

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО–КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет-магазин з продажу мобільних пристроїв»

Виконав студент групи ПЗ-21-5
САВЧУК Владислав Вікторович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ГРОМСЬКИЙ Олександр Олександрович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **САВЧУК Владислав Вікторович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Інтернет-магазин з продажу мобільних пристроїв»**,
затверджена наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: Процес організації та функціонування Інтернет-магазину у сфері електронної комерції мобільними пристроями, що передбачає дослідження та застосування програмних засобів, технологій та архітектурних підходів для реалізації вебдодатку на базі стеку Angular та ASP.NET Core.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	27.02.2025р.	27.02.2025р.
2	Власенко О.В.	27.02.2025р.	27.02.2025р.
3	Власенко О.В.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Бакалаврська кваліфікаційна робота присвячена створенню програмного забезпечення для інтернет–магазину, який спеціалізується на продажу мобільних пристроїв. До складу роботи входить вебзастосунок і пояснювальна записка.

У пояснювальній записці представлено 65 сторінок, 38 ілюстрацій, 18 таблиць та перелік із 30 використаних джерел.

Основна мета роботи – спроектувати та реалізувати функціональний вебзастосунок, який дозволяє користувачам зручно переглядати асортимент мобільних телефонів, фільтрувати товари за технічними параметрами, порівнювати моделі, додавати їх до кошика й оформлювати покупки. Для адміністратора передбачено інструменти для керування контентом – редагування товарів, категорій і специфікацій, а також можливість переглядати замовлення.

Під час виконання проєкту було проведено аналіз предметної галузі та аналогічних систем, після чого обґрунтовано вибір архітектурного підходу й технологічного стеку. Система побудована за принципами клієнт–серверної архітектури: фронтенд реалізовано з використанням Angular, а для створення інтерфейсу застосовано бібліотеки Bootstrap і Material 3. Серверна частина написана на ASP.NET Core із реалізацією RESTful API. Для збереження даних використовується реляційна база даних MS SQL Server. Аутентифікація й авторизація користувачів реалізована за допомогою JWT. Також інтегровано платіжну систему WayForPay для прийому онлайн–оплат.

У результаті виконаної роботи розроблено вебзастосунок, готовий до впровадження, який забезпечує основну функціональність для звичайних користувачів і адміністратора магазину.

КЛЮЧОВІ СЛОВА: ANGULAR, ASP.NET CORE, SQL SERVER, JWT, WAYFORPAY.

					ІПЗ.КР.Б – 121 – 25 – ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Савчук В.В.			Інтернет-магазин з продажу мобільних пристроїв			Лім.	Арк.	Аркушів		
Керівник		Власенко О.В.								3	65	
								Житомирська політехніка, група ІПЗ-21-5				
Н. контр.												
Затверд.		Вакалюк Т.А.										

ABSTRACT

The bachelor's thesis is devoted to the creation of software for an online store specializing in the sale of mobile devices. The thesis consists of a web application and an explanatory note.

The explanatory note contains 65 pages, 38 illustrations, 18 tables, and a list of 30 sources used.

The main goal of the work is to design and implement a functional web application that allows users to conveniently view the range of mobile phones, filter products by technical parameters, compare models, add them to the basket and make purchases. The administrator is provided with tools for content management – editing products, categories, and specifications, as well as the ability to view orders.

During the project, an analysis of the subject area and similar systems was carried out, after which the choice of architectural approach and technology stack was justified. The system is built on the principles of client–server architecture: the front end is implemented using Angular, and the Bootstrap and Material 3 libraries are used to create the interface. The server part is written in ASP.NET Core with RESTful API implementation. An MS SQL Server relational database is used for data storage. User authentication and authorisation are implemented using JWT. The WayForPay payment system is also integrated for accepting online payments.

As a result of the work performed, a web application ready for implementation has been developed, which provides the basic functionality for regular users and the store administrator.

KEYWORDS: ANGULAR, ASP.NET CORE, SQL SERVER, JWT, WAYFORPAY.

		Савчук В.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Власенко О.В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

Зміст

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ САЙТУ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Обґрунтування вибору архітектури	14
1.4. Обґрунтування вибору інструментальних засобів	16
Висновки до першого розділу.....	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИН У З ПРОДАЖУ МОБІЛЬНИХ ПРИСТРОЇВ.....	19
2.1. Визначення варіантів використання та структури системи	19
2.2. Проєктування бази даних	22
2.3. Проєктування архітектури програмного забезпечення.....	27
2.4. Реалізація функціоналу.....	33
Висновки до другого розділу	35
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	36
3.1 Порядок встановлення та налаштування системи	36
3.2 Структура інтерфейсу та порядок взаємодії з Інтернет-магазином	38
3.3. Тестування програмного продукту	59
Висновки до третього розділу.....	61
ВИСНОВКИ.....	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	63
ДОДАТКИ.....	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – інтерфейс прикладного програмування, що дозволяє обмінюватися даними між різними частинами програмного забезпечення.

SPA (Single Page Application) – односторінковий вебзастосунок, у якому контент оновлюється без повного перезавантаження сторінки.

JWT (JSON Web Token) – відкритий стандарт для безпечної передачі даних між клієнтом і сервером у вигляді зашифрованого токена.

REST (Representational State Transfer) – архітектурний стиль взаємодії в розподілених системах, що використовує HTTP-протокол.

SSR (Server-Side Rendering) – серверний рендеринг контенту, який дозволяє формувати HTML-сторінки на стороні сервера.

CRUD (Create, Read, Update, Delete) – базові операції над даними в інформаційних системах.

UI (User Interface) – інтерфейс користувача.

UX (User Experience) – досвід взаємодії користувача із застосунком або сайтом.

CDN (Content Delivery Network) – мережа доставки контенту для швидкої роздачі статичних файлів.

DBMS (Database Management System) – система керування базами даних.

ER (Entity-Relationship) – відносини між сутностями.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку цифрових технологій електронна комерція потребує ефективних, надійних та масштабованих програмних рішень. Особливо це актуально для сфери продажу мобільних пристроїв, яка характеризується високою динамікою, великим обсягом даних і потребою в зручному користувацькому інтерфейсі. Розробка вебзастосунку для онлайн-магазину вимагає системного підходу до проектування програмного забезпечення: вибору відповідної архітектури, побудови структурованої моделі даних, реалізації безпечного механізму автентифікації та авторизації, організації взаємодії з базою даних і сторонніми сервісами (зокрема платіжними системами). Актуальність теми полягає в необхідності створення програмного рішення, яке не лише задовольнить потреби кінцевих користувачів, а й відповідатиме вимогам до якості, масштабованості та підтримуваності, що є ключовими критеріями у професійній діяльності фахівця з інженерії програмного забезпечення.

Випускна кваліфікаційна робота бакалавра присвячена розробці Інтернет-магазину у мобільних пристроїв з повноцінним функціоналом як для кінцевих користувачів, так і для адміністратора системи. Реалізований вебзастосунок забезпечує можливість перегляду товарів, фільтрації за технічними параметрами, додавання до кошика, оформлення замовлення та його оплати, а також надає інтерфейс для керування товарами, категоріями, специфікаціями та замовленнями. Застосунок базується на сучасній клієнт–серверній архітектурі, що дозволяє легко масштабувати рішення та інтегрувати зовнішні сервіси.

Метою цієї роботи є створення стабільного, захищеного та функціонального вебзастосунку для онлайн–продажу мобільних пристроїв. Для досягнення поставленої мети було визначено такі основні завдання:

1. Провести аналіз предметної області та сформулювати вимоги до функціональності системи.
2. Обґрунтувати вибір технологій та архітектурного підходу, адаптованого під завдання Інтернет-магазину.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Розробити клієнтську частину вебзастосунку з використанням Angular[1] та бібліотек інтерфейсу.
4. Реалізувати серверну логіку на основі ASP.NET Core[2], побудувати REST API, підключити базу даних та впровадити авторизацію через JWT[9].
5. Інтегрувати платіжну систему WayForPay[5], а також протестувати готову систему на предмет коректності, безпеки та зручності використання.

Об’єктом дослідження є процес розробки вебзастосунку з архітектурою типу клієнт–сервер, який орієнтований на електронну торгівлю мобільними пристроями. Увага зосереджена на організації взаємодії користувача із застосунком, структурі бази даних, захисті персональних даних, а також адмініструванні товарного каталогу та замовлень.

Предметом дослідження є методи й засоби побудови вебзастосунку, зокрема реалізація фронтенду на Angular[1], розробка бекенду на ASP.NET Core[2], структура RESTful[10] API, впровадження механізмів аутентифікації й авторизації, а також підключення зовнішньої платіжної системи. Дослідження охоплює повний цикл створення сучасного програмного продукту, що відповідає вимогам зручності, надійності та розширюваності.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ САЙТУ

1.1. Постановка задачі

Стрімкий розвиток цифрової торгівлі зумовлює необхідність створення ефективних вебзастосунків, які забезпечують зручність здійснення покупок, безпечну взаємодію з персональними даними та гнучке адміністрування. Особливої актуальності набувають спеціалізовані Інтернет-магазини, орієнтовані на продаж мобільних пристроїв, оскільки попит на таку продукцію постійно зростає, а вимоги користувачів до якості сервісу стають дедалі вищими. Сучасний споживач очікує не лише на інформативну та зручну в користуванні платформу, але й на функціонал, що охоплює всі етапи взаємодії з товаром – від вибору до доставки. З огляду на це, створення ефективного інструменту для електронної комерції стає не лише питанням бізнес-інтересу, а й технологічним викликом.

У межах розробки вебзастосунку основна увага зосереджена на реалізації повного циклу взаємодії між користувачем та платформою. Покупець повинен мати змогу зручно переглядати каталог товарів, фільтрувати їх за категоріями, брендом і характеристиками, переходити до детальної інформації, оцінювати товари, читати або залишати відгуки, додавати товари до кошика, а також швидко переходити до оформлення замовлення. На сторінці замовлення важливо забезпечити інтуїтивну форму для введення контактних даних, а також підключення API служби доставки – у цьому випадку Нової пошти – для зручного вибору населеного пункту та відділення. У результаті користувач отримує не лише зручний інтерфейс, а й персоналізований досвід замовлення.

Окремим функціональним блоком є можливість порівняння кількох товарів за основними параметрами, що дозволяє споживачу прийняти обґрунтоване рішення при виборі між подібними моделями. Для підвищення зручності доступу реалізовано функцію авторизації та реєстрації, а також механізм відновлення пароля через електронну пошту. З метою захисту персональних даних усі облікові операції супроводжуються автентифікацією з

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

використанням токенів JWT[9], що дозволяє уникнути збереження паролів у сесіях. Надсилання електронних повідомлень (наприклад, при відновленні доступу) виконується через підключення до SMTP-сервера за допомогою бібліотеки MailKit[20].

З боку адміністратора реалізовано розгалужену панель управління, що включає інструменти для CRUD-операцій над товарами, категоріями, характеристиками, групами характеристик, користувачами, відгуками й замовленнями. Також реалізовано функціонал для моніторингу статусу замовлень і управління їхнім виконанням. Окрім основної функціональності, у систему вбудовано можливість створення та відновлення резервних копій бази даних, що критично важливо для збереження цілісності інформації. Для адміністраторів також доступна візуальна аналітика щодо замовлень та продажів, що дозволяє приймати більш обґрунтовані управлінські рішення.

Ключовими вимогами до системи є висока стабільність, продуктивність, безпечна авторизація з використанням JSON Web Token, масштабованість та підтримка інтеграції з зовнішніми сервісами. Для реалізації онлайн-оплати в систему інтегровано платіжну платформу WayForPay[5], яка забезпечує надійність та безпеку транзакцій. Особливістю обраного підходу є розділення застосунку на дві частини: фронтенд, реалізований на Angular[1], та бекенд – на ASP.NET Core[2]. Така структура забезпечує гнучкість розгортання, зручність оновлення, масштабованість і відповідність сучасним архітектурним підходам до створення вебзастосунків.

Таким чином, сформовано задачу створення багатofункціонального вебзастосунку для продажу мобільних пристроїв, який забезпечує як зручність користування для кінцевого споживача, так і широкі можливості управління для адміністратора. Система повинна підтримувати сучасні інструменти взаємодії, дотримуватися вимог інформаційної безпеки, бути стабільною у роботі та готовою до розширення у разі збільшення обсягів даних або кількості користувачів.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

Перед початком розробки власного вебзастосунку доцільно проаналізувати існуючі програмні рішення, які вже функціонують на ринку в аналогічному сегменті. Це дозволяє виявити сильні та слабкі сторони конкурентів, визначити типові підходи до реалізації функціональності, структуру інтерфейсу та можливості для удосконалення. У рамках аналізу було розглянуто три популярні українські Інтернет-магазини: Avic, Stylus та МТА, які спеціалізуються на продажу мобільних пристроїв, комп'ютерної техніки та електроніки.

Avic (Рис. 1.1) - це великий український Інтернет-магазин електроніки, який спеціалізується на продажу мобільних телефонів, побутової техніки, комп'ютерної периферії та аксесуарів. Сайт має розгалужену структуру категорій та потужну систему фільтрації, що дозволяє користувачам швидко знаходити потрібний товар за характеристиками, брендом чи ціновим діапазоном. Оформлення замовлення реалізовано через покроковий процес із можливістю вибору доставки та оплати.

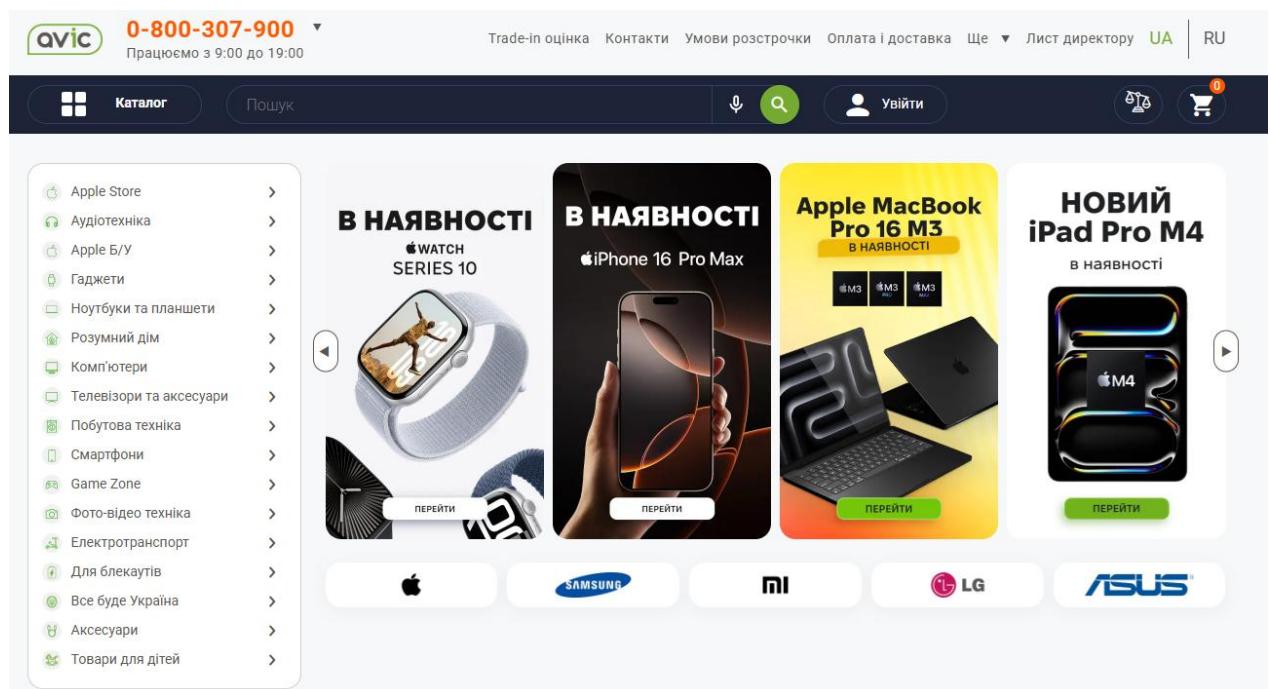


Рис. 1.1. Інтернет-магазин Avic

Серед переваг ресурсу можна відзначити багатий асортимент, наявність фільтрів за технічними параметрами, реалізовану систему відгуків, порівняння

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

товарів, підтримку особистого кабінету та можливість відновлення паролю. Крім того, реалізовано інтеграцію з Новою поштою для вибору міста та відділення доставки. Однак інтерфейс дещо перевантажений, і новому користувачеві може знадобитися час, щоб зорієнтуватися в його структурі.

Stylus (Рис. 1.2) - Інтернет-магазин, що орієнтований переважно на продаж смартфонів, ноутбуків та аксесуарів. Цей ресурс вирізняється простим і зрозумілим інтерфейсом, що фокусується на ключових діях користувача: вибір товару, перегляд його характеристик, додавання до кошика та оформлення замовлення.

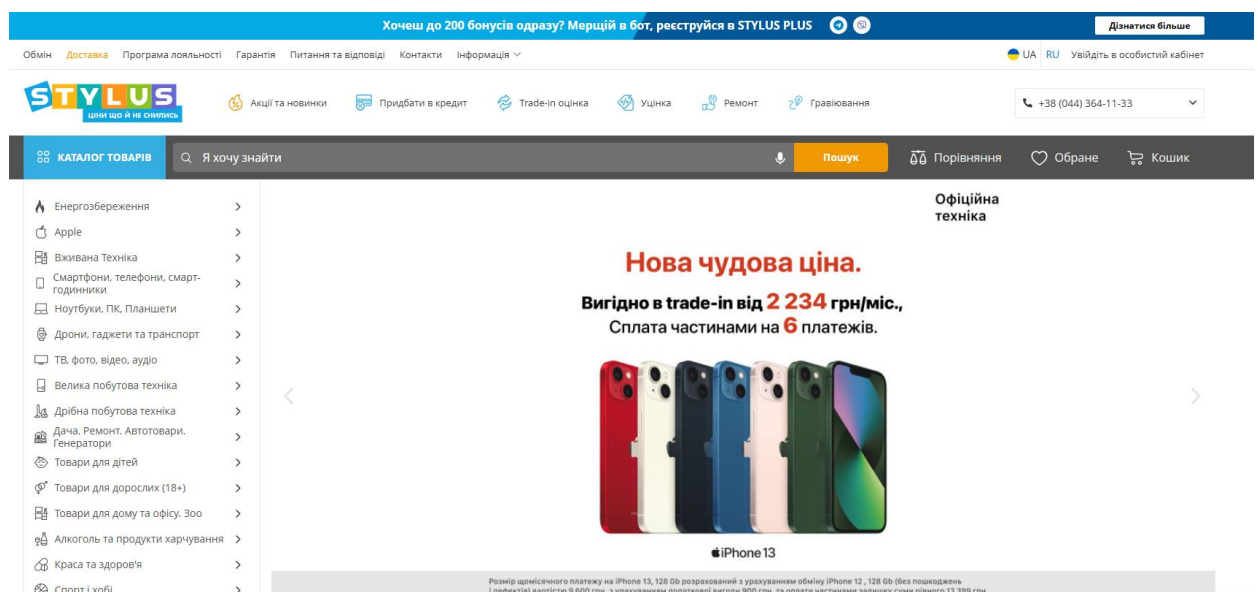


Рис. 1.2. Інтернет-магазин Stylus

До сильних сторін цього сайту можна віднести чистий, сучасний дизайн, швидке завантаження сторінок, адаптивність до мобільних пристроїв, зручну фільтрацію та підвантаження даних Нової пошти при оформленні доставки. Серед недоліків можна відзначити обмежений функціонал особистого кабінету, відсутність розширених опцій адміністрування та менш виражену інтеграцію з маркетинговими інструментами, як-от програми лояльності або спеціальні пропозиції.

МТА (Рис. 1.3) - відомий український Інтернет-магазин, який спеціалізується на продажу широкого спектру електроніки, включаючи смартфони, комп'ютери, техніку для дому та аксесуари. Сайт вирізняється

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

розгалуженою структурою, великим асортиментом і розширеним функціоналом для постійних користувачів.

