

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ

«ІНТЕРНЕТ-МАГАЗИН КОМП'ЮТЕРНИХ

КОМПЛЕКТУЮЧИХ»

DESIGN OF AN ONLINE STORE OF COMPUTER

COMPONENTS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-222

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ПЕТРОВ О.І.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК.Т.В

(ПРИЗВИЩЕ та ініціали)

Рецензент БАБЕНКО Н.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПЕТРОВА Олексія Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення програмної системи «Інтернет-магазин комп'ютерних комплектуючих». Design of an Online Store of Computer Components.

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року
№ 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмної системи. 3. Реалізація програмної системи. 4. Тестування та оцінка ефективності програмної системи.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ФЕДОРОНЧАК Т.В., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олексій ПЕТРОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ФЕДОРОНЧАК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
80 с., 5 табл., 42 рис., 2 дод., 14 джерел.

ІНТЕРНЕТ-МАГАЗИН, ВЕБСАЙТ, КОМП'ЮТЕРНІ
КОМПЛЕКТУЮЧІ, ІНТЕРНЕТ-МАГАЗИН КОМП'ЮТЕРНИХ
КОМПЛЕКТУЮЧИХ, ОНЛАЙН МАГАЗИН

Об'єкт дослідження – процес побудови програмних систем електронної комерції для обробки замовлень користувачів.

Предмет дослідження – методи і програмні засоби для розробки інтернет-магазинів.

Мета роботи – зменшення часу та підвищення зручності пошуку комп'ютерних комплектуючих користувачами в мережі інтернет, а також пришвидшення обробки замовлень адміністраторами шляхом розробки вебзастосунку інтернет магазину.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, технології розробки клієнт-серверних вебзастосунків, платформа Node.js, бібліотека React, операційна система Microsoft Windows 11.

Результати. Створено програмну систему інтернет-магазину комп'ютерних комплектуючих, реалізований як клієнт-серверний вебзастосунок, що забезпечує взаємодію користувача з каталогом товарів, пошуку, фільтрації та управління даними через зручний інтерфейс.

Висновки. Розроблено програмну систему інтернет-магазину комп'ютерних комплектуючих, яка зменшує час обробки запитів користувачів та пошуку товарів за рахунок застосування оптимізованих алгоритмів обробки даних і ефективної організації клієнт-серверної взаємодії.

Галузь використання – онлайн-продаж комп'ютерних комплектуючих для широкого кола користувачів через вебінтерфейс.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 5 tables, 42 figures, 2 appendixes, 14 sources.

ONLINE STORE, WEBSITE, COMPUTER PARTS, ONLINE STORE FOR COMPUTER PARTS, ONLINE SHOP

Object of study is e-commerce software systems for processing data on computer components.

Subject of study is software tools for processing user queries and managing the product catalogue in an online shop.

The purpose of the work is to reduce the time taken to process user requests and search for products in the online shop system for computer components by applying effective data processing algorithms and optimising the information storage structure.

Materials, methods, and tools: object-oriented programming, client-server web application development technologies, the Node.js platform, the React library, the JavaScript programming language, the Microsoft Windows 11 operating system.

Results. A software system for an online shop selling computer components has been created, implemented as a client-server web application that enables user interaction with the product catalogue, search, filtering and data management via a user-friendly interface.

Conclusions A software system for an online shop selling computer components has been developed, which reduces the time taken to process user requests and search for products through the use of optimised data processing algorithms and the effective organisation of client-server interaction.

The field of use is a online sales of computer components to a wide range of users via a web interface.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Дослідження актуальності проблеми	9
1.2 Аналіз існуючих інтернет-магазинів комп'ютерних комплектуючих	10
1.3 Вимоги до програмної системи	15
1.4 Постановка задачі.....	17
2 Проєктування програмної системи	19
2.1 Проєктування структури програмної системи	19
2.2 Проєктування бази даних	20
2.3 Проєктування функціоналу системи.....	23
2.4 Проєктування інтерфейсу користувача	24
2.5 Вибір технології розробки.....	27
3 Реалізація програмної системи	29
3.1 Реалізація клієнтської частини системи	29
3.2 Реалізація серверної частини та взаємодія з базою даних	32
3.3 Алгоритм роботи програмної системи.....	46
4 Тестування та оцінка ефективності програмної системи.....	51
4.1 Методика тестування програмної системи.....	51
4.2 Тестування верстки програмної системи.....	52
4.3 Функціональне тестування системи.....	54
4.4 Тестування адаптивності вебзастосунку	55
4.5 Аналіз результатів тестування	57
Висновки	59
Перелік джерел посилання	61
Додаток А Текст програми.....	63
Додаток Б Слайди презентації	75

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CLI	– Command Line Interface;
CSS	– Cascading Style Sheets;
DB	– Database;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
IDE	– Integrated Development Environment;
JS	– JavaScript;
JSON	– JavaScript Object Notation;
REST	– Representational State Transfer;
SPA	– Single Page Application;
UI	– User Interface;
UX	– User Experience;
БД	– база даних;
ID	– ідентифікатор;
ІС	– інформаційна система;
ПЗ	– програмне забезпечення;
СУБД	– система управління базами даних.

ВСТУП

У сучасному світі стрімкий розвиток інформаційних технологій та інтернету суттєво змінив підходи до ведення бізнесу. Електронна комерція стала одним із ключових напрямів економіки, забезпечуючи швидкий доступ до товарів і послуг, а також значно спрощуючи процес купівлі для користувачів. Інтернет-магазини набули широкого поширення завдяки зручності використання, великому асортименту продукції та можливості порівняння товарів у режимі реального часу.

Яскравими прикладами успішних інтернет-магазинів є Rozetka, Amazon та eBay. Зокрема, Rozetka є одним із найбільших онлайн-ритейлерів в Україні, що починав як магазин комп'ютерної техніки та електроніки і з часом перетворився на універсальний маркетплейс із мільйонами товарів і мільйонами щоденних відвідувачів [1]. Успіх подібних платформ обумовлений впровадженням сучасних технологій, зручним інтерфейсом, ефективною логістикою та орієнтацією на потреби користувачів.

Актуальність даного дипломного проєкту полягає у необхідності розробки сучасного, зручного та функціонального вебресурсу, який забезпечить ефективний процес купівлі комп'ютерних комплектуючих.

Метою дипломного проєкту є розробка інтернет-магазину комп'ютерних комплектуючих, який забезпечить користувачам зручний пошук, вибір та придбання товарів. Реалізація такого проєкту передбачає створення інтуїтивно зрозумілого інтерфейсу, ефективної системи навігації та фільтрації товарів.

Практичне значення роботи полягає у можливості використання розробленого інтернет-магазину як повноцінного комерційного продукту або як основи для подальшого вдосконалення та масштабування в умовах сучасного ринку електронної комерції.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

У сучасному світі інтернет-магазини стали одним із ключових інструментів ведення бізнесу. Вони дозволяють здійснювати продаж товарів у режимі онлайн, забезпечуючи доступ до широкого асортименту продукції, інформації про неї, а також можливість оформлення замовлення у будь-який час. Інтернет-магазин визначається як вебсайт, на якому користувач може ознайомитися з товарами, їх характеристиками, цінами та оформити покупку [2].

Стрімкий розвиток цифрових технологій та глобальна інформатизація суспільства суттєво змінили підходи до організації торгівлі. Традиційні форми продажу дедалі частіше доповнюються або навіть повністю замінюються електронною комерцією, яка забезпечує більшу гнучкість, доступність та швидкість обслуговування. Для бізнесу це відкриває можливість розширення цільової аудиторії без територіальних обмежень, а для споживачів – комфортний доступ до товарів незалежно від місця перебування.

Актуальність створення інтернет-магазину комп'ютерних комплектуючих обумовлена постійним зростанням попиту на комп'ютерну техніку. Це пов'язано з розвитком ІТ-сфери, дистанційної роботи та навчання, а також популяризацією ігрової індустрії. Користувачі дедалі частіше обирають онлайн-покупки через їх зручність та економію часу.

Окрему роль у цьому процесі відіграє специфіка комп'ютерних комплектуючих як категорії товарів. На відміну від багатьох інших груп продукції, при виборі таких товарів покупець звертає увагу не лише на ціну чи бренд, а й на технічні характеристики, продуктивність, сумісність із іншими компонентами та перспективу подальшого оновлення системи. Саме тому для даної сфери необхідний інтернет-магазин, який не лише продає товари, а й допомагає користувачу зробити обґрунтований вибір.

Важливим фактором є також те, що сучасні інтернет-магазини повинні забезпечувати не лише продаж товарів, а й зручний пошук, фільтрацію та структуровану подачу інформації. Як зазначається у матеріалах щодо створення e-commerce платформ, правильна структура каталогу та ефективні фільтри суттєво впливають на прийняття рішення про покупку [3].

