань розробки

На підставі проведеного аналізу предметної області та виявлених недоліків конкурентних рішень сформульовано технічне завдання на розробку вебзастосунку інтернет-магазину електронних пристроїв. Запропоноване рішення орієнтоване на усунення зафіксованих проблем і має забезпечити зручний, швидкий та функціонально повноцінний інструмент – окремо для покупців і для адміністраторів системи.

Функціональні вимоги для покупців охоплюють:

– перегляд каталогу товарів із детальними технічними характеристиками та фотоматеріалами;

– пошук за назвою, брендом або моделлю з підтримкою фільтрації за технічними параметрами й ціновим діапазоном;

– управління кошиком додавання позицій, коригування кількості та видалення товарів;

– оформлення замовлення з вибором способу доставки та методу оплати;

– реєстрацію й автентифікацію з підтримкою особистого кабінету;

– перегляд історії замовлень та моніторинг їхніх статусів.

Адміністративний функціонал включає:

– управління каталогом додавання, редагування та видалення товарних позицій;

– управління категоріями й атрибутами товарів;

– обробку замовлень і оновлення їхніх статусів;

– контроль залишків на складі.

Нефункціональні вимоги визначають чотири ключові критерії якості системи: адаптивність – інтерфейс коректно відображається на десктопах, планшетах і смартфонах; швидкодія – сторінки завантажуються швидко, система реагує на дії без помітних затримок; зручність – інтерфейс є інтуїтивно зрозумілим, кількість кроків для ключових сценаріїв мінімізована; надійність – система забезпечує цілісність даних і коректну обробку виняткових ситуацій.

Архітектура застосунку будується на принципі розподілу відповідальності між рівнями. Клієнтська частина реалізована як односторінковий застосунок (SPA) на сучасному JavaScript-фреймворку. Серверна частина інкапсулює бізнес-логіку та надає RESTful API для взаємодії з клієнтом. База даних зберігає всі структуровані дані системи: товари, користувачів, замовлення та кошики. Підсистема управління замовленнями реалізує повний цикл обробки – від оформлення до доставки – з підтримкою статусної моделі й автоматичним коригуванням складських залишків.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області електронної комерції та існуючих програмних рішень у сфері інтернет-торгівлі електронними пристроями. Визначено структурні та функціональні особливості інтернет-магазинів електроніки як складних багаторівневих систем, що поєднують каталог товарів, системи пошуку, управління замовленнями та адміністративний модуль.

Проведений огляд п'яти провідних українських платформ – ROZETKA, TELEMART, COMFY, FOXTROT та CITRUS – дозволив виявити спільні недоліки існуючих рішень: перевантаженість інтерфейсу, складну навігацію, недостатню оптимізацію для мобільних пристроїв та обмежені можливості персоналізації. Ці недоліки підтверджують актуальність розробки нового вебзастосунку, що усуне виявлені проблеми.

На основі аналізу сформульовано завдання розробки: створення вебзастосунку інтернет-магазину електронних пристроїв з зручним адаптивним інтерфейсом, ефективною системою пошуку та фільтрації, повноцінним управлінням каталогом і замовленнями. Визначено функціональні вимоги для двох основних категорій користувачів: покупців та адміністраторів системи, а також нефункціональні вимоги щодо адаптивності, швидкодії та надійності застосунку.

2 МОДЕЛЮВАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ЗАСТОСУНКУ

У цьому розділі проаналізовано інструментарій, методи і технології, застосовані під час розробки вебзастосунку електронної комерції, та сформовано специфікацію вимог до програмного забезпечення. Визначення технологічного стеку й детальний опис вимог є необхідною передумовою для проєктування архітектури системи та її практичної реалізації.

2.1 Аналіз інструментарію та методів розробки

Вибір технологічного стеку визначається специфікою предметної області: обробкою замовлень, управлінням каталогом товарів, автентифікацією користувачів та взаємодією через REST API. Аналіз актуального стану галузі та наукових публікацій дозволив обґрунтувати рішення для кожного рівня системи.

2.1.1 Серверна частина (Backend)

Технологічною основою серверної частини є мова C# 12 і платформа .NET 8 [6] – актуальна LTS-версія з довгостроковою підтримкою та підтвердженою продуктивністю. Для побудови API використовується ASP.NET Core Web API: він забезпечує маршрутизацію HTTP-запитів через атрибути контролерів, вбудований механізм Dependency Injection, гнучкий Middleware-конвеєр і уніфіковану конфігурацію. Крос-платформна архітектура ASP.NET Core суттєво підвищує продуктивність серверних застосунків порівняно з класичним ASP.NET.

Взаємодія між клієнтом і сервером базується на архітектурному стилі REST [7]. Кожен ресурс системи – товари, замовлення, користувачі, кошики – доступний через окремий маршрут із підтримкою методів GET, POST, PUT, PATCH і DELETE відповідно до їхньої HTTP-семантики. Чіткий поділ відповідальності між рівнями забезпечується через Service Layer Pattern.

Рівень доступу до даних (Data Access Layer) реалізовано засобами Entity Framework Core 8 [8] з підходом Code-First і керуванням схемою через міграції. LINQ використовується для побудови типізованих запитів, зокрема до складних

вибірок каталогу товарів. Патерни DbContext і Repository організують доступ до даних, гарантуючи транзакційну цілісність під час інтенсивних CRUD-операцій. Як СКБД обрано Microsoft SQL Server [9] – платформу, що забезпечує надійну роботу з реляційними даними, підтримує розвинуте індексування, повнотекстовий пошук і механізми забезпечення високої доступності. Для адміністрування бази даних використовується SQL Server Management Studio (SSMS).

Для захисту конфіденційних даних реалізовано автентифікацію на основі JWT [10]: після успішного входу сервер генерує підписаний токен, що складається з трьох компонентів – header, payload і signature (структуру токена наведено на рис. 2.1). Валідацію виконує бібліотека Microsoft.AspNetCore.Authentication.JwtBearer. Stateless-архітектура токен-орієнтованого підходу знижує навантаження на сервер і спрощує горизонтальне масштабування порівняно з класичними сесійними механізмами. Паролі зберігаються у вигляді BCrypt-хешів [11] через бібліотеку BCrypt.Net-Next – алгоритм із адаптивним коефіцієнтом складності, стійкий до атак методом перебору. Авторизація контролюється атрибутами [Authorize] та [Authorize(Roles = "Admin")] на рівні контролерів. Перетворення між шарами Entity і DTO виконує AutoMapper 13.

Рисунок 2.1 – Структура JSON Web Token

Загальна архітектура серверного коду організована за патернами Repository [12], Generic Repository, Unit of Work та Service Layer [13]. Generic Repository у поєднанні з Unit of Work забезпечує атомарність операцій і спрощує модульне тестування. Service Layer інкапсулює бізнес-логіку й формує чітку межу між рівнем представлення і рівнем даних. Структурно серверна частина поділена на три проєкти: ElectronicStore.API (контролери, сервіси, маппінги),

ElectronicStore.Domain (моделі, DTO, інтерфейси) та ElectronicStore.Data (репозиторії, DbContext, міграції).

2.1.2 Клієнтська частина (Frontend)

Клієнтську частину реалізовано на Vanilla JavaScript ES6+ без підключення великих фреймворків – це мінімізує обсяг клієнтського коду та скорочує час завантаження сторінок. Код організовано за class-based архітектурою: кожен функціональний модуль системи – каталог, кошик, профіль, авторизація – інкапсульований в окремий JavaScript-клас із власною зоною відповідальності.

Для побудови адаптивного інтерфейсу застосовується Bootstrap [14]. Його система сітки, готові UI-компоненти та утилітарні класи забезпечують коректне відображення на пристроях різних розмірів без написання медіа-запитів для стандартних контрольних точок (breakpoints). HTTP-запити до серверного API виконуються через бібліотеку Axios, яка реалізує зручний Promise-інтерфейс і підтримує interceptors – завдяки їм JWT-токен автоматично додається до заголовка кожного вихідного запиту відповідно до принципів REST-взаємодії [7]. Для незареєстрованих користувачів стан голосування у відгуках зберігається через Cookie. Додатково підключено: Font Awesome 6.5 (іконки), Toastify.js (toast-сповіщення), Swiper.js 11 (слайдери) та AOS (анімації прокрутки).

Серверна частина надає два сервіси для інтеграції з клієнтом. Swagger/OpenAPI автоматично генерує інтерактивну документацію API на основі атрибутів контролерів, що спрощує узгодження контрактів між шарами. CORS налаштовано на стороні сервера для дозволу крос-доменних запитів із клієнтського домену.

2.2 Моделі та методи реалізації застосунку

Реалізацію застосунку структуровано навколо триланкової клієнт-серверної архітектури з чітким поділом на рівень представлення (HTML/CSS/JavaScript), рівень бізнес-логіки (ASP.NET Core) і рівень даних (Microsoft SQL Server).

Незалежність шарів гарантує, що зміни в одному рівні не спричиняють каскадних модифікацій в інших, що безпосередньо впливає на зручність супроводу системи.

Конвеєр обробки запиту. Клієнт ініціює HTTP-запит через Axios-interceptor, який автоматично додає JWT-токен до заголовка Authorization: Bearer. На сервері Middleware-конвеєр ASP.NET Core спершу валідує підпис токена: за позитивного результату з payload витягуються ідентифікатор і роль користувача. Атрибути [Authorize] та [Authorize(Roles = "Admin")] на рівні контролерів порівнюють роль із вимогами конкретного ендпоїнта. Після проходження авторизації запит потрапляє до Service Layer, який виконує бізнес-логіку і звертається до відповідного репозиторію. AutoMapper перетворює Entity-модель на DTO перед поверненням відповіді клієнту. Цей підхід унеможливлює витік службових або конфіденційних полів (наприклад, BCrypt-хешу пароля) через API – ProductDto містить лише публічні атрибути товару, тоді як Entity-модель Product може включати додаткові внутрішні поля.

Мапінг HTTP-методів на операції репозиторію. REST-контракт системи будується на прямій відповідності між HTTP-семантикою та CRUD-операціями: POST > INSERT, GET > SELECT, PUT/PATCH > UPDATE, DELETE > DELETE. Кожен ресурс – товари, категорії, замовлення, користувачі – доступний через окремий маршрут: /api/products і /api/products/{id}, /api/orders і /api/orders/{id}. Репозиторій виступає єдиною точкою доступу до даних, а Unit of Work забезпечує атомарність операцій, що охоплюють кілька сутностей. Контроль доступу інтегровано безпосередньо в цей ланцюжок: адміністратор має доступ до POST, PUT і DELETE для товарів, тоді як покупець обмежений методами GET і POST для замовлень.

Алгоритм динамічної фільтрації та захист від SQL-ін'єкцій. Пошук і фільтрація каталогу реалізовані через побудову динамічних LINQ-специфікацій. Кожен активний фільтр – категорія, ціновий діапазон, бренд, наявність – додає умову до ланцюжка IQueryable<T> перед виконанням запиту. Entity Framework Core транслює цей ланцюжок у параметризовані SQL-запити: фільтр за ціною

перетворюється на WHERE Price >= @min AND Price <= @max, пошук за назвою – на WHERE Name LIKE @query. Параметризація унеможливлює SQL-ін'єкції на рівні ORM, без додаткових санітаційних механізмів. Сортування за ціною, рейтингом або датою додавання реалізується через ORDER BY у динамічно побудованому запиті. Ключові поля, що беруть участь у фільтрації та сортуванні, індексовано в базі даних для забезпечення прийнятної продуктивності за зростаючого обсягу каталогу.

2.3 Специфікація вимог до програмного забезпечення

Специфікація вимог описує повну поведінку розроблюваного вебзастосунку та включає функціональні й нефункціональні вимоги до системи.

1) Призначення та межі проєкту

1.1) Призначення системи

Вебзастосунок інтернет-магазину електронних пристроїв забезпечує зручний процес вибору, порівняння та придбання товарів через Інтернет. Покупцям надається доступ до каталогу з детальними характеристиками, інструменти пошуку, фільтрації, кошик та особистий кабінет. Адміністраторам – інструменти управління каталогом та замовленнями.

1.2) Погодження, ухвалені в програмній документації

– специфікація розроблена на основі аналізу потреб покупців та адміністраторів;

– взаємодія між клієнтом і сервером – через REST API у форматі JSON (UTF-8);

– автентифікація – на основі JWT без збереження сесій на сервері;

– серверна частина – C# 12 / .NET 8, клієнтська – Vanilla JavaScript ES6+.

1.3) Межі проєкту

– розробляється вебінтерфейс та REST API; мобільний застосунок не передбачено;

– інтеграція з реальними платіжними шлюзами відсутня в поточній версії;

– модуль управління логістикою не входить до складу системи.

2) Загальний опис

2.1) Сфера застосування

Платформа роздрібної онлайн-торгівлі електронними пристроями для приватних покупців та адміністраторів, що використовують браузер на ПК або мобільному пристрої.

2.2) Характеристики користувачів

– гість: перегляд каталогу, пошук, сесійні голоси у відгуках через Cookie; без доступу до замовлень;

– зареєстрований покупець: повний функціонал – кошик, замовлення, особистий кабінет;

– адміністратор: управління каталогом, категоріями та замовленнями через адміністративний модуль.

2.3) Загальна структура і склад системи

– клієнтська частина: HTML5, CSS3, JavaScript ES6+, Bootstrap, Axios;

– серверна частина: ASP.NET Core Web API (.NET 8), EF Core 8, Repository/UoW/Service Layer;

– база даних: Microsoft SQL Server (Code-First міграції EF Core);

– безпека: JWT-автентифікація, BCrypt-хешування паролів;

– документація: Swagger/OpenAPI.

2.4) Загальні обмеження

– потребує стабільного інтернет-з'єднання та сучасного браузера (ES6+);

– реальна обробка платежів не реалізована;

– одночасне редагування одного запису кількома адміністраторами не захищено.

3) Функції системи

3.1) Реєстрація та автентифікація

3.1.1) Опис функції

Реєстрація нових користувачів та вхід зареєстрованих з видачею JWT-токена для доступу до захищених ресурсів.

3.1.2) Вхідна і вихідна інформація

– вхідна: ім'я, email, пароль (реєстрація); email, пароль (вхід);

– вихідна: JWT-токен, дані профілю (ідентифікатор, ім'я, роль).

3.1.3) Функціональні вимоги

– валідація email та мінімальної довжини пароля (8+ символів);

– перевірка унікальності email при реєстрації;

– хешування пароля BCrypt перед збереженням;

– генерація підписаного JWT з полями: ID, роль, термін дії;

– блокування після 5 невдалих спроб входу протягом 10 хвилин.

3.2) Перегляд каталогу та пошук

3.2.1) Опис функції

Перегляд каталогу товарів, пошук за ключовими словами та фільтрація результатів.

3.2.2) Вхідна і вихідна інформація

– вхідна: пошуковий запит, параметри фільтрації (категорія, ціна, бренд, наявність), сортування, сторінка;

– вихідна: посторінковий список товарів (назва, зображення, ціна, рейтинг, наявність).

3.2.3) Функціональні вимоги

– повнотекстовий пошук за назвою та описом;

– фільтрація за категорією, брендом, ціновим діапазоном, наявністю;

– сортування за ціною, рейтингом, датою додавання;

– пагінація з налаштовуваним розміром сторінки та відображенням загальної кількості результатів.

3.3) Перегляд картки товару

3.3.1) Опис функції

Відображення детальної інформації про товар: характеристики, зображення, відгуки, статус наявності.

3.3.2) Вхідна і вихідна інформація

– вхідна: ідентифікатор товару;

– вихідна: назва, опис, технічні характеристики, ціна, залишок, зображення, рейтинг, відгуки.