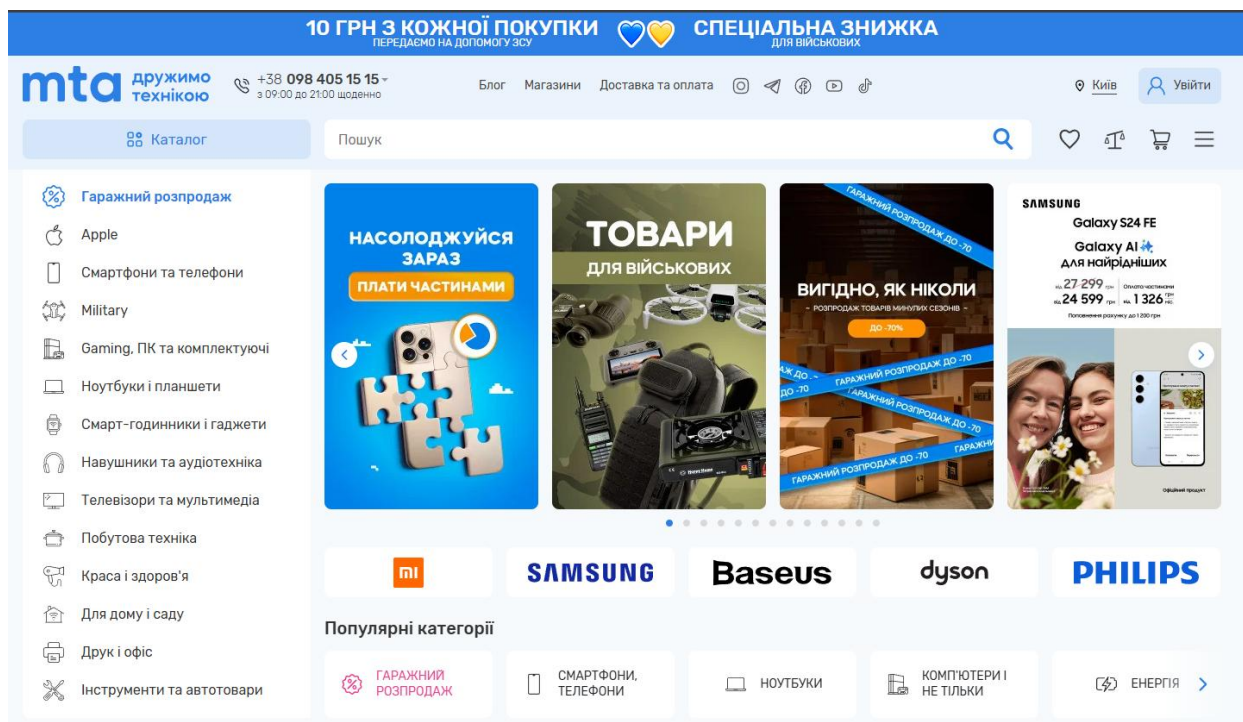


Рис. 1.3. Інтернет-магазин МТА

Сайт має інтуїтивно зрозумілий інтерфейс, реалізовану систему знижок, бонусну програму, можливість порівняння товарів і детальні картки з відгуками та характеристиками. Крім цього, реалізовано авторизацію, збереження товарів у списку бажаного, інтерактивний кошик та зручну систему оформлення замовлення з вибором доставки Новою поштою. Попри переваги, інтерфейс може здатися перевантаженим, особливо при першому ознайомленні. Також навігація в деяких розділах потребує звикання через багаторівневу структуру.

Аналіз сайтів Avic, Stylus та МТА показав, що всі вони реалізують базовий функціонал Інтернет-магазину: каталог товарів, фільтрацію, порівняння, відгуки, оформлення замовлень, інтеграцію з Новою поштою та мобільну версію. Водночас кожен із них має свої особливості: Avic орієнтований на широкий асортимент, Stylus - на простоту, а МТА - на розширену функціональність.

Отримані результати дозволяють врахувати найкращі рішення та уникнути недоліків під час створення власного вебзастосунку для продажу мобільних пристроїв.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1

Порівняльна характеристика Інтернет-магазинів аналогів

Характеристика	Avic	Stylus	MTA
Детальна фільтрація	+/-	+	+
Порівняння товарів	+	+	+
Відгуки користувачів	+/-	+	+
Особистий кабінет	+	+/-	+
Онлайн-оплата	-	-	+
Відновлення паролю	+	-	+
API Нової пошти	+	+	+
Мобільна версія	+	+	+

1.3. Обґрунтування вибору архітектури

Вибір архітектури є визначальним етапом у процесі проєктування вебзастосунку, оскільки саме вона визначає спосіб організації логіки, обміну даними між компонентами, масштабованість і можливість подальшого розвитку системи. Архітектура повинна відповідати як функціональним, так і нефункціональним вимогам до застосунку, враховувати його цільове навантаження, тип взаємодії з користувачами, потреби в безпеці та інтеграції зі сторонніми сервісами.

У межах розробки вебзастосунку для Інтернет-магазину мобільних пристроїв було обрано клієнт-серверну архітектуру (Рис. 1.4), яка передбачає чітке розділення ролей між клієнтською та серверною частинами. Такий підхід дозволяє забезпечити масштабованість, гнучкість у розробці та підтримці системи, а також полегшує інтеграцію з зовнішніми API.

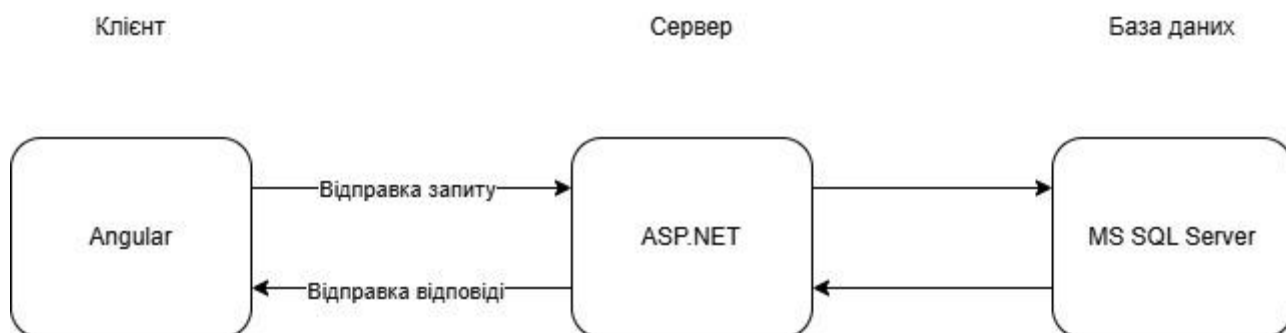


Рис. 1.4. Діаграма архітектури проєкту

Клієнтська частина реалізується у вигляді односторінкового застосунку (SPA), який виконується на стороні користувача у браузері та взаємодіє з сервером через HTTP-запити до API. У результаті відбувається динамічне оновлення даних без необхідності повного перезавантаження сторінки, що забезпечує швидку реакцію інтерфейсу на дії користувача.

Серверна частина відповідає за обробку запитів, реалізацію бізнес-логіки, роботу з базою даних і забезпечення безпеки. Вона надає набір RESTful-ендпоінтів для взаємодії з клієнтом. Така архітектура дозволяє відокремити інтерфейс від логіки обробки даних, що позитивно впливає на стабільність та масштабованість системи. API сервер взаємодіє з базою даних, сторонніми сервісами (наприклад, платіжною системою або службою доставки) та відповідає за аутентифікацію користувачів.

Додатковою архітектурною перевагою є можливість горизонтального масштабування: клієнтські та серверні частини можуть бути розгорнуті окремо, що спрощує управління навантаженням і дозволяє ефективніше використовувати ресурси. Такий підхід також відкриває можливість розміщення частин системи на різних фізичних або віртуальних серверах, а також підключення CDN для прискорення завантаження клієнтської частини.

Особливу увагу при виборі архітектури було приділено безпеці та підтримці ролей користувачів. Серверна частина реалізує розмежування прав доступу залежно від ролі (адміністратор або звичайний користувач), а передача конфіденційної інформації, зокрема токенів авторизації, відбувається через захищені канали. Вся бізнес-логіка виконується на сервері, що мінімізує ризики маніпуляцій на клієнтському рівні.

Архітектурне рішення також враховує потенціал подальшого розвитку проєкту. Завдяки модульності та гнучкості застосованого підходу, у майбутньому можна безболісно додавати нові функціональні модулі, змінювати структуру інтерфейсу або адаптувати систему до нових технологічних вимог без суттєвих змін в існуючому коді.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

1.4. Обґрунтування вибору інструментальних засобів

Успішна реалізація вебзастосунку залежить не лише від архітектурних рішень, а й від правильного вибору інструментальних засобів. У межах цього проєкту було обрано сучасні, перевірені часом технології, які забезпечують надійність, безпеку, масштабованість та зручність у розробці й підтримці системи.

Для клієнтської частини обрано фреймворк Angular[1], що базується на мові TypeScript. Він дозволяє створювати SPA-застосунки з реактивною структурою, підтримує двостороннє зв'язування даних, маршрутизацію та модульність, що значно спрощує супровід та масштабування застосунку. У якості стилізації використано Bootstrap[7] та Angular Material[8], які забезпечують сучасний зовнішній вигляд інтерфейсу без необхідності ручного опрацювання адаптивності.

Серверна частина реалізована з використанням ASP.NET Core[2], який є кросплатформеним фреймворком для створення REST API. Цей стек було обрано завдяки високій продуктивності, широкій підтримці інструментів для безпеки, логування, DI-контейнерів і масштабування. Він надає зручну інтеграцію з Entity Framework Core[3] – ORM-системою, яка автоматизує роботу з реляційною базою даних MS SQL Server[4], дозволяючи розробникам працювати із сутностями напряду через LINQ-запити.

Для автентифікації та захисту API використовується механізм JWT[9] (JSON Web Token), що забезпечує безстейтову авторизацію. Паролі зберігаються у вигляді хешу з використанням сервісу хешування. Надсилання листів, наприклад, для скидання паролю, реалізовано за допомогою бібліотеки MailKit[20], яка підтримує сучасні протоколи SMTP/IMAP з автентифікацією.

Платіжна система WayForPay[5] забезпечує можливість інтеграції захищеної онлайн-оплати. Для доставки використовується API Нової пошти, що дозволяє динамічно отримувати список міст і відділень для зручного оформлення замовлення. Усі запити до сторонніх сервісів реалізовано через окремі сервіси у клієнтській і серверній частинах.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Для підтвердження доцільності вибору основних інструментальних засобів в таблиці 1.2 наведено порівняльну характеристику з обраними інструментами.

Таблиця 1.2

Порівняльна характеристика інструментальних засобів

Категорія	Обраний інструмент	Альтернатива 1	Альтернатива 2	Причина вибору
Фреймворк (клієнт)	Angular	React	Vue.js	Компонентна структура, SPA, підтримка Google.
Мова програмування (клієнт)	TypeScript	JavaScript	Dart	Статична типізація, інтеграція з Angular
Фреймворк (сервер)	ASP.NET Core	Spring Boot (Java)	Express (Node.js)	Висока продуктивність, безпечна маршрутизація
Мова програмування (сервер)	C#	Java	Node.js	Сумісність з .NET екосистемою
ORM	Entity Framework Core	Dapper	Hibernate	Автоматизація міграцій, інтеграція з LINQ
СУБД	MS SQL Server	PostgreSQL	MySQL	Глибока інтеграція з .NET, висока надійність
Аутентифікація	JWT	OAuth2	Cookie Auth	Безсерверна, масштабована автентифікація
Надсилення email	MailKit	SmtpClient (.NET)	Mailgun API	Гнучкість, підтримка шифрування
Платіжна система	WayForPay	LiqPay	Stripe	Простота інтеграції, підтримка callback
API для доставки	Нова Пошта API	Justin API	Meest Express API	Поширеність, точність даних

Висновки до першого розділу.

У першому розділі здійснено загальний аналіз напрямків використання інформаційних технологій для створення вебзастосунків у сфері електронної комерції. Проведено постановку задачі, що охоплює як користувацький функціонал, так і потреби адміністративного управління контентом, оплатою та доставкою.

Аналіз аналогічних вебресурсів дав змогу виявити ефективні рішення, реалізовані в існуючих Інтернет-магазинах, а також визначити недоліки, яких варто уникнути. Було обґрунтовано вибір клієнт-серверної архітектури як найбільш придатної для реалізації масштабованого середовища взаємодії.

У розділі також наведено причини вибору технологій і бібліотек для клієнтської та серверної частин, з урахуванням питань безпеки та інтеграції зі сторонніми сервісами, такими як WayForPay[5] та API Нової пошти. Проведене дослідження дало змогу сформулювати технічну основу майбутнього застосунку та створити цілісне бачення подальших етапів розробки.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИН У З ПРОДАЖУ МОБІЛЬНИХ ПРИСТРОЇВ

2.1. Визначення варіантів використання та структури системи

Проектування вебзастосунку доцільно починати з визначення ролей користувачів та варіантів їхньої взаємодії з системою. Такий підхід дозволяє сформувати чітку логіку доступу до функціоналу, врахувати особливості кожного типу користувача та створити гнучку структуру для масштабування системи в майбутньому.

У межах реалізації Інтернет-магазин у було визначено три основні ролі: гість, користувач та адміністратор. Кожна з них має свій набір дозволених дій (Рис. 2.1).

Гість (неавторизований користувач) має можливість переглядати головну сторінку, здійснювати навігацію по категоріях, переглядати картки товарів, ознайомлюватися з характеристиками, цінами, читати відгуки, а також додавати товари до кошика. Також для нього доступна функція пошуку й порівняння товарів. Оформити замовлення або залишити відгук гість не може – для цього необхідна авторизація.

Користувач (авторизований) має доступ до розширеного функціоналу. Після входу до облікового запису користувач отримує можливість оформлювати замовлення, обирати місто та відділення доставки через API Нової пошти, здійснювати оплату за допомогою WayForPay[5], залишати відгуки, переглядати історію власних замовлень, змінювати особисті дані та відновлювати пароль у разі потреби.

Адміністратор має доступ до внутрішньої панелі керування, де може створювати, редагувати та видаляти товари, змінювати категорії та характеристики, переглядати список усіх замовлень, керувати їхнім статусом і мати доступ до аналітичної інформації. Крім того, адміністратор має можливість модерації відгуків та контролю даних користувачів.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

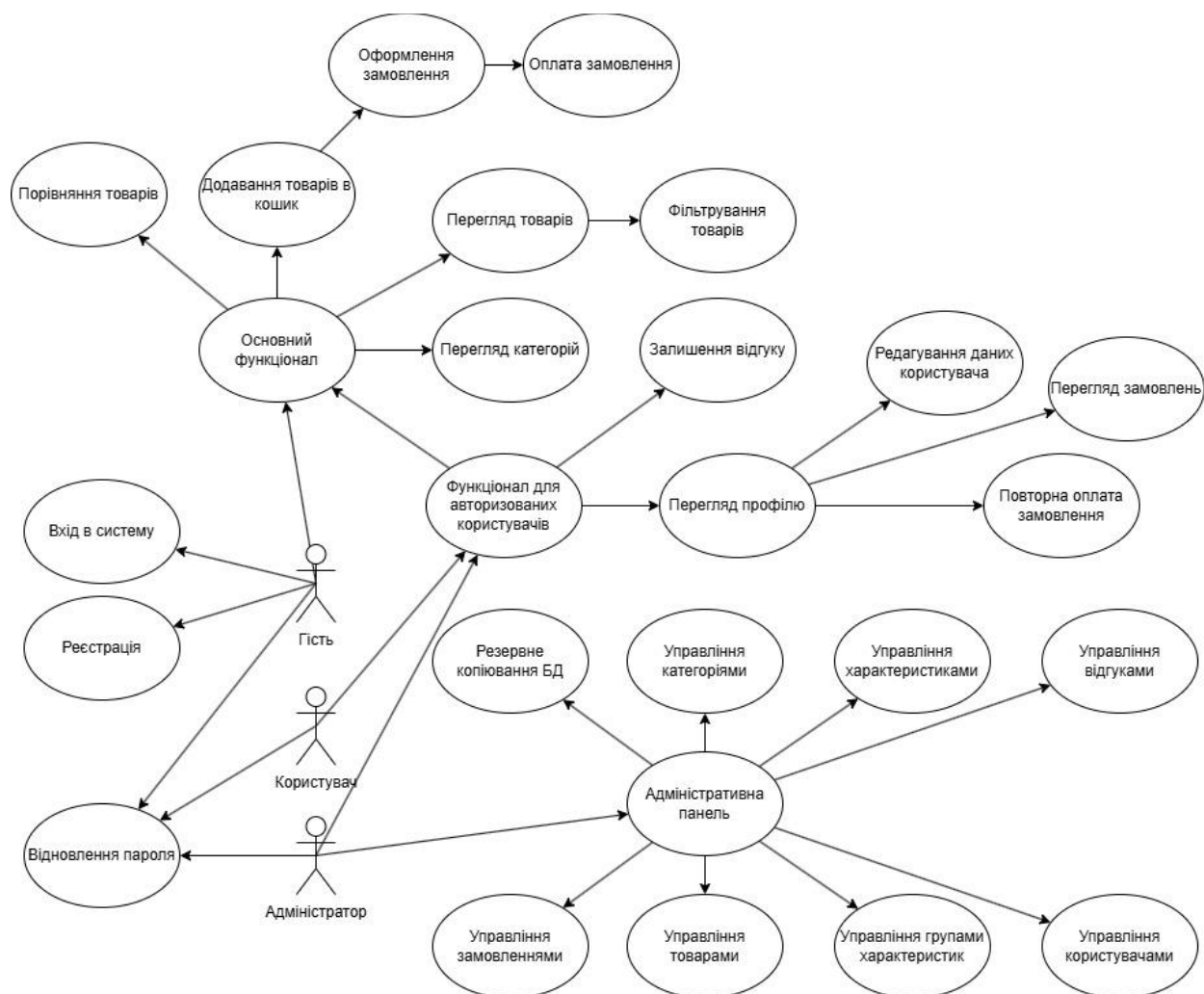


Рис. 2.1. Діаграма варіантів використання

На основі ролей сформовано загальну структуру вебзастосунку, яка охоплює клієнтську частину, серверну логіку, базу даних та взаємодію з зовнішніми сервісами. Така структура дозволяє чітко розмежовувати відповідальність між окремими компонентами, спрощує реалізацію захищеного доступу та забезпечує ефективне управління системою в цілому.

Опис варіантів використання для кожної ролі наведено в таблицях 2.1 – 2.3.

Таблиця 2.1

Варіанти використання для ролі «Гість»

№	Варіант використання	Опис
1	Перегляд категорій	Отримання доступу до списку категорій товарів
2	Перегляд товарів	Ознайомлення з інформацією про товари у вибраній категорії
3	Фільтрування товарів	Застосування фільтрів за характеристиками, брендом, ціною тощо
4	Додавання до кошика	Тимчасове збереження товарів для можливого замовлення

Продовження таблиці 2.1

5	Оформлення замовлення	Підтвердження кошика і введення даних для замовлення
6	Оплата замовлення	Проведення онлайн-оплати через WayForPay
7	Порівняння товарів	Вибір кількох товарів і перегляд їхніх характеристик у порівнянні
8	Вхід у систему	Введення логіна й пароля для переходу до особистого кабінету
9	Реєстрація	Створення нового облікового запису
10	Відновлення паролю	Відновлення доступу до акаунта через електронну пошту

Таблиця 2.2

Варіанти використання для ролі «Користувач»

№	Варіант використання	Опис
1	Перегляд профілю	Доступ до особистої інформації та історії замовлень
2	Редагування профілю	Зміна контактної інформації або паролю
3	Перегляд замовлень	Ознайомлення з переліком оформлених раніше замовлень
4	Повторна оплата замовлення	Повторне здійснення оплати у випадку, якщо попередня була перервана
5	Залишення відгуків	Написання коментаря до товару після покупки

Таблиця 2.3

Варіанти використання для ролі «Адміністратор»

№	Варіант використання	Опис
1	Доступ до адміністративної панелі	Вхід до захищеного середовища адміністрування
2	Управління товарами	Додавання, редагування, видалення товарів
3	Управління категоріями	Створення та оновлення категорій товарів
4	Управління характеристиками	Зміна можливих технічних характеристик товарів
5	Управління групами характеристик	Зв'язування характеристик у групи для зручності фільтрації
6	Управління відгуками	Модерація користувацьких відгуків
7	Управління користувачами	Перегляд зареєстрованих користувачів, блокування, редагування
8	Управління замовленнями	Перевірка та оновлення статусів оформлених замовлень
9	Резервне копіювання БД	Створення резервної копії бази даних для збереження інформації

Таким чином, на основі визначених ролей користувачів та їхніх можливостей сформовано загальну логіку роботи системи. Це забезпечує чітке розмежування функціоналу та створює основу для подальшого проектування бази даних, серверної логіки та інтерфейсу вебзастосунку.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Проєктування бази даних

База даних є фундаментальною частиною будь-якого інформаційного застосунку, що відповідає за збереження, обробку та структуровану взаємодію даних. Для ефективної реалізації функціональності інтернет-магазину мобільних пристроїв необхідно спроектувати логічну модель бази даних, яка враховує як основні об'єкти предметної області, так і зв'язки між ними. Якість проєктування бази даних безпосередньо впливає на швидкодію застосунку, його масштабованість, надійність і здатність ефективно обробляти великі обсяги даних у реальному часі.

Під час проєктування було обрано реляційну модель даних, що реалізується за допомогою СУБД Microsoft SQL Server[4]. Ця система керування базами даних підтримує складні запити, транзакції, індексацію, а також надає високий рівень захищеності даних. В основі структури бази лежать таблиці, які відповідають за зберігання інформації про користувачів, ролі, товари, категорії, характеристики, замовлення, відгуки, сесії, доставку, оплату та інші пов'язані сутності. Таблиці створені з урахуванням принципів нормалізації до третьої нормальної форми, що дозволило уникнути дублювання інформації та забезпечити її логічну цілісність. Наприклад, окремо виділено таблиці для типів характеристик, груп характеристик, способів оплати та статусів замовлень.

Зв'язки між сутностями реалізовано з дотриманням принципів зовнішніх ключів (foreign key), що забезпечують коректність залежностей і унеможливають появу «висячих» записів. Наприклад, кожен товар належить до певної категорії, може мати декілька характеристик, бути частиною кількох замовлень, містити відгуки користувачів тощо. Замовлення, у свою чергу, пов'язані з користувачами, які їх створюють, із товарами, які замовляються, а також із записами про статус оплати, доставку та поточний стан. Таким чином, вся бізнес-логіка системи знаходить відображення в чітко структурованих таблицях із правильно реалізованими зв'язками.