Крім цього, значну роль відіграє якість користувацького інтерфейсу. Інтуїтивно зрозумілий дизайн, швидке завантаження сторінок, адаптивність до мобільних пристроїв та логічна навігація є невід'ємними складовими успішного вебресурсу. Якщо користувач витрачає занадто багато часу на пошук потрібного товару або стикається зі складним процесом оформлення замовлення, це може призвести до втрати потенційного клієнта.

Сучасний ринок електронної комерції вимагає від розробників створення платформ, які поєднують функціональність, надійність та зручність використання. Особливо це актуально для спеціалізованих магазинів, де покупець очікує не лише доступу до каталогу, а й додаткових інструментів для аналізу товарів, порівняння моделей та отримання консультаційної інформації.

Таким чином, створення спеціалізованого інтернет-магазину комп'ютерних комплектуючих є актуальним завданням, оскільки дозволяє вирішити проблему зручного підбору товарів, їх сумісності та спрощення процесу покупки для користувача. Реалізація такого проєкту сприятиме підвищенню якості обслуговування клієнтів, забезпечить конкурентоспроможність на ринку електронної комерції та відповідатиме сучасним тенденціям розвитку цифрового бізнесу.

1.2 Аналіз існуючих інтернет-магазинів комп'ютерних комплектуючих

У сучасних умовах стрімкого розвитку електронної комерції інтернет-магазини стали одним із ключових каналів реалізації технічної продукції та комп'ютерних комплектуючих. Зростання попиту на онлайн-покупки зумовлює

необхідність створення зручних, функціональних і ефективних вебресурсів, здатних забезпечити користувачеві швидкий доступ до потрібної інформації та комфортний процес вибору товару. Особливо актуальним є питання організації інтернет-магазинів у сфері комп'ютерної техніки, де покупець потребує не лише стандартного каталогу, а й інструментів для аналізу характеристик, порівняння товарів та перевірки їх сумісності. Саме тому важливим етапом у розробці власного програмного продукту є дослідження існуючих рішень, що дозволяє визначити їх сильні та слабкі сторони, а також сформулювати вимоги до майбутньої системи. Для цього було проведено аналіз популярних інтернет-магазинів електроніки та комп'ютерної техніки, які працюють на українському ринку.

Для аналізу було обрано такі інтернет-магазини:

- TTT.ua;
- Exe.ua;
- Peshka.com.ua.

TTT.ua – інтернет-магазин електроніки, що пропонує широкий асортимент техніки: смартфони, ноутбуки, аксесуари, комп'ютерну техніку та інші товари. Компанія також співпрацює з відомими брендами та виступає офіційним дистриб'ютором деяких виробників. Сайт має розгалужену структуру каталогу з великою кількістю категорій. Головне меню сайту зображено на рис. 1.1 [4].

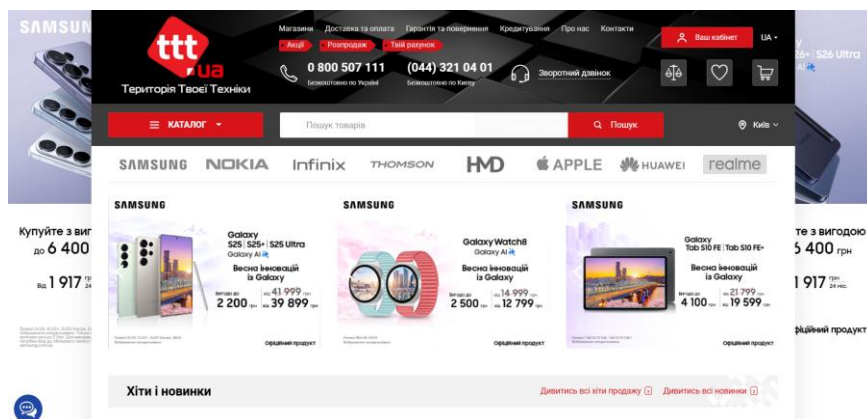


Рисунок 1.1 – Головне меню сайту TTT.ua [4]

Ехе.ua – спеціалізований інтернет-магазин комп'ютерної техніки та комплектуючих, який працює понад 20 років і пропонує понад 50 тисяч товарів. Магазин забезпечує можливість онлайн-замовлення, різні способи оплати, швидку доставку та наявність системи відгуків для користувачів. Головне меню сайту зображено на рис. 1.2 [5].

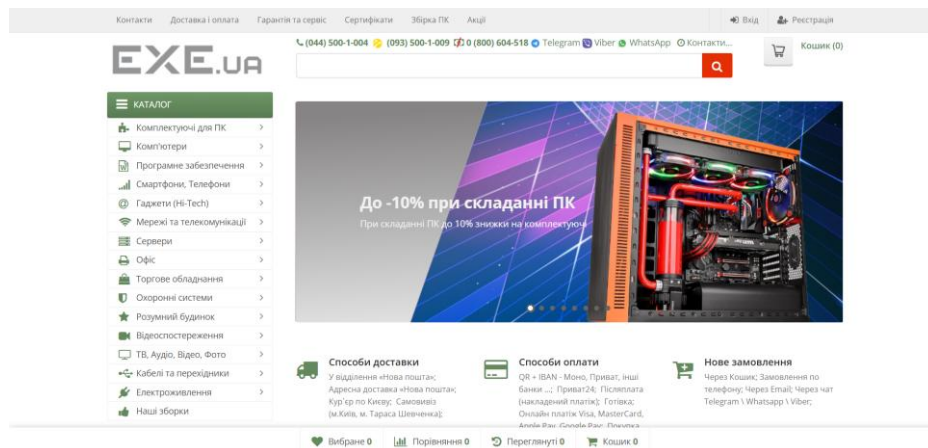


Рисунок 1.2 – Головне меню сайту Ехе.ua [5]

Peshka.com.ua – інтернет-магазин електроніки та комплектуючих, орієнтований на роздрібний продаж техніки. Основною особливістю подібних магазинів є наявність каталогу товарів, системи пошуку, фільтрів та описів продукції, що дозволяє користувачам швидко знайти потрібний товар та здійснити покупку онлайн. Головне меню сайту наведено на рис. 1.3 [6].

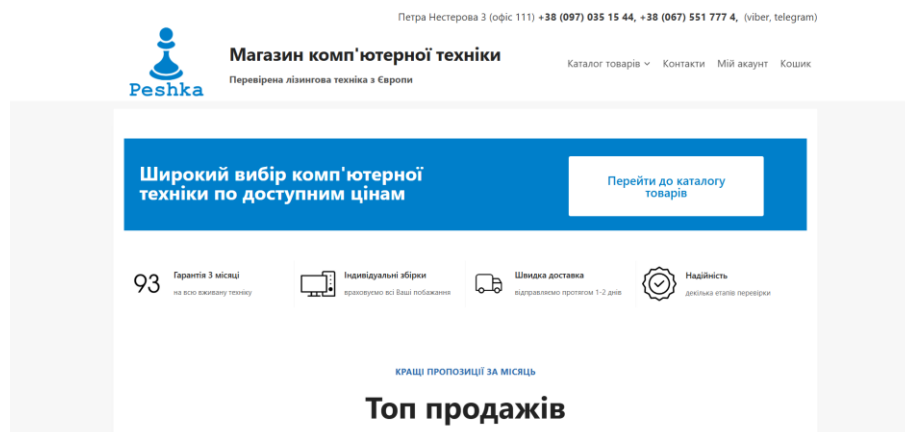


Рисунок 1.3 – Головне меню сайту Peshka.com.ua [6]

Для більш ґрунтового оцінювання особливостей розглянутих інтернет-магазинів доцільно здійснити їх порівняння за низкою ключових критеріїв, які безпосередньо впливають на ефективність роботи вебресурсу та зручність взаємодії користувача із системою. Зокрема, увага приділяється таким аспектам, як зручність використання, функціональні можливості, дизайн інтерфейсу та швидкодія сайту. Саме ці характеристики дозволяють визначити рівень конкурентоспроможності кожного магазину, а також виявити переваги й недоліки існуючих рішень. Узагальнені результати проведеного аналізу наведено в табл.1.1, де представлено порівняльну характеристику досліджуваних інтернет-магазинів.

Таблиця 1.1 – Порівняльна характеристика інтернет-магазинів

Критерій	ТТТ.ua	Ехе.ua	Peshka.com.ua
Зручність використання	Висока – зручне меню, багато категорій, але іноді перевантажений інтерфейс	Середня – більш простий інтерфейс, орієнтований на функціональність	Середня – стандартна структура каталогу
Функціонал	Широкий (фільтри, категорії, акції, бренди)	Розширений (відгуки, акції, різні способи оплати, великий каталог)	Базовий (каталог, пошук, оформлення замовлення)
Дизайн	Сучасний, але перевантажений елементами	Простий, менш сучасний	Простий, мінімалістичний
Швидкодія	Середня (через велику кількість елементів)	Висока (простий інтерфейс)	Середня

Проведений аналіз існуючих інтернет-магазинів електроніки та комп'ютерної техніки показав, що більшість із них мають достатньо широкий функціонал, значний асортимент товарів та різноманітні сервіси для покупців. До таких можливостей належать системи пошуку, фільтрації, сортування, акційні пропозиції, різні способи оплати та доставки. Проте, незважаючи на широкий спектр функцій, не всі досліджені ресурси орієнтовані саме на зручний та ефективний підбір комп'ютерних комплектуючих. У більшості випадків користувач змушений самотійно аналізувати технічні характеристики товарів, порівнювати їх між собою та перевіряти сумісність компонентів, що ускладнює процес вибору та збільшує час на прийняття рішення.