3.3.3) Функціональні вимоги

– відображення повного переліку технічних характеристик за категорією товару;

– галерея зображень із можливістю збільшення;

– середній рейтинг та перелік відгуків покупців;

– кнопка додавання до кошика з вибором кількості.

3.4) Управління кошиком

3.4.1) Опис функції

Формування списку обраних товарів перед оформленням замовлення.

3.4.2) Вхідна і вихідна інформація

– вхідна: ідентифікатор товару, кількість, ID користувача;

– вихідна: вміст кошика (товари, кількість, ціна, загальна сума).

3.4.3) Функціональні вимоги

– додавання товару з перевіркою наявності на складі;

– зміна кількості та видалення позицій, очищення кошика;

– збереження кошика між сеансами для зареєстрованих користувачів;

– відображення кількості позицій у навігаційному меню.

3.5) Оформлення замовлення

3.5.1) Опис функції

Оформлення замовлення на основі кошика з введенням даних доставки та підтвердженням.

3.5.2) Вхідна і вихідна інформація

– вхідна: вміст кошика, адреса та контактні дані, спосіб доставки;

– вихідна: підтвердження з номером замовлення, загальна сума, деталі доставки.

3.5.3) Функціональні вимоги

– перевірка наявності товарів перед підтвердженням;

– автоматичне оновлення залишків після оформлення;

– збереження замовлення зі статусом «Нове»; очищення кошика.

3.6) Управління профілем

3.6.1) Опис функції

Доступ до особистого кабінету з персональними даними та історією замовлень.

3.6.2) Вхідна і вихідна інформація

– вхідна: оновлені персональні дані;

– вихідна: актуальні дані профілю, список замовлень зі статусами.

3.6.3) Функціональні вимоги

– перегляд та редагування персональних даних;

– перегляд історії замовлень з деталями та статусами;

– зміна пароля з підтвердженням поточного.

3.7) Адміністративне управління каталогом

3.7.1) Опис функції

Управління товарним асортиментом, категоріями та залишками для адміністраторів.

3.7.2) Вхідна і вихідна інформація

– вхідна: дані товару (назва, опис, ціна, категорія, характеристики, зображення, кількість);

– вихідна: підтвердження операції, оновлений каталог.

3.7.3) Функціональні вимоги

– повний CRUD для товарів і категорій; завантаження зображень;

– управління залишками; доступ лише для ролі Admin.

3.8) Обробка замовлень адміністратором

3.8.1) Опис функції

Перегляд усіх замовлень та оновлення їх статусів.

3.8.2) Вхідна і вихідна інформація

– вхідна: ідентифікатор замовлення, новий статус;

– вихідна: оновлений статус, список замовлень.

3.8.3) Функціональні вимоги

– перегляд замовлень з фільтрацією за статусом і датою;

– перегляд деталей замовлення;

– оновлення статусу: Нове, В обробці, Відправлено, Доставлено, Скасовано.

4) Вимоги до інформаційного забезпечення

4.1) Джерела і зміст вхідної інформації

– дані користувачів – через реєстраційні форми;

– дані товарів – через адміністративний модуль;

– дані замовлень – автоматично з кошика та форми доставки.

4.2) Нормативно-довідкова інформація

– довідник категорій товарів;

– довідник статусів замовлень;

– довідник ролей користувачів (User, Admin).

4.3) Вимоги до організації та збереження інформації

– дані зберігаються в Microsoft SQL Server;

– схема керується міграціями EF Core;

– паролі зберігаються виключно у вигляді BCrypt-хешів;

– зображення зберігаються у файловій системі сервера з посиланнями в БД;

– цілісність зв'язків забезпечується зовнішніми ключами та каскадними операціями.

5) Вимоги до технічного забезпечення

– сервер: ОС з підтримкою .NET 8 Runtime (Windows Server або Linux);

– СКБД: Microsoft SQL Server 2022+;

– клієнт: сучасний браузер (Chrome 90+, Firefox 88+, Edge 90+, Safari 14+);

– мережа: з'єднання з Інтернетом від 1 Мбіт/с;

– інструменти розробки: Visual Studio 2022, SSMS.

6) Вимоги до програмного забезпечення

6.1) Архітектура програмної системи

Триланкова клієнт-серверна архітектура: SPA-клієнт на Vanilla JS, REST API на ASP.NET Core з Service Layer і Repository Pattern, база даних Microsoft SQL Server.

6.2) Системне програмне забезпечення

– мова: C# 12 / .NET 8;

– фреймворк: ASP.NET Core Web API;

– ORM: Entity Framework Core 8;

– СКБД: Microsoft SQL Server.

6.3) Мережне програмне забезпечення

– протокол HTTPS для всіх запитів;

– CORS налаштований для клієнтського домену;

– формат даних: JSON (UTF-8).

6.4) Програмне забезпечення ведення інформаційної бази

– СКБД: Microsoft SQL Server 2022+;

– інструмент: SSMS;

– версіонування: міграції EF Core.

6.5) Мова і технологія розробки

– backend: C# 12, .NET 8, ASP.NET Core Web API, EF Core 8, AutoMapper 13, JWT, BCrypt.Net-Next;

– frontend: HTML5, CSS3, JavaScript ES6+, Bootstrap, Axios, Font Awesome 6.5, Toastify.js, Swiper.js 11, AOS;

– інструменти: Visual Studio 2022, SSMS, Swagger/OpenAPI.

7) Вимоги до зовнішніх інтерфейсів

7.1) Інтерфейс користувача

– адаптивний дизайн від 320 px до 1920 px;

– інтуїтивна навігація: головна, каталог, картка товару, кошик, кабінет;

– сповіщення через Toastify.js (успіх, помилка, попередження).

7.2) Апаратний інтерфейс

Спеціалізоване обладнання не потрібне. Застосунок працює на будь-якому пристрої з браузером.

7.3) Програмний інтерфейс

– REST API з JSON-відповідями для всіх операцій;

– документація через Swagger UI (/swagger);

– JWT передається у заголовку Authorization: Bearer.

7.4) Комунікаційний протокол

– HTTPS для всіх запитів;

– HTTP-методи: GET, POST, PUT, PATCH, DELETE.

8) Властивості програмного забезпечення

8.1) Доступність

Час доступності – не менше 99% на місяць; цілодобова робота без планових перерв у робочий час.

8.2) Супроводжуваність

– модульна структура (API / Domain / Data);

– документація через Swagger/OpenAPI;

– версіонування схеми БД через міграції EF Core.

8.3) Переносимість

– серверна частина: будь-яка платформа з .NET 8 (Windows, Linux, macOS);

– клієнт: будь-який сучасний браузер без додаткового ПЗ.

8.4) Продуктивність

– відповідь API на стандартний запит – до 1 секунди;

– завантаження сторінок – до 3 секунд;

– підтримка 100+ одночасних користувачів; ефективна робота з каталогами понад 10 000 товарів.

8.5) Надійність

– транзакційна цілісність через Unit of Work;

– обробка виключень з HTTP-статус кодами;

– валідація даних на клієнті та сервері.

8.6) Безпека

– підписані JWT-токени з обмеженим терміном дії;

– BCrypt-хешування паролів;

– захист від SQL-ін'єкцій через параметризовані запити EF Core;

– захист від XSS через екранування вхідних даних;

– HTTPS; розмежування прав User/Admin через атрибути авторизації.

9) Інші вимоги

– Cookie для сесійних голосів гостей у відгуках;

– структуровані JSON-відповіді про помилки з кодом і описом;

– адміністративний модуль без необхідності прямого доступу до БД.

Висновки до розділу 2

У другому розділі обґрунтовано вибір технологічного стеку та визначено методи реалізації вебзастосунку. Серверна частина побудована на C# 12 / .NET 8 з ASP.NET Core Web API, Entity Framework Core 8 та Microsoft SQL Server; архітектура базується на патернах Repository, Unit of Work і Service Layer. Автентифікація реалізована через JWT з BCrypt-хешуванням паролів. Клієнтська частина виконана на Vanilla JavaScript ES6+ з Bootstrap без великих фреймворків.

Взаємодія клієнта і сервера організована за REST-принципами. Ключові методи реалізації: CRUD через репозиторії, динамічна LINQ-фільтрація, stateless JWT-авторизація, DTO-патерн з AutoMapper. Сформована специфікація вимог (SRS) охоплює призначення системи, характеристики трьох ролей користувачів, вісім функцій з вхідними/вихідними даними та функціональними вимогами, вимоги до інформаційного, технічного й програмного забезпечення, інтерфейсів та нефункціональних властивостей. Отримані результати є основою для проєктування архітектури у наступному розділі.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

У цьому розділі сформовано проєктні рішення вебзастосунку інтернет-магазину електронних пристроїв: визначено архітектуру програмного забезпечення, спроєктовано структуру бази даних, побудовано комплекс UML-діаграм та наведено опис інтерфейсів системи.

3.1 UML-моделювання системи

Для формалізованого опису поведінки та статичної структури вебзастосунку розроблено комплекс UML-діаграм. Склад і типи діаграм визначено специфікою системи: трьома рольовими профілями користувачів, складним бізнес-сценарієм оформлення замовлень та багаторівневою архітектурою серверної частини на базі ASP.NET Core з Repository Pattern і Service Layer. Побудовано діаграму варіантів використання, шість діаграм послідовностей, три діаграми активності та діаграму класів.

Межі системи і функціональний периметр кожного актора зафіксовано на діаграмі варіантів використання (рис. 3.1). Вона охоплює три актори: гостя, зареєстрованого покупця (User) та адміністратора.

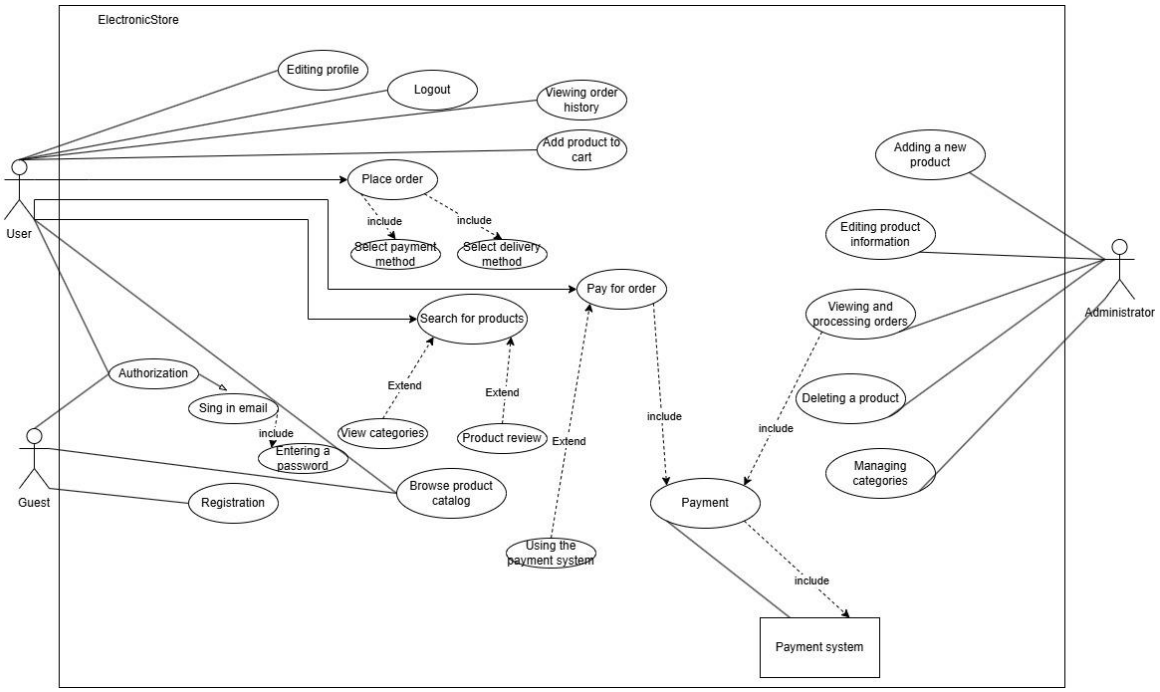

Рисунок 3.1 – Діаграма варіантів використання

Гість взаємодіє з відкритими ресурсами системи: переглядає каталог, виконує пошук і фільтрацію, відкриває картку товару, а також ініціює реєстрацію або автентифікацію.

Зареєстрований покупець наслідує весь функціонал гостя та отримує доступ до захищених операцій: управління кошиком і його вмістом, оформлення замовлення з вибором доставки та оплати, перегляду архіву замовлень, редагування профілю та зміни пароля, формування списків бажань і порівняння, написання відгуків та голосування за корисність відгуків інших покупців.

Адміністратор взаємодіє з окремим адміністративним модулем: виконує CRUD-операції над товарами й категоріями, переглядає всі замовлення та оновлює їхні статуси, управляє обліковими записами користувачів. Варіант використання «Оформлення замовлення» включає (include) підваріанти «Вибір способу доставки» та «Вибір способу оплати»; варіант «Пошук товарів» розширюється (extend) варіантами «Перегляд категорій» та «Перегляд відгуків».

Діаграми послідовностей деталізують часовий порядок міжкомпонентної взаємодії в межах конкретних сценаріїв. Кожна діаграма відображає повний пайплайн обробки запиту відповідно до прийнятої архітектури: HTTP-запит від Axios-клієнта > API Controller > Service Layer > Repository (Unit of Work / DbContext) > MS SQL Server.

Процедуру верифікації та ідентифікації користувача деталізовано на рис. 3.2. Axios-клієнт надсилає POST-запит із обліковими даними до AuthController. Контролер делегує виклик до AuthService, який через UserRepository звертається до MS SQL Server для пошуку облікового запису за email. Після отримання запису AuthService виконує BCrypt-порівняння хешів. За позитивного результату генерується підписаний JWT-токен, який повертається клієнту через контролер; NotificationService фіксує факт успішного входу. Після завершення сеансу токен знищується на стороні клієнта.

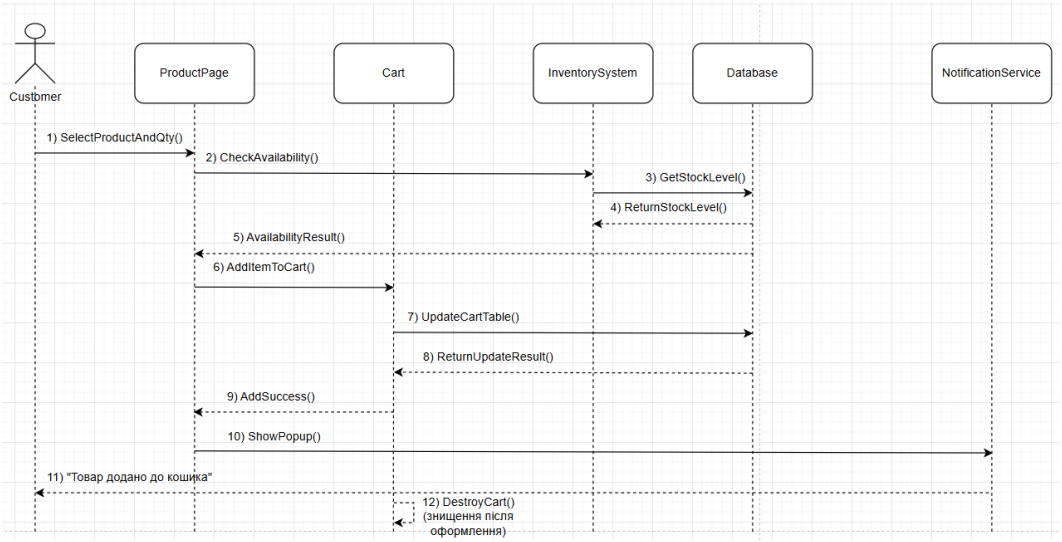

Рисунок 3.2 – Діаграма послідовностей автентифікація користувача

Динаміку міжкомпонентної взаємодії при модифікації вмісту кошика ілюструє рис. 3.3. ProductPage через Axios надсилає POST-запит до CartController, передаючи ідентифікатор товару та кількість. CartService направляє запит до InventoryRepository, який через DbContext перевіряє залишок у таблиці складу MS SQL Server. За наявності достатньої кількості CartService викликає CartRepository для запису рядка до таблиці CartItems. Успішне збереження ініціює відповідь клієнту: NotificationService відображає popup-сповіщення, лічильник кошика в навігаційному меню оновлюється.