Для забезпечення масштабованості та зручності підтримки, структура бази даних організована у вигляді моделей, які використовуються у коді застосунку

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

через ORM Entity Framework Core[3]. Це дозволяє автоматично створювати та оновлювати міграції бази даних, адаптувати схему до змін у логіці застосунку, а також підтримувати зв'язок між моделями та контролерами. Завдяки цьому реалізовано автоматичну генерацію таблиць, зв'язків і обмежень без необхідності ручного написання SQL-коду, що підвищує швидкість розробки та знижує ймовірність помилок.

У результаті база даних проєкту забезпечує повну підтримку всіх основних процесів інтернет-магазину, включаючи зберігання інформації про товари, обробку замовлень, керування користувачами та їхніми діями, а також збереження історії взаємодії користувача з системою. Така структурована і логічно узгоджена модель є надійною основою для подальшого масштабування функціоналу та впровадження нових бізнес-можливостей.

Було створено ER-діаграму (Рис. 2.2), яка ілюструє загальну логічну структуру бази даних та взаємозв'язки між основними таблицями.

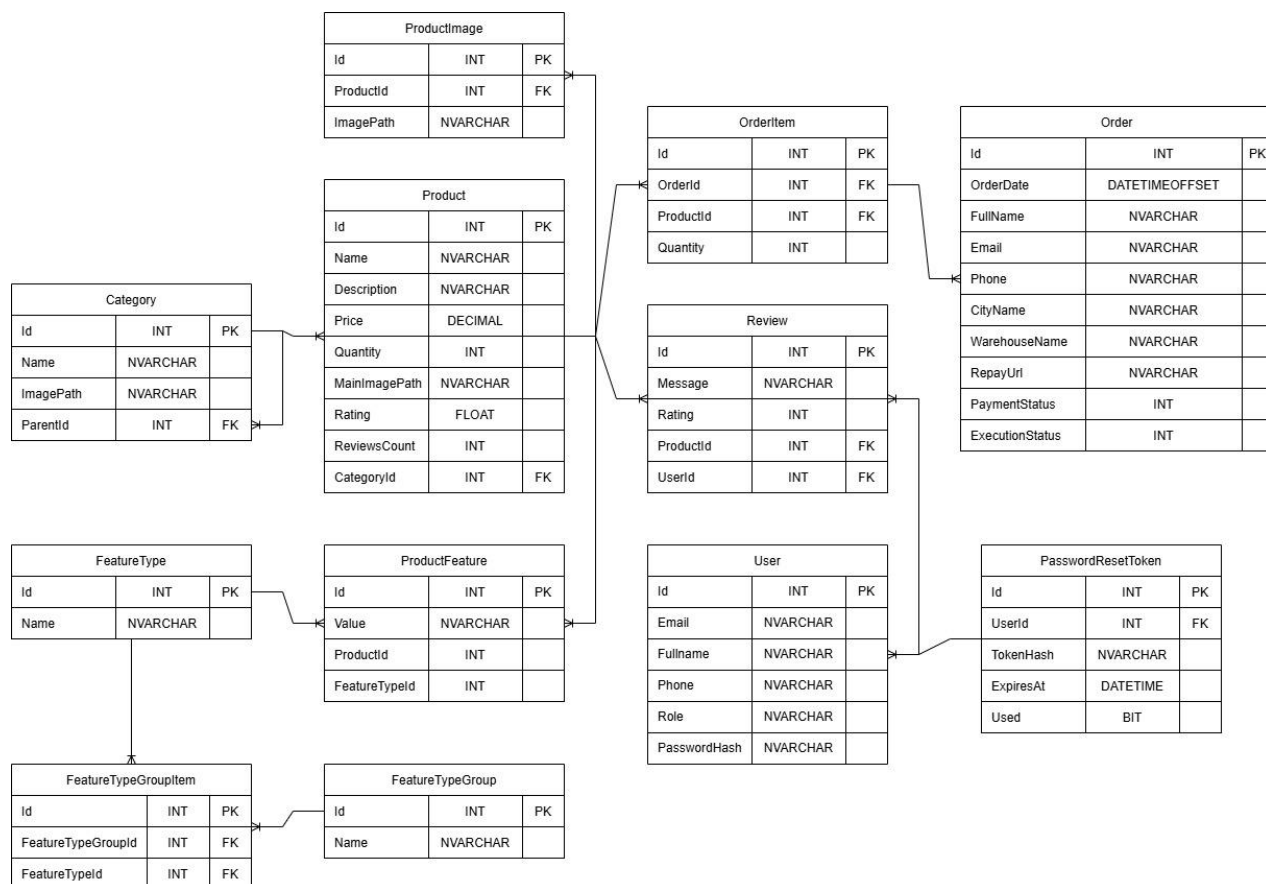


Рис. 2.2. ER-діаграма структури бази даних Інтернет-магазину.

Після побудови логічної моделі відбувається проєктування фізичної структури бази даних, де визначаються типи даних для кожного поля, обмеження цілісності, індекси для пришвидшення пошуку, а також ураховуються вимоги до безпеки та захисту персональної інформації.

У таблицях 2.4 – 2.15 детально описано кожну сутність бази даних, що реалізована у проєкті.

Таблиця 2.4

Таблиця: User

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Email	string	NVARCHAR	Електронна пошта користувача
Fullname	string	NVARCHAR	Повне ім'я користувача
Phone	string	NVARCHAR	Телефон користувача (необов'язкове)
Role	string	NVARCHAR	Роль користувача (admin/user)
PasswordHash	string	NVARCHAR	Хеш пароля користувача

Таблиця 2.5

Таблиця: Category

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Name	string	NVARCHAR	Назва категорії
ImagePath	string	NVARCHAR	Шлях до зображення категорії
ParentId	int	INT	Зовнішній ключ на батьківську категорію (nullable)

Таблиця 2.6

Таблиця: Product

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Name	string	NVARCHAR	Назва товару
Description	string	NVARCHAR	Опис товару (необов'язковий)
Price	decimal	DECIMAL	Ціна товару
Quantity	int	INT	Кількість на складі
MainImagePath	string	NVARCHAR	Основне зображення товару
Rating	double	FLOAT	Середній рейтинг товару
ReviewsCount	int	INT	Кількість відгуків
CategoryId	int	INT	Зовнішній ключ на категорію

Таблиця 2.7

Таблиця: ProductImage

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
ProductId	int	INT	Зовнішній ключ на товар
ImagePath	string	NVARCHAR	Шлях до зображення

Таблиця 2.8

Таблиця: ProductFeature

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Value	string	NVARCHAR	Значення характеристики
ProductId	int	INT	Зовнішній ключ на товар
FeatureTypeId	int	INT	Зовнішній ключ на тип характеристики

Таблиця 2.9

Таблиця: FeatureType

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Name	string	NVARCHAR	Назва характеристики

Таблиця 2.10

Таблиця: FeatureTypeGroup

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Name	string	NVARCHAR	Назва групи характеристик

Таблиця 2.11

Таблиця: FeatureTypeGroupItem

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
FeatureTypeGroupId	int	INT	Зовнішній ключ на групу характеристик
FeatureTypeId	int	INT	Зовнішній ключ на тип характеристики

Таблиця 2.12

Таблиця: Order

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ

Продовження таблиці 2.12

OrderDate	DateTimeOffset	DATETIMEOFFSET	Дата та час замовлення
FullName	string	NVARCHAR	Ім'я та прізвище покупця
Email	string	NVARCHAR	Електронна пошта покупця
Phone	string	NVARCHAR	Телефон покупця
CityName	string	NVARCHAR	Назва міста доставки (Нова пошта)
WarehouseName	string	NVARCHAR	Відділення Нової пошти
RepayUrl	string	NVARCHAR	Посилання для повторної оплати
PaymentStatus	enum	INT	Статус оплати
ExecutionStatus	enum	INT	Статус виконання замовлення

Таблиця 2.13

Таблиця: OrderItem

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
OrderId	int	INT	Зовнішній ключ на замовлення
ProductId	int	INT	Зовнішній ключ на товар
Quantity	int	INT	Кількість товару у замовленні

Таблиця 2.14

Таблиця: Review

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
Message	string	NVARCHAR	Текст відгуку
Rating	int	INT	Оцінка (1–5)
ProductId	int	INT	Зовнішній ключ на товар
UserId	int	INT	Зовнішній ключ на користувача

Таблиця 2.15

Таблиця: PasswordResetToken

Поле	Тип C#	Тип SQL	Опис
Id	int	INT	Первинний ключ
UserId	int	INT	Зовнішній ключ на користувача
TokenHash	string	NVARCHAR	Хеш токена
ExpiresAt	DateTime	DATETIME	Дата завершення дії токена
Used	bool	BIT	Чи був токен використаний

Після побудови логічної моделі відбувається проєктування фізичної структури бази даних, де визначаються типи даних для кожного поля, обмеження

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

цілісності, індекси для пришвидшення пошуку, а також ураховуються вимоги до безпеки та захисту персональної інформації.

2.3. Проєктування архітектури програмного забезпечення

Функціонування вебзастосунку передбачає не лише структуроване зберігання даних та взаємодію з інтерфейсом, але й реалізацію логіки роботи окремих процесів, які забезпечують повноцінний цикл взаємодії користувача з системою. У процесі розробки було спроектовано та реалізовано низку ключових алгоритмів, які охоплюють такі аспекти, як авторизація, відновлення доступу, оформлення замовлень та інтеграція з платіжною платформою.

Під час входу до системи користувач вводить електронну пошту та пароль, після чого ці дані надсилаються на сервер. У разі успішної автентифікації генерується JSON Web Token, який зберігається на клієнтській стороні, зазвичай у локальному сховищі браузера. Подальші запити до захищених ресурсів супроводжуються цим токеном, що дозволяє серверу перевіряти ідентичність користувача без повторного введення облікових даних. На діаграмі послідовності (Рис. 2.3), ілюструється взаємодія між клієнтом, сервером і системою збереження токенів у процесі авторизації.

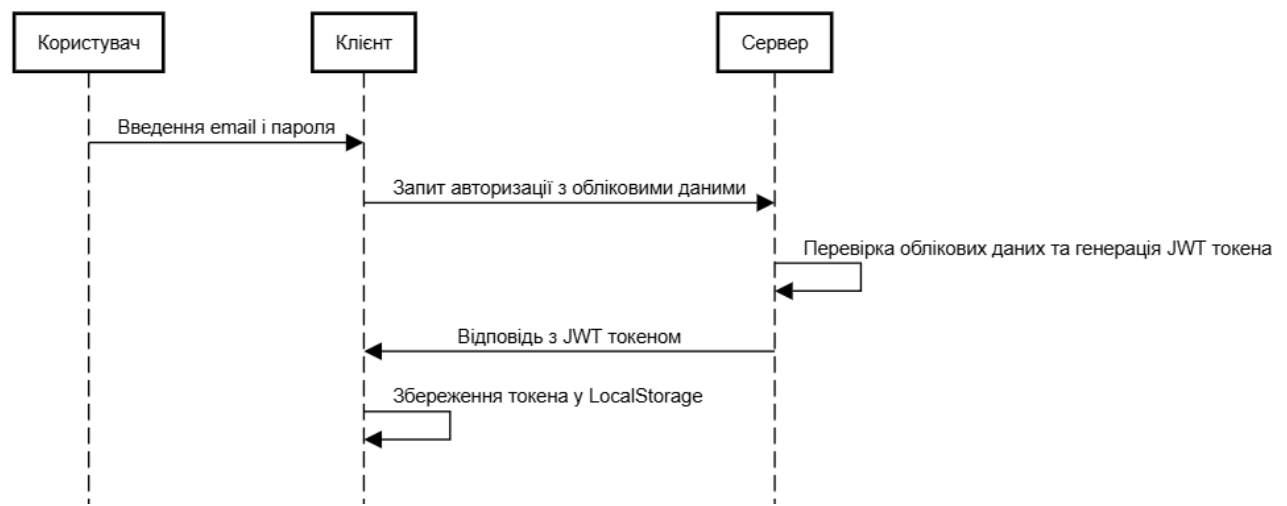


Рис. 2.3. Діаграма послідовності процесу авторизації користувача

Сценарій оформлення замовлення починається з додавання товарів до кошика. Користувач переглядає список категорій, обирає товар, додає його до кошика та переходить до форми замовлення. Під час цього він вводить контактні дані, обирає населений пункт і відділення доставки через інтеграцію з API Нової

пошти. Після підтвердження замовлення клієнт перенаправляється на платіжну сторінку. У разі успішної оплати система отримує відповідь від WayForPay[5] та оновлює статус замовлення. Уся логіка цього процесу послідовно відображена на діаграмі активності (Рис. 2.4).



Рис. 2.4. Діаграма активності оформлення замовлення користувачем

У випадку, якщо користувач забув пароль, система дозволяє ініціювати процедуру його відновлення. Користувач вводить електронну пошту, після чого сервер генерує унікальний токен для скидання пароля та надсилає його на вказану адресу. Посилання з токеном дозволяє перейти до форми введення нового пароля. Після підтвердження новий хеш зберігається в базі, а токен

вважається використаним. Детальний перебіг цього процесу наведено у вигляді діаграми активності (Рис. 2.5).

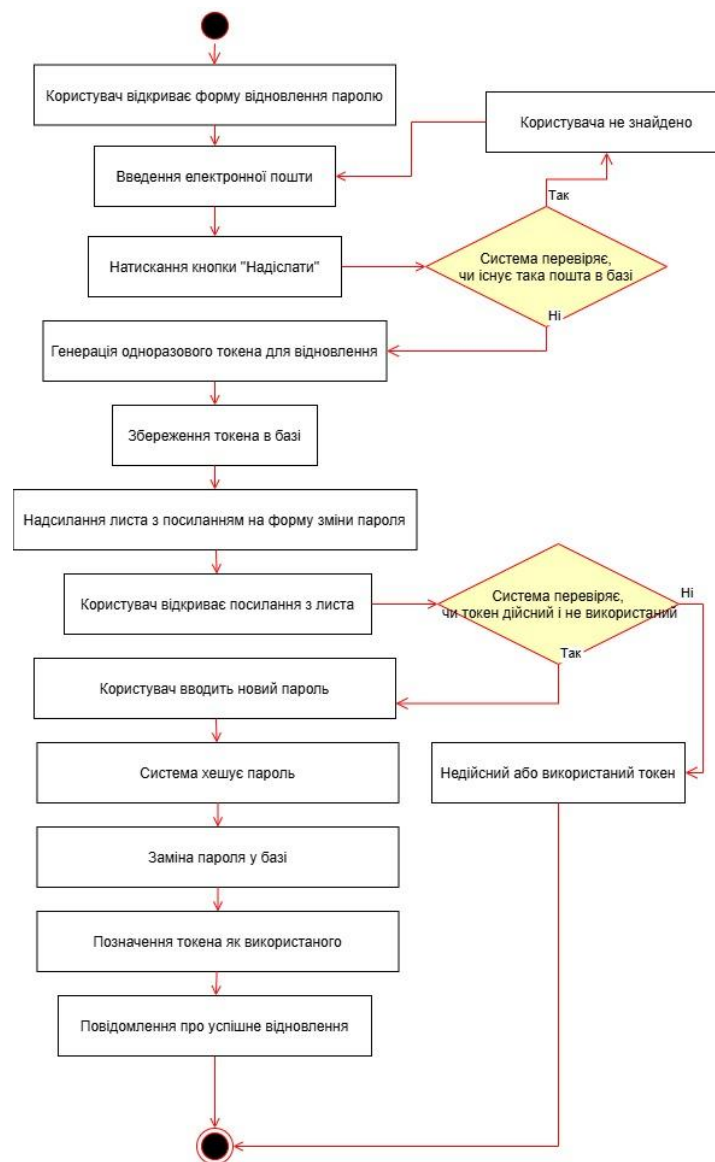


Рис. 2.5. Діаграма активності відновлення паролю користувача

Інтеграція з платіжною системою WayForPay[5] дозволяє здійснювати онлайн-оплати безпосередньо в межах застосунку. Після підтвердження замовлення формується запит із даними транзакції, який перенаправляє користувача на сторінку оплати. У разі успішної операції платіжна система викликає callback-метод на сервері з передачею статусу. Сервер обробляє відповідь, зберігає підтвердження оплати та змінює статус замовлення на відповідний. Цей механізм зображено у вигляді діаграми послідовності (Рис. 2.6), що демонструє взаємодію між користувачем, застосунком і стороннім платіжним сервісом.

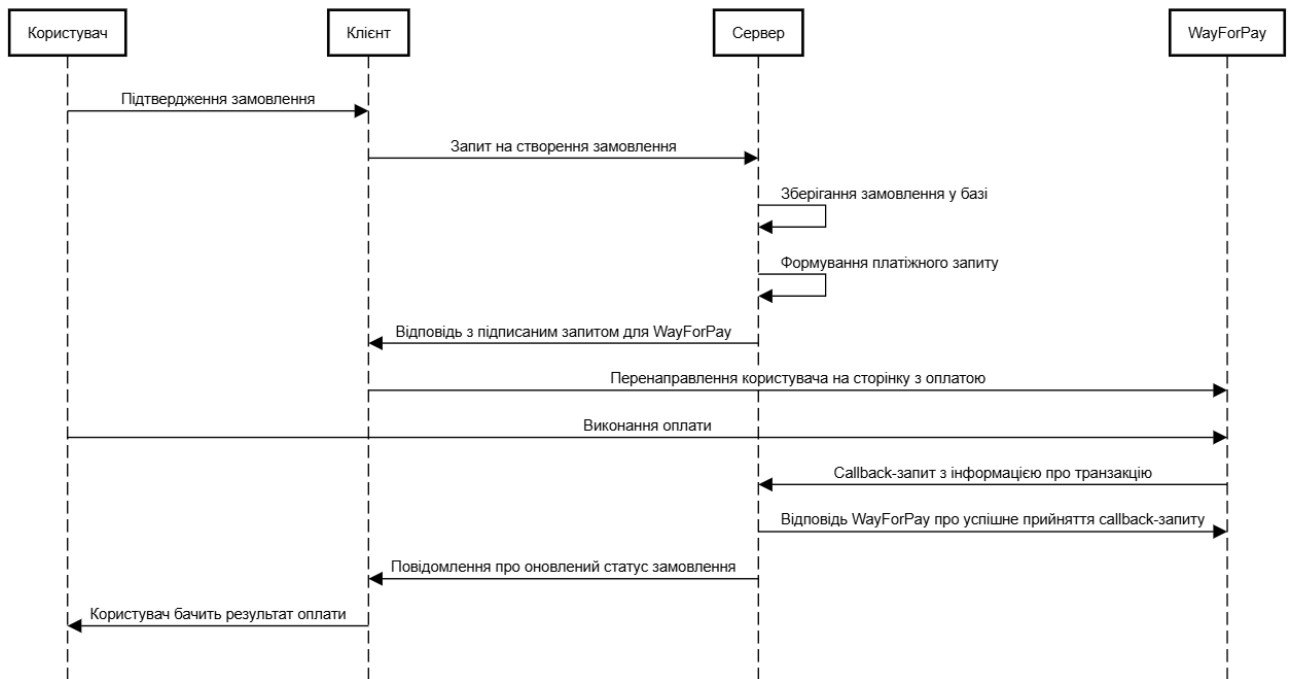


Рис. 2.6. Діаграма послідовності процесу оплати замовлення

Крім діаграм, що відображають алгоритмічну логіку, у процесі проєктування було створено низку діаграм класів, які демонструють структуру програмних компонентів, реалізованих у середовищі ASP.NET Core[2]. На цих діаграмах представлено взаємозв'язки між моделями, контролерами та допоміжними структурами, які формують основу серверної логіки вебзастосунку.

Представлено структуру моделей бази даних (Рис. 2.7), де кожна сутність (наприклад, Product, Category, Order) відображає відповідну таблицю в базі даних. Моделі пов'язані між собою за допомогою навігаційних властивостей, що дозволяє використовувати механізми ORM Entity Framework Core[3] для зручного доступу до пов'язаних даних. Це забезпечує інтуїтивно зрозумілий спосіб навігації між сутностями, наприклад, отримання списку товарів певної категорії або доступ до користувача, який створив конкретне замовлення.