Серед основних недоліків проаналізованих платформ можна виокремити перевантаженість інтерфейсу великою кількістю елементів, недостатню спеціалізацію на комплектуючих, а також обмежені можливості для автоматизованої перевірки сумісності деталей. Для користувачів, які прагнуть самотійно зібрати персональний комп'ютер або оновити окремі компоненти, такі обмеження створюють додаткові труднощі та підвищують ризик помилкового вибору.

Отже, розробка власного інтернет-магазину, який буде орієнтований саме на підбір комп'ютерних комплектуючих, є доцільною та актуальною в сучасних умовах розвитку електронної комерції. Такий вебресурс повинен забезпечувати не лише широкий каталог продукції, а й зручний, інтуїтивно зрозумілий інтерфейс, адаптований до потреб користувачів. Крім того, важливо реалізувати покращений функціонал, зокрема інструменти для перевірки сумісності компонентів, порівняння характеристик та формування оптимальних конфігурацій.

Створення подібної системи дозволить підвищити якість обслуговування клієнтів, спростити процес вибору техніки та забезпечити більш ефективну взаємодію між покупцем і продавцем. Саме це стане конкурентною перевагою майбутнього проєкту та визначить його практичну цінність у сфері продажу комп'ютерних комплектуючих.

1.3 Вимоги до програмної системи

Під час розробки інтернет-магазину комп'ютерних комплектуючих важливо чітко визначити основні вимоги до програмної системи, оскільки саме вони формують базу для її подальшого проєктування, реалізації та тестування. Наявність детально сформульованих вимог дозволяє забезпечити ефективність роботи системи, зручність її використання для кінцевих користувачів, а також відповідність сучасним тенденціям у сфері електронної комерції. Крім того, правильно визначені вимоги сприяють підвищенню конкурентоспроможності майбутнього програмного продукту, оскільки дозволяють врахувати як технічні, так і користувацькі аспекти функціонування вебресурсу. Усі вимоги до системи доцільно поділити на дві основні групи: функціональні та нефункціональні.

До функціональних вимог належать можливості, які система повинна реалізовувати для забезпечення повноцінної та безперебійної роботи інтернет-магазину. Насамперед, це наявність структурованого каталогу товарів із детальним описом, технічними характеристиками, якісними зображеннями та актуальними цінами. Такий каталог повинен бути логічно організованим за категоріями та підкатегоріями, що дозволить користувачам швидко знаходити потрібну продукцію. Важливою вимогою є також реалізація пошуку товарів за назвою, артикулом або ключовими словами, а також системи фільтрації за категоріями, брендами, ціновим діапазоном, технічними параметрами та іншими характеристиками.

Окрему увагу слід приділити можливості порівняння товарів, що є особливо актуальним для сфери комп'ютерних комплектуючих, де користувачі часто аналізують декілька моделей перед здійсненням покупки. Також система повинна підтримувати формування кошика, оформлення замовлення, вибір способу оплати та доставки. Для авторизованих користувачів доцільно передбачити особистий кабінет із можливістю перегляду історії покупок, збереження обраних товарів та відстеження статусу замовлень. Не менш

важливою є наявність адміністративної панелі для управління каталогом, обробки замовлень, оновлення інформації про товари та контролю за роботою ресурсу.

До нефункціональних вимог відносяться характеристики, що визначають якість програмної системи та її відповідність сучасним стандартам розробки. Насамперед, це зручність інтерфейсу та адаптивність дизайну для коректного відображення на різних пристроях – комп'ютерах, планшетах і смартфонах. Інтерфейс має бути інтуїтивно зрозумілим, логічно структурованим і не перевантаженим зайвими елементами, що сприятиме позитивному користувацькому досвіду.

Крім того, важливими нефункціональними вимогами є швидкодія та стабільність системи. Завантаження сторінок повинно відбуватися без затримок навіть за великої кількості товарів у каталозі. Система має забезпечувати безпечне зберігання персональних даних користувачів, захист від несанкціонованого доступу та відповідати базовим вимогам кібербезпеки. Надійність роботи ресурсу є критично важливою, оскільки будь-які технічні збої можуть призвести до втрати клієнтів та зниження довіри до магазину.

Ще однією важливою характеристикою є масштабованість програмної системи. Це означає, що у майбутньому її функціональні можливості повинні легко розширюватися без суттєвих змін у базовій архітектурі. Такий підхід дозволить адаптувати інтернет-магазин до нових потреб ринку, інтегрувати додаткові сервіси, збільшувати кількість користувачів та розширювати асортимент товарів.

Отже, дотримання зазначених функціональних і нефункціональних вимог є необхідною умовою для створення якісного, надійного та конкурентоспроможного інтернет-магазину комп'ютерних комплектуючих. Саме комплексний підхід до визначення вимог дозволить розробити сучасну програмну систему, яка відповідатиме очікуванням користувачів, забезпечить ефективне управління бізнес-процесами та відповідатиме актуальним стандартам веброзробки.

1.4 Постановка задачі

На основі проведеного аналізу предметної області, дослідження існуючих рішень та визначення ключових вимог до програмної системи було сформульовано основну задачу дипломного проєкту. Вона полягає у розробці сучасного інтернет-магазину комп'ютерних комплектуючих, який забезпечить користувачам зручний пошук, вибір та придбання товарів, а також надасть ефективні інструменти для адміністрування каталогу продукції. Особливістю такого вебресурсу має стати орієнтація саме на сферу комп'ютерної техніки, де важливими є деталізовані характеристики товарів, можливість порівняння моделей та зручна навігація між великою кількістю категорій.

Актуальність поставленої задачі зумовлена постійним зростанням попиту на онлайн-продаж технічної продукції, а також необхідністю створення спеціалізованих платформ, які враховують специфіку вибору комп'ютерних комплектуючих. На відміну від універсальних торговельних майданчиків, розроблюваний інтернет-магазин має забезпечити не лише базовий функціонал електронної комерції, а й підвищений рівень зручності для користувачів, які потребують технічної інформації та швидкого доступу до неї.

Для досягнення поставленої мети необхідно вирішити низку взаємопов'язаних задач, кожна з яких спрямована на забезпечення повноцінної реалізації програмного продукту:

- провести аналіз сучасного ринку електронної комерції у сфері комп'ютерних комплектуючих та визначити актуальні тенденції розвитку цієї галузі;
- дослідити функціональні можливості існуючих інтернет-магазинів з метою виявлення їх сильних і слабких сторін;
- сформулювати перелік функціональних та нефункціональних вимог до майбутньої програмної системи;
- спроектувати структуру вебзастосунку, логіку взаємодії його модулів та архітектуру бази даних;

- реалізувати користувацьку частину інтернет-магазину з каталогом товарів, системою пошуку, фільтрації та оформлення замовлення;
- створити адміністративну панель для керування товарами, категоріями, замовленнями та іншими елементами системи;
- забезпечити адаптивність інтерфейсу та зручність використання на різних типах пристроїв;
- провести тестування системи, оцінити її працездатність, ефективність та відповідність поставленим вимогам.

Результатом виконання дипломного проєкту має стати повноцінний вебзастосунок, орієнтований на продаж комп'ютерних комплектуючих, який відповідатиме сучасним вимогам до електронної комерції, стандартам веброзробки та принципам зручного користувацького досвіду. Реалізація такого проєкту сприятиме підвищенню ефективності онлайн-продажу технічної продукції та стане прикладом практичного застосування сучасних технологій у сфері електронного бізнесу.

2 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

Проектування програмної системи є важливим етапом розробки, оскільки дозволяє заздалегідь передбачити логіку роботи застосунку, оптимізувати архітектуру та забезпечити ефективність подальшої реалізації. Для інтернет-магазину комп'ютерних комплектуючих це особливо актуально, оскільки система повинна поєднувати зручність для користувача, швидкодію, масштабованість і надійність.

2.1 Проектування структури програмної системи

Для реалізації інтернет-магазину було обрано клієнт-серверну архітектуру, яка передбачає розподіл системи на клієнтську частину, серверну логіку та базу даних. Такий підхід забезпечує гнучкість, масштабованість і зручність супроводу системи. На рис. 2.1 представлена структурна схема сайту.



Рисунок 2.1 – Структурна схема сайту

Побудована структурна схема відображає логічну організацію інтернет-магазину та взаємозв'язок між основними функціональними модулями системи. Центральним елементом структури є головна сторінка, яка виступає основною точкою входу для користувача та забезпечує доступ до всіх ключових розділів вебзастосунку.