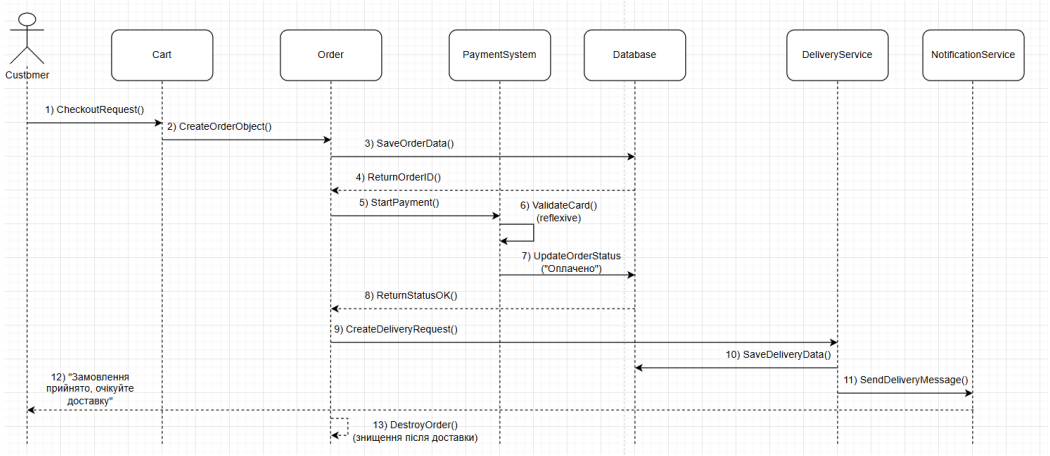

Рисунок 3.3 – Діаграма послідовностей додавання товару до кошика

Управління життєвим циклом замовлення на етапі білінгу відображено на рис. 3.4. Після ініціювання оформлення OrderController приймає HTTP-запит та

передає його до OrderService. Сервіс формує об’єкт замовлення і через OrderRepository зберігає його в MS SQL Server у межах транзакції Unit of Work. Після фіксації ідентифікатора замовлення OrderService звертається до PaymentSystem, який виконує валідацію платіжних даних та оновлює статус замовлення до «Оплачено» через репозиторій. Паралельно DeliveryService отримує дані доставки та зберігає їх. Завершальним кроком NotificationService формує і надсилає клієнту підтвердження з деталями замовлення.

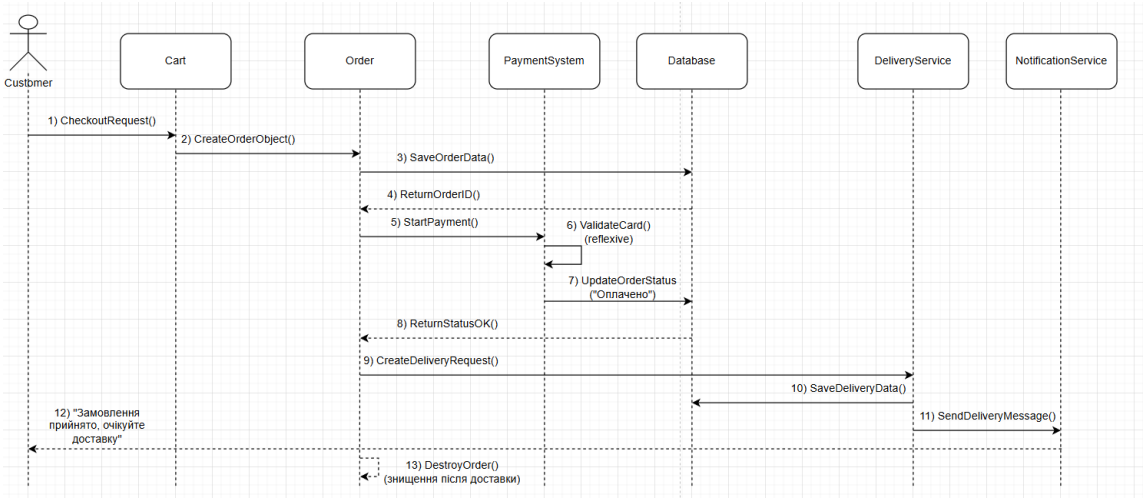

Рисунок 3.4 – Діаграма послідовностей оформлення замовлення

Діаграми активності описують покроковий перебіг процесів із розгалуженнями та альтернативними гілками виконання. Побудовано три діаграми.

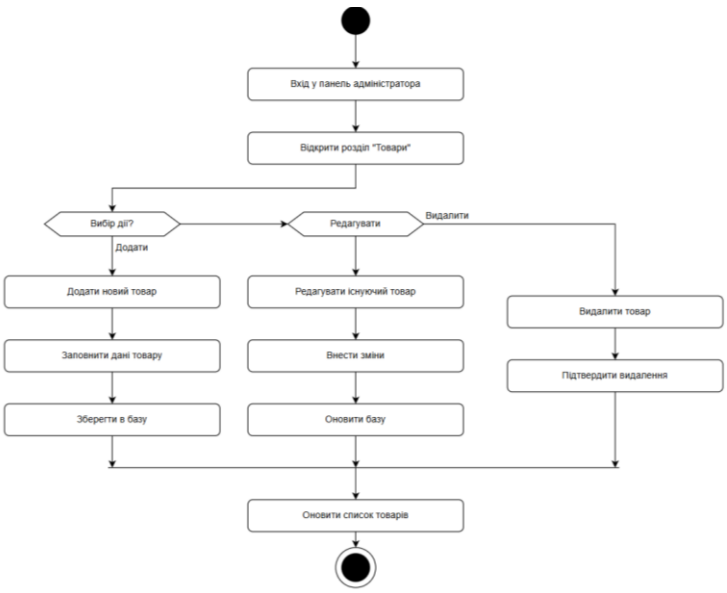

Рисунок 3.5 – Діаграми діяльності

Процес адміністративного управління товарним асортиментом схематично представлено на рис. 3.5. Після входу до панелі адміністратора й відкриття розділу «Товари» виникає точка розгалуження між трьома гілками: «Додати» (заповнення форми нового товару > збереження запису), «Редагувати» (внесення змін > оновлення запису) та «Видалити» (запит підтвердження > видалення запису). Незалежно від обраної гілки список товарів оновлюється, і процес завершується в єдиному фінальному вузлі.

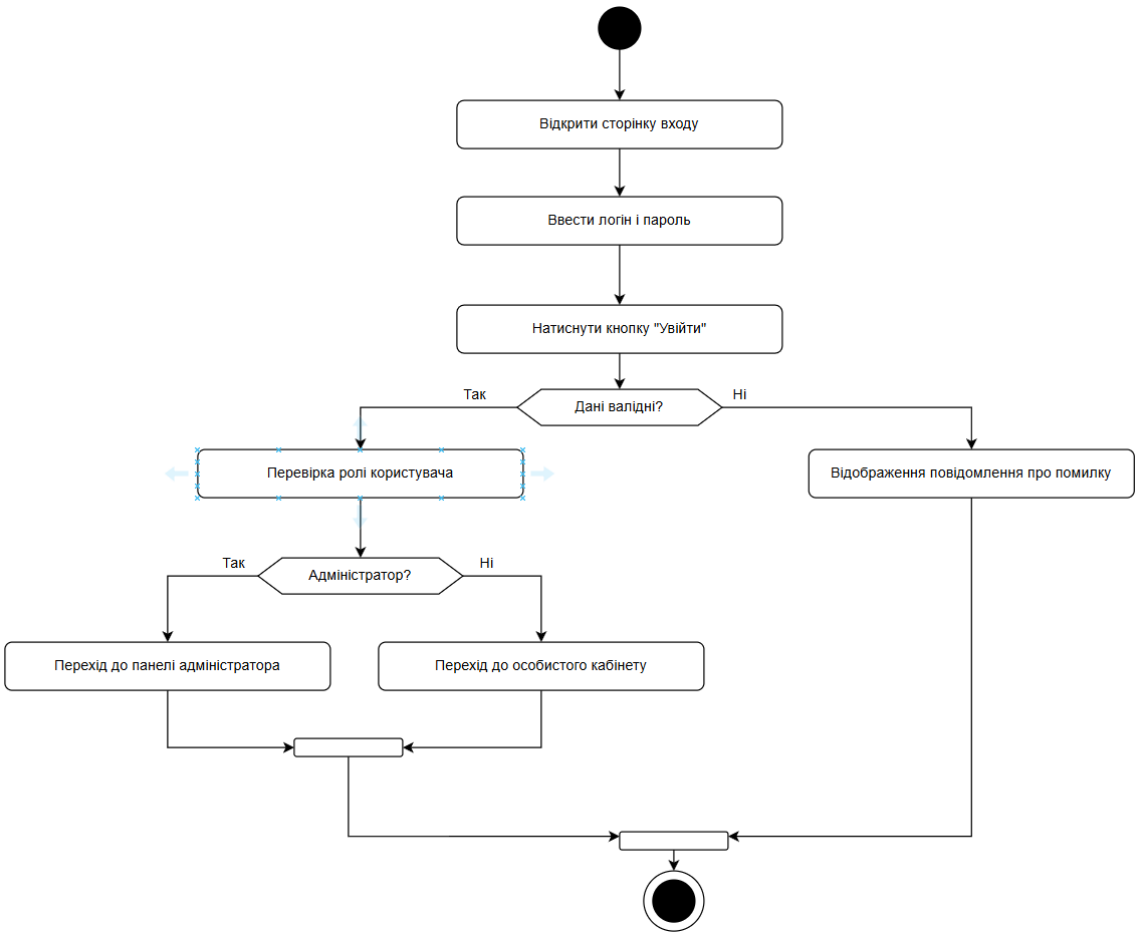

Рисунок 3.6 – Діаграми діяльності

Відповідно до розробленої моделі (рис. 3.6), процес автентифікації розпочинається із введення логіна та пароля на сторінці входу. Система виконує валідацію: у разі невдачі відображається повідомлення про помилку й виконання повертається до початкового вузла введення. За успішної валідації перевіряється роль облікового запису: адміністратор перенаправляється до адміністративної панелі, покупець – до особистого кабінету.

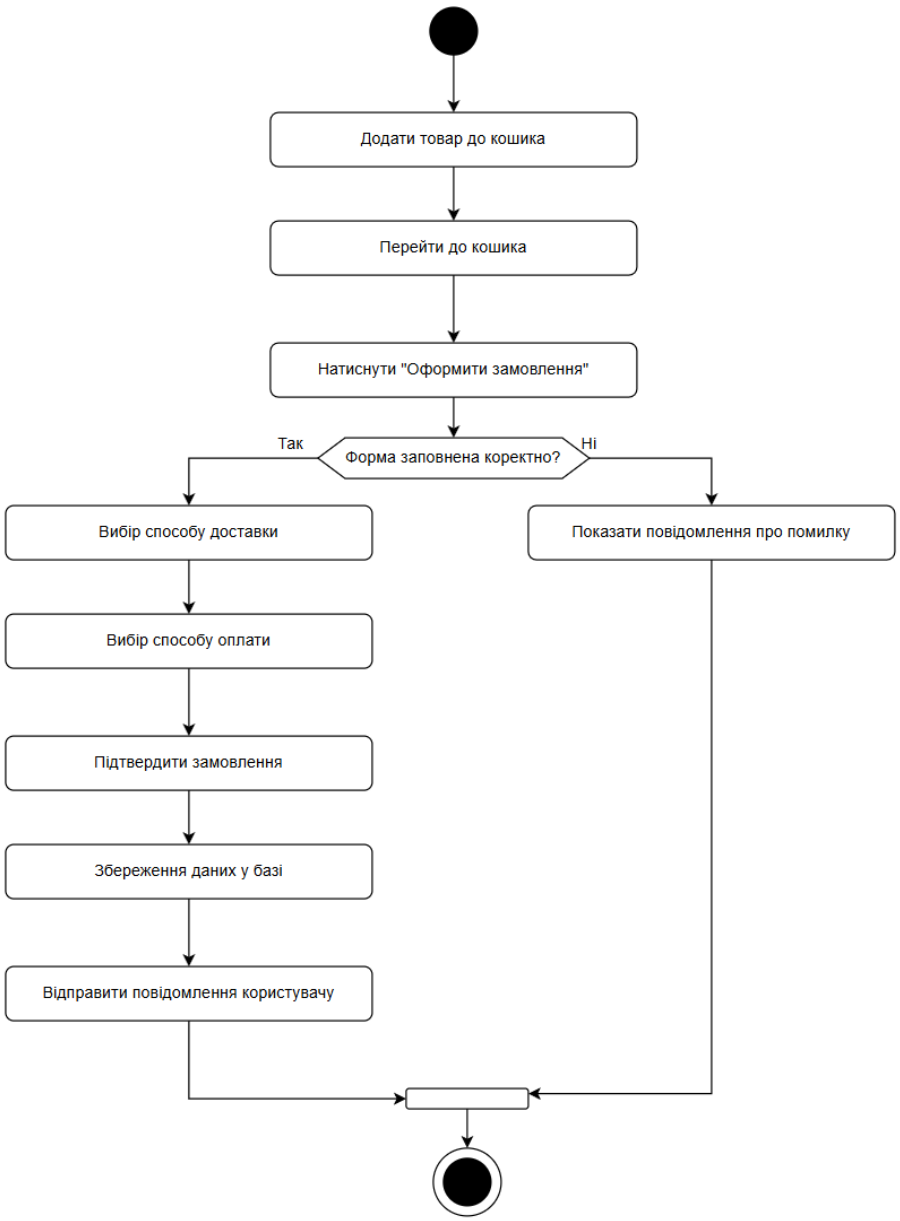

Рисунок 3.7 – Діаграми діяльності

Алгоритм оформлення замовлення покупцем зображено на рис. 3.7. Після переходу до оформлення з кошика система перевіряє коректність заповнення форми доставки: помилка повертає користувача на крок виправлення. Пройшовши перевірку, покупець обирає спосіб доставки та оплати й підтверджує замовлення. Система атомарно зберігає дані в MS SQL Server, надсилає підтвердження і досягає фінального вузла процесу.

Статичну структуру серверної частини відображає діаграма класів (рис. 3.8), що охоплює тринадцять Entity-моделей проєкту ElectronicStore.Domain, які безпосередньо відповідають таблицям бази даних ElectronicStoreDB.