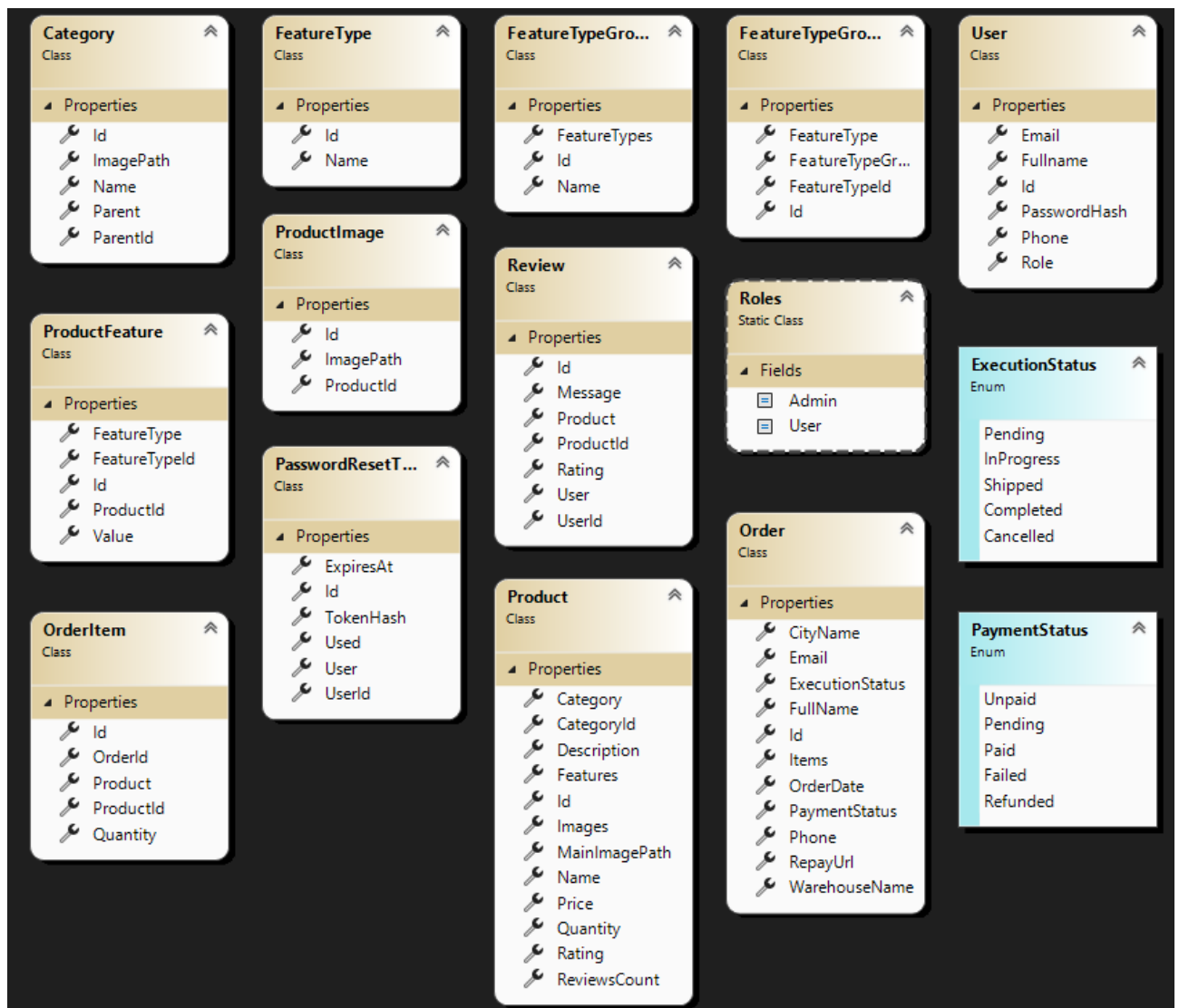


Рис. 2.7. Діаграма класів моделей бази даних

Для обробки HTTP-запитів було реалізовано набір контролерів, кожен з яких відповідає за окремий функціональний модуль системи (наприклад, ProductController, OrderController, AuthController). Нижче зображено їх структуру та основні методи (Рис. 2.8), включаючи Get, Post, Put і Delete, які реалізують CRUD-операції.

Кожен контролер використовує залежності, що передаються через конструктор, відповідно до принципу інверсії керування (IoC). Валідація вхідних даних здійснюється за допомогою DTO-моделей і анотацій даних. Для забезпечення безпеки до окремих методів застосовуються атрибути авторизації з перевіркою ролей.

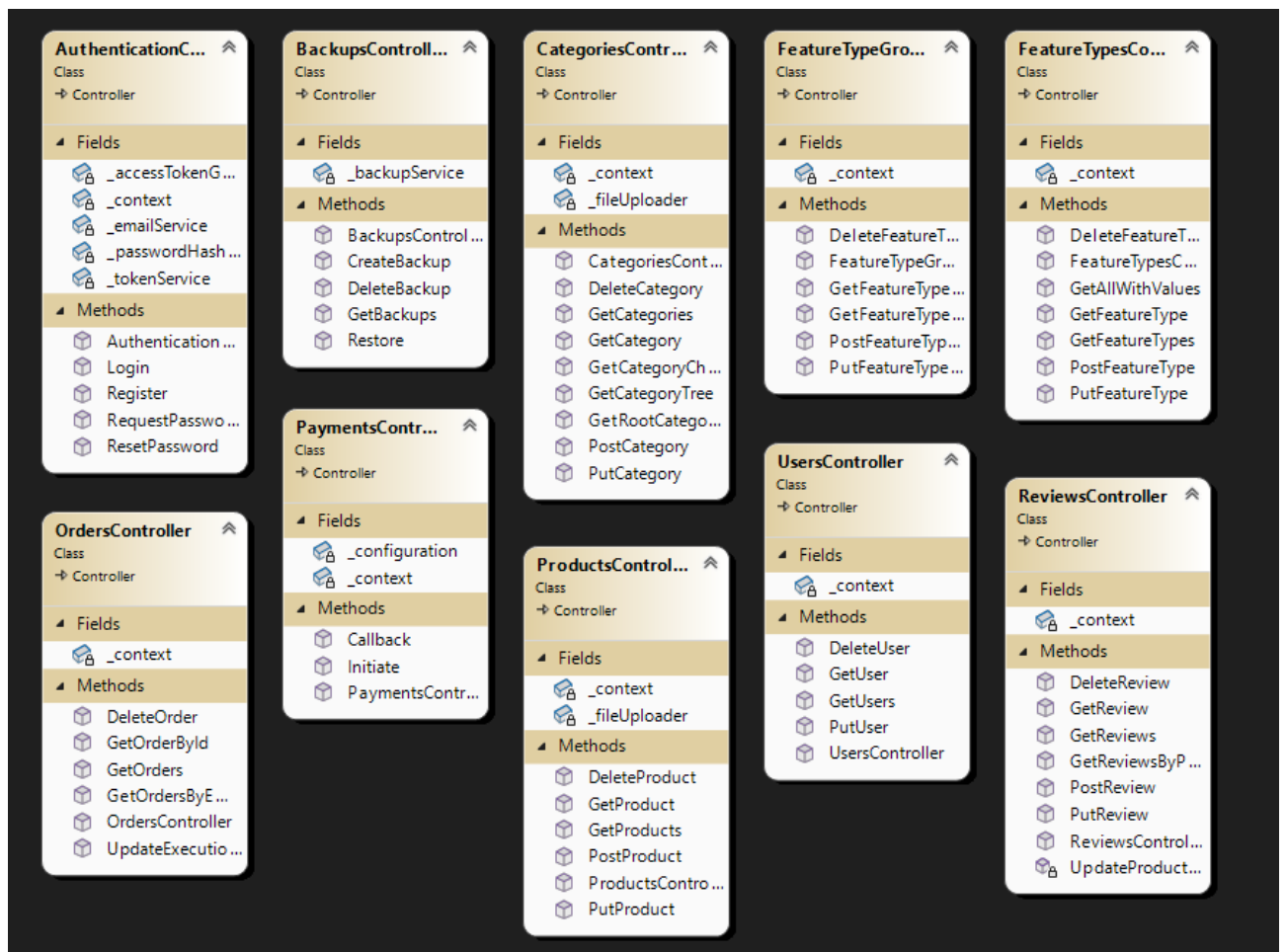


Рис. 2.8. Діаграма класів контролерів ASP.NET Core

Особливу роль у системі відіграють моделі запитів (Рис. 2.9), які використовуються для приймання вхідних даних від клієнта. Вони дозволяють чітко розділяти зовнішню контрактну структуру даних від внутрішніх моделей збереження. Наведено приклади таких моделей, як `CreateProductRequest`, `RegisterRequest`, `ResetPasswordRequest` тощо, з відповідними властивостями.

Ці моделі забезпечують централізовану валідацію введених даних і дозволяють уникнути надмірної складності в контролерах. Вони використовуються у зв'язці з атрибутами типу `[Required]`, `[EmailAddress]` тощо. Такий підхід підвищує безпеку системи та спрощує її підтримку.



Рис. 2.9. Діаграма класів моделей запитів

Використання таких діаграм дозволяє краще структурувати код, забезпечити інкапсуляцію логіки в межах окремих шарів застосунку та полегшити супровід і розвиток системи.

2.4. Реалізація функціоналу

На основі результатів проєктування було реалізовано вебзастосунок, що охоплює повний цикл взаємодії користувача з Інтернет-магазином – від ознайомлення з товарами до оформлення замовлення й адміністрування даних. Реалізацію виконано відповідно до принципів клієнт-серверної архітектури з розподілом логіки між Angular-застосунком на стороні клієнта та сервером, розробленим на базі ASP.NET Core[2]. Основна логіка клієнтської частини реалізована у вигляді Angular-компонентів, зокрема `cart.component.ts`, `compare.component.ts`, а також сервісів `auth.service.ts` і `order.service.ts`, приклади яких наведено у Додатку А.

Користувач має змогу переглядати категорії, ознайомлюватися з товарами, переглядати детальні характеристики, відгуки, рейтинг і зображення. Реалізовано повнотекстовий пошук і динамічну фільтрацію за характеристиками, ціною, категорією та іншими параметрами. Усі ці функції побудовано на основі реактивних форм і асинхронних запитів до API без перезавантаження сторінки. У Додатку А наведено код компонентів, які реалізують цю взаємодію.

Після вибору товару його можна додати до кошика, перейти до оформлення замовлення, вказати персональні дані та обрати відділення доставки через інтеграцію з API Нової пошти. Взаємодія з API реалізована у вигляді окремого сервісу `nova-poshta.service.ts`, приклад якого подано в Додатку Г. Сервер обробляє запит, створює запис замовлення та формує посилання для переходу на сторінку оплати, що здійснюється через `WayForPay[5]`. Обробка замовлення і платежів на сервері реалізована за допомогою контролерів `PaymentsController.cs` і `AuthenticationController.cs`, що наведені в Додатку Б.

Механізм авторизації реалізовано на основі JWT-токенів, що дозволяє забезпечити безпечний доступ до захищених маршрутів. У разі втрати доступу до акаунта користувач може скористатися функцією відновлення паролю: після введення email надсилається лист із посиланням на форму скидання. Сервер генерує токен, зберігає його у базі та дозволяє оновити пароль. Логіка цього процесу реалізована у сервісах `PasswordHasherService.cs` і `PasswordResetTokenService.cs`, а також у `EmailService.cs`, фрагменти яких наведено у Додатках Б і Г відповідно.

Користувач має доступ до особистого кабінету, де може переглядати історію замовлень, відслідковувати статуси доставки, редагувати контактні дані. Реалізація цього функціоналу інтегрована у компоненти Angular-застосунку, які працюють із сервісами, поданими в Додатку А. Окремим компонентом реалізовано функцію порівняння товарів без залучення сервера – усі дані обробляються локально.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Для адміністратора реалізовано панель керування із CRUD-операціями для основних сутностей: товарів, категорій, характеристик, груп характеристик, відгуків і платежів. Усі дії адміністрування виконуються через Angular-інтерфейс, що спілкується з REST API. Серверна частина реалізована через контролери. Моделі бази даних, з якими вони працюють, наведено в Додатку В.

Таким чином, вебзастосунок охоплює всі ключові аспекти функціонування сучасного Інтернет-магазину, поєднуючи зручність клієнтської взаємодії з надійністю серверної логіки. Детальні реалізації основних компонентів і сервісів представлено у Додатках А–Г.

Висновки до другого розділу

У цьому розділі було виконано комплексне проєктування та реалізацію основних компонентів Інтернет-магазину у мобільних пристроїв. Сформовано структуру системи з урахуванням ролей користувачів, визначено варіанти використання, спроектовано базу даних, реалізовано ER-модель та створено таблиці з відповідними полями й зв'язками. Детально описано основні бізнес-процеси системи у вигляді діаграм послідовності й активності, що дозволяє чітко візуалізувати логіку взаємодії між компонентами.

Здійснено реалізацію функціональних можливостей, зокрема: динамічного пошуку товарів, фільтрації, порівняння, управління контентом, оформлення замовлень, оплати через інтеграцію з WayForPay[5], а також реалізовано механізми зворотного зв'язку (відгуки, рейтинги) та відновлення доступу. Завдяки чіткому проєктуванню та ефективній реалізації, система є масштабованою та готовою до подальшого впровадження і розвитку.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

Для коректної роботи інтернет-магазину мобільних пристроїв необхідно виконати низку дій, спрямованих на розгортання серверної та клієнтської частин, налаштування бази даних, інтеграцій із зовнішніми сервісами та забезпечення захищеної взаємодії між компонентами системи. Застосунок побудований за архітектурною моделлю «клієнт-сервер», тому процес встановлення включає окремі кроки для кожної складової. Взаємодія між елементами системи схематично представлена на діаграмі компонентів (Рис. 3.1).

Серверна частина реалізована на основі ASP.NET Core[2]. Для її запуску необхідно мати встановлене середовище .NET[12] SDK (версія 7.0 або новіша), а також Microsoft SQL Server[4] як систему управління базами даних. Після клонування проєкту необхідно налаштувати конфігураційний файл appsettings.json, де вказується рядок підключення до бази даних, секретний ключ JWT[9], SMTP-параметри для надсилання пошти, а також параметри інтеграції з платіжною системою WayForPay[5].

Далі встановлюються всі необхідні залежності проєкту за допомогою NuGet Package Manager або через команду dotnet restore. Серед обов'язкових бібліотек – Entity Framework Core[3] (для доступу до БД), MailKit[20] (для роботи з поштою), бібліотеки для JWT-автентифікації та серіалізації JSON. Створення таблиць бази даних здійснюється автоматично через команду Update-Database (виконується в Package Manager Console) або dotnet ef database update у CLI. Усі таблиці створюються на основі моделей, описаних у серверному коді.

На сервері реалізовано також API-інтеграцію з WayForPay[5] та Новою Поштою, що дозволяє обробляти платежі та підвантажувати відділення доставки у реальному часі. Усі зовнішні виклики реалізовані через сервіси, які діють як посередники між контролерами й сторонніми API. Ці зв'язки також представлені на діаграмі компонентів (Рис. 3.1).

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнтська частина побудована з використанням Angular[1]. Для її запуску потрібно встановити Node.js (версія 18 або новіша) та менеджер пакунків npm. Після встановлення залежностей через npm install, у файлі environment.ts необхідно вказати актуальну адресу API сервера. Збірка застосунку відбувається командою ng build, а запуск у режимі розробки – через ng serve.

Angular-застосунок містить компоненти, що відповідають за відображення інтерфейсу користувача, та сервіси, які взаємодіють із сервером через HTTP-запити у форматі JSON. Комунікація між Angular[1] та сервером здійснюється за допомогою Angular-сервісів, які автоматично додають JWT-токен до заголовків запиту. Користувачі автентифікуються, після чого отриманий токен дозволяє доступ до захищених ресурсів API. Цей механізм відображено у вигляді взаємодії компонентів (Рис. 3.1).

Інтеграція з API Нової Пошти реалізована на клієнтському рівні – Angular-застосунок напряму надсилає HTTP-запити до відповідного API, отримуючи список міст і відділень у форматі JSON. Це дозволяє користувачеві динамічно обирати адресу доставки під час оформлення замовлення.

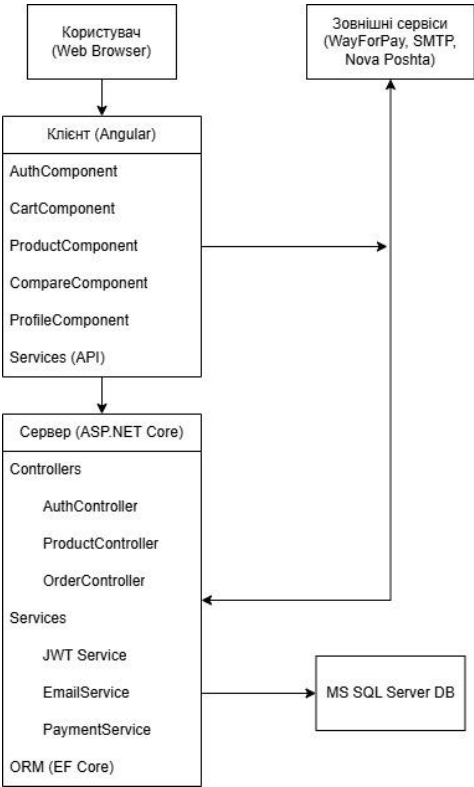


Рис. 3.1. Діаграма компонентів вебзастосунку

Також представлено діаграму розгортання (Рис. 3.2), яка демонструє взаємозв'язки між фізичними вузлами системи: клієнтський браузер, сервер API, SQL Server[4] та зовнішні сервіси (пошта та оплата). Такий підхід дозволяє масштабувати застосунок та легко адаптувати його до хмарної інфраструктури.

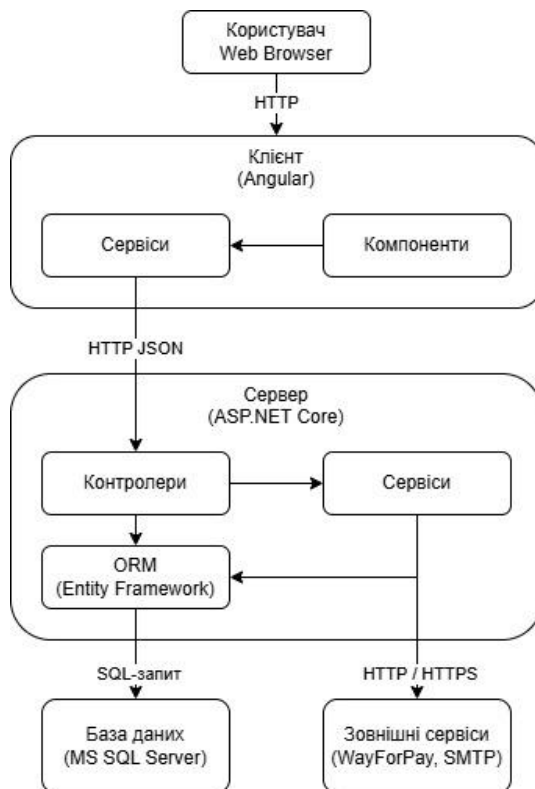


Рис. 3.2. Діаграма розгортання системи

Таким чином, інсталяція системи є послідовною та включає налаштування конфігураційних файлів, встановлення залежностей, створення структури бази даних та запуск фронтенду. Завдяки поділу на компоненти та сервісну архітектуру, усі частини проєкту можуть оновлюватися або масштабуватися незалежно одна від одної, що відповідає сучасним вимогам до вебзастосунків.

3.2 Структура інтерфейсу та порядок взаємодії з Інтернет-магазином

На головній сторінці Інтернет-магазину (Рис. 3.3), зосереджено ключові елементи взаємодії користувача із системою. У верхній частині розташована горизонтальна навігаційна панель, яка містить назву магазину, посилання на основні розділи сайту, такі як «Головна», «Товари» та «Порівняння», а також елементи керування обліковим записом, включно з кнопками входу, реєстрації та переходу до кошика.

Нижче на сторінці представлено блок категорій товарів. Кожна категорія оформлена у вигляді плитки із зображенням і назвою, що дозволяє здійснювати швидкий перехід до відповідного розділу магазину. Такий підхід забезпечує зручну навігацію, покращує візуальне сприйняття й орієнтує користувача у структурі каталогу.

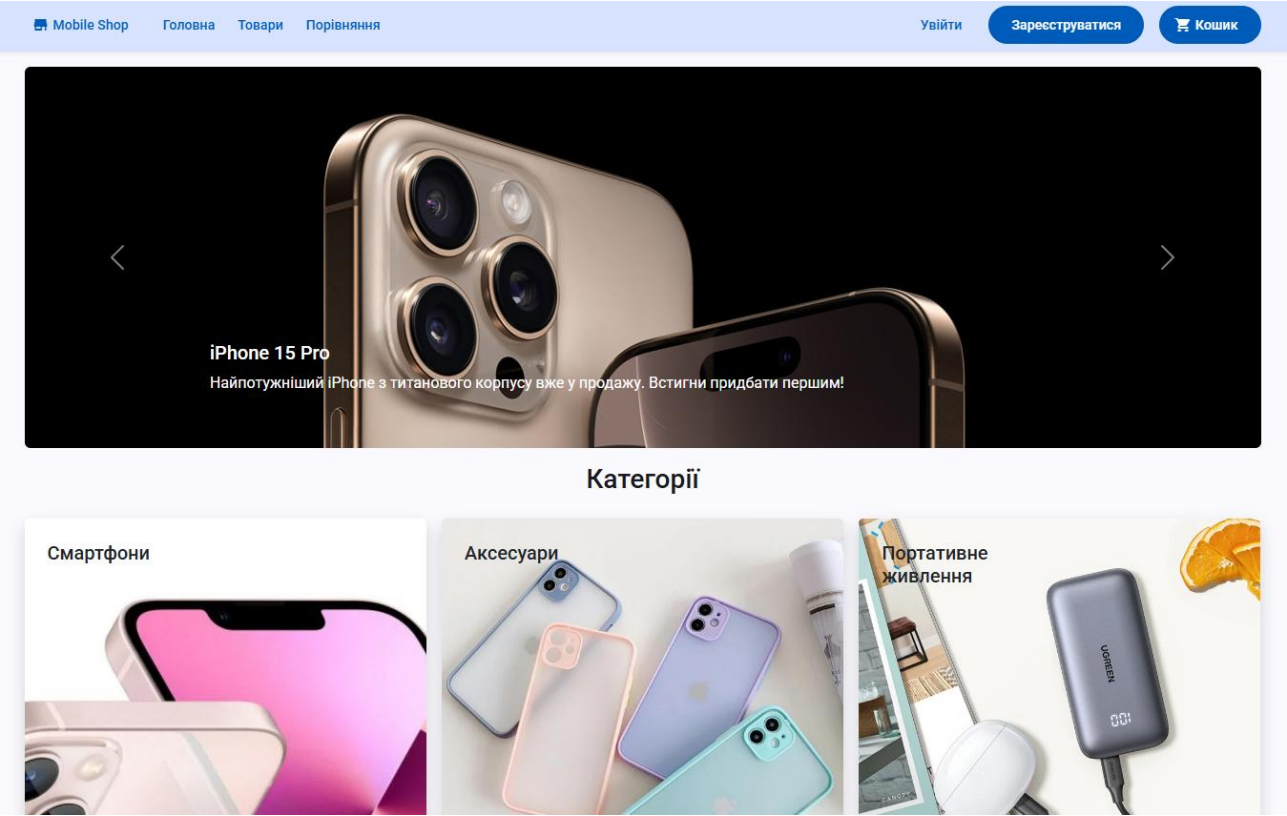


Рис. 3.3. Головна сторінка

Нижче представлено інтерфейс сторінки реєстрації (Рис. 3.4), яка забезпечує створення нового облікового запису в системі. Сторінка містить форму з обов’язковими полями для введення персональних даних, зокрема повного імені, електронної адреси, пароля та підтвердження пароля. Кожне поле супроводжується відповідною піктограмою для полегшення сприйняття.