Каталог товарів, містить перелік доступних комп'ютерних комплектуючих, згрупованих за відповідними категоріями. Для детальнішого

ознайомлення з характеристиками передбачено перехід на сторінку товару, де розміщується повний опис, технічні параметри, ціна та можливість додавання до кошика.

Модуль пошуку товарів забезпечує швидке знаходження необхідної продукції за ключовими словами, що значно підвищує зручність користування сайтом.

Система фільтрів дозволяє сортувати продукцію за категоріями, брендами, ціною, технічними характеристиками та іншими параметрами.

Кошик призначений для збереження обраних товарів. У цьому модулі користувач може переглядати перелік доданих позицій, змінювати їх кількість або видаляти непотрібні товари.

Електронний кабінет забезпечує персоналізований доступ до інформації користувача.

2.2 Проєктування бази даних

База даних є одним із ключових компонентів будь-якої програмної системи, оскільки саме вона забезпечує структуроване зберігання, обробку та доступ до інформації. У вебзастосунках електронної комерції роль бази даних є особливо важливою, адже саме через неї реалізується зберігання інформації про товари, користувачів, замовлення, кошики, відгуки та інші сутності, що формують основу функціонування системи. Коректно спроектована база даних забезпечує не лише стабільну роботу програмного продукту, а й можливість його подальшого масштабування, оптимізації та інтеграції з іншими сервісами.

Під час розробки даного проєкту було обрано систему керування базами даних PostgreSQL, яка є однією з найпоширеніших реляційних СКБД у сучасній веброзробці. Вибір саме цієї технології обумовлений її надійністю, високою продуктивністю, підтримкою складних SQL-запитів, масштабованістю та широкими можливостями для забезпечення цілісності даних. Крім того,

PostgreSQL є відкритим програмним забезпеченням, що робить її доступною для використання у навчальних та комерційних проєктах.

Важливою перевагою обраної СКБД є підтримка реляційної моделі даних, що дозволяє ефективно організовувати зв'язки між сутностями та забезпечувати логічну структуру збереження інформації. Це особливо актуально для інтернет-магазину, де існує велика кількість взаємопов'язаних об'єктів: товари пов'язані з категоріями та брендами, користувачі – із кошиками та оцінками, а система фільтрації – з характеристиками товарів. На рис. 2.2 представлено діаграму бази даних інтернет-магазину комп'ютерних комплектуючих.

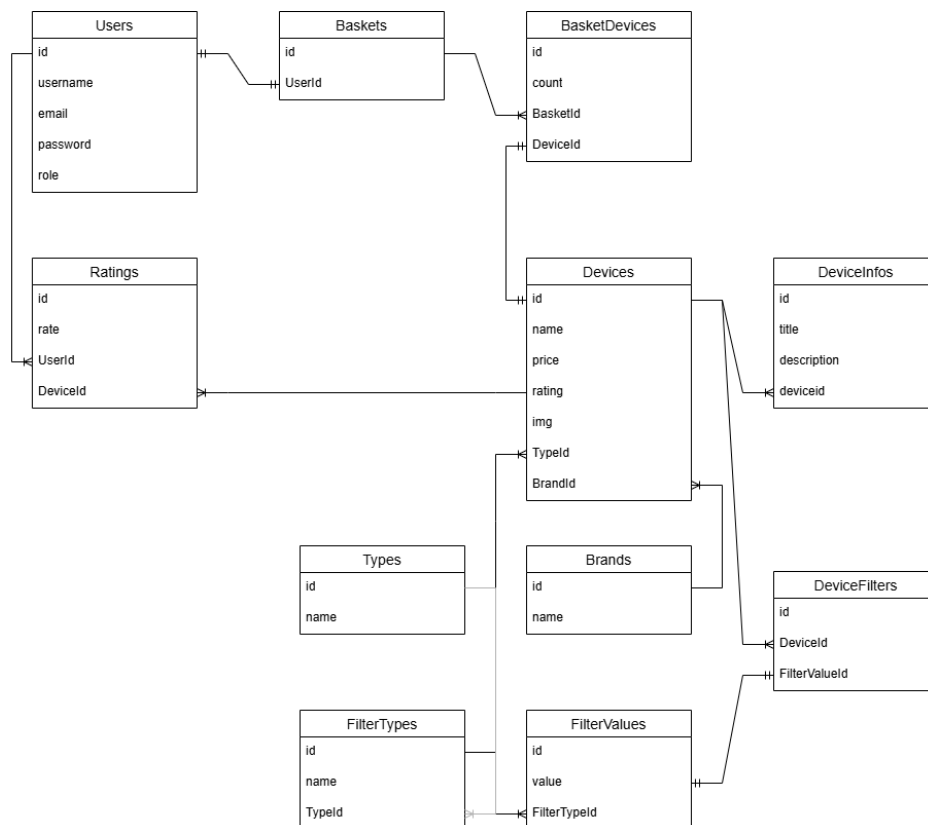


Рисунок 2.2 – Діаграма бази даних інтернет-магазину комп'ютерних комплектуючих

Дана БД побудована за реляційним принципом та містить основні сутності, необхідні для реалізації функціоналу каталогу товарів, кошика, оцінювання продукції та системи фільтрації. Структура бази даних була

розроблена таким чином, щоб забезпечити зручність роботи із великими обсягами інформації, мінімізувати дублювання даних та підтримувати цілісність зв'язків між таблицями.

Devices – таблиця, яка зберігає основну інформацію про товари. До її атрибутів належать назва товару, ціна, рейтинг, зображення, а також зовнішні ключі на таблиці Types та Brands. Це дозволяє класифікувати товари за типами та виробниками. Завдяки такій структурі забезпечується можливість швидкого пошуку продукції за категоріями та формування логічного каталогу товарів.

DeviceInfos – містить додаткові параметри пристроїв у форматі назви характеристики та її значення. Дана таблиця реалізує гнучкий механізм збереження технічних характеристик, що особливо важливо для комп'ютерних комплектуючих, де кожен тип товару має специфічний набір параметрів. Такий підхід дозволяє зберігати необмежену кількість характеристик без зміни структури основної таблиці товарів.

Types / Brands – містять перелік категорій товарів та брендів. Така структура забезпечує зручне групування товарів та спрощує реалізацію пошуку, сортування і навігації по каталогу. Розділення категорій і брендів в окремі таблиці також відповідає принципам нормалізації бази даних та дозволяє уникнути дублювання інформації.

Механізм фільтрації реалізовано через таблиці FilterTypes, FilterValues, DeviceFilters:

- FilterTypes визначає тип фільтра та прив'язується до певного типу товару;
- FilterValues містить конкретні значення для кожного фільтра;
- DeviceFilters створює зв'язок між товарами та значеннями фільтрів.

Users – таблиця для роботи з користувачами. Вона містить облікові дані: ім'я користувача, електронну пошту, пароль та роль. Наявність ролей дозволяє розмежувати права доступу між звичайними користувачами та адміністраторами системи. Це є важливим елементом безпеки та забезпечує контроль за виконанням адміністративних функцій.

Кошик реалізовано за допомогою таблиць Basket та BasketDevices:

- Baskets зберігає інформацію про кошик конкретного користувача;
- BasketDevices реалізує зв'язок між кошиком і товарами, а також містить кількість одиниць кожного товару.

Така реалізація дозволяє підтримувати функціонал додавання, оновлення та видалення товарів у кошику, а також забезпечує гнучкість при роботі з декількома позиціями одночасно.

Ratings – таблиця для реалізації системи оцінювання товару. Вона дозволяє користувачам залишати оцінки товарам, що сприяє формуванню рейтингу продукції та підвищує довіру інших покупців до магазину. Реалізація такої функції також дозволяє аналізувати популярність товарів та використовувати ці дані для подальшого вдосконалення каталогу.

Таким чином, структура бази даних інтернет-магазину комп'ютерних комплектуючих є логічно побудованою, нормалізованою та адаптованою до потреб вебзастосунку. Вона забезпечує ефективне зберігання інформації, підтримує основні бізнес-процеси системи та створює надійну основу для подальшого розширення функціональних можливостей програмного продукту.

2.3 Проектування функціоналу системи

Функціонал інтернет-магазину комп'ютерних комплектуючих був спроектований з урахуванням потреб основних категорій користувачів – покупців та адміністраторів. Основною метою є забезпечення зручного процесу пошуку, вибору та придбання товарів, а також ефективного керування асортиментом продукції. На рисунку 2.3 представлена Use Case діаграма, яка відображає варіанти використання сайту для користувача, покупця та адміністратора.

Для звичайного користувача система надає можливість перегляду каталогу товарів, використання пошуку та фільтрації за різними

характеристиками. Це дозволяє швидко знаходити необхідні комплектуючі відповідно до заданих параметрів.