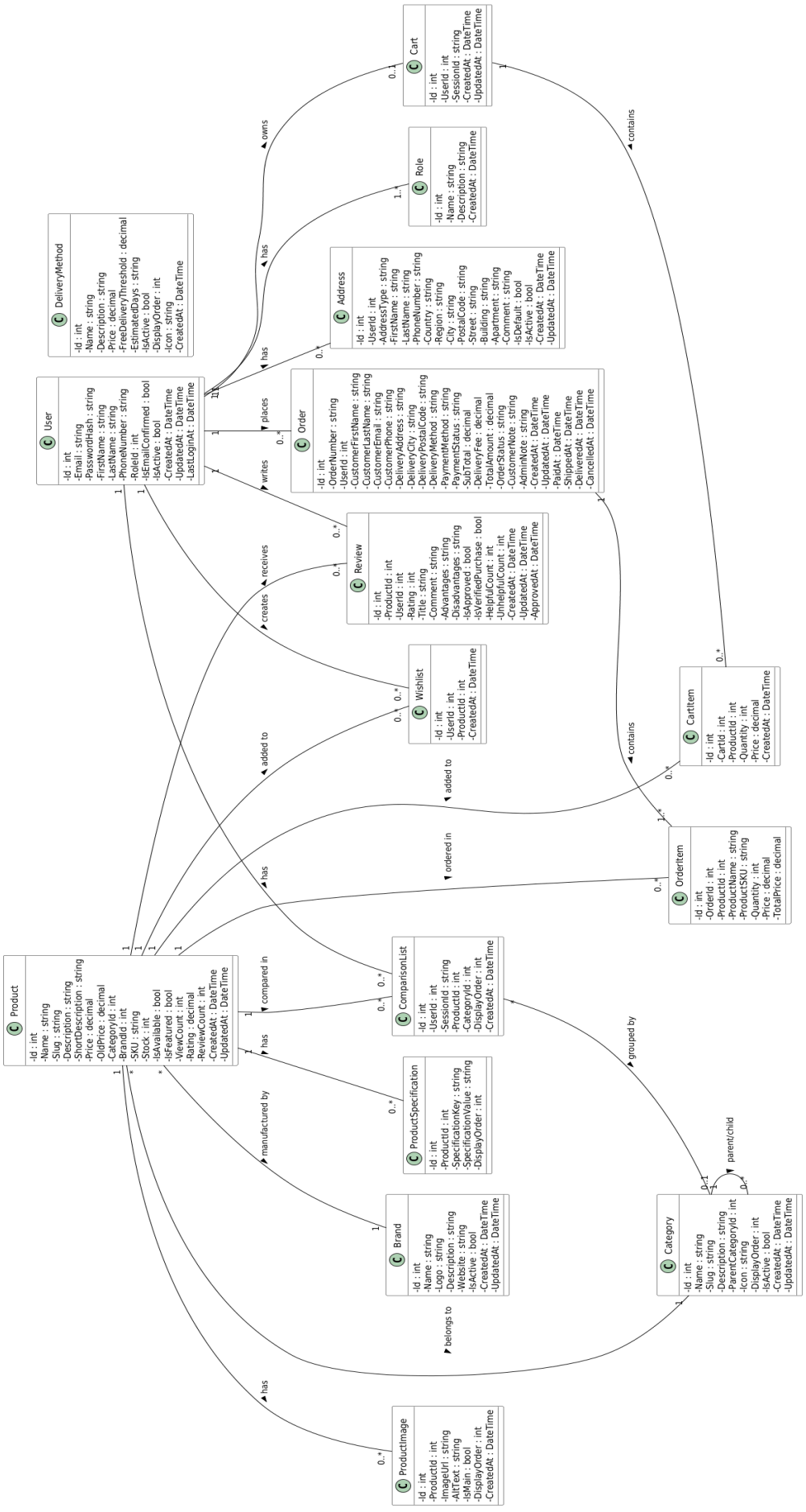

Рисунок 3.8 – Діаграма класів

Центральним елементом моделі предметної області є клас Product. З класом Category він пов’язаний відношенням асоціації з кратністю 1..*: одна категорія агрегує довільну кількість товарів; клас Category має самопосилальний зв’язок parent/child для реалізації ієрархії підкатегорій довільної глибини. Аналогічне відношення асоціації 1..* зв’язує Product із класом Brand. Зі своїми медіа- і специфікаційними даними Product пов’язаний відношенням композиції: ProductImage (кратність 1..*) та ProductSpecification (кратність 0..*) не існують у системі незалежно від батьківського товару; видалення Product каскадно знищує пов’язані записи.

Клас User пов’язаний із Role відношенням асоціації з кратністю 0..1 – кожен обліковий запис має рівно одну роль. Від User відходять чотири відношення: Order (creates, 0..*), Review (writes, 0..*), Cart (has, 0..1) та Wishlist/ComparisonList (0..*). Клас Order є агрегатом: він пов’язаний з OrderItem відношенням композиції (1..*) та з Address відношенням асоціації (0..1). Клас Cart утримує CartItem через композицію (0..*). Клас Review пов’язаний із Product відношенням асоціації (0..*) та отримує голоси покупців через клас ReviewVote (receives, 0..*). Усі зв’язки між класами відображають зовнішньоключові обмеження, визначені у схемі ElectronicStoreDB.

3.2 Проєктування бази даних

База даних ElectronicStoreDB спроєктована за реляційною моделлю з використанням підходу Code-First у Entity Framework Core 8 і розгорнута на Microsoft SQL Server 2022. Концептуальну схему всіх сутностей та їхніх взаємозв’язків розгорнуто на ER-діаграмі (рис. 3.9). Структурно схему організовано у чотири функціональних домени: «Користувачі», «Каталог», «Продажі» та «Клієнтський досвід».

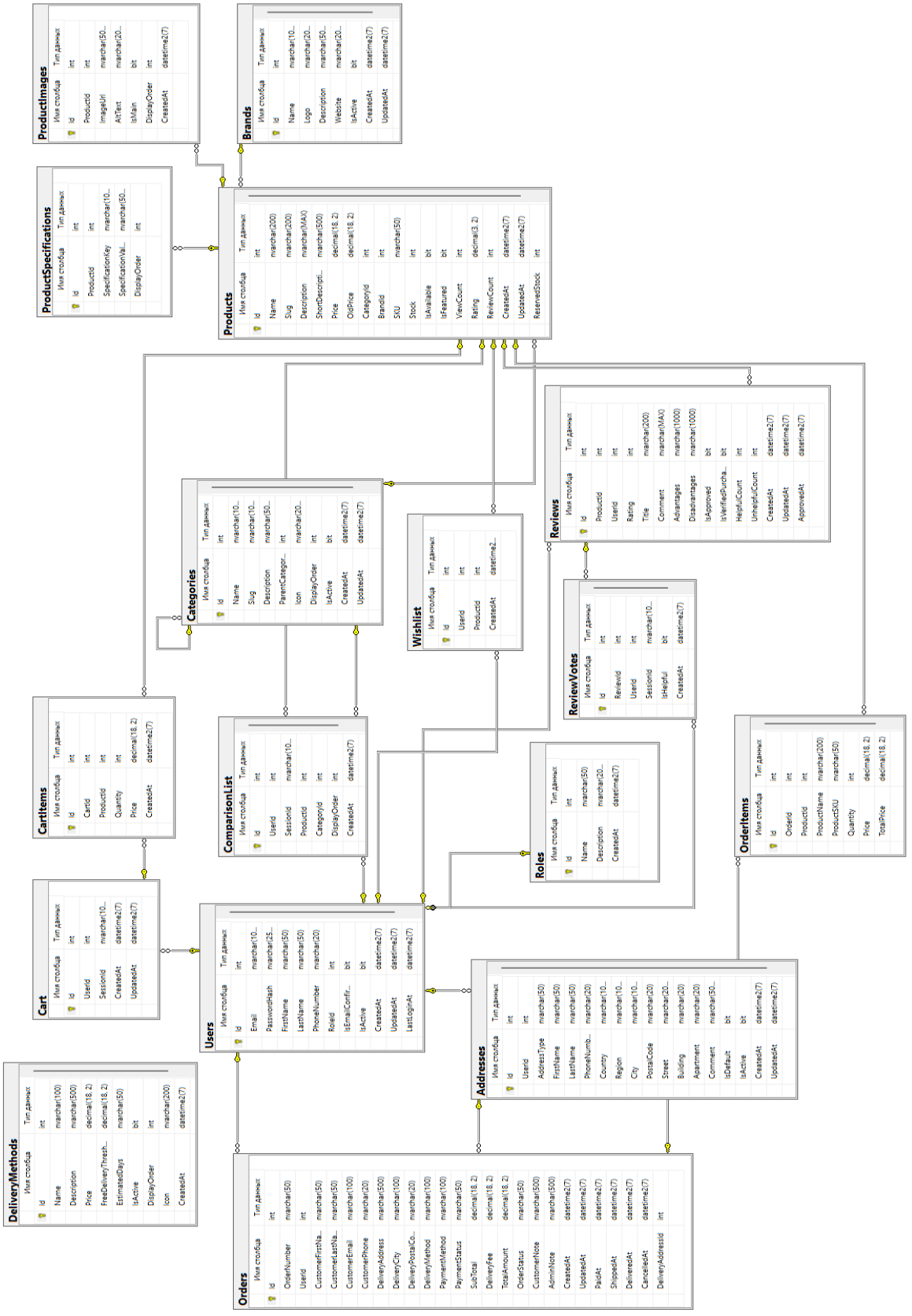

Рисунок 3.9 – ER-діаграма бази даних ElectronicStoreDB

Домен «Користувачі» формують дві таблиці з чітким розмежуванням обов’язків. Users зберігає облікові записи зареєстрованих користувачів; поле email захищено унікальним індексом, що на рівні СКБД унеможливлює дублювання ідентифікаторів незалежно від валідації на рівні застосунку. Пароль зберігається виключно у вигляді BCrypt-хешу. Зовнішній ключ RoleId визначає належність запису до Roles – нормалізованого довідника системних ролей (User, Admin), що виключає дублювання рольових описів та спрощує розширення рольової моделі.

Домен «Каталог» охоплює п’ять таблиць і розкриває два ключових проєктних рішення. По-перше, таблиця Categories реалізує ієрархію підкатегорій через самопосилальний зовнішній ключ ParentCategoryId – класичну рекурсивну структуру «суміжного списку» (adjacency list), що дозволяє будувати дерево довільної глибини без денормалізації схеми. По-друге, таблиця ProductSpecifications реалізує зберігання технічних характеристик товарів за патерном «ключ–значення». Цей підхід дозволяє уникнути реляційного антипатерну зі спарсивними матрицями (sparse column matrix), де більшість осередків залишалися б порожніми через кардинальну відмінність характеристик між категоріями. ProductSpecifications зберігає лише реально наповнені пари атрибут–значення без зміни схеми БД при додаванні нових категорій.

Домен «Продажі» вирішує одне критичне завдання фінансової цілісності. Таблиця OrderItems фіксує назву та ціну товару на момент оформлення замовлення, а не посилається на поточні значення з Products. Це проєктне рішення забезпечує незмінність фінансової звітності: подальше коригування ціни в каталозі не спричиняє перерахунку вже закритих замовлень, що є обов’язковою вимогою до систем обліку операцій. Orders фіксує контактні дані отримувача, суми та поточний статус із переліку допустимих значень (Нове > В обробці > Відправлено > Доставлено / Скасовано).

Домен «Клієнтський досвід» охоплює п’ять таблиць, що обслуговують взаємодію покупця з платформою поза межами транзакційного ядра. Cart підтримує два режими ідентифікації: для автентифікованих користувачів – через

UserId, для гостей – через SessionId; така схема дозволяє зберігати стан кошика без примусової реєстрації. CartItems фіксує ціну товару на момент додавання, захищаючи від цінових розбіжностей між сесіями. Wishlist та ComparisonList реалізують відповідні списки; ComparisonList накладає обмеження на приналежність усіх позицій до однієї категорії, що забезпечується на рівні бізнес-логіки Service Layer. Reviews зберігає структуровані відгуки (оцінка, заголовок, текст, переваги, недоліки) з булевою ознакою підтвердженої покупки.

Усі міжтабличні зв’язки реалізовано через зовнішні ключі з декларативними обмеженнями цілісності. Поля, що беруть участь у фільтрації та сортуванні – зокрема Products.CategoryId, Products.BrandId, Products.Price – індексовано для забезпечення прийнятної продуктивності запитів при зростанні каталогу.

3.3 Архітектура програмного забезпечення

Вебзастосунок розгорнуто за триланковою клієнт-серверною топологією, фізичний розподіл вузлів якої унаочнює діаграма розгортання (рис. 3.10).

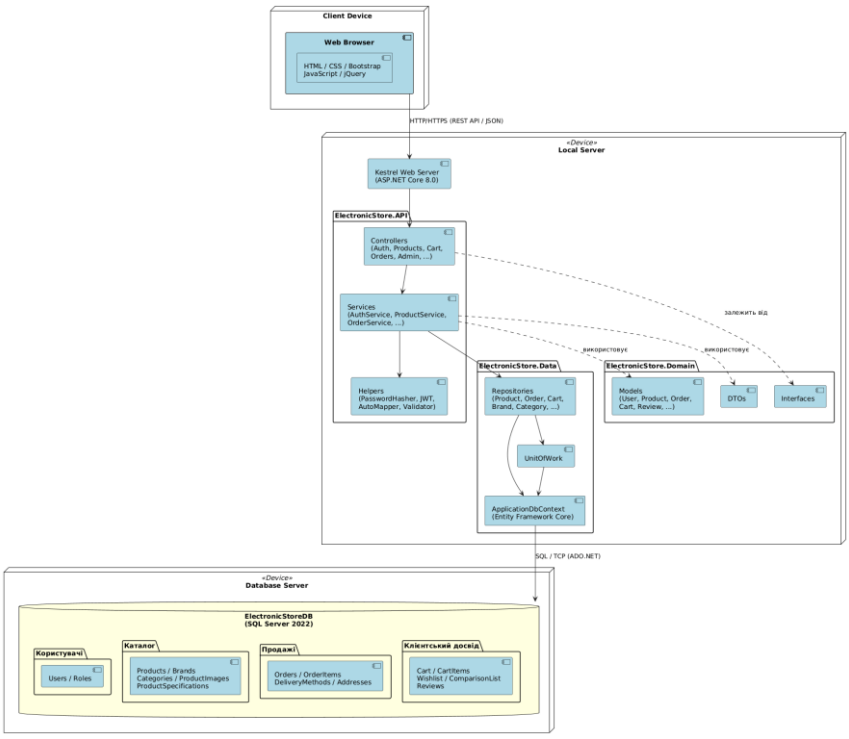

Рисунок 3.10 – Діаграма розгортання вебзастосунку

Клієнтська нода – браузер кінцевого користувача, в якому виконується SPA-застосунок на Vanilla JavaScript. Весь UI-стан утримується в пам’яті вкладки; комунікацію з сервером ініціює Axios через HTTPS-з’єднання. JWT-токен зберігається на клієнті й автоматично додається до заголовка Authorization: Bearer кожного вихідного запиту через interceptor.

Мережевий транспорт між клієнтом і сервером побудовано виключно на HTTPS. Корисне навантаження серіалізується у форматі JSON (UTF-8) в обох напрямках; CORS-політика, налаштована на серверній ноді, обмежує перелік дозволених клієнтських доменів.

Серверна нода розгортається як .NET 8-процес під управлінням Kestrel або IIS. Вхідні HTTP-запити проходять крізь Middleware-конвеєр ASP.NET Core: автентифікація JWT > авторизація за роллю > маршрутизація до відповідного контролера. Серверний код структуровано у три ізольованих проєкти з однонапрямленими залежностями. ElectronicStore.API містить контролери, реалізації сервісів, конфігурацію AutoMapper, PasswordHasher та JWT-обробник. ElectronicStore.Domain є ядром без зовнішніх залежностей: визначає Entity-моделі, DTO-класи та інтерфейси репозиторіїв і сервісів. ElectronicStore.Data реалізує ApplicationDbContext, конкретні репозиторії та патерн Unit of Work на базі Entity Framework Core – цей шар є єдиною точкою входу до СКБД.