Поля пароля та підтвердження пароля мають функцію візуалізації введених символів, яка активується натисканням на іконку у вигляді ока. Це дозволяє уникнути помилок при введенні складних паролів. Після заповнення всіх полів користувач може натиснути кнопку «Підтвердити», що ініціює процес реєстрації з передачею даних на сервер. У разі успішного завершення операції

формується новий обліковий запис і відкривається сторінка з авторизацією користувача.

Рис. 3.4. Сторінка реєстрації користувача

Нижче можна переглянути сторінку авторизації (Рис. 3.5), яка дозволяє користувачеві увійти до системи для доступу до персоналізованого функціоналу. Інтерфейс містить два основні поля: електронну адресу та пароль. Кожне поле супроводжується відповідною піктограмою, що вказує на тип введених даних. Для зручності передбачено можливість візуалізації введеного пароля шляхом натискання на іконку «Око».

Під формою розташоване посилання «Скидання пароля», яке ініціює процес відновлення доступу до облікового запису у разі втрати пароля. Після заповнення форми натискання кнопки «Підтвердити» запускає перевірку облікових даних і, у разі успіху, надає користувачеві доступ до особистого кабінету та відповідних можливостей.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Рис. 3.5. Сторінка авторизації користувача

Перейдемо до сторінки з формою для ініціації скидання пароля (Рис. 3.6). Вона є частиною механізму відновлення доступу до облікового запису у випадку, якщо користувач забув пароль або не може увійти в систему. Інтерфейс сторінки максимально спрощений і містить єдине поле для введення електронної адреси, що прив'язана до облікового запису.

Після введення коректної адреси та натискання кнопки «Підтвердити» відбувається надсилання запиту на сервер, який генерує унікальний токен і відправляє на вказану адресу листа з посиланням для встановлення нового пароля. Якщо адреса не зареєстрована, користувач отримає відповідне повідомлення про помилку.

Механізм скидання пароля реалізований із дотриманням вимог безпеки: токен має обмежений строк дії та може бути використаний лише один раз. Це дозволяє захистити облікові записи від несанкціонованого доступу.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Рис. 3.6. Сторінка з запитом на скидання пароля

Представлено інтерфейс сторінки для скидання пароля (Рис. 3.7), яка відкривається за унікальним посиланням, надісланим на електронну пошту користувача після ініціації запиту на відновлення доступу. Сторінка містить два поля введення: новий пароль та його підтвердження. Кожне поле супроводжується іконкою, яка дозволяє відобразити або приховати введені символи.

Після заповнення обох полів і натискання кнопки «Підтвердити» введені дані передаються на сервер, де відбувається перевірка токена та відповідність обох паролів. У разі успішної перевірки зберігається новий хеш пароля в базі даних, а токен скидання позначається як використаний, що унеможливорює повторне звернення за цим же посиланням.

У разі виникнення помилок – наприклад, якщо токен недійсний або строки паролів не збігаються – користувач отримує відповідне повідомлення з поясненням. Це дозволяє уникнути неправильного заповнення та забезпечує зручний досвід відновлення доступу.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 3.7. Сторінка з скидання пароля

Зображено сторінку з підкатегоріями товарів (Рис. 3.8), яка відкривається після вибору однієї з основних категорій. У наведеному прикладі відображено підкатегорії категорії «Смартфони», що згруповані за брендами – iPhone, Samsung та Xiaomi. Кожна підкатегорія оформлена у вигляді окремої плитки з ілюстрацією продукції відповідного бренду, назвою та кнопкою «Переглянути товари».

При натисканні на кнопку користувач перенаправляється до сторінки з повним списком товарів, які належать до вибраної підкатегорії. Вся взаємодія відбувається динамічно, без перезавантаження сторінки, завдяки використанню Angular[1].

Такий підхід дозволяє зручно організувати навігацію по товарах великого каталогу, а також візуально виокремлює пропозиції кожного виробника, що позитивно впливає на зручність користувача під час вибору техніки.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

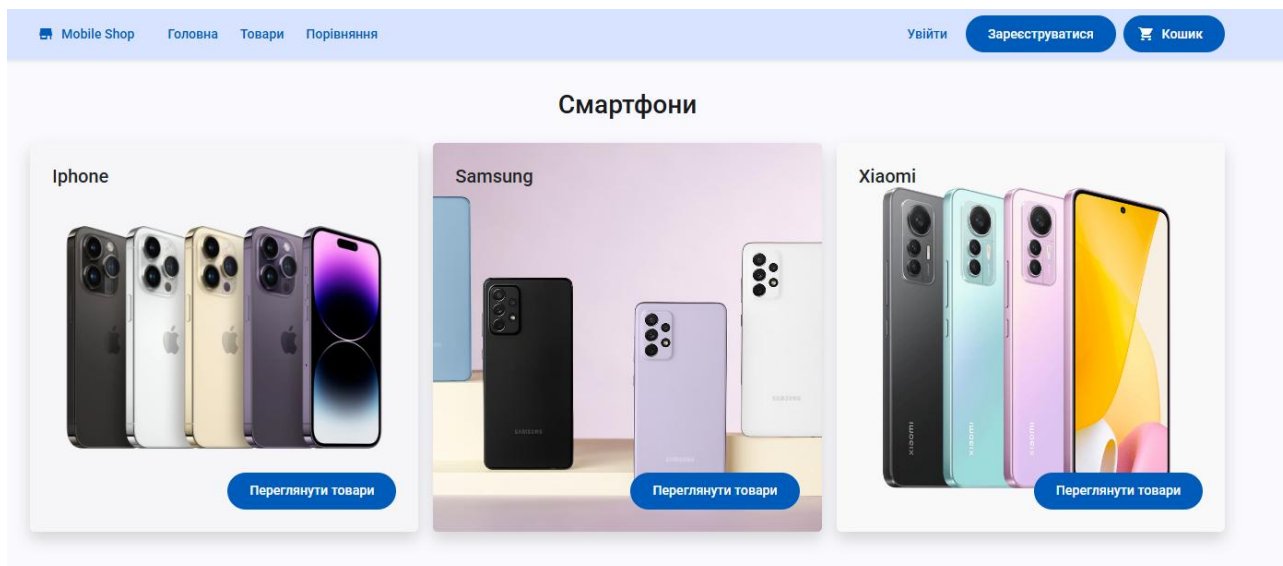


Рис. 3.8. Сторінка з підкатегоріями

Представлено сторінку з переліком товарів (Рис. 3.9), яка реалізує основну функціональність Інтернет-магазин у – перегляд, фільтрацію та вибір продукції. Ліворуч розміщена панель фільтрів, яка дозволяє налаштувати відображення товарів за низкою параметрів, таких як ціна, категорія, підкатегорія та характеристики. Доступні фільтри оновлюються залежно від вибраної категорії, що забезпечує динамічну адаптацію інтерфейсу під вміст каталогу.

Крім фільтрації, у верхній частині сторінки реалізовано можливість сортування товарів за різними критеріями, зокрема за ціною. По центру виводиться перелік товарів у вигляді карток, кожна з яких містить зображення, назву, рейтинг, ціну, а також кнопку для додавання товару до кошика. Над картками також доступне поле для текстового пошуку товарів за назвою.

Усі дії – фільтрація, пошук, сортування та додавання до кошика – виконуються без перезавантаження сторінки завдяки використанню Angular[1] та реактивного підходу до оновлення інтерфейсу. Такий підхід забезпечує високу швидкість взаємодії та зручність користування.

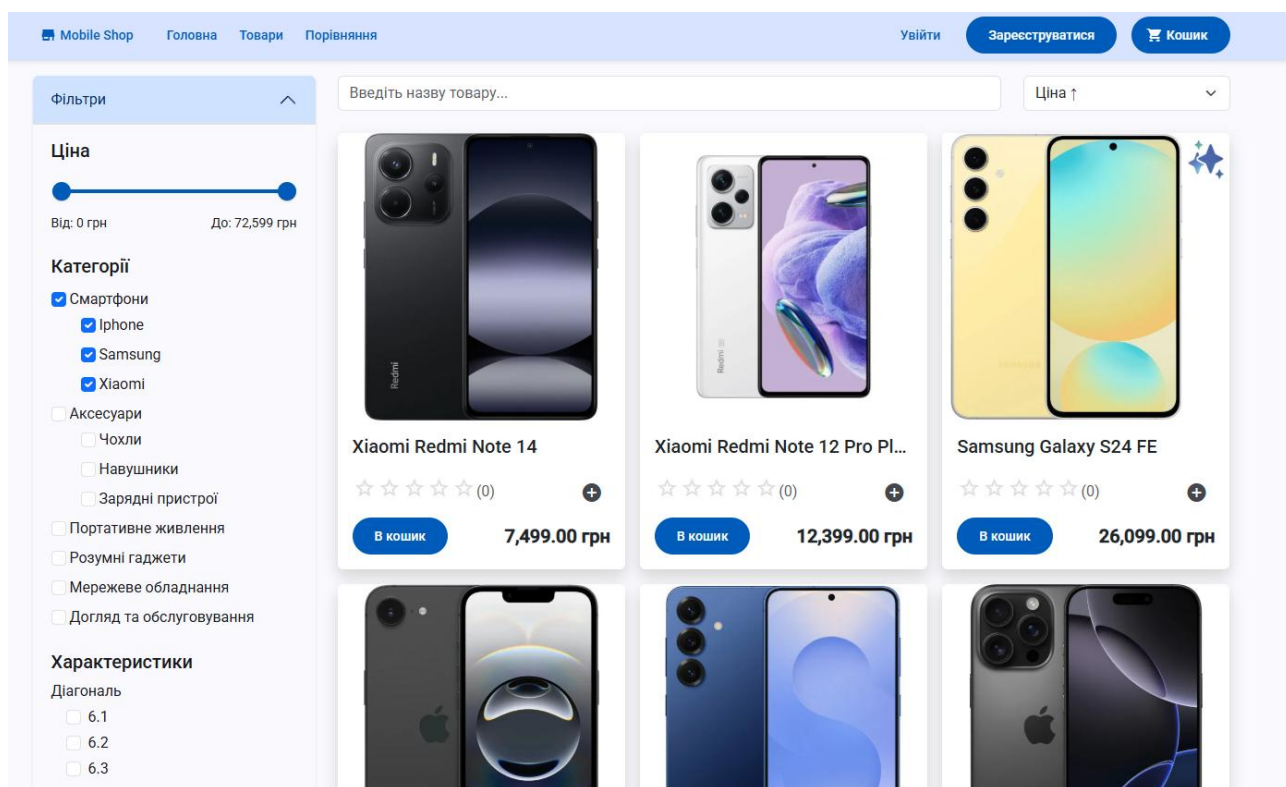


Рис. 3.9. Сторінка з товарами

Зображено сторінку з деталями про товар (Рис. 3.10), яка відкривається після вибору конкретної позиції з каталогу. Інтерфейс сторінки складається з двох основних блоків: зображення товару ліворуч і текстового опису праворуч. У верхній частині правого блоку відображається назва товару, його рейтинг, кількість доступних одиниць, ціна та кнопка «В кошик» для додавання вибраної моделі до кошика.

Нижче подано короткий опис пристрою, який дозволяє ознайомитися з основними перевагами моделі. Розгорнута інформація про товар доступна у вигляді вкладок – «Опис», «Характеристики» та «Відгуки». Кожна вкладка відповідає за виведення відповідної інформації без перезавантаження сторінки. Така структура дозволяє швидко перемикатися між деталями про товар, технічними параметрами та думками інших покупців.

Крім того, реалізовано галерею зображень з можливістю перегортання слайдів, що дозволяє краще ознайомитися з виглядом продукту. Сторінка є динамічною і повністю побудована на Angular[1], що забезпечує миттєве реагування на дії користувача.



Рис. 3.10. Сторінка з деталями про товар

Зображено вкладку «Характеристики» на сторінці товару (Рис. 3.11). Цей розділ містить структурований перелік технічних параметрів обраної моделі, що дозволяє користувачеві швидко ознайомитися з основними особливостями пристрою. Дані представлені у вигляді таблиці, де в першому стовпці вказано назву характеристики, а в другому – її значення.

У прикладі наведено такі характеристики: категорія, діагональ екрана, обсяг вбудованої та оперативної пам'яті, роздільна здатність основної й фронтальної камери, тип роз'ємів, ємність акумулятора та колір корпусу. Такий формат подання інформації є зручним для сприйняття, особливо під час порівняння кількох моделей.

Інтерфейс реалізовано таким чином, щоб забезпечити однакову структуру для всіх товарів, незалежно від їхньої категорії. Система підтримує динамічне відображення характеристик, які автоматично формуються на основі пов'язаних значень у базі даних.

		Опис	Характеристики	Відгуки
Категорія		Iphone		
Діагональ		6.3		
Вбудована пам'ять, ГБ		1024		
Оперативна пам'ять, ГБ		8		
Основна камера, Мп		48		
Фронтальна камера, Мп		12		
Роз'єми		USB Type-C		
Ємність акумулятора, мАг		3355		
Колір		Чорний		

Рис. 3.11. Характеристики товару

Зображено вкладку «Відгуки» на сторінці товару (Рис. 3.12). Цей інтерфейс дає змогу переглядати думки інших користувачів, а також залишати власний відгук. Відгуки виводяться у вигляді карток, що містять ім'я автора, текст коментаря та виставлений рейтинг у вигляді зірочок.

Праворуч від зони перегляду розташовано форму для додавання нового відгуку. Вона містить поле для введення тексту, інтерактивну панель вибору оцінки та кнопку «Додати». Усі нові коментарі додаються асинхронно, без перезавантаження сторінки, що забезпечує зручну й швидку взаємодію з користувачем.

Механізм збереження відгуків реалізовано через REST API, а авторизація користувача є обов'язковою умовою для публікації. Така реалізація дозволяє підтримувати зворотний зв'язок між покупцями та сайтом, а також формувати загальний рейтинг товару на основі оцінок усіх користувачів.

		Опис	Характеристики	Відгуки
Vlad Savchuk Чудова камера ★★★★★		Залиште відгук ☆☆☆☆☆ Ваш відгук Додати		

Рис. 3.12. Відгуки про товар

Представлено сторінку порівняння товарів (Рис. 3.13), що дає змогу візуально зіставити технічні характеристики кількох моделей. Інтерфейс складається з таблиці, у якій товари відображено у вигляді стовпців, а характеристики – рядками. У верхній частині кожного стовпця розміщено зображення товару, його назву та кнопку для видалення зі списку порівняння.

У прикладі показано три моделі: Apple iPhone 16e, Apple iPhone 16 Pro та Xiaomi Redmi Note 12 Pro Plus. Для кожного пристрою вказано діагональ екрана, обсяг вбудованої та оперативної пам'яті, роздільну здатність камер, тип роз'ємів, ємність акумулятора та колір корпусу. Завдяки однаковій структурі таблиці користувач може легко оцінити відмінності між моделями й прийняти зважене рішення щодо покупки.

Mobile Shop

Головна

Товари

Порівняння

Адмін-панель

Vlad Savchuk

Кошик



Apple iPhone 16e

×



Apple iPhone 16 Pro

×



Xiaomi Redmi Note 12 Pro...

×

Діагональ	6.1	6.3	6.67
Вбудована пам'ять, ГБ	128	1024	256
Оперативна пам'ять, ГБ	8	8	8
Основна камера, Мп	48	48	200
Фронтальна камера, Мп	8	12	16
Роз'єми	USB Type-C	USB Type-C	USB Type-C
Ємність акумулятора, мАг	4005	3355	5000
Колір	Чорний	Чорний	Білий

Рис. 3.13. Сторінка з порівнянням товарів

Представлено інтерфейс сторінки кошика (Рис. 3.14), який відображає вибрані товари перед оформленням замовлення. Ліва частина сторінки містить перелік доданих позицій у вигляді карток, кожна з яких включає зображення товару, назву, поточну ціну та поле для зміни кількості. Праворуч від кожного товару розміщено кнопку з іконкою смітника, яка дозволяє видалити позицію з кошика.

У правій частині сторінки знаходиться блок із загальним підсумком. У ньому відображається кількість доданих товарів, а також загальна сума до сплати. Нижче розміщено кнопку «Оформити замовлення», яка переводить користувача до етапу заповнення форми доставки й оплати.

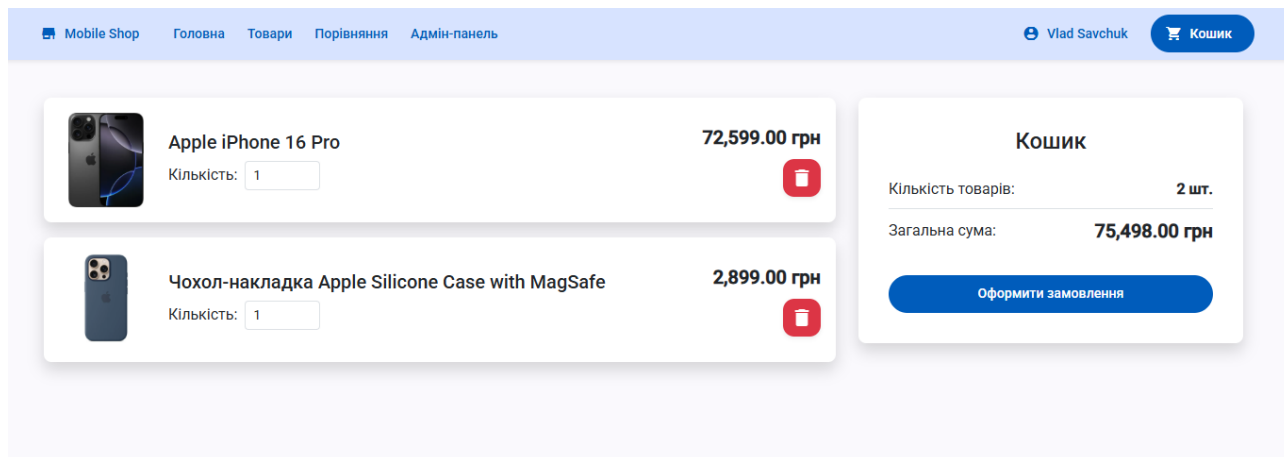


Рис. 3.14. Кошик товарів

Зображено інтерфейс сторінки оформлення замовлення (Рис. 3.15), яка є завершальним етапом процесу покупки в Інтернет-магазині. Сторінка поділена на дві основні частини: ліворуч розташовано форму для введення даних покупця, а праворуч – підсумкова інформація про замовлення.

У блоці «Дані покупця» користувач заповнює поля з ПІБ, електронною адресою та номером телефону. Нижче розміщено секцію «Відділення Нової Пошти», де потрібно вибрати місто й відділення доставки. Дані для цих полів підтягуються динамічно через API служби доставки «Нова Пошта»[6], що дозволяє швидко знайти потрібне відділення без зайвого введення вручну.

Правий блок «Ваш кошик» містить перелік товарів, обраних для замовлення, разом із їх кількістю та вартістю. Наприкінці виводиться загальна сума до оплати. Після перевірки всіх даних користувач натискає кнопку «Перейти до оплати», що ініціює формування платіжного запиту та перенаправлення на сторінку WayForPay[5] для здійснення транзакції.

Такий розподіл дозволяє користувачеві швидко оформити покупку з мінімальною кількістю дій, а також гарантує точність введених даних завдяки інтегрованим сервісам.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Mobile Shop

Головна

Товари

Порівняння

Адмін-панель

Vlad Savchuk

Кошик

Оформлення замовлення

Дані покупця

ПІБ

Vlad Savchuk

Електронна адреса

fedqus@gmail.com

Телефон

+380963746811

Ваш кошик

1 × Apple iPhone 16 Pro

72,599.00 грн

1 × Чохол-накладка Apple Silicone Case with MagSafe

2,899.00 грн

Всього: 75,498.00 грн

Відділення Нової Пошти

Місто

Житомир

Відділення

Відділення №1: вул. Гранітна, 16

Перейти до оплати

Рис. 3.15. Сторінка з оформленням замовлення

Зображено сторінку профілю користувача (Рис. 3.16), яка дозволяє переглядати та змінювати особисту інформацію. Інтерфейс поділений на дві основні зони: редагування персональних даних і керування безпекою облікового запису.

У верхній частині сторінки відображено форму з полями електронної адреси, ПІБ і номера телефону. Усі поля редаговані, що дозволяє користувачеві за потреби змінити свої контактні дані. Після внесення змін необхідно натиснути кнопку «Зберегти та вийти», яка надсилає оновлену інформацію на сервер для збереження.