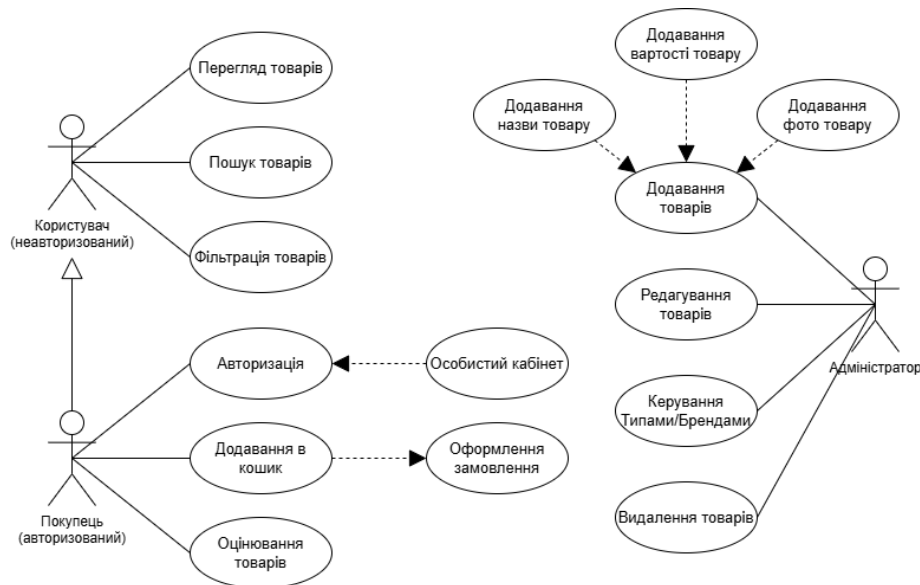


Рисунок 2.3 – Use Case діаграма

Для зручності користувачів передбачено функціонал реєстрації та авторизації, що дозволяє отримати доступ до особистого кабінету. Це надає звичайного користувача можливості оцінювати товар, додавати його в кошик та оформлювати замовлення.

Для адміністратора реалізовано функціонал управління товарами, категоріями, брендами та характеристиками. Це забезпечує можливість актуалізації каталогу, редагування інформації та підтримання коректності даних у системі.

2.4 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача є одним із ключових етапів створення сучасного інтернет-магазину, оскільки саме від якості взаємодії користувача із системою залежить зручність користування, швидкість пошуку товарів та рівень задоволеності клієнтів. У сфері електронної комерції інтерфейс має не лише привабливий вигляд, а й забезпечувати інтуїтивно

зрозумілу навігацію, швидкий доступ до інформації та безперервний процес оформлення покупки [7].

Згідно з сучасними принципами UI/UX-дизайну для e-commerce платформ, важливими складовими якісного інтерфейсу є простота, логічність, послідовність оформлення та адаптивність. Простий і зрозумілий інтерфейс дозволяє користувачеві швидко орієнтуватися в системі, а єдина стилістика всіх сторінок формує відчуття цілісності та професійності вебресурсу. Крім того, важливою є доступність інтерфейсу для різних категорій користувачів, зокрема з урахуванням стандартів доступності та зручності взаємодії з елементами сайту [8].

Під час проектування інтерфейсу було спроектовано декілька основних сторінок системи: головну сторінку, каталог товарів, сторінку окремого товару, кошик, особистий кабінет користувача та адміністративну панель. Кожна сторінка виконує окрему функцію та має чітко визначену структуру. На рис. 2.4 – 2.7 представлено макети інтерфейсу інтернет-магазину комп'ютерних комплектуючих.

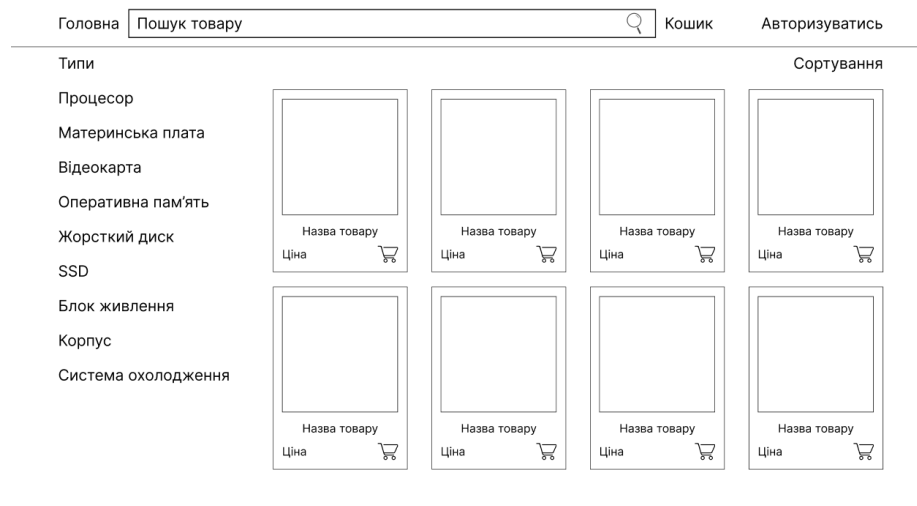


Рисунок 2.4 – Макет головної сторінки сайту

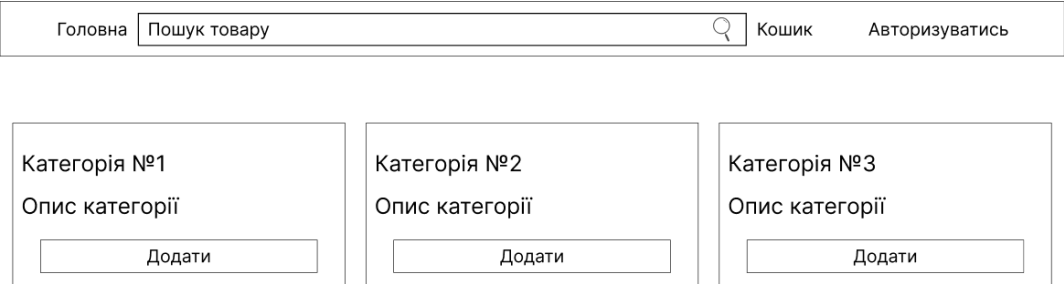


Рисунок 2.5 – Макет адміністративної панелі

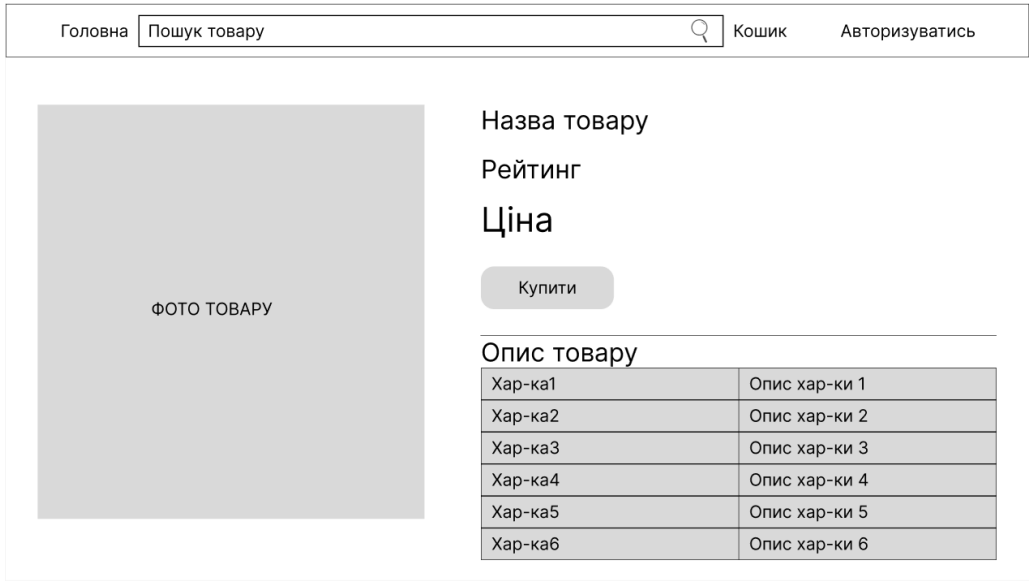


Рисунок 2.6 – Макет сторінки опису товару

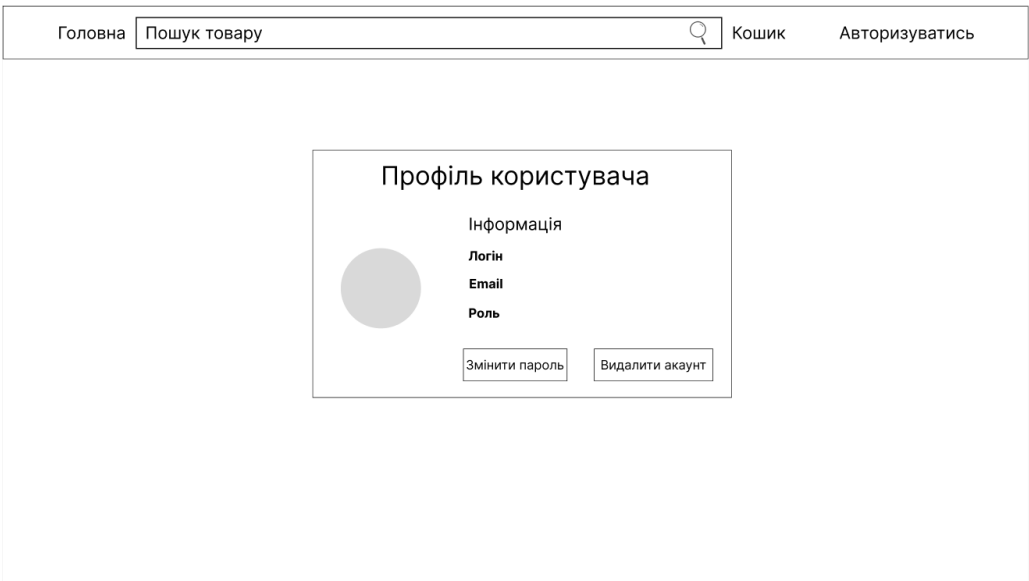


Рисунок 2.7 – Макет сторінки профілю користувача

У результаті було сформовано інтерфейс, який поєднує функціональність, простоту та орієнтацію на потреби кінцевого користувача. Це дозволяє підвищити ефективність взаємодії із системою та створити позитивний користувацький досвід.