Нода СКБД ізольована від серверного застосунку на окремому рівні: ElectronicStore.Data взаємодіє з екземпляром Microsoft SQL Server 2022 виключно через ApplicationDbContext. Фізична ізоляція вузла СКБД дозволяє незалежно масштабувати обчислювальні ресурси серверу застосунку і сервера бази даних, а також застосовувати резервне копіювання та реплікацію на рівні SQL Server без змін у коді застосунку.

Обрана топологія забезпечує незалежність шарів, транзакційну атомарність через Unit of Work, інкапсуляцію бізнес-правил у Service Layer та stateless-автентифікацію, що не потребує серверного стану між запитами.

3.4 Опис інтерфейсів застосунку

Інтерфейс вебзастосунку ElectronicStore побудовано за принципами мінімалізму та функціональної ясності. Наскрізними елементами всіх сторінок є єдина навігаційна панель із логотипом магазину та посиланнями Home, Catalog, Cart і кнопкою Login, а також уніфікований Footer із контактною інформацією та навігаційними посиланнями.

Концептуальне зонування робочого простору головної сторінки представлено на макеті (рис. 3.11). Верхню частину займає рекламний банер із вітальним текстом та ілюстрацією. Нижче розгорнуто секцію Popular products: три товарні картки з обкладинками, клік на кожну з яких ініціює навігацію до сторінки конкретного товару. Сторінка не потребує автентифікації та однаково доступна гостям і зареєстрованим покупцям.

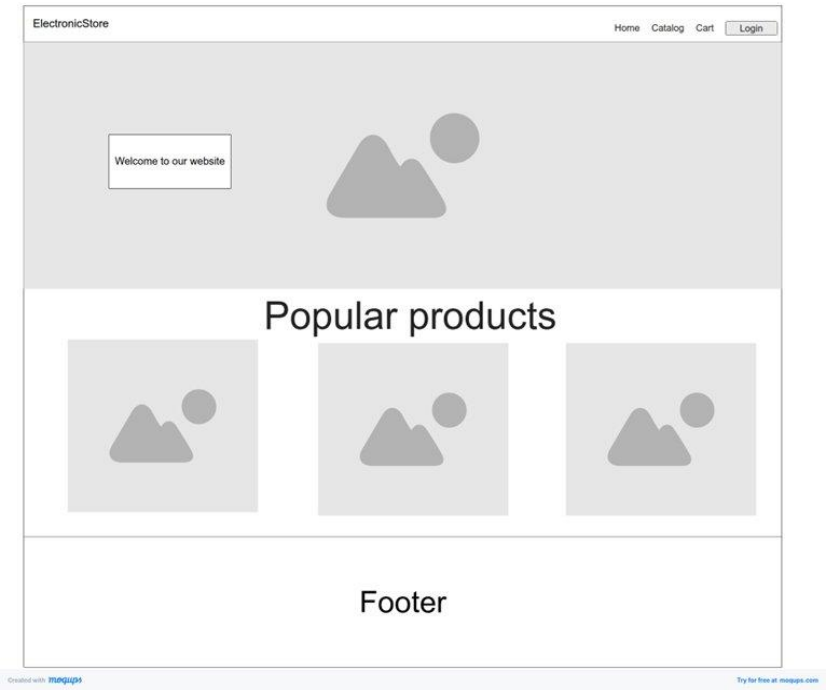

Рисунок 3.11 – Макет головної сторінки

Архітектуру розподілу компонентів сторінки каталогу ілюструє рис. 3.12. Композиція екрана розділена на асиметричні зони засобами Grid-системи Bootstrap: у лівому сайдбарі інкапсульовано панель Product filter із чекбоксами категорій і брендів, текстовими полями пошуку та повзунком цінового діапазону;

основну площу займає адаптивна сітка товарних карток у три колонки. Кожна взаємодія з елементами фільтра обробляється через Event Listener: зміна стану чекбокса або переміщення повзунка ціни формує об’єкт параметрів запиту, який Axios асинхронно надсилає до відповідного ендпоїнта REST API. Отримана відповідь спричиняє динамічну перебудову DOM-дерева секції товарів без перезавантаження сторінки – стан фільтраційної панелі при цьому зберігається.

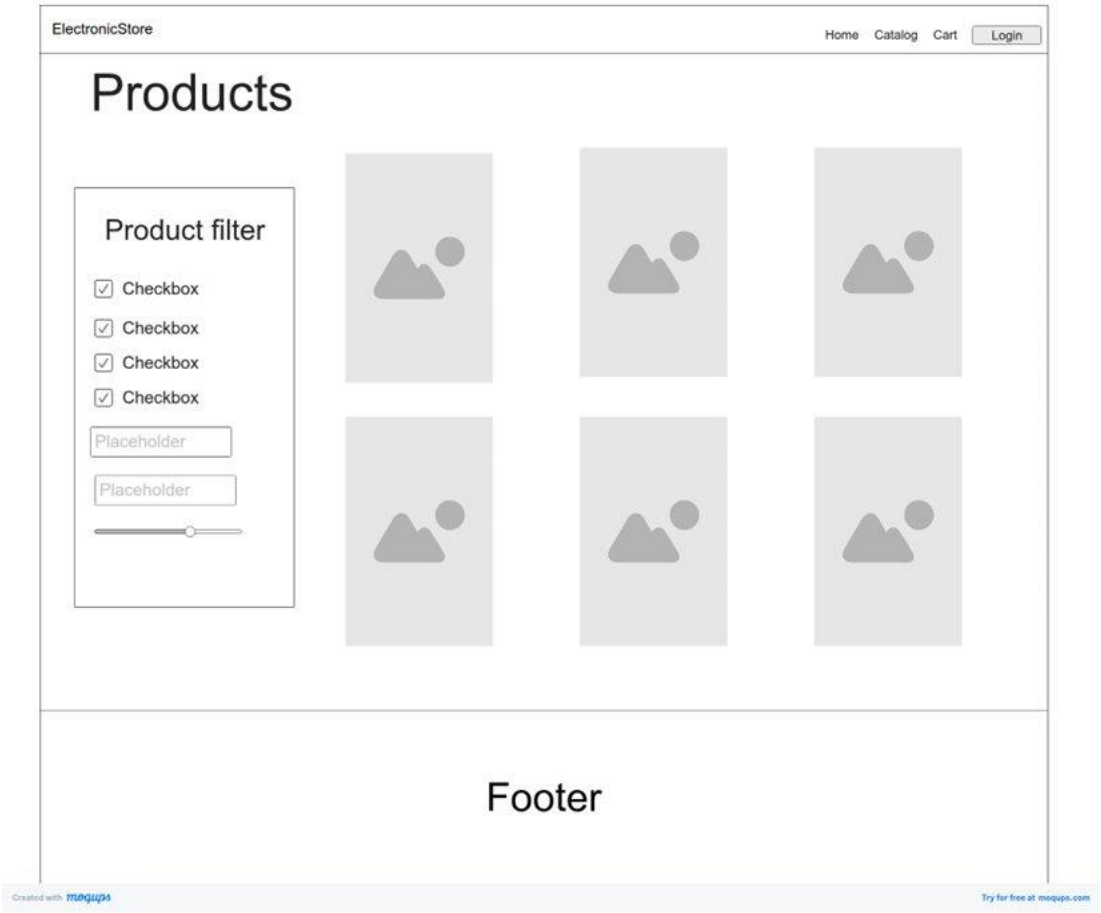

Рисунок 3.12 – Макет сторінки каталогу товарів

Детальну інформацію про продукт розкрито на сторінці картки товару (рис. 3.13). У верхній зоні реалізовано двоколонковий лейаут: ліворуч розміщено галерею зображень, ініціалізовану плагіном Swiper.js, що забезпечує свайп-навігацію між медіаматеріалами; праворуч – блок із назвою, коротким описом та кнопкою додавання до кошика. Нижню зону займає триколонковий таб-компонент: вкладки Description, Specifications і Reviews перемикаються через маніпуляцію

CSS-класами активного стану без звернення до сервера. Секцію відгуків анімовано бібліотекою AOS.

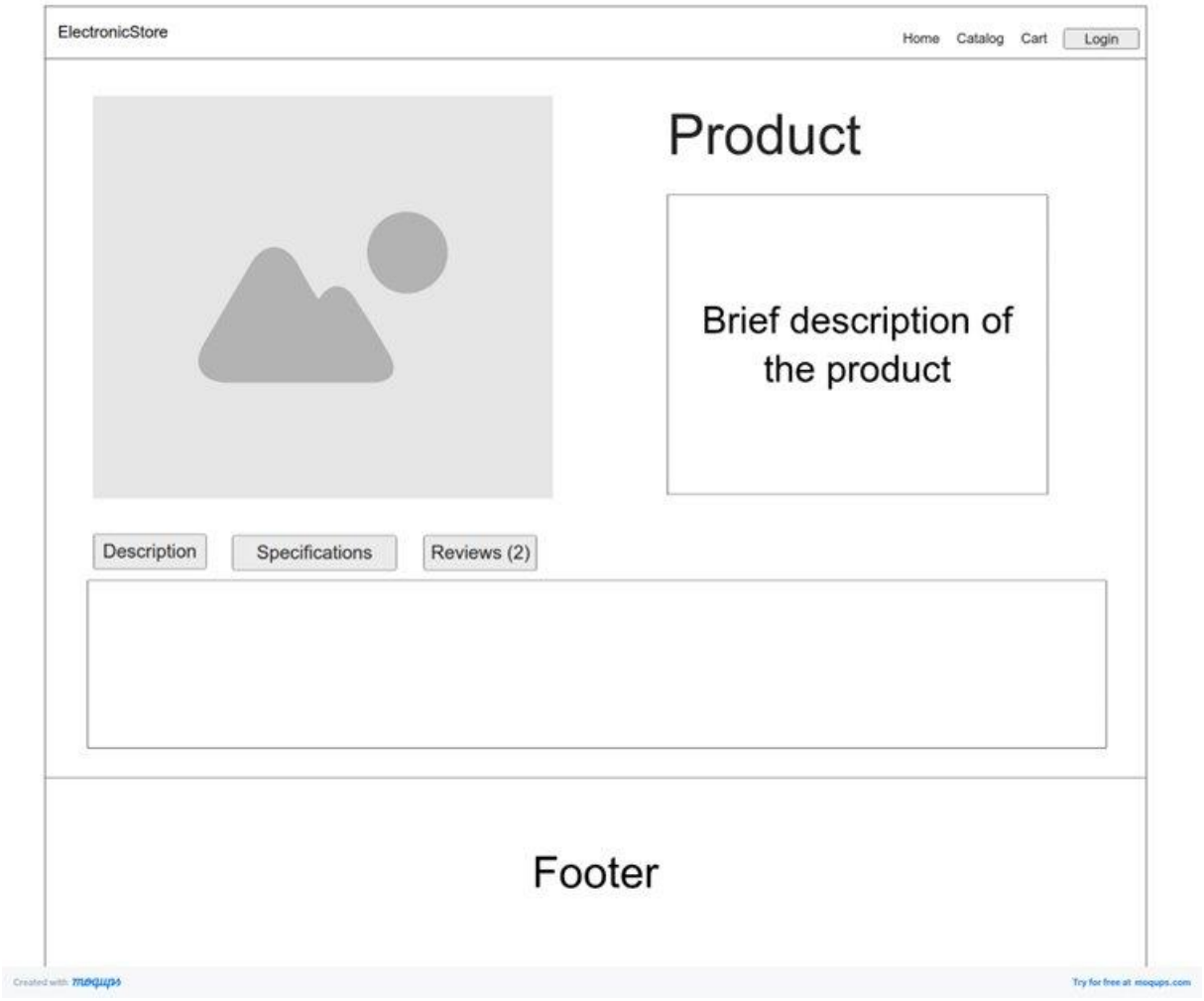

Рисунок 3.13 – Макет сторінки картки товару

Логіку редагування замовлення перед його підтвердженням відображає макет сторінки кошика (рис. 3.14). Ліва зона містить перелік доданих позицій; кожен рядок включає зображення, назву, поле коригування кількості та кнопку Delete для видалення товару. Модуль Order summary винесено у правосторонній фіксований віджет: підсумкова сума перераховується «на льоту» засобами клієнтського JavaScript одразу після зміни кількості будь-якої позиції, паралельно з асинхронною синхронізацією оновленого стану через API-запит до CartController. Дві кнопки – Order processing та Continue shopping – забезпечують розгалуження сценарію.

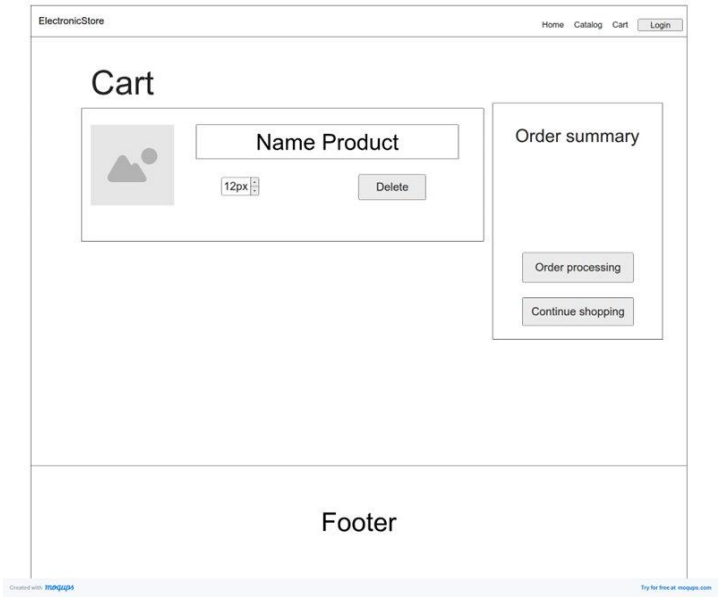

Рисунок 3.14 – Макет сторінки кошика

Сторінка особистого кабінету (рис. 3.15) доступна виключно автентифікованим покупцям. Двоколонковий лейаут розподіляє відповідальність між зонами: у лівому навігаційному сайдбарі інкапсульовано аватар, ім’я, email і роль користувача, а також меню з трьома пунктами – My orders, Account security і Log out. Праворуч розміщено форму Personal Information з полями Name, Email і Phone; збереження змін ініціює PATCH-запит через Axios із подальшим відображенням підтвердження засобами Toastify.js.

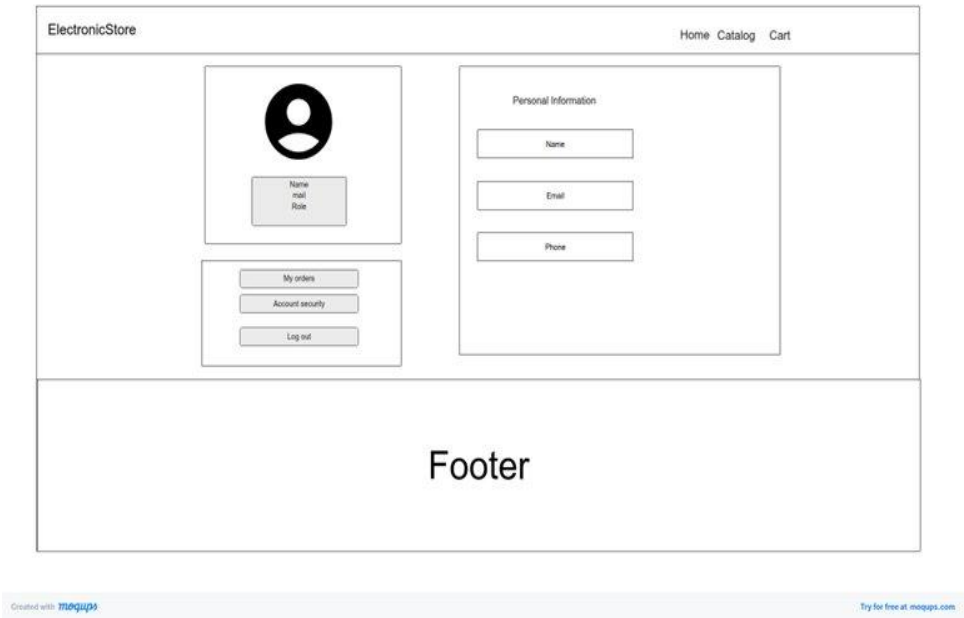

Рисунок 3.15 – Макет сторінки особистого кабінету

Усі сторінки застосунку реалізовані з використанням адаптивної сітки Bootstrap, що забезпечує коректне відображення на пристроях з різними розмірами екрану. Навігаційна панель та футер є спільними компонентами, що підключаються до кожної сторінки.