У нижній частині сторінки розміщено блок, присвячений скиданню пароля. Користувачеві пропонується надіслати запит на зміну пароля, натиснувши відповідне посилання. Після цього на вказану електронну адресу надсилається лист із токеном і посиланням на сторінку, де можна встановити новий пароль.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Mobile Shop Головна Товари Порівняння Адмін-панель Vlad Savchuk Кошик

Головна Дані користувача

Замовлення

Електронна адреса
fedqus@gmail.com

ПІБ
Vlad Savchuk

Телефон
+380967346181

Зберегти та вийти

Скидання пароля

Натисніть кнопку нижче, щоб отримати лист із посиланням для скидання пароля.

Надіслати лист

Рис. 3.16. Профіль користувача

Зображено сторінку перегляду замовлень (Рис. 3.17), яка доступна авторизованим користувачам. Інтерфейс призначений для ознайомлення з історією покупок, відображенням поточного статусу оплати та виконання кожного з оформлених замовлень.

Сторінка містить перелік замовлень, де кожне з них виводиться у вигляді окремого блоку з номером, датою створення, статусом оплати (наприклад, «Оплачено» або «Повернуто кошти») і статусом виконання (наприклад, «Завершено» або «В очікуванні обробки»). Додатково виводиться заголовок «Товари в замовленні», під яким користувач може натиснути кнопку «Детально» для перегляду повного списку товарів, що входили до конкретного замовлення.

Mobile Shop Головна Товари Порівняння Адмін-панель Vlad Savchuk Кошик

Головна Ваші замовлення

Замовлення

Замовлення №11004
Дата: 17.06.2025, 23:19
Оплата: Повернуто кошти
Статус виконання: В очікуванні обробки

Товари в замовленні:
Детально

Замовлення №10004
Дата: 16.06.2025, 20:48
Оплата: Оплачено
Статус виконання: Завершено

Товари в замовленні:
Детально

Рис. 3.17. Замовлення користувача

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

Зображено інтерфейс сторінки резервного копіювання бази даних (Рис. 3.18), який доступний лише адміністраторам у межах адмінпанелі. Цей розділ дозволяє створювати, переглядати, завантажувати, відновлювати або видаляти наявні резервні копії системи.

Інтерфейс складається з таблиці, в якій для кожної резервної копії відображається її назва, розмір файлу та дата створення. Праворуч від кожного запису розміщено іконки для виконання дій – відновлення з резервної копії або її видалення. У верхній частині сторінки знаходиться кнопка «Створити», яка ініціює створення нової резервної копії, що зберігається на сервері у форматі .bak.

Даний функціонал критично важливий для забезпечення надійності роботи системи. Він дозволяє адміністраторам оперативно реагувати на технічні збої, відновлюючи дані до попереднього стану. Процес резервного копіювання реалізовано через інтеграцію з сервером SQL, що дозволяє працювати безпосередньо з файлами бази даних.

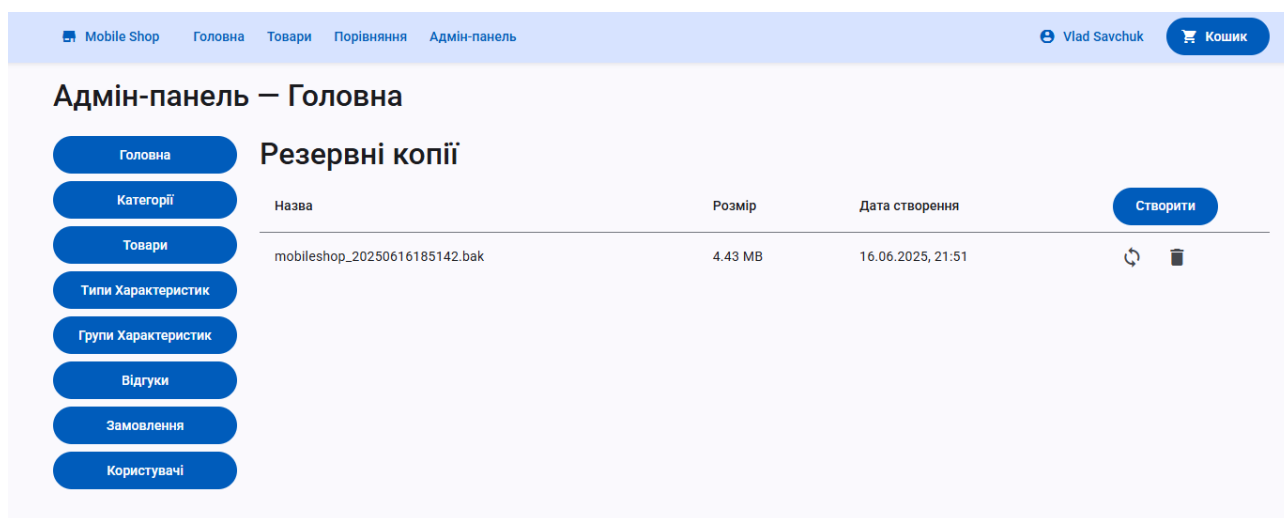


Рис. 3.18. Сторінка з резервними копіями для адміністратора

Зображено сторінку керування категоріями товарів (Рис. 3.19), що входить до функціоналу адмінпанелі. Цей інтерфейс дозволяє адміністраторам переглядати наявні категорії, створювати нові, а також редагувати або видаляти вже існуючі. Кожен запис у таблиці містить зображення, назву категорії та, за потреби, посилання на батьківську категорію, якщо йдеться про підкатегорію.

У правій частині таблиці біля кожного запису розміщено дві дії: кнопка з іконкою олівця для редагування іконки, назви або ієрархії категорії, а також кнопка видалення. Над таблицею розміщено поле пошуку для швидкої фільтрації списку, а також кнопка «Створити», яка відкриває форму для додавання нової категорії.

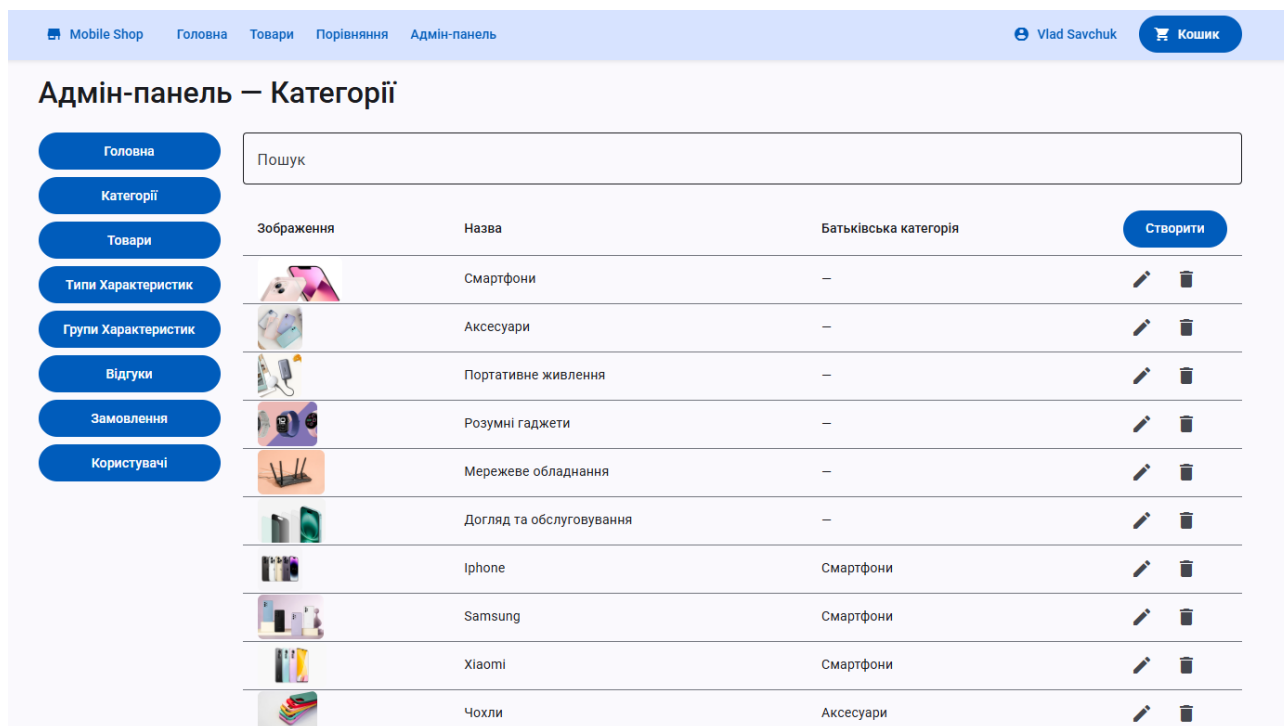


Рис. 3.19. Сторінка з категоріями для адміністратора

Зображено інтерфейс сторінки керування товарами в адмінпанелі (Рис. 3.20). Цей розділ надає адміністраторам можливість переглядати перелік усіх товарів, які представлені в Інтернет-магазині, а також виконувати базові CRUD-операції – створення, редагування й видалення.

У таблиці відображаються ключові параметри кожного товару: зображення, назва, ціна, категорія, до якої він належить, а також кількість доступних одиниць. У правій частині кожного рядка містяться кнопки з іконками – олівець для редагування товару та смітник для його видалення. Над таблицею реалізовано поле пошуку, що дозволяє швидко фільтрувати список за назвою, і кнопка «Створити», яка відкриває форму додавання нового товару.

Цей функціонал є важливим для оперативного оновлення інформації про асортимент магазину. Усі дії адміністратор виконує без потреби перезавантаження сторінки, що підвищує зручність керування товарною базою.

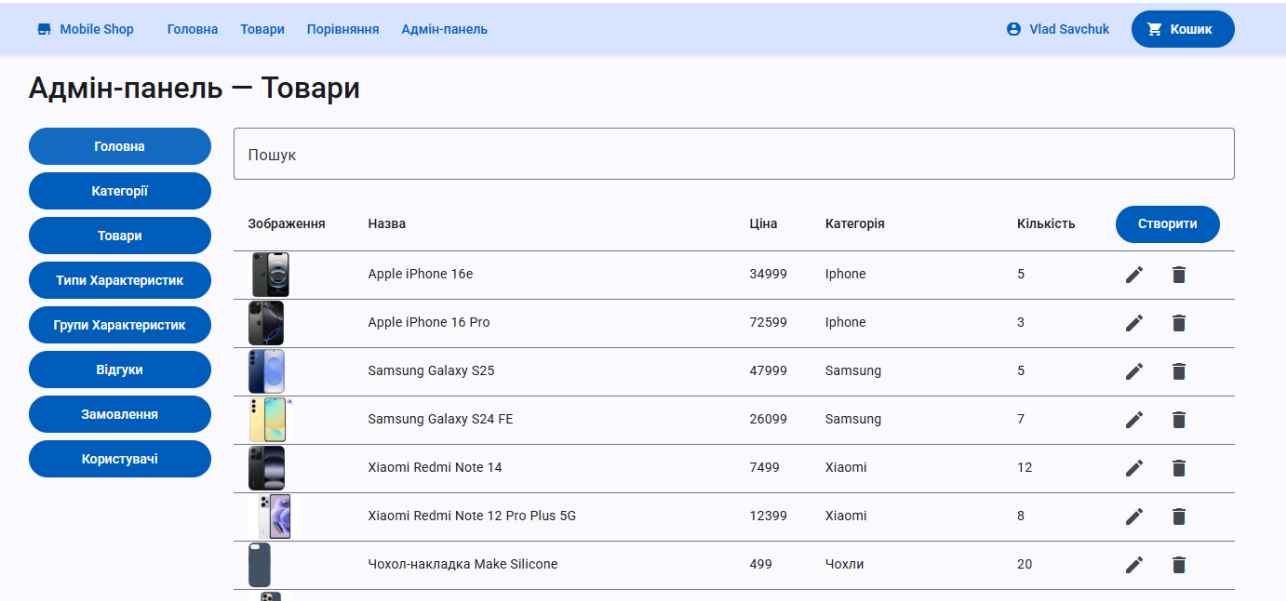


Рис. 3.20. Сторінка з товарами для адміністратора

Зображено сторінку керування типами характеристик товарів у межах адмінпанелі (Рис. 3.21). Даний інтерфейс дозволяє адміністраторам визначати перелік характеристик, які згодом будуть використовуватися для опису продукції та реалізації фільтрації на стороні користувача.

У таблиці відображено список характеристик із назвами, такими як «Діагональ», «Вбудована пам'ять, ГБ», «Оперативна пам'ять, ГБ», «Основна камера, Мп», «Роз'єми» тощо. Кожен запис має дві дії – редагування за допомогою іконки олівця та видалення через іконку смітника. Над таблицею реалізовано текстове поле для пошуку, а також кнопка «Створити» для додавання нових характеристик.

Цей розділ забезпечує гнучкість у структуризації характеристик товарів, дозволяє швидко оновлювати або розширювати їх перелік без необхідності втручання в код або базу даних вручну.

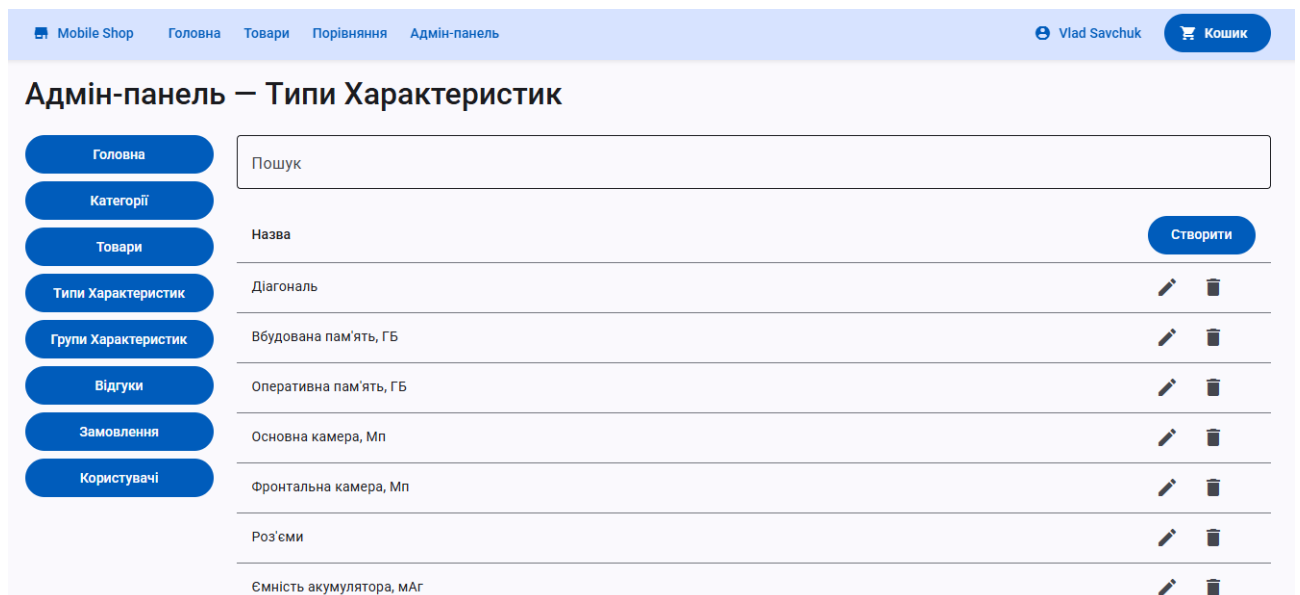


Рис. 3.21. Сторінка з типами характеристик для адміністратора

Представлено сторінку керування групами характеристик (Рис. 3.22), яка є частиною функціоналу адмінпанелі. Даний інтерфейс дозволяє адміністраторам створювати логічні об'єднання типів характеристик для різних категорій товарів. Це спрощує подальшу прив'язку параметрів до товарів.

У таблиці виведено назви груп (наприклад, «Смартфони», «Чохол», «Навушники» тощо), а також перелік типів характеристик, що входять до кожної групи. Наприклад, група «Смартфони» містить характеристики діагоналі, пам'яті, камер, роз'ємів тощо. Для кожного запису доступні дії редагування та видалення, які розміщено праворуч. У верхній частині інтерфейсу реалізовано поле пошуку та кнопка «Створити», що відкриває форму для створення нової групи з відповідними характеристиками.

Такий підхід дозволяє адміністраторам централізовано управляти структурою параметрів і забезпечує масштабованість системи при додаванні нових товарів.

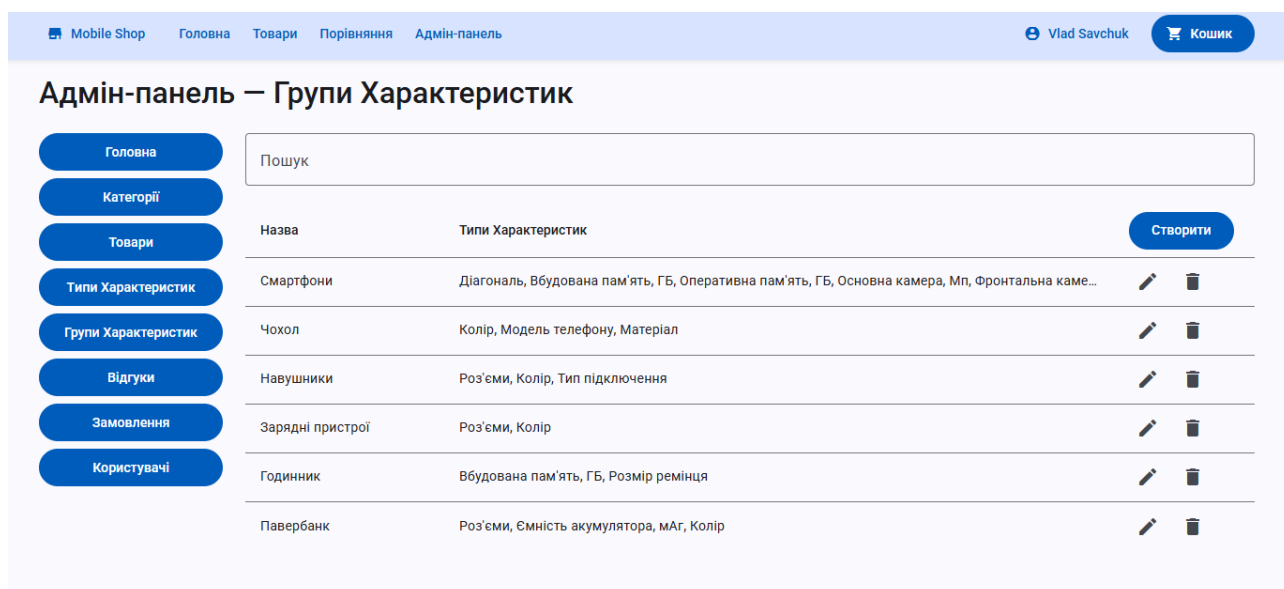


Рис. 3.22. Сторінка з групами характеристик для адміністратора

Зображено сторінку керування відгуками (Рис. 3.23), яка доступна адміністраторам у межах адмінпанелі. Цей інтерфейс призначений для перегляду всіх користувацьких відгуків, залишених на сайті, а також для їх модерації.

У таблиці представлено інформацію про товар, до якого належить відгук, ім'я користувача, зміст самого коментаря, рейтинг, виставлений у вигляді зірочок, а також доступні дії. Адміністратор має змогу редагувати або видаляти відгуки за допомогою відповідних іконок у стовпці «Дії». Над таблицею розташовано поле для пошуку, що дозволяє швидко знайти потрібний відгук за ключовими словами або іменем користувача.

Такий функціонал є важливим для підтримання якісного зворотного зв'язку в системі, забезпечення прозорості й контролю за публікаціями. Адміністратор може оперативно видалити неприйнятний або неправдивий відгук, тим самим покращуючи загальний досвід взаємодії користувачів із платформою.

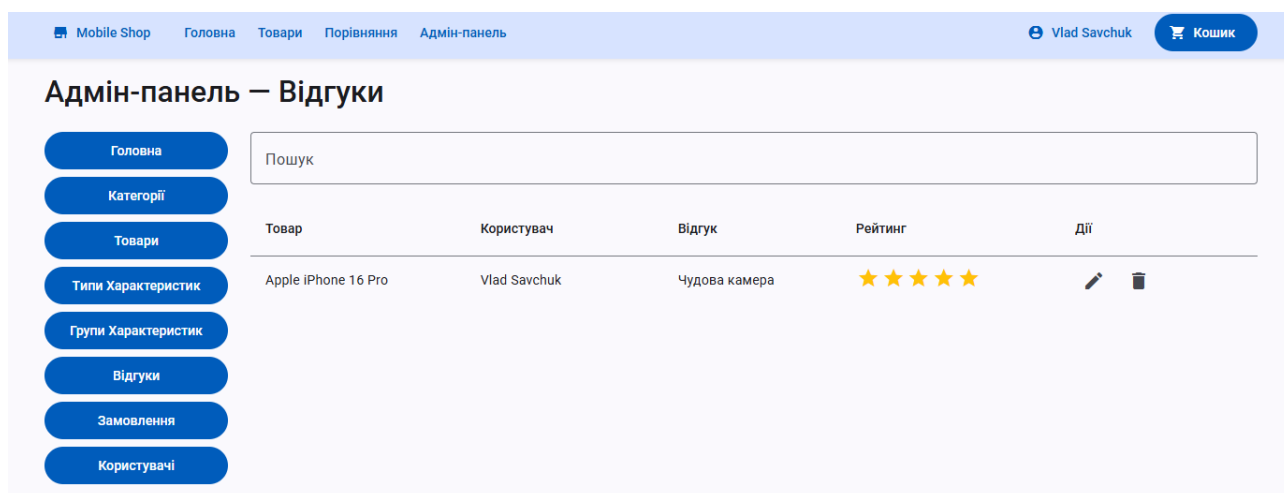


Рис. 3.23. Сторінка з відгуками для адміністратора

Представлено інтерфейс сторінки керування замовленнями (Рис. 3.24), який входить до складу адміністративної панелі. Даний функціонал забезпечує адміністратора можливістю переглядати всі замовлення користувачів, контролювати їх статуси та в разі потреби змінювати відповідні дані.