2.5 Вибір технології розробки

Вибір технологічного стеку є одним із ключових етапів проєктування програмної системи, оскільки саме від нього залежить продуктивність, масштабованість, зручність підтримки та подальший розвиток вебзастосунку. Для створення інтернет-магазину комп'ютерних комплектуючих доцільно використовувати сучасні технології, які забезпечують швидку обробку запитів, зручну роботу з базою даних та якісний користувацький інтерфейс.

Для клієнтської частини системи було обрано React – популярну JavaScript-бібліотеку для створення динамічних користувацьких інтерфейсів. Її головною перевагою є компонентний підхід, який дозволяє розбивати інтерфейс на незалежні багаторазові модулі. Це спрощує підтримку коду, пришвидшує розробку та забезпечує масштабованість проєкту. React також підтримує одно сторінкові застосунки (SPA), що позитивно впливає на швидкодію та зручність взаємодії користувача із системою [9].

Серверна частина реалізується на основі Node.js. Такий підхід є особливо актуальним для інтернет-магазинів, де сервер постійно взаємодіє з базою даних, платіжними системами та користувачами. Перевагою також є можливість використання JavaScript як на клієнтській, так і на серверній стороні, що спрощує розробку та скорочує витрати часу [10].

Для зберігання даних було обрано PostgreSQL – потужну реляційну систему керування базами даних із відкритим вихідним кодом. Вона добре підходить для систем електронної комерції, де важливими є цілісність даних, швидкий доступ до інформації про товари, користувачів і замовлення, а також стабільність при високому навантаженні [11].

Поєднання React, Node.js та PostgreSQL формує сучасний повноцінний стек розробки, який дозволяє створити продуктивний, масштабований та зручний у використанні вебзастосунок. Також наведена таблиця 2.1, яка порівнює популярні стеки.

Таблиця 2.1 – Порівняльна характеристика популярних стеків розробки

Критерій	React + Node.js + PostgreSQL	Angular + Java Spring + MySQL	Vue.js + Laravel + MySQL
Швидкість розробки	Висока	Середня	Висока
Продуктивність	Висока	Висока	Середня
Масштабованість	Висока	Дуже висока	Середня
Зручність підтримки	Висока	Складніша структура	Висока
Єдина мова програмування	Так (JavaScript)	Ні	Частково
Підходить для SPA	Так	Так	Так
Робота з великим навантаженням	Висока	Висока	Середня
Гнучкість архітектури	Висока	Середня	Висока

Аналіз показує, що стек Node.js + React + PostgreSQL є оптимальним вибором для розробки інтернет-магазину комп'ютерних комплектуючих. Його основними перевагами є висока продуктивність, використання єдиної мови програмування, зручність масштабування та сучасний підхід до побудови вебзастосунків. Саме цей стек забезпечує баланс між швидкістю розробки, технічною ефективністю та можливістю подальшого розвитку системи.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Реалізація клієнтської частини системи

Архітектура клієнтської частини побудована у вигляді окремих функціональних компонентів, кожен із яких відповідає за конкретний елемент інтерфейсу. Такий підхід забезпечує повторне використання коду, спрощує підтримку та масштабування проєкту. Для розробки користувацького інтерфейсу було використано бібліотеку React-Bootstrap, яка поєднує можливості React із готовими адаптивними компонентами Bootstrap. Це дозволило прискорити процес створення інтерфейсу, забезпечити його сучасний вигляд та реалізувати зручну взаємодію користувача із системою.

У межах реалізації було створено основні сторінки системи: головну сторінку, каталог товарів, сторінку окремого товару, кошик, форму авторизації, особистий кабінет користувача та адміністративну панель.

Головна сторінка інтернет-магазину є центральним елементом користувацького інтерфейсу. У структурі головної сторінки реалізовано навігаційне меню, яке дозволяє користувачеві перейти до кошика та особистого кабінету, для адміністраторів з'являється, також, адмін панель. Головна сторінка сайту представлена на рис. 3.1.

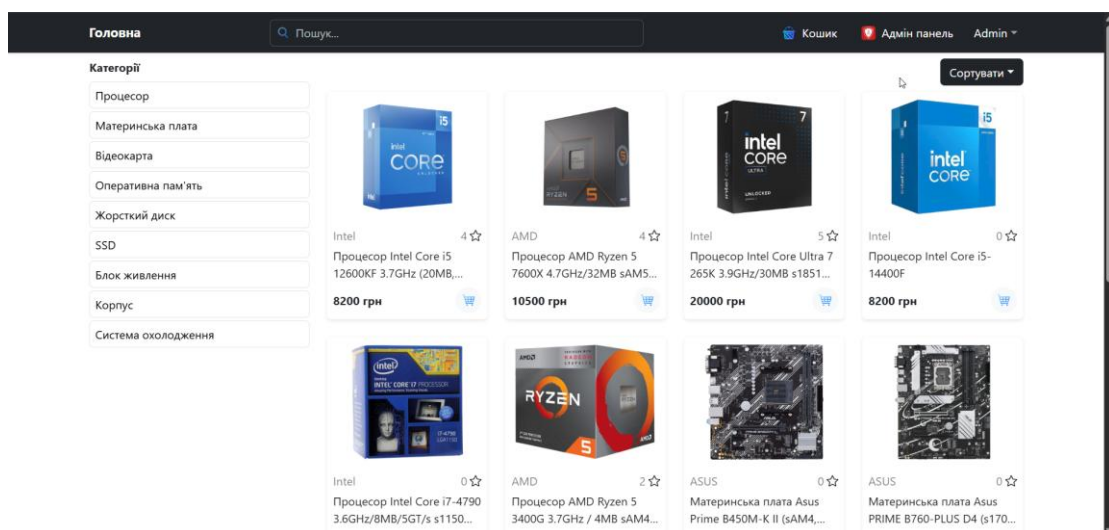


Рисунок 3.1 – Головна сторінка сайту

Інтерфейс сторінки окремого товару побудований із урахуванням принципів зручності та візуальної ієрархії. На сторінці передбачено відображення назви товару, його рейтингу, а також можливості поставити свій, ціна та детальний опис його характеристик. Також, з даної сторінки можна додати товар до кошика. На рис. 3.2 представлено реалізацію сторінки.

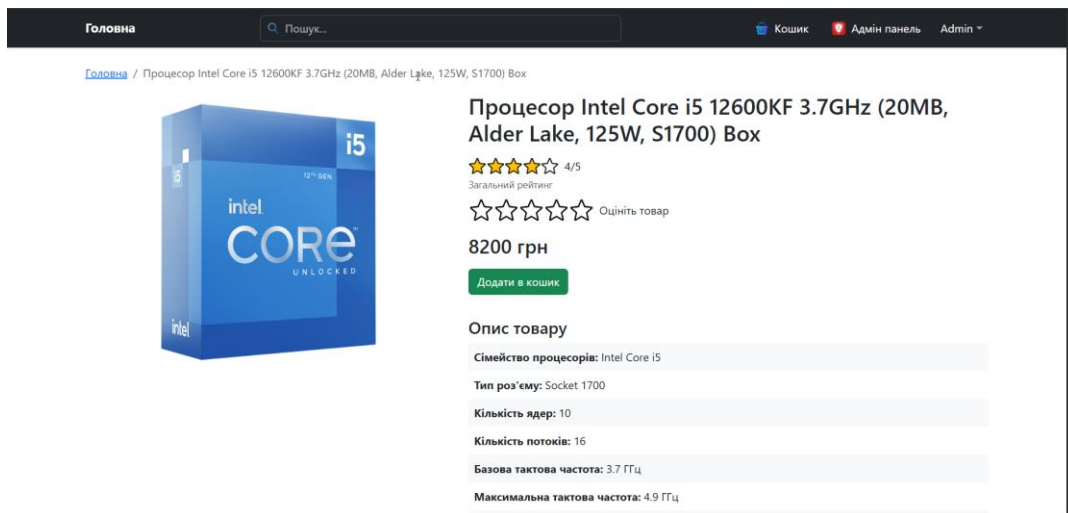


Рисунок 3.2 – Сторінка окремого товару

Інтерфейс сторінки кошика побудований таким чином, щоб забезпечити швидке сприйняття інформації та мінімізувати кількість дій, необхідних для переходу до оформлення замовлення. Для зручності користувача, інформація представлена у вигляді таблиці. Сторінка кошика представлена на рис. 3.3.