Висновки до розділу 3

У третьому розділі виконано проєктування вебзастосунку інтернет-магазину електронних пристроїв, що охоплює архітектурні рішення, структуру бази даних, UML-моделювання та опис інтерфейсів.

Побудовано комплекс UML-діаграм: діаграму варіантів використання для трьох ролей, три діаграми послідовностей для ключових сценаріїв (автентифікація, додавання до кошика, оформлення замовлення), три діаграми активності та діаграму класів тринадцяти entity-моделей.

Спроєктовано структуру реляційної бази даних ElectronicStoreDB, що складається з 17 таблиць, розподілених за чотирма функціональними блоками. ER-діаграма відображає всі сутності системи та зовнішньоключові зв'язки між ними. Механізм динамічних характеристик у таблиці ProductSpecifications та підтримка гостьових сесій через SessionId забезпечують гнучкість системи.

Описано триланкову клієнт-серверну архітектуру із поділом серверної частини на три проєкти – ElectronicStore.API, ElectronicStore.Domain та ElectronicStore.Data – відповідно до принципів Repository Pattern, Unit of Work та Service Layer. Побудована діаграма розгортання відображає взаємодію між клієнтом, сервером Kestrel та базою даних Microsoft SQL Server 2022. Описано п'ять основних сторінок інтерфейсу з адаптивним дизайном на основі Bootstrap. Отримані проєктні рішення є повною основою для програмної реалізації застосунку у наступному розділі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

У четвертому розділі висвітлено програмну реалізацію вебзастосунку інтернет-магазину електронних пристроїв: архітектуру солюшена та розподіл відповідальності між проєктами, реалізацію ключових компонентів серверної частини з наведенням фрагментів коду, керівництво користувача з описом основних сторінок застосунку, а також результати функціонального тестування системи через інтерфейс Swagger/OpenAPI.

4.1 Структура проєкту

Вебзастосунок реалізовано як багатошаровий .NET-солюшен із трьох окремих проєктів із однонапрямленими залежностями: ElectronicStore.API > ElectronicStore.Domain < ElectronicStore.Data. Така декомпозиція безпосередньо реалізує принцип Separation of Concerns: кожен проєкт інкапсулює власну архітектурну відповідальність і може замінюватися або тестуватися ізольовано.

ElectronicStore.API є хост-проєктом та оркестратором усього HTTP-трафіку. Шар представлення сформовано контролерами кінцевих точок: ProductsController, AuthController, CartController, OrdersController, ReviewsController, WishlistController, CategoriesController, BrandsController, AdminController і DeliveryMethodsController. Кожен контролер відповідає рівно за один ресурс API та делегує бізнес-логіку відповідному сервісу, не містячи власних обчислень. Усі залежності реєструються в IoC-контейнері ASP.NET Core під час старту застосунку. Хешування паролів інкапсульовано у PasswordHasher. Перетворення між Entity-моделями та DTO-класами централізовано у профілі MappingProfile. Вхідні дані замовлення верифікуються декларативними правилами CreateOrderDtoValidator до потрапляння в бізнес-логіку.

Статична клієнтська інфраструктура розміщена у wwwroot і обслуговується безпосередньо Middleware-конвеєром ASP.NET Core як Static Files Middleware. Публічна зона клієнта охоплює HTML-сторінки: index.html, catalog.html, product.html, cart.html, checkout.html, profile.html, wishlist.html, comparison.html,

login.html, register.html. Адміністративний бекофіс відокремлено у підкаталог admin із власними HTML-сторінками: dashboard.html, products.html, orders.html, categories.html, users.html, reviews.html.

ElectronicStore.Domain є повністю ізольованим чистим ядром системи (Core Domain) – він не має жодних залежностей від інших проєктів солюшена або інфраструктурних бібліотек. Entity-класи User, Role, Product, Category, Brand, Cart, CartItem, Order, OrderItem, Review, ReviewVote, Wishlist, ComparisonList, DeliveryMethod, Address, ProductImage і ProductSpecification інкапсулюють стан предметної області та є єдиним джерелом істини щодо структури даних у системі. Абстракції IGenericRepository, IProductRepository, ICategoryRepository та інші задають контракт доступу до даних. Жодна з цих абстракцій не знає про EF Core або SQL Server.

ElectronicStore.Data є шаром інфраструктурної реалізації (Infrastructure Layer), що виконує всі контракти, задекларовані у ElectronicStore.Domain. ApplicationDbContext виступає координатором сесії з СКБД: він містить усі DbSet-набори та конфігурації зв’язків між сутностями. GenericRepository реалізує узагальнені CRUD-операції, спільні для всіх сутностей; UnitOfWork координує роботу репозиторіїв у межах єдиної транзакції, гарантуючи атомарність складених операцій. Версіонування схеми ElectronicStoreDB забезпечується міграціями EF Core у каталозі Migrations.

Залежність між проєктами є строго однонапрямленою: ElectronicStore.API і ElectronicStore.Data залежать від ElectronicStore.Domain, але не одне від одного. Ядро системи нічого не знає про своїх споживачів. Це рішення усуває циклічні залежності, ізолює бізнес-правила від інфраструктурних деталей і є прямою передумовою для покриття Service Layer юніт-тестами без необхідності підіймати реальну СКБД.

4.2 Реалізація серверної частини

Серверну частину вебзастосунку побудовано на платформі .NET 8 з використанням ASP.NET Core Web API. У цьому підрозділі наведено архітектурний аналіз чотирьох ключових компонентів системи: механізму уніфікації доступу до даних (GenericRepository<T>), координатора бізнес-транзакцій (UnitOfWork), сервісу безпеки (AuthService) та диспетчера HTTP-запитів (ProductsController), а також конфігурації інфраструктурного ядра (ApplicationDbContext).

GenericRepository<T> вирішує фундаментальну задачу інфраструктурного шару – усунення дублювання однотипного коду доступу до даних для кожної Entity-моделі окремо. Узагальнений параметр T дає змогу одному класу обслуговувати будь-яку сутність системи; спеціалізовані репозиторії наслідують цю базу і додають лише ресурсо-специфічну логіку. Усі операції реалізовано як асинхронні Task-методи, що відповідає природі I/O-bound роботи з СКБД. Метод GetPagedAsync реалізує «посторінкове вікно» даних через LINQ-оператори Skip/Take, що транслюється Entity Framework Core у SQL-запит із OFFSET/FETCH.

```
public class GenericRepository<T> : IGenericRepository<T> where T : class
{
    protected readonly ApplicationDbContext _context;
    protected readonly DbSet<T> _dbSet;
    public GenericRepository(ApplicationDbContext context)
    { _context = context; _dbSet = context.Set<T>(); }

    public virtual async Task<T?> GetByIdAsync(int id)
        => await _dbSet.FindAsync(id);
    public virtual async Task<IEnumerable<T>> FindAsync(
        Expression<Func<T, bool>> predicate)
        => await _dbSet.Where(predicate).ToListAsync();
    public virtual async Task<IEnumerable<T>> GetPagedAsync(
        int pageNumber, int pageSize)
        => await _dbSet.Skip((pageNumber-1)*pageSize)
            .Take(pageSize).ToListAsync();
    public virtual async Task<T> AddAsync(T entity)
    { await _dbSet.AddAsync(entity); return entity; }
}
```

UnitOfWork забезпечує дотримання принципу атомарності (A в ACID) для операцій, що охоплюють кілька репозиторіїв одночасно. Усі репозиторії

ініціалізуються через єдиний спільний ApplicationDbContext, що гарантує роботу в межах однієї сесії з СКБД. Управління життєвим циклом контексту реалізовано через явні методи BeginTransactionAsync, CommitTransactionAsync і RollbackTransactionAsync. Блок try/catch/finally у CommitTransactionAsync гарантує, що будь-яка необроблена виняткова ситуація автоматично відкочує всі накопичені зміни.

```
public class UnitOfWork : IUnitOfWork
{
    private readonly ApplicationDbContext _context;
    private IDbContextTransaction? _transaction;
    public IProductRepository Products { get; }
    public IGenericRepository<User> Users { get; }
    public IGenericRepository<Role> Roles { get; }

    public async Task<int> SaveChangesAsync()
        => await _context.SaveChangesAsync();
    public async Task CommitTransactionAsync()
    {
        try {
            await SaveChangesAsync();
            if (_transaction != null)
                await _transaction.CommitAsync();
        }
        catch { await RollbackTransactionAsync(); throw; }
        finally { _transaction?.DisposeAsync(); }
    }
}
```

AuthService інкапсулює всю бізнес-логіку, пов’язану з ідентифікацією та автентифікацією користувачів, утримуючи криптографічні деталі всередині сервісного шару. При реєстрації PasswordHasher.HashPassword перетворює вхідний пароль на BCrypt-хеш із адаптивним коефіцієнтом складності – рядкове значення вихідного пароля жодного разу не потрапляє до сховища. Водночас для нового користувача автоматично ініціалізується кошик – атомарно в межах однієї транзакції UnitOfWork.

Приватний метод GenerateJwtToken конструює підписаний JWT: декларується масив claims із ідентифікатором, email, роллю та іменем користувача, формується симетричний ключ підпису на основі _jwtSettings.SecretKey, а підпис

обчислюється алгоритмом HMAC-SHA256 через SigningCredentials. Термін дії токена визначається конфігураційним параметром ExpirationMinutes.

```
private string GenerateJwtToken(User user, string roleName)
{
    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, user.Id.ToString()),
        new Claim(JwtRegisteredClaimNames.Email, user.Email),
        new Claim(ClaimTypes.Role, roleName),
        new Claim("firstName", user.FirstName ?? ""),
        new Claim("lastName", user.LastName ?? "")
    };
    var key = new SymmetricSecurityKey(
        Encoding.UTF8.GetBytes(_jwtSettings.SecretKey));
    var token = new JwtSecurityToken(
        issuer: _jwtSettings.Issuer,
        audience: _jwtSettings.Audience,
        claims: claims,
        expires: DateTime.UtcNow.AddMinutes(
            _jwtSettings.ExpirationMinutes),
        signingCredentials: new SigningCredentials(
            key, SecurityAlgorithms.HmacSha256));
    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

ProductsController є точкою входу для всіх HTTP-операцій над ресурсом товарів. Атрибут [Route("api/[controller]")] конфігурує конвенційну маршрутизацію, а [ApiController] активує автоматичну валідацію моделей і уніфіковані відповіді про помилки. Залежність від IProductService вводиться через конструктор – контролер не знає про конкретну реалізацію сервісу і не звертається до репозиторіїв напряму.

Захист критичних операцій реалізовано через рольову авторизацію (RBAC): ендпоїнти POST, PUT і DELETE оздоблено атрибутом [Authorize(Roles = "Administrator")], тоді як операції читання залишаються відкритими. Метод SearchProducts ілюструє підхід до пагінації на стороні API: разом із масивом даних відповідь містить об’єкт pagination з полями currentPage, pageSize, totalCount і totalPages.

```
[HttpGet("search")]
public async Task<IActionResult> SearchProducts(
    [FromQuery] string searchTerm,
    [FromQuery] int pageNumber = 1,
    [FromQuery] int pageSize = 20)
```