У таблиці виводиться основна інформація про кожне замовлення: ідентифікатор (#), дата оформлення, ПІБ покупця, місто доставки, статус оплати (наприклад, «Оплачено», «Повернуто кошти», «Помилка оплати») та статус виконання («В очікуванні обробки», «Завершено» тощо). У правій частині кожного рядка розміщено набір дій – редагування, перегляд деталей і видалення замовлення.

Над таблицею реалізовано текстове поле пошуку, що дозволяє адміністраторам швидко знаходити потрібне замовлення за ключовими параметрами. Інтерфейс адаптовано для швидкої взаємодії з великою кількістю записів, що актуально при високому потоці клієнтів.

При відкритті деталей конкретного замовлення адміністратор може переглянути перелік товарів, контактну інформацію користувача, метод оплати та обрану доставку. Це дозволяє оперативно реагувати на запити користувачів, уточнювати дані та вносити необхідні корективи.

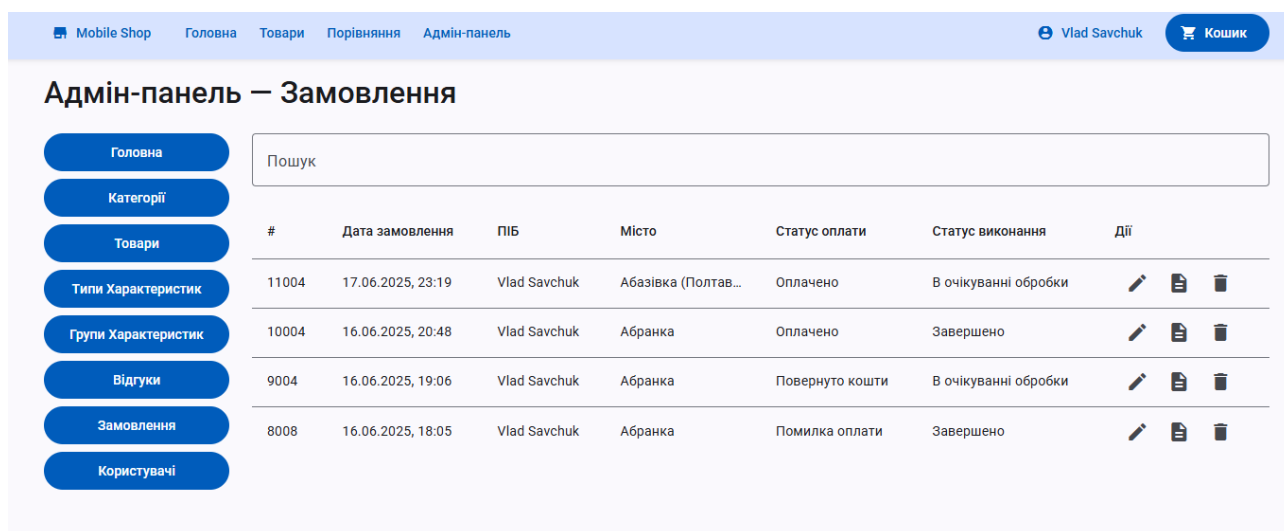


Рис. 3.24. Сторінка з замовленнями для адміністратора

Зображено сторінку керування користувачами (Рис. 3.25), яка є частиною адміністративної панелі. Цей функціонал дозволяє адміністраторам переглядати список зареєстрованих користувачів, а також виконувати базові дії з їхніми обліковими записами.

У таблиці відображаються ключові атрибути кожного користувача: повне ім'я (ПІБ), електронна адреса, номер телефону та роль (наприклад, «admin» або «user»). Праворуч від кожного запису розміщено кнопки для редагування та видалення користувача. Над таблицею реалізовано поле пошуку, яке дозволяє фільтрувати користувачів за іменем або поштою.

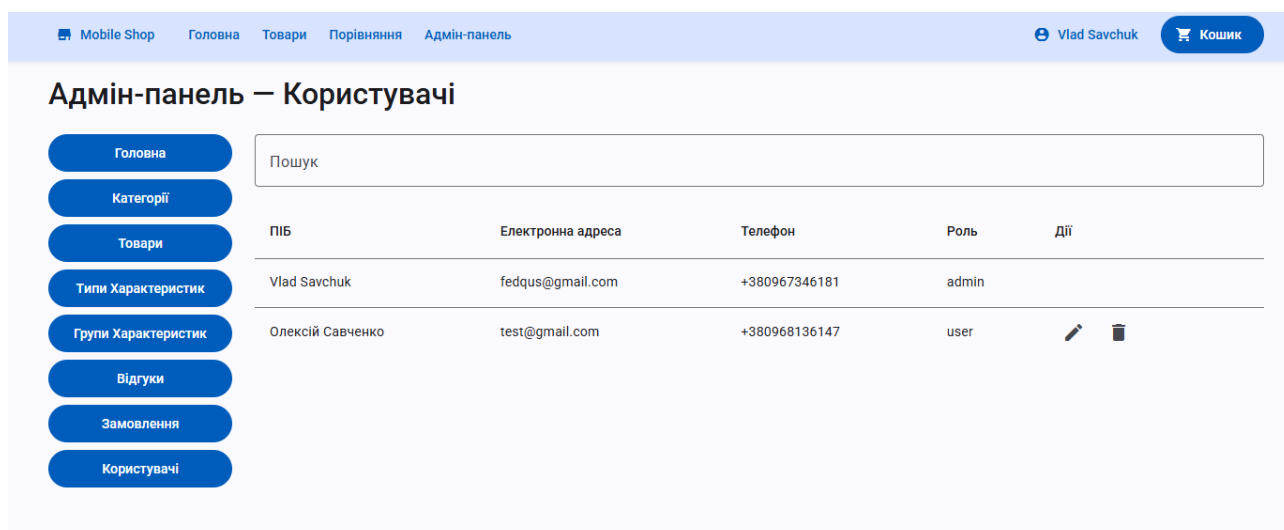


Рис. 3.25. Сторінка з користувачами для адміністратора

У результаті аналізу інтерфейсів вебзастосунку було продемонстровано логічну та послідовну організацію всіх сторінок для забезпечення зручної

взаємодії з користувачем. Інтерфейс поділено на окремі зони відповідно до ролей користувачів, що дозволяє чітко розмежовувати функціональні можливості між звичайними користувачами та адміністраторами.

Для користувачів реалізовано інтуїтивно зрозумілі сторінки реєстрації, авторизації, перегляду товарів, їх фільтрації, порівняння, додавання до кошика та оформлення замовлення. Особистий кабінет дає змогу змінювати особисті дані, переглядати замовлення й ініціювати скидання пароля. Усі ці елементи інтерфейсу розроблені з урахуванням принципів зручності, простоти та функціональності.

Інтерфейси адміністративної панелі охоплюють повний набір інструментів для керування категоріями, товарами, характеристиками, замовленнями, відгуками, користувачами та резервним копіюванням бази даних. Кожен розділ побудований у вигляді таблиці з можливістю пошуку, редагування та видалення, що забезпечує зручне адміністрування та масштабованість системи.

Таким чином, реалізований інтерфейс вебзастосунку повністю відповідає вимогам функціональності, забезпечує зручність для користувачів і ефективність для адміністраторів, а також сприяє цілісному та злагодженому досвіду взаємодії з системою.

3.3. Тестування програмного продукту

У процесі розробки вебзастосунку було проведено комплексне тестування з метою перевірки коректності роботи основних функціональних модулів, стабільності системи, відповідності до вимог і відсутності критичних помилок. Тестування охоплювало як клієнтську частину, створену на Angular[1], так і серверну частину, реалізовану за допомогою ASP.NET Core[2]. Окрему увагу приділено перевірці взаємодії з базою даних, правильності механізмів автентифікації та авторизації, роботі інтеграцій з платіжною системою WayForPay[5], а також з API служби доставки Нової Пошти.

Усі перевірки здійснювалися на локальному сервері, а також у тестовому середовищі з емульованим потоком користувачів. Тестувалися різноманітні сценарії використання системи: починаючи від стандартної взаємодії

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

користувача з інтерфейсом і закінчуючи обробкою критичних ситуацій, наприклад, втрати підключення до зовнішніх сервісів. Було перевірено роботу реєстрації та входу в систему, зміну даних профілю, додавання товарів у кошик, оформлення замовлення, генерацію платіжного посилання, отримання callback-відповіді від платіжного сервісу, а також підвантаження списків міст і відділень через API служби доставки. Водночас було протестовано процес скидання паролю, написання й відображення відгуків, фільтрацію товарів за заданими параметрами та механізм порівняння.

На боці адміністратора перевірялася можливість створення, редагування та видалення товарів, категорій і характеристик, управління обліковими записами користувачів, перегляд і обробка замовлень, а також створення та відновлення резервної копії бази даних. Усі ці дії перевірялися на відповідність очікуваному результату, правильність реакції інтерфейсу та стабільність логіки роботи системи.

Функціональне тестування проводилося вручну на кожному етапі реалізації. Особлива увага приділялася валідації форм, коректному відображенню повідомлень про помилки та успішні дії, а також перевірці переходів між сторінками. Ретельно перевірялися ситуації, коли користувач вводив некоректні дані, здійснював повторні дії або намагався отримати доступ до закритих ділянок системи без відповідних прав. Під час тестування використовувалися також інструменти Postman і Swagger[15] для прямої перевірки HTTP-запитів до REST API, що дозволило ізолювати серверну частину від клієнтського інтерфейсу і оцінити правильність обробки запитів на кожному рівні.

У результаті тестування були виявлені дрібні логічні помилки, пов'язані з дублюванням даних, некоректною валідацією полів або відсутністю повідомлень про результат дії. Всі виявлені недоліки були оперативно усунені під час реалізації. Повторне тестування підтвердило, що зміни не вплинули на інші частини системи.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Крім функціонального, проводилось також поверхнєве навантажувальне тестування. Було змодельовано ситуації з підвищеним потоком одночасних запитів до API, активною роботою декількох користувачів у клієнтській частині, великим обсягом записів у базі даних. Система продемонструвала стабільність роботи в межах очікуваних навантажень, адекватну швидкість відповіді й відсутність критичних збоїв.

У підсумку, результати тестування підтвердили працездатність вебзастосунку відповідно до вимог. Система стабільно функціонує, забезпечує очікувану логіку взаємодії та готова до впровадження в реальні умови з подальшою перспективою масштабування та розвитку.

Висновки до третього розділу

У третьому розділі було детально розглянуто інтерфейсні рішення, порядок налаштування системи, її функціональні компоненти, а також проведено тестування для підтвердження коректності роботи. Було описано, як саме користувач взаємодіє з інтерфейсом вебзастосунку, від перегляду товарів до оформлення замовлення, та як адміністратор керує контентом за допомогою зручної адмінпанелі.

Система має модульну структуру, що дозволяє легко масштабувати її функціональність. Усі ключові елементи – особистий кабінет, реєстрація, аутентифікація, робота з характеристиками, кошиком, відгуками, порівнянням товарів та обробкою замовлень реалізовані відповідно до функціональних вимог.

Проведене тестування підтвердило, що вебзастосунок функціонує стабільно, реагує коректно на дії користувача та адміністратора, а також демонструє надійність у роботі з зовнішніми інтеграціями – такими як платіжна система WayForPay[5] та API служби доставки «Нова Пошта»[6].

У підсумку, реалізований програмний продукт повністю відповідає поставленим завданням і готовий до практичного використання.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи було реалізовано повнофункціональний вебзастосунок для Інтернет-магазин у з продажу мобільних пристроїв. Робота охоплювала всі етапи життєвого циклу програмного забезпечення – від аналізу предметної галузі та формування технічних вимог до проєктування архітектури, реалізації ключових модулів, налаштування інтерфейсу та проведення тестування.

На основі проведеного аналізу існуючих аналогів було сформульовано функціональні вимоги, які дозволили створити систему, що поєднує зручність користування з розширеною функціональністю. Система реалізована за архітектурною моделлю «клієнт-сервер», де клієнтська частина побудована з використанням Angular[1], а серверна частина – на основі ASP.NET Core[2] із застосуванням REST API.

У проєкті впроваджено механізми авторизації та аутентифікації на основі JWT[9], реалізовано взаємодію з реляційною базою даних Microsoft SQL Server[4] через Entity Framework Core[3], інтегровано платіжну систему WayForPay[5], а також API служби доставки «Нова Пошта»[6] для динамічного формування адреси доставки. Особливу увагу приділено функціоналу фільтрації, порівняння товарів, обробки відгуків, резервного копіювання та адміністрування даних.

Результати тестування підтвердили працездатність усіх основних компонентів системи. Реалізоване рішення відповідає вимогам до сучасного електронного комерційного застосунку та має достатній потенціал для подальшого розширення функціоналу й масштабування.

Реалізований вебзастосунок є завершеним прикладом сучасного інтернет-магазину з повним циклом електронної комерції, який об'єднує зручний інтерфейс, надійну архітектуру, інтеграцію з зовнішніми сервісами та гнучку систему адміністрування, що підтверджує готовність системи до практичного використання та подальшого розвитку.

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Angular – Official Documentation [Електронний ресурс]. – Режим доступу: <https://angular.io/docs>
2. ASP.NET Core Documentation – Microsoft Docs [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core>
3. Entity Framework Core Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/ef/core>
4. Microsoft SQL Server Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/sql/sql-server>
5. WayForPay Payment Gateway API [Електронний ресурс]. – Режим доступу: <https://wiki.wayforpay.com/en>
6. Нова Пошта API v2.0 [Електронний ресурс]. – Режим доступу: <https://developers.novaposhta.ua/documentation>
7. Bootstrap – The most popular HTML, CSS, and JS library [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com/>
8. Material Design for Angular (Angular Material) [Електронний ресурс]. – Режим доступу: <https://material.angular.io/>
9. JWT Introduction – jsonwebtoken.io [Електронний ресурс]. – Режим доступу: <https://jwt.io/introduction>
10. RESTful API Design – Microsoft Docs [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
11. Visual Studio – IDE Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/visualstudio>
12. Microsoft Docs – .NET Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/>
13. RxJS Documentation – Reactive Extensions Library for JavaScript [Електронний ресурс]. – Режим доступу: <https://rxjs.dev/>
14. GitLab – The DevSecOps Platform [Електронний ресурс]. – Режим доступу: <https://gitlab.com/>

		Савчук В.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Власенко О.В.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

15. Swagger – OpenAPI Specification Documentation [Електронний ресурс]. – Режим доступу: <https://swagger.io/specification/>
16. draw.io – Diagrams and ER Modeling [Електронний ресурс]. – Режим доступу: <https://app.diagrams.net/>
17. Google Fonts – Typography for Web [Електронний ресурс]. – Режим доступу: <https://fonts.google.com/>
18. CSS Tricks – Frontend Development Guide [Електронний ресурс]. – Режим доступу: <https://css-tricks.com/>
19. Stack Overflow – Developer Q&A Community [Електронний ресурс]. – Режим доступу: <https://stackoverflow.com/>
20. MailKit Documentation [Електронний ресурс]. – Режим доступу: <https://github.com/jstedfast/MailKit>
21. Git – Official Documentation [Електронний ресурс]. – Режим доступу: <https://git-scm.com/doc>
22. Angular Routing and Navigation [Електронний ресурс]. – Режим доступу: <https://angular.io/guide/router>
23. Jason Watmore. JWT Authentication in ASP.NET Core [Електронний ресурс]. – Режим доступу: <https://jasonwatmore.com/post/2021/07/18/net-5-jwt-authentication-tutorial-with-example-api>
24. Fowler, M. Refactoring: Improving the Design of Existing Code. – 2nd ed. – Boston: Addison-Wesley, 2018. – 448 p.
25. Freeman, E., Robson, E. Head First Design Patterns. – 2nd ed. – Sebastopol: O'Reilly Media, 2020. – 672 p.
26. Richter, J. CLR via C#. – 4th ed. – Redmond: Microsoft Press, 2012. – 826 p.
27. Esposito, D. Programming ASP.NET Core. – 1st ed. – Microsoft Press, 2018. – 432 p.
28. Seemann, M. Dependency Injection in .NET. – Manning Publications, 2011. – 528 p.
29. Freeman, A. Pro Angular: Build Powerful and Dynamic Web Apps. – 5th ed. – Berkeley: Apress, 2022. – 880 p.

		Савчук В.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Власенко О.В.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

30. Troelsen, A., Japikse, P. Pro C# 10 with .NET 6. – 11th ed. – Apress, 2022. – 1535 p.

		Савчук В.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Власенко О.В.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Клієнтська частина Angular

categories.component.ts

```
@Component({
  selector: 'app-categories',
  imports: [CategoryTilesComponent],
  templateUrl: './categories.component.html',
  styleUrls: ['./categories.component.scss']
})
export class CategoriesComponent {
  @ViewChild("categoryTiles") categoryTiles!: CategoryTilesComponent;

  route = inject(ActivatedRoute)
  router = inject(Router)
  categoryService = inject(CategoryService)

  title = signal("Kateropii")

  ngOnInit(): void {
    this.route.paramMap.subscribe(params => {
      const id = +params.get("id")!;

      if (id) {
        this.categoryService.getChildren(id).subscribe(categories => {
          if (categories.length == 0) {
            this.router.navigate(['/products'], { queryParams: { categoryId: id } });
          }
          this.categoryTiles.setCategories(categories);
        })
        this.categoryService.getById(id).subscribe(category => {
          this.title.set(category.name)
        })
      }
      else {
        this.categoryService.getRoots().subscribe(categories => {
          this.categoryTiles.setCategories(categories);
        })
      }
    })
  }
}
```

```
}
```

compare.component.ts

```
interface FeatureCell {  
  productIndex: number;  
  feature?: ProductFeature;  
  numericValue?: number;  
}
```

```
@Component({  
  selector: 'app-compare',  
  imports: [CommonModule, MatButtonModule, MatIconModule, RouterModule],  
  templateUrl: './compare.component.html',  
  styleUrls: ['./compare.component.scss']  
})
```

```
export class CompareComponent {  
  apiUrl = environment.apiUrl;  
  products: Product[] = [];  
  featureKeys: FeatureType[] = [];  
  compareService = inject(CompareService);
```

```
  ngOnInit(): void {  
    this.compareService.items$.subscribe(items => {  
      this.products = items;  
      this.buildFeatureKeys();  
    });  
  }  
}
```

```
  private buildFeatureKeys(): void {  
    const map = new Map<number, FeatureType>();  
    this.products.forEach(p => p.features?.forEach(f => {  
      if (f.featureType) {  
        map.set(f.featureType.id, f.featureType);  
      }  
    }));  
    this.featureKeys = Array.from(map.values())  
  }
```

```
  getFeatureValue(product: Product, feature: FeatureType): string {  
    const f = product.features?.find(x => x.featureTypeId === feature.id);
```

```
    return f?.value ?? '-';  
  }  
}
```

```
private buildFeatureMap(): Map<number, FeatureCell[]> {  
  const map = new Map<number, FeatureCell[]>();  
  
  this.featureKeys.forEach(ft => {  
    const cells: FeatureCell[] = this.products.map((p, idx) => {  
      const f = p.features?.find(x => x.featureTypeId === ft.id);  
      const num = f ? parseFloat(f.value) : NaN;  
      return { productIndex: idx, feature: f, numericValue: !isNaN(num) ? num : undefined };  
    });  
    map.set(ft.id, cells);  
  });  
  
  return map;  
}
```

```
public shouldHighlight(product: Product, feature: FeatureType): boolean {  
  const cells = this.buildFeatureMap().get(feature.id)!;  
  
  const presentCells = cells.filter(c => !!c.feature);  
  
  const first = presentCells[0]?.feature!.value;  
  if (presentCells.length  
    && presentCells.every(c => c.feature!.value === first)) {  
    return !!product.features?.find(f => f.featureTypeId === feature.id);  
  }  
}
```

```
const nums = presentCells  
  .map(c => parseFloat(c.feature!.value))  
  .filter(n => !isNaN(n));  
if (nums.length === presentCells.length && nums.length > 0) {  
  const max = Math.max(...nums);  
  return parseFloat(this.getFeatureValue(product, feature)) === max;  
}  
  
return false;  
}
```

```

remove(prod: Product) {
  this.compareService.removeFromCompare(prod.id);
}
}

```

cart.component.ts

```

@Component({
  selector: 'app-cart',
  imports: [CommonModule, RouterModule, MatButtonModule, MatIconModule],
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.scss']
})
export class CartComponent {
  apiUrl = environment.apiUrl;

  cartService = inject(CartService);
  notificationsService = inject(NotificationsService);
  cartItems$ = this.cartService.cartItems$;

  remove(productId: number) {
    this.cartService.removeFromCart(productId);
  }

  handleQuantityChange(event: Event, productId: number): void {
    const input = event.target as HTMLInputElement;
    const quantity = input.valueAsNumber;

    try {
      this.cartService.updateQuantity(productId, quantity);
    } catch (error) {
      const previous = this.cartService.getCartItems().find(i => i.product.id === productId)?.quantity ?? 1;
      input.value = previous.toString();

      const message = error instanceof Error ? error.message : 'Помилка при зміні кількості товару';
      this.notificationsService.addNotification({ message, style: NotificationStyles.Error });
    }
  }
}

```

auth.service.ts

```

@Injectable({

```

```

providedIn: 'root'
})
export class AuthService {
  private key: string = "accessToken";

  private http = inject(HttpClient);
  private jwtHelper = new JwtHelperService();

  private userSubject: BehaviorSubject<IdentityUser | null>;
  public user$: Observable<IdentityUser | null>;

  constructor(@Inject(PLATFORM_ID) platformId: Object) {
    this.userSubject = new BehaviorSubject<IdentityUser | null>(isPlatformBrowser(platformId) ? this.getUser() : null)
    this.user$ = this.userSubject.asObservable();
  }

  login(loginRequest: LoginRequest): Observable<LoginResponse> {
    return this.http.post<LoginResponse>(`${environment.apiUrl}/auth/login`, loginRequest).pipe(
      tap(response => {
        this.setAccessToken(response.accessToken);
      })
    );
  }

  register(registerRequest: RegisterRequest): Observable<Object> {
    return this.http.post(`${environment.apiUrl}/auth/register`, registerRequest)
  }

  requestPasswordReset(request: PasswordResetRequest): Observable<void> {
    return this.http.post<void>(`${environment.apiUrl}/auth/request-password-reset`, request)
  }

  resetPassword(request: NewPasswordRequest): Observable<void> {
    return this.http.post<void>(`${environment.apiUrl}/auth/reset-password`, request)
  }

  logout(): void {
    this.removeAccessToken();
  }
}

```

```
isLoggedIn(): boolean {  
    return !this.isTokenExpired();  
}
```

```
private getUser(): IdentityUser | null {  
    return !this.isTokenExpired() ? this.getDecodedToken() : null;  
}
```

```
private getAccessToken(): string | null {  
    if (typeof localStorage !== 'undefined') {  
        return localStorage.getItem(this.key);  
    }  
    return null;  
}
```

```
private setAccessToken(token: string): void {  
    if (typeof localStorage !== 'undefined') {  
        localStorage.setItem(this.key, token);  
        this.userSubject.next(this.getUser());  
    }  
}
```

```
private removeAccessToken(): void {  
    if (typeof localStorage !== 'undefined') {  
        localStorage.removeItem(this.key);  
        this.userSubject.next(this.getUser());  
    }  
}
```

```
private getDecodedToken(): IdentityUser | null {  
    const token = this.getAccessToken();  
    return token ? this.jwtHelper.decodeToken<IdentityUser>(token) : null;  
}
```

```
private isTokenExpired(): boolean {  
    const token = this.getAccessToken();  
  
    return token ? this.jwtHelper.isTokenExpired(token) : true;  
}
```

```
}  
}
```

Додаток Б.