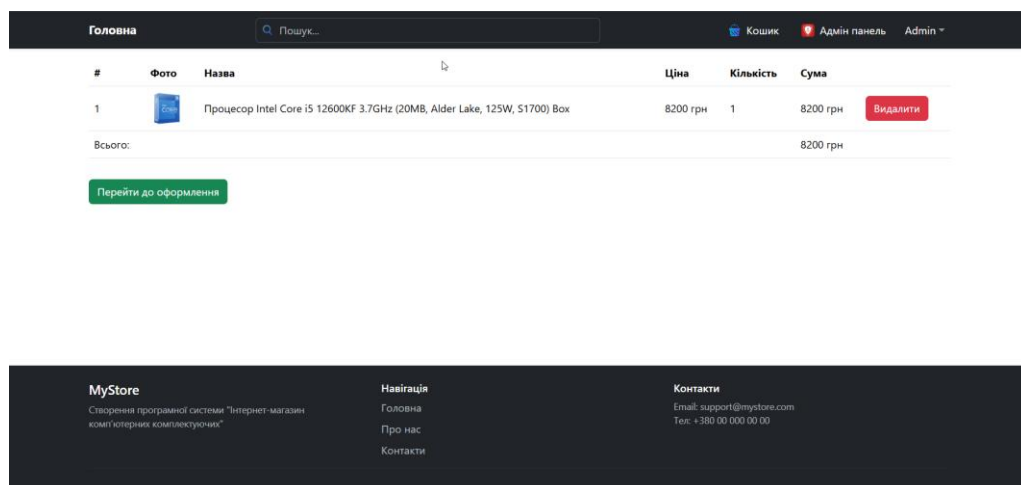
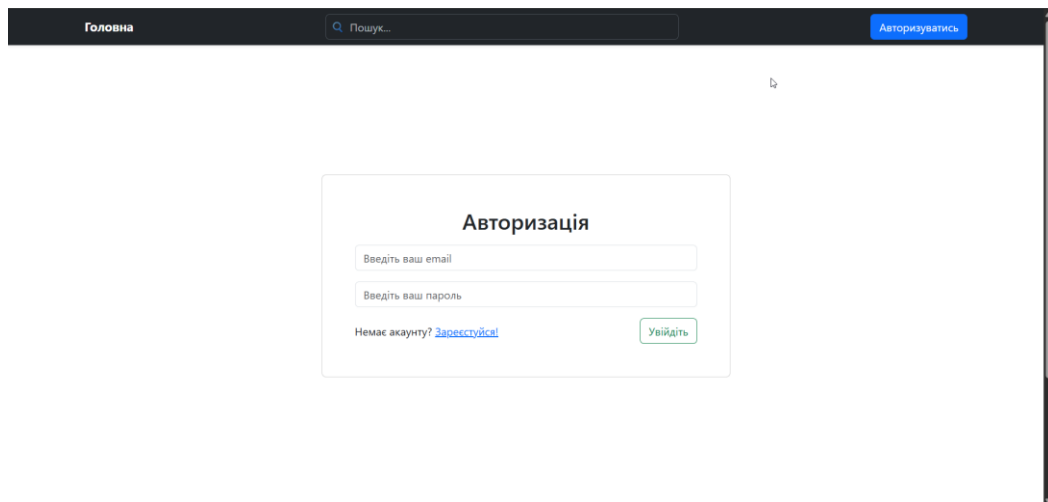


Рисунок 3.3 – Сторінка кошику

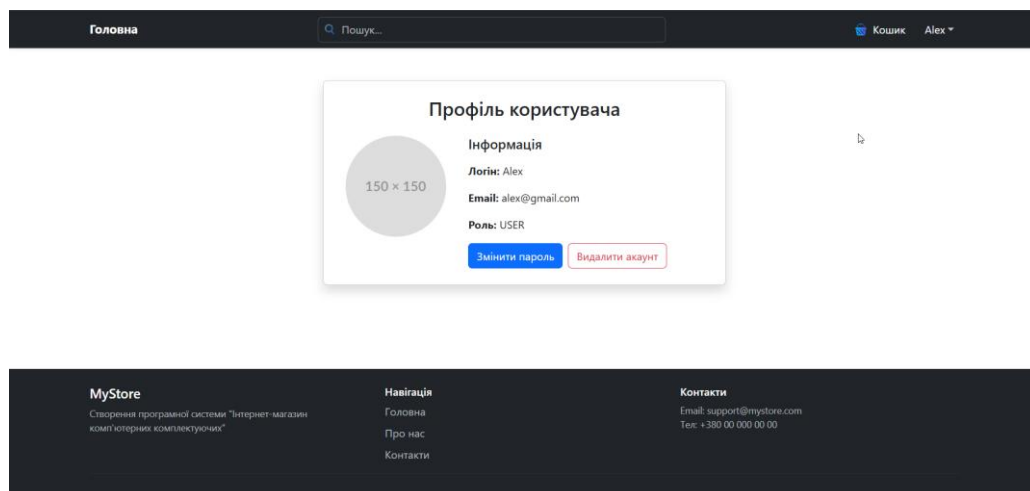
У межах реалізації сторінки передбачено форму введення основних даних для входу, зокрема електронної пошти та пароля. Після введення інформації система виконує перевірку коректності даних та надсилає запит до серверної частини для підтвердження автентичності користувача. Форма авторизації представлена на рис. 3.4.



The screenshot shows a web application interface with a dark header. On the left, there is a link 'Головна'. In the center, there is a search bar with a magnifying glass icon and the text 'Пошук...'. On the right, there is a blue button labeled 'Авторизуватись'. Below the header, centered on the page, is a white box titled 'Авторизація'. Inside this box, there are two input fields: 'Введіть ваш email' and 'Введіть ваш пароль'. Below these fields, there is a link 'Немає акаунту? Зареєструйся!' and a green button labeled 'Увійдіть'.

Рисунок 3.4 – Форма авторизації

Сторінка особистого кабінету надає доступ до індивідуальної інформації, дозволяє змінити пароль, а також надає змогу видалити свій акаунт. На рис. 3.5 представлена її реалізація.



The screenshot shows a web application interface with a dark header. On the left, there is a link 'Головна'. In the center, there is a search bar with a magnifying glass icon and the text 'Пошук...'. On the right, there is a shopping cart icon labeled 'Кошик' and a user profile icon labeled 'Alex'. Below the header, centered on the page, is a white box titled 'Профіль користувача'. Inside this box, there is a circular placeholder for a profile picture with the text '150 x 150'. To the right of the placeholder, there is a section titled 'Інформація' containing the following details: 'Логін: Alex', 'Email: alex@gmail.com', and 'Роль: USER'. Below this information, there are two buttons: a blue button labeled 'Змінити пароль' and a red button labeled 'Видалити акаунт'. At the bottom of the page, there is a dark footer. On the left, there is the 'MyStore' logo and the text 'Створення програмної системи "Інтернет-магазин комп'ютерних комплектуючих"'. In the center, there is a section titled 'Навігація' with links 'Головна', 'Про нас', and 'Контакти'. On the right, there is a section titled 'Контакти' with the text 'Email: support@mystore.com' and 'Тел: +380 00 000 00 00'.

Рисунок 3.5 – Особистий кабінет

У межах реалізації адміністративної панелі передбачено можливість додавання, редагування та видалення товарів, керування категоріями, брендами та технічними характеристиками продукції. Це дозволяє підтримувати актуальність асортименту та оперативно вносити зміни до каталогу. Адміністративна панель представлена на рис. 3.6.