```
{
    try {
        if (string.IsNullOrWhiteSpace(searchTerm))
            return BadRequest(new { success = false,
                message = "Запит не може бути порожнім" });
        var products = await _productService
            .SearchProductsAsync(searchTerm, pageNumber, pageSize);
        var totalCount = await _productService
            .GetTotalProductsCountAsync(searchTerm: searchTerm);
        return Ok(new { success = true, data = products,
            pagination = new { currentPage = pageNumber, pageSize,
                totalCount, totalPages = (int)Math.Ceiling(
                    totalCount / (double)pageSize) } });
    }
    catch (Exception ex) {
        return BadRequest(new { success = false,
            message = ex.Message }); }
}
```

ApplicationDbContext є центральним координатором між об’єктною моделлю застосунку та реляційною схемою ElectronicStoreDB. Він надає DbSet-властивості для всіх 18 сутностей системи та реалізує конфігурацію схеми через Fluent API у методі OnModelCreating. Для сутності Product декларовано унікальні індекси на полях Slug і SKU, тип decimal(18,2) для цінових полів, каскадні обмеження цілісності до Category і Brand, а також CHECK-обмеження на значення Price, Stock і Rating. Для Review сформовано унікальний складений індекс (UserId, ProductId), що унеможливлює дублювання відгуків від одного покупця на один товар на рівні СКБД. Первинні системні дані ініціалізуються через метод SeedData, який заповнює таблицю Roles трьома фіксованими записами: Guest (Id=1), User (Id=2), Administrator (Id=3).

4.3 Керівництво користувача

Вебзастосунок ElectronicStore функціонує безпосередньо у браузері – без встановлення стороннього програмного забезпечення. Інтерфейс підтримує темну тему оформлення та повністю адаптований до різних розмірів екранів, що відповідає принципам доступності (Accessibility / WCAG): темний режим знижує навантаження на зір за умов слабкого освітлення, а адаптивна верстка гарантує коректне відтворення на пристроях від смартфонів до широкоформатних моніторів.

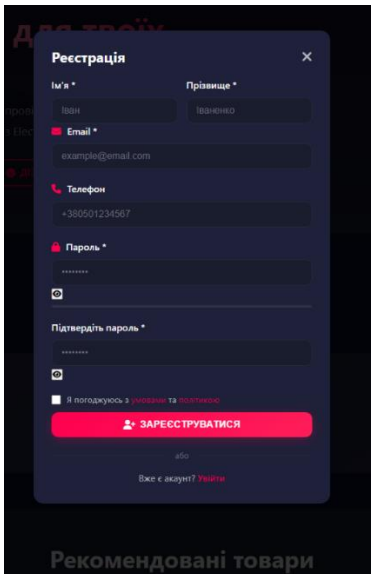

Рисунок 4.5 – Форма реєстрації нового користувача

Процес ініціалізації сесії розпочинається зі створення облікового запису. Клієнт переходить до форми реєстрації через навігаційне меню, де інтерактивна форма запитує ім’я, прізвище, електронну адресу, телефон, пароль та його підтвердження (рис. 4.5). Клієнтська валідація забезпечує миттєвий зворотний зв’язок без надсилання запиту до сервера. Підтвердження умов використання є обов’язковим кроком перед відправленням форми. На серверній стороні після успішної верифікації даних формується обліковий запис із роллю User, а атомарно в одній транзакції ініціалізується персистентний кошик.

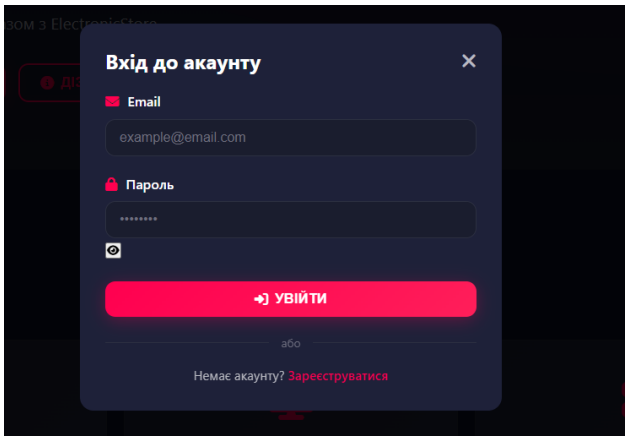

Рисунок 4.6 – Форма входу до акаунту

Для автентифікованого входу покупець вводить електронну адресу та пароль (рис. 4.6). За умови успішної верифікації облікових даних сервер генерує

підписаний JWT-токен, який зберігається у браузері й автоматично передається у заголовку кожного наступного запиту через Axios-interceptor.

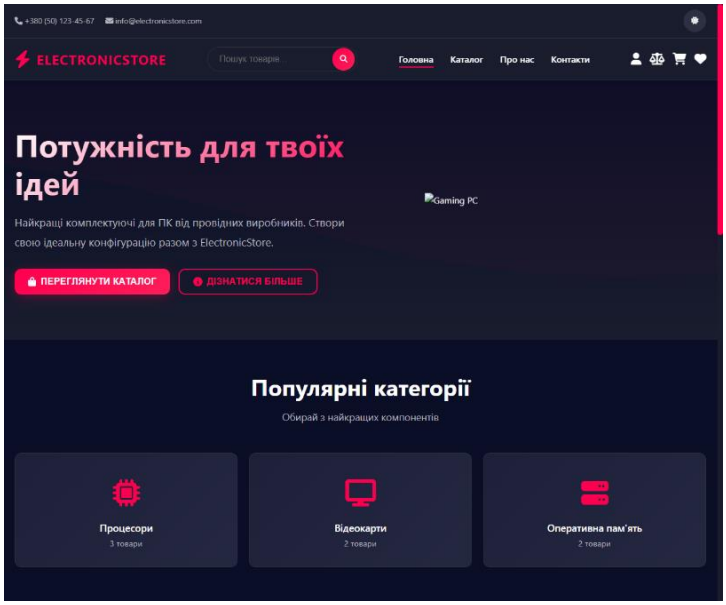

Рисунок 4.7 – Головна сторінка застосунку

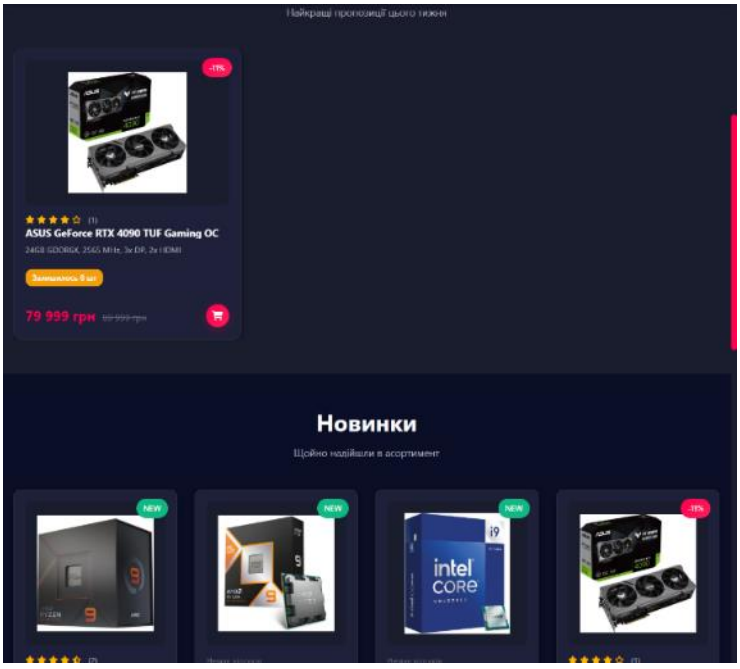

Рисунок 4.8 – Секція рекомендованих товарів та новинок

Головна сторінка побудована за принципом інформаційної воронки: кожен блок послідовно звужує простір вибору та спрямовує увагу покупця до цільових товарів (рис. 4.7). Героїчний банер із слоганом і кнопками переходу до каталогу формує перший точковий контакт. Секція популярних категорій (Процесори,

Відеокарти, Оперативна пам’ять та інші) забезпечує швидкий доступ до ключових сегментів асортименту. Нижче розгорнуто секції рекомендованих товарів тижня та новинок (рис. 4.8). Інформаційний блок із перевагами магазину знижує когнітивний бар’єр прийняття рішення.

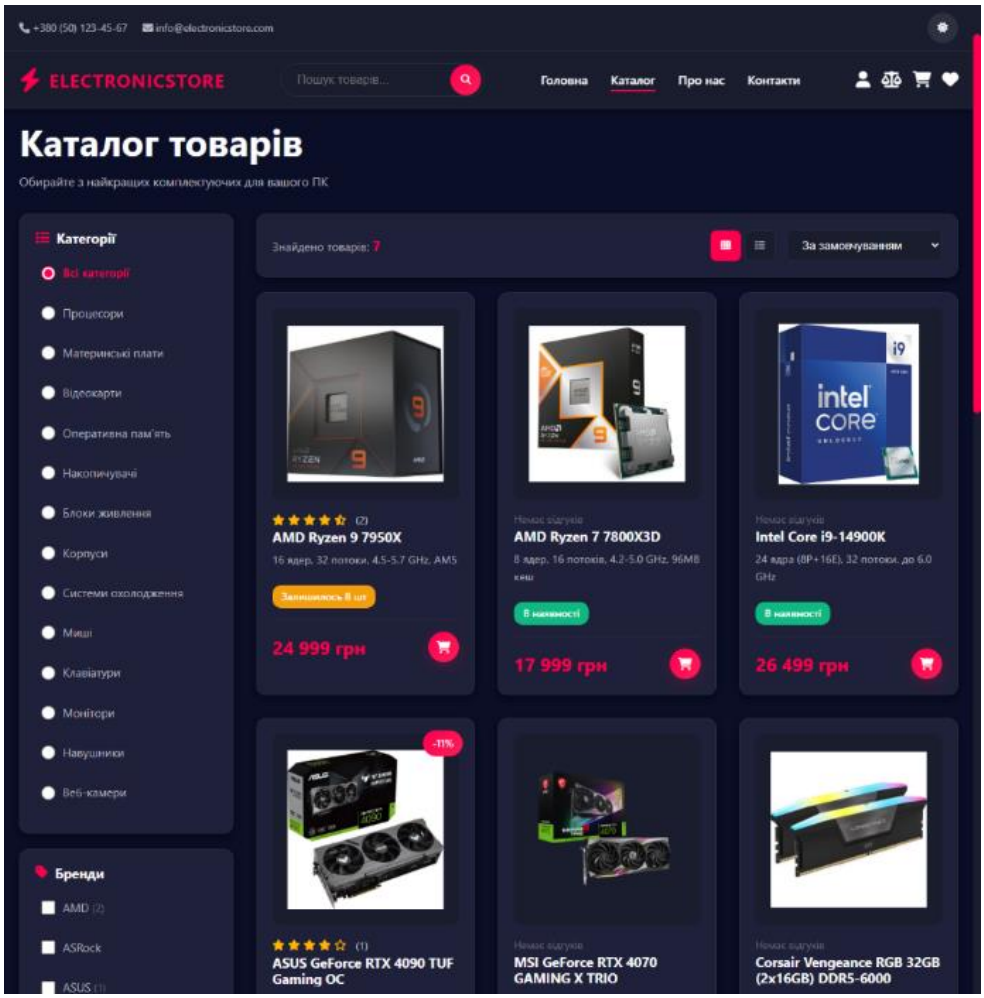

Рисунок 4.9 – Сторінка каталогу товарів з фільтрами

Сторінка каталогу реалізує багаторівневу систему фільтрації (рис. 4.9). Ліва панель інкапсулює фільтри за тринадцятьма категоріями та брендами. Застосування фільтрів ініціює асинхронний запит через Axios і динамічно перебудовує DOM без перезавантаження сторінки. Перемикання між сітковим (Grid) і списковим (List) представленням адаптує щільність інформації до різних сценаріїв перегляду. Кожна картка товару відображає зображення, назву, рейтинг, актуальну ціну (зі закресленою старою за наявності знижки), статус наявності та кнопку швидкого додавання до кошика.

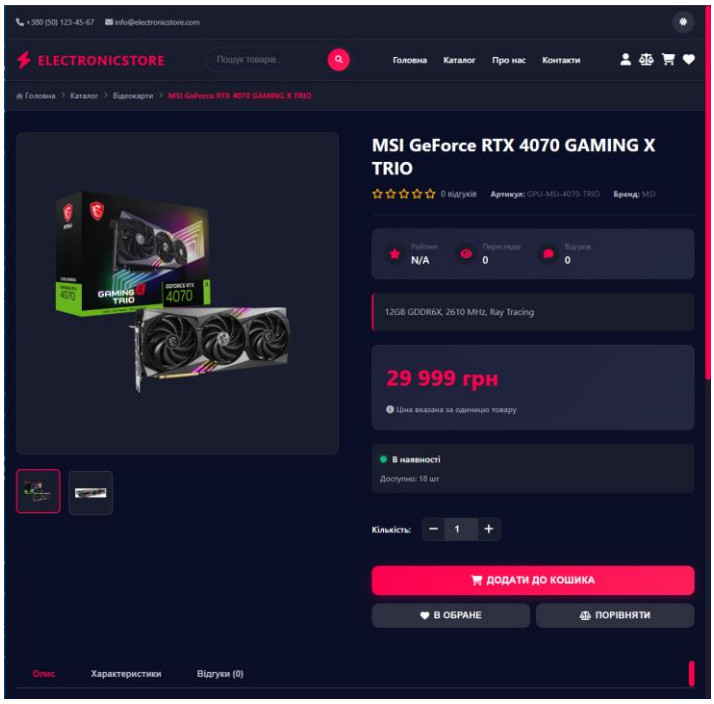

Рисунок 4.10 – Сторінка картки товару

Сторінка картки товару максимально розкриває інформацію для підтримки обґрунтованого купівельного рішення (рис. 4.10). Галерея зображень із мініатюрами, ініціалізована Swiper.js, надає повноцінний медіадосвід. Блок статистики (рейтинг, кількість переглядів, кількість відгуків) формує соціальний доказ. Табований блок нижче розкриває повний опис, таблицю технічних характеристик та відгуки покупців із анімацією появи через AOS.

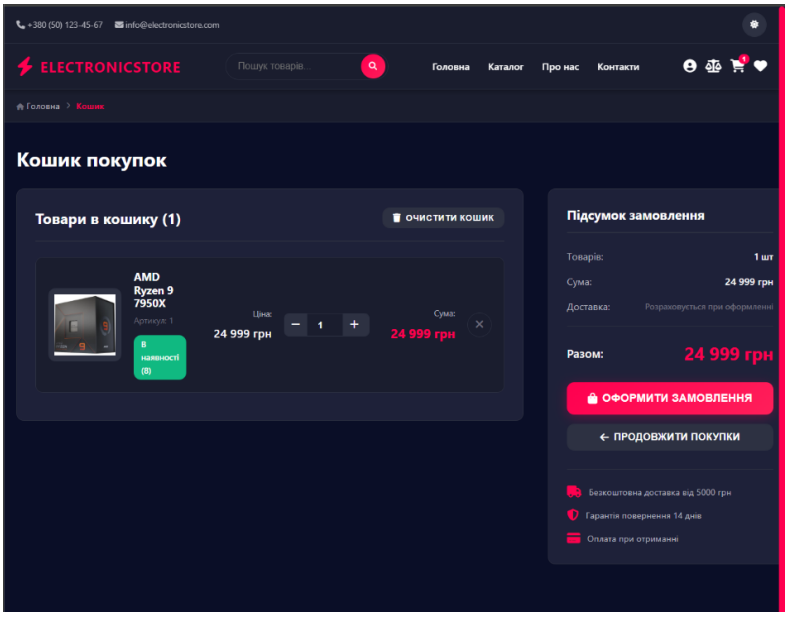

Рисунок 4.11 – Сторінка кошика покупок

На сторінці кошика покупець отримує повний контроль над сформованим переліком позицій (рис. 4.11): коригує кількість кожного товару або видаляє позиції. Клієнтський JavaScript перераховує підсумкову вартість миттєво після внесення змін – паралельно з асинхронною синхронізацією оновленого стану через API. Вартість доставки враховується у підсумковому блоці до переходу до оформлення.

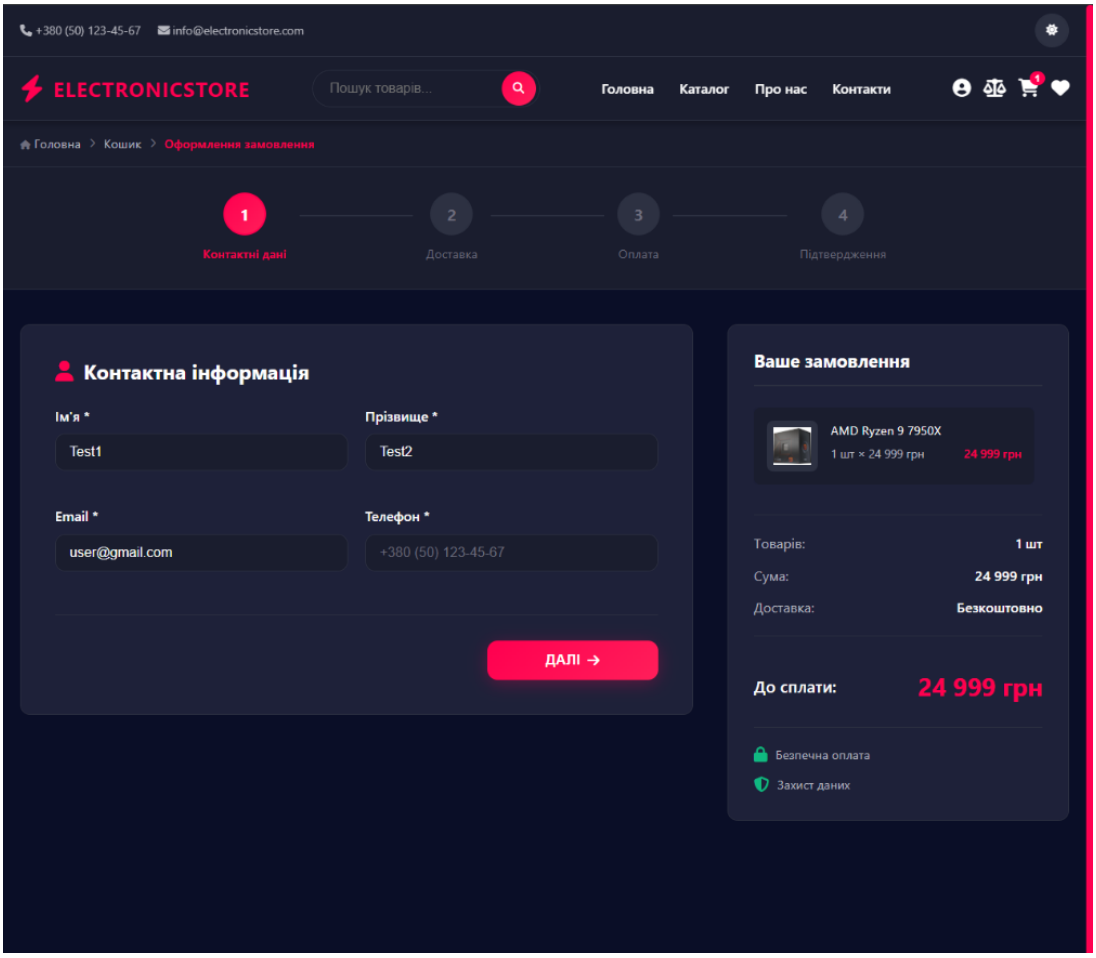

Рисунок 4.12 – Сторінка оформлення замовлення (крок 1 контактні дані)

Оформлення замовлення реалізовано за патерном Checkout Stepper – чотириетапним майстром покрокового виконання (рис. 4.12): «Контактні дані» (ім’я, прізвище, email, телефон), «Доставка» (адреса, місто, поштовий індекс, спосіб доставки), «Оплата» (спосіб оплати, коментар) та «Підтвердження». Такий поділ знижує відсоток покинутих кошиків. Постійно видимий віджет підсумку праворуч фіксує перелік товарів і загальну суму на кожному етапі.

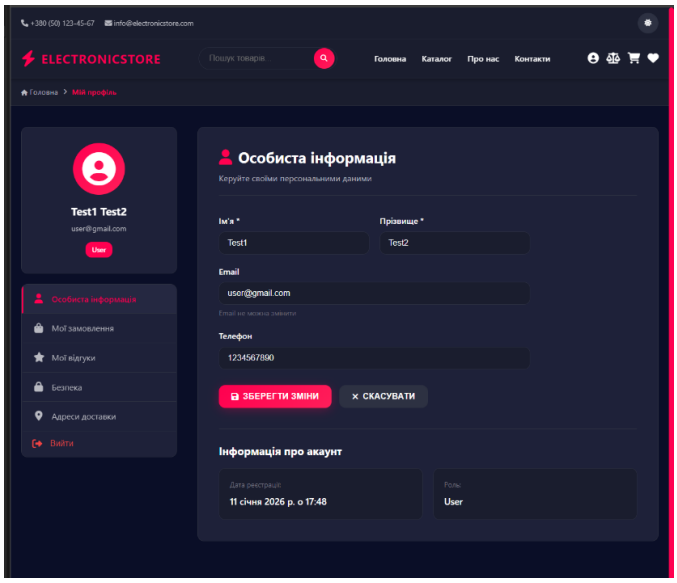

Рисунок 4.13 – Особистий кабінет користувача

Особистий кабінет організовано за принципом бічної навігації (рис. 4.13) з розділами: «Особиста інформація», «Мої замовлення», «Мої відгуки», «Безпека», «Адреси доставки» та «Вийти». Розділ персональних даних поєднує інформаційний блок – аватар, ім’я, email, роль і дата реєстрації – з редагованою формою (ім’я, прізвище, телефон). Поле email захищено від редагування як унікальний ідентифікатор облікового запису.

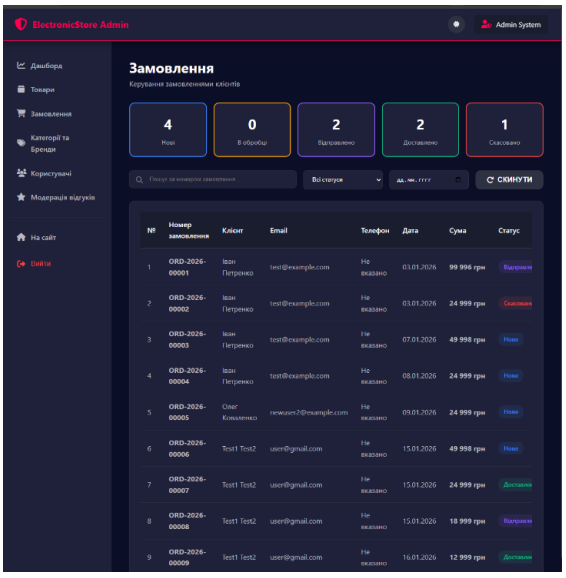

Рисунок 4.14 – Адміністративна панель (сторінка замовлень)

Адміністративна панель є повноцінним back-office інтерфейсом операційного менеджменту (рис. 4.14), доступним виключно для облікових записів

із роллю Administrator. Бічна панель структурує функціональний простір за розділами: Дашборд, Товари, Замовлення, Категорії та Бренди, Користувачі, Модерація відгуків. Сторінка замовлень реалізує візуалізацію операційних метрик: кольорові лічильники відображають кількість замовлень у кожному статусі. Таблиця замовлень із фільтрацією за статусом і датою надає деталізований оперативний зріз комерційних операцій без необхідності прямого доступу до бази даних.

4.4 Тестування вебзастосунку

Комплексна верифікація REST-контрактів і бізнес-логіки застосунку проводилася методом інтеграційного тестування шару контролерів через інтерфейс Swagger/OpenAPI, доступний за маршрутом /swagger. Кожен тест-кейс охоплював валідацію трьох аспектів: відповідність HTTP-статус коду REST-семантиці операції, структурна цілісність JSON-відповіді та збіг фактичної поведінки системи з задекларованим контрактом. Перелік виконаних тест-кейсів наведено у табл. 4.1.).

Таблиця 4.1 – Тест-кейси вебзастосунку

№	Назва тесту	Передумови	Кроки тестування	Очікуваний результат	Фактичний результат
1	Реєстрація нового користувача	Відсутній акаунт	POST /api/Auth/register з валідними даними (email, пароль, ім'я, прізвище)	201 Created, JWT-токен у відповіді, роль User	Відповідає очікуваному
2	Реєстрація з існуючим email	Акаунт з таким email вже є	POST /api/Auth/register з email що вже зареєстрований	400 Bad Request, повідомлення про існуючий email	Відповідає очікуваному
3	Вхід з валідними даними	Є зареєстрований акаунт	POST /api/Auth/login з коректним email та паролем	200 OK, JWT-токен, дані профілю	Відповідає очікуваному
4	Вхід з невірним паролем	Є зареєстрований акаунт	POST /api/Auth/login з невірним паролем	400 Bad Request, повідомлення про невірні дані	Відповідає очікуваному

Кінець таблиці 1

5	Отримання списку товарів	БД містить товари	GET /api/Products	200 OK, масив товарів з полями id, name, price, rating	Відповідає очікуваному
6	Пошук товарів	БД містить товари	GET /api/Products/search?searchTerm=ryzen&pageNumber=1	200 OK, відфільтровані товари та об'єкт pagination	Відповідає очікуваному