Серверна частина ASP.NET Core

AuthenticationController.cs

```
namespace MobileShopApi.Controllers
```

```
{
```

```
    [Route("api/auth")]
```

```
    public class AuthenticationController : Controller
```

```
    {
```

```
        private readonly MobileShopDbContext _context;
```

```
        private readonly PasswordHasherService _passwordHasher;
```

```
        private readonly AccessTokenGeneratorService _accessTokenGenerator;
```

```
        private readonly PasswordResetTokenService _tokenService;
```

```
        private readonly EmailService _emailService;
```

```
        public AuthenticationController(MobileShopDbContext context, PasswordHasherService passwordHasher,
```

```
        AccessTokenGeneratorService accessTokenGenerator, PasswordResetTokenService tokenService, EmailService emailService)
```

```
        {
```

```
            _context = context;
```

```
            _passwordHasher = passwordHasher;
```

```
            _accessTokenGenerator = accessTokenGenerator;
```

```
            _tokenService = tokenService;
```

```
            _emailService = emailService;
```

```
        }
```

```
        [HttpPost("register")]
```

```
        public async Task<ActionResult> Register([FromBody] RegisterRequest request)
```

```
        {
```

```
            if (!ModelState.IsValid)
```

```
            {
```

```
                return BadRequest(new ErrorResponse(ModelState.Values.SelectMany(v => v.Errors.Select(e => e.ErrorMessage))));
```

```
            }
```

```
            var existingUser = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);
```

```
            if (existingUser != null)
```

```
            {
```

```
                return Conflict(new ErrorResponse("Електронна адреса вже існує!"));
```

```

    }

    var newUser = new User
    {
        Email = request.Email,
        Fullname = request.Fullname,
        PasswordHash = _passwordHasher.HashPassword(request.Password!)
    };

    await _context.Users.AddAsync(newUser);
    await _context.SaveChangesAsync();

    return Ok();
}

[HttpPost("login")]
public async Task<ActionResult<LoginResponse>> Login([FromBody] LoginRequest request)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(new ErrorResponse(ModelState.Values.SelectMany(v => v.Errors.Select(e => e.ErrorMessage))));
    }

    var authenticatedUser = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);

    if (authenticatedUser == null)
    {
        return Unauthorized(new ErrorResponse("Недійсна електронна адреса!"));
    }

    if (!_passwordHasher.VerifyPassword(request.Password!, authenticatedUser.PasswordHash!))
    {
        return Unauthorized(new ErrorResponse("Неправильний пароль!"));
    }

    var accessToken = _accessTokenGenerator.GenerateToken(authenticatedUser);

    return new LoginResponse { AccessToken = accessToken };
}

```

```

[HttpPost("request-password-reset")]
public async Task<ActionResult> RequestPasswordReset([FromBody] PasswordResetRequest request)
{
    try
    {
        Console.WriteLine(request.Email);

        if (!ModelState.IsValid)
        {
            return BadRequest(new ErrorResponse(ModelState.Values.SelectMany(v => v.Errors.Select(e => e.ErrorMessage))));
        }

        var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);
        if (user == null) return Ok();

        var token = await _tokenService.GenerateAsync(user.Id);

        var scheme = Request.Scheme;
        var host = Request.Host.Value;

        var resetLink = $"{scheme}://{host}/reset-password?token={Uri.EscapeDataString(token)}";

        var htmlBody = @$"
        <p>Ви запросили скидання паролю.</p>
        <p>Щоб встановити новий пароль, перейдіть за посиланням:</p>
        <p><a href='{resetLink}'>Скинути пароль</a></p>
        <p>Якщо ви не робили цього запиту, просто проігноруйте цей лист.</p>";

        await _emailService.SendEmailAsync(
            to: request.Email,
            subject: "Скидання паролю",
            htmlBody: htmlBody
        );

        return Ok();
    }
    catch (Exception ex)
    {

```

```

        return BadRequest(new ErrorResponse(ex.Message));
    }
}

[HttpPost("reset-password")]
public async Task<IActionResult> ResetPassword([FromBody] NewPasswordRequest request)
{
    Console.WriteLine(request.Token);
    Console.WriteLine(request.NewPassword);
    Console.WriteLine(request.ConfirmNewPassword);
    if (!ModelState.IsValid)
    {
        return BadRequest(new ErrorResponse(ModelState.Values.SelectMany(v => v.Errors.Select(e => e.ErrorMessage))));
    }

    var userId = await _tokenService.ValidateAsync(request.Token);
    if (userId == null)
    {
        return BadRequest(new ErrorResponse("Невірний або прострочений токен."));
    }

    var user = await _context.Users.FindAsync(userId.Value);
    if (user == null)
    {
        return NotFound(new ErrorResponse("Користувача не знайдено."));
    }

    user.PasswordHash = _passwordHasher.HashPassword(request.NewPassword);
    await _context.SaveChangesAsync();

    await _tokenService.ConsumeAsync(request.Token);

    return Ok();
}
}
}

```

PaymentsController.cs

```

namespace MobileShopApi.Controllers
{

```

```

[Route("api/[controller]")]
public class PaymentsController : Controller
{
    private readonly MobileShopDbContext _context;
    private readonly IConfiguration _configuration;

    public PaymentsController(MobileShopDbContext context, IConfiguration configuration)
    {
        _context = context;
        _configuration = configuration;
    }

    [HttpPost("initiate")]
    public async Task<ActionResult<WayForPayInitResponse>> Initiate([FromBody] OrderRequest request)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(new ErrorResponse(ModelState.Values.SelectMany(v => v.Errors).Select(e => e.ErrorMessage)));
        }

        foreach (var item in request.Items)
        {
            var product = await _context.Products.FirstOrDefaultAsync(p => p.Id == item.ProductId);
            if (product != null)
            {
                if (item.Quantity > product.Quantity)
                {
                    return BadRequest(new ErrorResponse("Кількість товару в замовленні не може перевищувати кількість на складі!"));
                }
            }
        }

        var order = new Order
        {
            OrderDate = DateTimeOffset.UtcNow,
            FullName = request.FullName,
            Email = request.Email,
            Phone = request.Phone,

```

```

    CityName = request.CityName,
    WarehouseName = request.WarehouseName
};
_context.Orders.Add(order);
await _context.SaveChangesAsync();

var orderItems = request.Items
    .Select(i => new OrderItem
    {
        OrderId = order.Id,
        ProductId = i.ProductId,
        Quantity = i.Quantity
    })
    .ToList();
_context.OrderItems.AddRange(orderItems);
await _context.SaveChangesAsync();

foreach (var item in orderItems)
{
    var product = await _context.Products.FirstOrDefaultAsync(p => p.Id == item.ProductId);
    if (product != null)
    {
        product.Quantity -= item.Quantity;
    }
}
await _context.SaveChangesAsync();

var scheme = Request.Scheme;
var host = Request.Host.Value;

var merchantDomainName = host;
var merchantAccount = _configuration["WayForPay:MerchantAccount"]!;
var merchantKey = _configuration["WayForPay:SecretKey"]!;

var details = orderItems
    .Join(_context.Products,
        oi => oi.ProductId,
        p => p.Id,
        (oi, p) => new { p.Name, p.Price, oi.Quantity })

```

```

        .ToList();

var orderReference = order.Id.ToString();
var orderDateUnix = order.OrderDate.ToUnixTimeSeconds().ToString();
const string currency = "UAH";
var total = details.Sum(x => x.Price * x.Quantity).ToString("F2", CultureInfo.InvariantCulture);

var productNames = details.Select(x => x.Name).ToArray();
var productCounts = details.Select(x => x.Quantity.ToString()).ToArray();
var productPrices = details.Select(x => x.Price.ToString("F2", CultureInfo.InvariantCulture)).ToArray();

var signParts = new List<string>
{
    merchantAccount,
    merchantDomainName,
    orderReference,
    orderDateUnix,
    total,
    currency
};

signParts.AddRange(productNames);
signParts.AddRange(productCounts);
signParts.AddRange(productPrices);
var signString = string.Join(";", signParts);

string signature;
using (var hmac = new HMACMD5(Encoding.UTF8.GetBytes(merchantKey)))
{
    var hash = hmac.ComputeHash(Encoding.UTF8.GetBytes(signString));
    signature = BitConverter.ToString(hash).Replace("-", "").ToLowerInvariant();
}

var serviceUrl = $"{scheme}://{host}/api/payments/callback";
var returnUrl = $"{scheme}://{host}/checkout/confirmation";

var response = new WayForPayInitResponse
{
    MerchantAccount = merchantAccount,
    MerchantDomainName = merchantDomainName,

```

```

        OrderReference = orderReference,
        OrderDate = long.Parse(orderDateUnix),
        Amount = total,
        Currency = currency,

        ProductName = productNames,
        ProductCount = productCounts,
        ProductPrice = productPrices,

        ServiceUrl = serviceUrl,
        ReturnUrl = returnUrl,
        Signature = signature
    };

    await _context.SaveChangesAsync();

    return Ok(response);
}

[HttpPost("callback")]
public async Task<ActionResult> Callback()
{
    var body = await new StreamReader(Request.Body).ReadToEndAsync();

    WayForPayServiceCallbackRequest request;
    try
    {
        request = JsonSerializer.Deserialize<WayForPayServiceCallbackRequest>(
            body,
            new JsonSerializerOptions { PropertyNameCaseInsensitive = true }
        );
    }
    catch (Exception ex)
    {
        Console.WriteLine("JSON parse error: " + ex);
        return BadRequest("Unable to parse callback JSON");
    }

    var order = await _context.Orders.FirstOrDefaultAsync(o => o.Id.ToString() == request.OrderReference);

```

```

        if (order == null) return NotFound();

        order.PaymentStatus = request.TransactionStatus switch
        {
            "Approved" => PaymentStatus.Paid,
            "Pending" => PaymentStatus.Pending,
            "Refunded" => PaymentStatus.Refunded,
            _ => PaymentStatus.Failed
        };
        order.RepayUrl = request.RepayUrl;

        Console.WriteLine(request.TransactionStatus);

        await _context.SaveChangesAsync();

        var response = new WayForPayServiceCallbackResponse
        {
            OrderReference = request.OrderReference,
            Status = "accept",
            Time = DateTimeOffset.UtcNow.ToUnixTimeSeconds()
        };
        return Ok(response);
    }
}

```

AccessTokenGeneratorService.cs

```

using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using Microsoft.IdentityModel.Tokens;
using MobileShopApi.Models;

namespace MobileShopApi.Services
{
    public class AccessTokenGeneratorService
    {
        private readonly AuthenticationConfiguration _authenticationConfiguration;

        public AccessTokenGeneratorService(AuthenticationConfiguration authenticationConfiguration)

```

```

{
    _authenticationConfiguration = authenticationConfiguration;
}

public string GenerateToken(User user)
{
    IEnumerable<Claim> claims = new List<Claim>
    {
        new Claim("id", user.Id.ToString()),
        new Claim("emailaddress", user.Email),
        new Claim("phone", user.Phone),
        new Claim("name", user.Fullname),
        new Claim("role", user.Role)
    };

    SecurityKey key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_authenticationConfiguration.AccessTokenSecret!));
    SigningCredentials credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

    JwtSecurityToken token = new JwtSecurityToken(
        _authenticationConfiguration.Issuer,
        _authenticationConfiguration.Audience,
        claims,
        DateTime.UtcNow,
        DateTime.UtcNow.AddMinutes(_authenticationConfiguration.AccessTokenExpirationMinutes),
        credentials);

    return new JwtSecurityTokenHandler().WriteToken(token);
}
}

```

PasswordResetTokenService.cs

```

using Microsoft.EntityFrameworkCore;
using MobileShopApi.Models;
using System.Security.Cryptography;
using System.Text;

namespace MobileShopApi.Services
{
    public class PasswordResetTokenService

```

```

{
    private readonly MobileShopDbContext _context;
    private readonly TimeSpan _tokenLifetime = TimeSpan.FromHours(1);

    public PasswordResetTokenService(MobileShopDbContext context)
    {
        _context = context;
    }

    public async Task<string> GenerateAsync(int userId)
    {
        await RevokeAllAsync(userId);

        var bytes = RandomNumberGenerator.GetBytes(32);
        var token = Convert.ToBase64String(bytes);

        using var sha = SHA256.Create();
        var hash = Convert.ToBase64String(
            sha.ComputeHash(Encoding.UTF8.GetBytes(token))
        );

        var entry = new PasswordResetToken
        {
            UserId = userId,
            TokenHash = hash,
            ExpiresAt = DateTime.UtcNow.Add(_tokenLifetime),
            Used = false
        };

        _context.PasswordResetTokens.Add(entry);
        await _context.SaveChangesAsync();

        return token;
    }

    public async Task<int?> ValidateAsync(string token)
    {
        using var sha = SHA256.Create();
        var hash = Convert.ToBase64String(

```

```

        sha.ComputeHash(Encoding.UTF8.GetBytes(token))
    );

    var entry = await _context.PasswordResetTokens
        .AsNoTracking()
        .FirstOrDefaultAsync(t =>
            t.TokenHash == hash &&
            !t.Used &&
            t.ExpiresAt > DateTime.UtcNow
        );

    return entry?.UserId;
}

public async Task ConsumeAsync(string token)
{
    using var sha = SHA256.Create();
    var hash = Convert.ToBase64String(
        sha.ComputeHash(Encoding.UTF8.GetBytes(token))
    );

    var entry = await _context.PasswordResetTokens
        .FirstOrDefaultAsync(t => t.TokenHash == hash && !t.Used);

    if (entry != null)
    {
        entry.Used = true;
        await _context.SaveChangesAsync();
    }
}

public async Task RevokeAllAsync(int userId)
{
    var tokens = await _context.PasswordResetTokens
        .Where(t => t.UserId == userId && !t.Used)
        .ToListAsync();

    if (tokens.Count > 0)
    {

```

```

        foreach (var t in tokens)
        {
            t.Used = true;
        }

        await _context.SaveChangesAsync();
    }
}
}
}

```

Додаток В.

Моделі бази даних

User.cs

```

namespace MobileShopApi.Models
{
    public static class Roles
    {
        public const string Admin = "admin";
        public const string User = "user";
    }

    public class User
    {
        public int Id { get; set; }
        public string Email { get; set; } = null!;
        public string Fullname { get; set; } = null!;
        public string? Phone { get; set; }
        public string Role { get; set; } = Roles.User;
        public string PasswordHash { get; set; } = null!;
    }
}

```

Product.cs

```

namespace MobileShopApi.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; } = null!;
        public string? Description { get; set; }
        public decimal Price { get; set; }
    }
}

```

```

    public int Quantity { get; set; }
    public string MainImagePath { get; set; } = null!;
    public double Rating { get; set; }
    public int ReviewsCount { get; set; }
    public int CategoryId { get; set; }

    public Category? Category { get; set; }
    public ICollection<ProductFeature> Features { get; set; } = null!;
    public IEnumerable<ProductImage> Images { get; set; } = null!;
}
}

```

Order.cs

```

namespace MobileShopApi.Models
{
    public enum PaymentStatus
    {
        Unpaid,
        Pending,
        Paid,
        Failed,
        Refunded
    }

    public enum ExecutionStatus
    {
        Pending,
        InProgress,
        Shipped,
        Completed,
        Cancelled
    }

    public class Order
    {
        public int Id { get; set; }
        public DateTimeOffset OrderDate { get; set; }

        public string FullName { get; set; } = null!;
        public string Email { get; set; } = null!;
    }
}

```

```

    public string Phone { get; set; } = null!;

    public string CityName { get; set; } = null!;
    public string WarehouseName { get; set; } = null!;

    public string? RepayUrl { get; set; }

    public PaymentStatus PaymentStatus { get; set; } = PaymentStatus.Unpaid;
    public ExecutionStatus ExecutionStatus { get; set; } = ExecutionStatus.Pending;

    public ICollection<OrderItem> Items { get; set; } = null!;
}
}

```

Додаток Г.

Інтеграції з зовнішніми сервісами

nova-poshta.service.ts

```

interface NPResponse<T> {
    success: boolean;
    data: T[];
    errors: string[];
}

export interface NPCity {
    Ref: string;
    Description: string
}

export interface NPWarehouse {
    Ref: string;
    Description: string
}

@Injectable({ providedIn: 'root' })
export class NovaPoshtaService {
    private http = new HttpClient(inject(HttpBackend));
    private apiKey = environment.npApiKey;
    private url = 'https://api.novaposhta.ua/v2.0/json/';

    getCities(findByString: string = ''): Observable<NPCity[]> {
        return this.http.post<NPResponse<NPCity>>(this.url, {

```

```

        apiKey: this.apiKey,
        modelName: 'Address',
        calledMethod: 'getCities',
        methodProperties: { FindByString: findByString, Limit: "50" }
    }).pipe(map(r => r.data));
}

getWarehouses(cityRef: string, findByString: string = ""): Observable<NPWarehouse[]> {
    return this.http.post<NPResponse<NPWarehouse>>(this.url, {
        apiKey: this.apiKey,
        modelName: 'AddressGeneral',
        calledMethod: 'getWarehouses',
        methodProperties: { CityRef: cityRef, FindByString: findByString }
    }).pipe(map(r => r.data));
}
}

```

EmailService.cs

```

namespace MobileShopApi.Services
{
    public class EmailService
    {
        private readonly IConfiguration _config;

        public EmailService(IConfiguration configuration)
        {
            _config = configuration;
        }

        public async Task SendEmailAsync(string to, string subject, string htmlBody)
        {
            var smtpHost = _config["EmailSettings:SmtpHost"]!;
            var smtpPort = int.Parse(_config["EmailSettings:SmtpPort"]!);
            var useSsl = bool.Parse(_config["EmailSettings:UseSsl"]!);
            var username = _config["EmailSettings:Username"]!;
            var password = _config["EmailSettings:Password"]!;
            var fromName = _config["EmailSettings:FromName"]!;
            var fromAddress = _config["EmailSettings:FromAddress"]!;

            var message = new MimeMessage();

```

```
message.From.Add(new MailboxAddress(fromName, fromAddress));  
message.To.Add(MailboxAddress.Parse(to));  
message.Subject = subject;
```

```
var builder = new BodyBuilder { HtmlBody = htmlBody };  
message.Body = builder.ToMessageBody();
```

```
using var smtp = new SmtpClient();  
await smtp.ConnectAsync(smtpHost, smtpPort,  
    useSsl ? SecureSocketOptions.SslOnConnect : SecureSocketOptions.StartTls);  
await smtp.AuthenticateAsync(username, password);  
await smtp.SendAsync(message);  
await smtp.DisconnectAsync(true);  
}  
  
}  
  
}
```