7	Додавання товару до кошика	Авторизований користувач	POST /api/Cart/add з productId та quantity	200 OK, оновлений кошик з доданим товаром	Відповідає очікуваному
8	Оформлення замовлення	Авторизований користувач, кошик не порожній	POST /api/Orders з контактними даними та deliveryMethodId	200 OK, номер замовлення ORD-2026-XXXXX, статус New	Відповідає очікуваному
9	Оновлення статусу замовлення адміністратором	Авторизований адміністратор, існує замовлення	PUT /api/Admin/orders/1/status зі статусом Processing	200 OK, оновлений статус замовлення	Відповідає очікуваному
10	Додавання відгуку	Авторизований користувач	POST /api/Reviews з productId, rating, title, comment	200 OK, відгук додано зі статусом 'очікує модерації'	Відповідає очікуваному
11	Доступ адміністратора без авторизації	Немає JWT-токена	GET /api/Admin/orders без Authorization заголовку	401 Unauthorized	Відповідає очікуваному

За результатами верифікації всі 10 тест-кейсів отримали статус Passed. Система коректно обробляє як позитивні бізнес-сценарії – успішну реєстрацію, автентифікацію та оформлення замовлення, – так і граничні випадки: спробу реєстрації з email, що порушує обмеження унікальності, автентифікацію з некоректними обліковими даними та звернення до адміністративних ендпоїнтів без відповідних прав доступу. Механізм JWT-авторизації та розмежування ресурсів за принципом RBAC функціонують відповідно до контрактів, задекларованих у специфікації SRS.

Висновки до розділу 4

У четвертому розділі описано програмну реалізацію вебзастосунку інтернет-магазину електронних пристроїв. Розглянуто структуру triprojectnого .NET-рішення: ElectronicStore.API (контролери, сервіси, конфігурація), ElectronicStore.Domain (моделі, DTO, інтерфейси) та ElectronicStore.Data (репозиторії, DbContext, міграції). Описано призначення кожного шару та його складових.

Детально розглянуто реалізацію ключових компонентів із наведенням фрагментів коду. GenericRepository<T> реалізує узагальнений доступ до даних для будь-якої сутності. UnitOfWork координує репозиторії та забезпечує транзакційну цілісність. AuthService реалізує реєстрацію та вхід з генерацією JWT-токенів і BCrypt-хешуванням паролів. ProductsController надає REST API з підтримкою пошуку, пагінації та авторизації за ролями.

Розроблено керівництво користувача з описом усіх основних сценаріїв роботи: реєстрація та вхід, перегляд головної сторінки та каталогу, перегляд картки товару, управління кошиком, оформлення замовлення покроковим майстром, робота з особистим кабінетом та адміністративною панеллю. Проведено функціональне мануальне тестування 11 тест-кейсів через Swagger/OpenAPI – всі перевірки пройдено успішно, що підтверджує відповідність реалізованої системи вимогам специфікації.

ВИСНОВКИ

У межах кваліфікаційної бакалаврської роботи розроблено повнофункціональний вебзастосунок інтернет-магазину електронних пристроїв, що забезпечує зручний процес вибору, порівняння та придбання товарів користувачами. На початковому етапі проведено комплексний аналіз предметної області електронної комерції та досліджено п'ять провідних українських платформ: ROZETKA, TELEMART, COMFY, FOXTROT та CITRUS. Порівняльний аналіз дозволив виявити спільні недоліки існуючих рішень – перевантаженість інтерфейсу, складну навігацію, недостатню мобільну оптимізацію та обмежені можливості фільтрації – і сформулювати вимоги до розроблюваної системи. \

На основі аналізу наукових джерел обґрунтовано вибір технологічного стеку: серверна частина реалізована на C# 12 / .NET 8 з ASP.NET Core Web API, Entity Framework Core 8 та Microsoft SQL Server; клієнтська – на Vanilla JavaScript ES6+ з Bootstrap. Архітектура системи побудована на патернах Repository, Unit of Work та Service Layer, що забезпечує слабку зв'язаність між компонентами та зручність підтримки. Автентифікація реалізована через JWT з BCrypt-хешуванням паролів відповідно до сучасних вимог безпеки. Сформовано повну специфікацію вимог до програмного забезпечення, що охоплює вісім функцій системи та три ролі користувачів: покупець, адміністратор і гість.

У процесі проєктування побудовано UML-діаграми варіантів використання, послідовностей, активності та класів, що відображають взаємодію користувачів із системою та логіку основних бізнес-процесів. Спроєктовано структуру реляційної бази даних з 18 таблицями та налаштуванням зв'язків, індексів і CHECK-обмежень через Fluent API Entity Framework Core.

На етапі програмної реалізації створено повнофункціональний REST API з контролерами для управління товарами, категоріями, кошиком, замовленнями, відгуками та адміністративним модулем. Реалізовано клієнтську частину з адаптивним інтерфейсом, що підтримує пошук і фільтрацію каталогу, управління кошиком, покроковий майстер оформлення замовлення та особистий кабінет

користувача. Проведено функціональне мануальне тестування 10 тест-кейсів через інтерфейс Swagger/OpenAPI. Усі перевірки пройдено успішно – система коректно обробляє як позитивні, так і негативні сценарії, механізм авторизації та розмежування прав за ролями функціонує відповідно до вимог специфікації.

Таким чином, поставлену мету роботи досягнуто – створено стабільно працюючий вебзастосунок інтернет-магазину електронних пристроїв, що забезпечує повний цикл онлайн-торгівлі від перегляду каталогу до оформлення замовлення. У перспективі застосунок може бути розширено шляхом інтеграції з реальними платіжними шлюзами, впровадження системи персоналізованих рекомендацій на основі штучного інтелекту та розробки мобільного застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ROZETKA: вебсайт. URL: https://rozetka.com.ua (дата звернення: 09.05.2026).

2. TELEMART: вебсайт. URL: https://telemart.ua (дата звернення: 09.05.2026).

3. COMFY: вебсайт. URL: https://comfy.ua (дата звернення: 09.05.2026).

4. FOXTROT: вебсайт. URL: https://foxtrot.com.ua (дата звернення: 09.05.2026).

5. CITRUS: вебсайт. URL: https://citrus.ua (дата звернення: 09.05.2026).

6. Valiveti S. S. Evolution of ASP.NET to ASP.NET Core: Tools, strategies, and implementation approaches. *2025 IEEE 2nd International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*. 2025. P. 1–7.

7. Resca S. Hands-On RESTful Web Services with ASP.NET Core 3: Design production-ready, testable, and flexible RESTful APIs for web applications and microservices. Packt Publishing Ltd, 2022. 508 p.

8. Smith J. P. Entity Framework Core in Action. 2nd ed. Simon and Schuster, 2021. 665 p.

9. Gorman K., Hirt A., Noderer D. et al. Introducing Microsoft SQL Server 2022: Reliability, scalability, and security both on premises and in the cloud. Packt Publishing Ltd, 2021. 453 p.

10. Wenz C. ASP.Net Core Security. Simon and Schuster, 2022. 345 p.

11. Obiri-Tetteh J. A., Turkson R. E., Frimpong E. A. et al. Bcrypt-Based Optimized Password Hashing Against Targeted Internet of Things Attacks. *Cureus*. 2024. Vol. 16, Issue 9. P. 1–13. DOI: https://doi.org/10.7759/cureus.69342

12. Ali A. M. Optimizing Software Architecture: Using the Repository Pattern in Decoupling Data Access Logic. *International Scientific Journal of Engineering and Management*. 2022. Vol. 01, Issue 01. P. 1–8. DOI: https://doi.org/10.55041/ISJEM00104

13. Farshidi S., Jansen S., Werf J. M. van der. Capturing software architecture knowledge for pattern-driven design. *Journal of Systems and Software*. 2022. Vol. 169. P. 1–18. DOI: https://doi.org/10.1016/j.jss.2020.110744

14. Naik P., Naik G. Unlocking the Power of Bootstrap for Exploring the Basics and Beyond. *Shashwat Publication*, 2024. 372 p.

15. FP A. Y., Wahyudi I., Styawati M. et al. Web-Based Altamart Store Sales System Design. *Formosa Journal of Computer and Information Science*. 2023. Vol. 2, Issue 2. P. 175–194. DOI: https://doi.org/10.55927/fjcis.v2i2.5412

16. Tadhani J. R., Vekariya V., Sorathiya V. et al. Securing web applications against XSS and SQLi attacks using a novel deep learning approach. *Scientific Reports*. 2024. Vol. 14, Issue 1. P. 1–17. DOI: https://doi.org/10.1038/s41598-024-52911-w

17. Nawrocki M., Kołodziej J. Vulnerabilities of web applications: Good practices and new trends. *Applied Cybersecurity & Internet Governance*. 2024. Vol. 3, Issue 2. P. 122–143. DOI: https://doi.org/10.5604/01.3001.0054.4150

18. Hoffman A. Web Application Security. O'Reilly Media, 2024. 407 p.

19. Sukartini N. W., Dewi K. C., Dewi N. I. K. User Interface/User Experience Design in Green E-Commerce to Enhance Brand Image: A Systematic Review Toward a Conceptual Model. *Proceedings of the International Conference on Applied Science and Technology on Social Science 2025 (iCAST-SS 2025)*. 2025. P. 55–62. DOI: https://doi.org/10.2991/978-94-6463-938-4_8

20. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd ed. O'Reilly Media, 2021. 319.