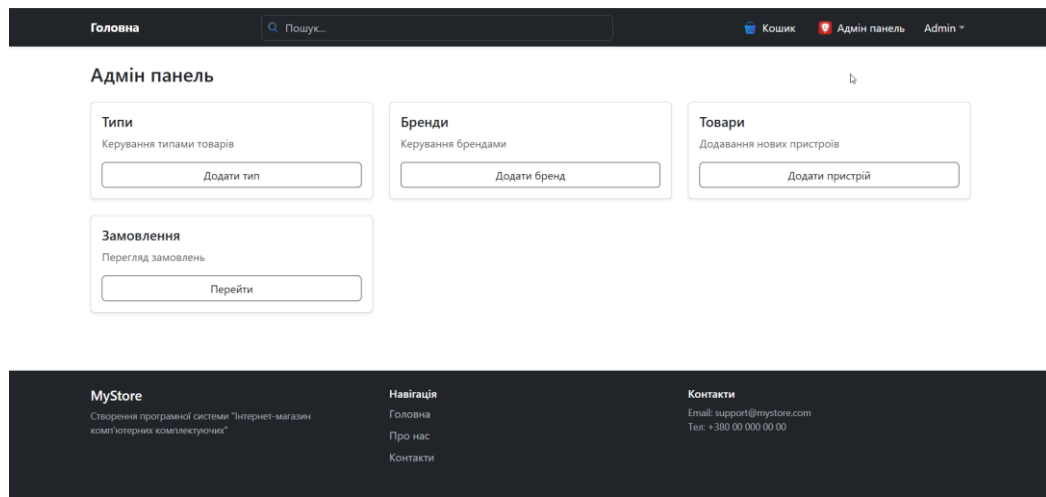


Рисунок 3.6 – Адміністративна панель

3.2 Реалізація серверної частини та взаємодія з базою даних

Серверна частина системи є одним із ключових компонентів програмного забезпечення, оскільки саме вона забезпечує обробку клієнтських запитів, реалізацію бізнес-логіки, контроль доступу до ресурсів та взаємодію з базою даних. У межах даного проекту серверна архітектура побудована із використанням середовища виконання JavaScript Node.js та фреймворку Express.js, що дозволило створити продуктивний, гнучкий і масштабований вебсервіс для обслуговування функціоналу інтернет-магазину комп'ютерних комплектуючих.

Вибір зазначеного технологічного стеку зумовлений його широким застосуванням у сучасній веброзробці та здатністю забезпечувати ефективну роботу із великою кількістю одночасних запитів. Node.js використовує асинхронну модель обробки подій, що є особливо важливим для серверних

застосунків електронної комерції, де необхідно швидко реагувати на дії користувачів. Використання Express.js значно спрощує створення REST API, дозволяє організувати маршрутизацію, інтегрувати middleware-компоненти та структурувати логіку застосунку.

Для зберігання та обробки інформації використовується система керування базами даних PostgreSQL. Дана СКБД була обрана завдяки своїй надійності, підтримці складних SQL-запитів, високій продуктивності та можливості масштабування. У контексті інтернет-магазину це особливо важливо, оскільки система повинна працювати з великим обсягом структурованих даних: каталогом товарів, обліковими записами користувачів, історією замовлень та інформацією про кошик.

Взаємодія між сервером і базою даних реалізована через відповідні моделі та SQL-запити, що забезпечують виконання CRUD-операцій для основних сутностей системи: товарів, користувачів, кошика та замовлень. Такий підхід дозволяє централізовано керувати доступом до даних, мінімізувати дублювання логіки та підвищити підтримуваність програмного коду.

Побудова серверної частини за модульним принципом дозволила розділити логіку на окремі функціональні блоки: маршрути, контролери, моделі та middleware-компоненти. Такий підхід спрощує підтримку, тестування та подальший розвиток системи. Кожен із модулів виконує чітко визначену роль у загальній архітектурі застосунку. Маршрути відповідають за приймання HTTP-запитів та їх перенаправлення до відповідних контролерів. Контролери реалізують бізнес-логіку, виконують перевірку вхідних даних та формують відповідь клієнту. Моделі забезпечують взаємодію з базою даних, а middleware-компоненти використовуються для авторизації, валідації, логування та обробки помилок.

Для забезпечення структурованості та масштабованості серверної частини було реалізовано модульну архітектуру проєкту. На рис. 3.7 представлено структуру основних директорій серверного застосунку.

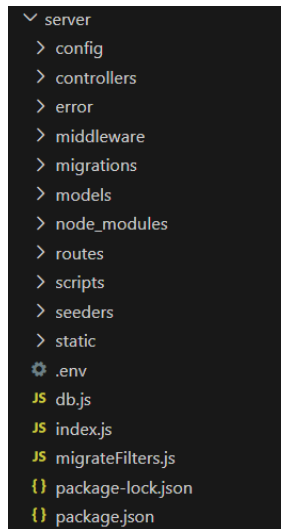


Рисунок 3.7 – Структура основних директорій серверного застосунка

Подібна структура дозволяє швидко орієнтуватися в кодовій базі та спрощує процес розробки. Крім того, модульність забезпечує можливість розширення функціоналу без необхідності суттєвого рефакторингу вже реалізованих компонентів.

Для організації взаємодії між клієнтською та серверною частинами було реалізовано систему REST API-маршрутів. Кожен маршрут відповідає за виконання конкретної операції, пов'язаної з обробкою даних. Наприклад, окремі маршрути передбачені для отримання списку товарів, створення нового запису, оновлення інформації про товар або його видалення. Така організація API відповідає сучасним стандартам побудови вебсервісів і забезпечує зручність інтеграції з клієнтською частиною застосунку. На рис. 3.8 представлено приклад реалізації маршрутів для роботи з товарами.

```
router.post('/', authMiddleware, checkRole('ADMIN'), deviceController.create)
router.get('/filters', deviceController.getFilters)
router.get('/', deviceController.getAll)
router.get('/:id', deviceController.getOne)
router.delete('/:id', authMiddleware, checkRole('ADMIN'), deviceController.deleteDevice)

module.exports = router
```

Рисунок 3.8 – Реалізація маршрутів для роботи з товарами

Для забезпечення повноцінного функціонування системи, окрім маршрутів для роботи з товарами, було реалізовано окремий набір API-маршрутів для керування користувачами. Вони відповідають за виконання таких операцій, як реєстрація, авторизація, зміна пароля та видалення облікового запису. Наявність окремого модуля маршрутів для користувачів забезпечує логічне розмежування функціоналу та підвищує безпеку системи. Приклад реалізації маршрутів для роботи з користувачами наведено на рис. 3.9.

```
router.post('/registration', userController.registration)
router.post('/login', userController.login)
router.get('/auth', authMiddleware, userController.check)
router.delete('/account', authMiddleware, userController.deleteUser)
router.post('/account', authMiddleware, userController.changePassword)

module.exports = router
```

Рисунок 3.9 – Реалізація маршрутів для роботи з користувачами

Процес взаємодії клієнта із сервером відбувається за наступною схемою: користувач виконує певну дію на клієнтському інтерфейсі, після чого формується HTTP-запит до відповідного API-маршруту. Сервер приймає цей запит, передає його до контролера, який виконує необхідну бізнес-логіку та звертається до моделі для роботи з базою даних. Після завершення операції формується JSON-відповідь, яка повертається клієнтському застосунку для подальшого відображення.

Бізнес-логіка системи реалізована у контролерах, які обробляють запити, виконують необхідні перевірки та формують відповідь клієнту. Саме контролери є центральним елементом серверної логіки, оскільки вони визначають правила взаємодії між користувачем, системою та базою даних. В проєкті контролери винесені в окрему директорію, для кожного з маршрутів передбачена функція в контролері. На рис. 3.10 наведено приклад контролера для обробки операцій із товарами.

```

class DeviceController {
  async create(req, res, next) {
    try {
      let { name, price, BrandId, TypeId, info } = req.body
      const { img } = req.files

      const fileName = uuid.v4() + ".jpg"
      img.mv(path.resolve(__dirname, '..', 'static', fileName))

      const device = await Device.create({
        name,
        price,
        BrandId,
        TypeId,
        img: fileName
      })

      if (info) {
        info = JSON.parse(info)

        await Promise.all(
          info.map(i =>
            DeviceInfo.create({
              title: i.title,
              description: i.description,
              deviceId: device.id
            })
          )
        )
      }

      return res.json(device)
    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }
}

```

Рисунок 3.10 – Контролер для створення нового товару

Контролер створення нового товару не лише приймає дані з клієнтської частини, але й виконує їх попередню валідацію. Зокрема, перевіряється наявність обов'язкових полів, коректність числових значень (ціни, кількості) та відповідність типів даних. У разі некоректного введення сервер повертає відповідь із відповідним HTTP-статусом помилки.

Після успішної валідації виконується звернення до моделі, яка формує SQL-запит для додавання нового запису до таблиці товарів. У відповідь клієнт отримує JSON-об'єкт із даними створеного товару, що дозволяє одразу відобразити його у користувацькому інтерфейсі.

Окрім створення нового товару, у системі реалізовано й інші важливі операції для роботи з пристроями. Зокрема, передбачено можливість отримання детальної інформації про окремий товар за його ідентифікатором, а також видалення запису з каталогу. Дані функції є необхідними для забезпечення повного циклу управління товарами в адміністративній панелі. Приклади реалізації контролерів для видалення пристрою та отримання одного товару наведено на рис. 3.11 та рис. 3.12.

```

async getOne(req, res, next) {
  try {
    const { id } = req.params

    const device = await Device.findOne({
      where: { id },
      include: [{ model: DeviceInfo, as: 'info' }]
    })

    return res.json(device)
  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}

```

Рисунок 3.11 – Контролер для отримання одного пристрою

```

async deleteDevice(req, res, next) {
  try {
    const { id } = req.params

    const device = await Device.findOne({
      where: { id }
    })

    if (!device) {
      return next(ApiError.badRequest('Пристрій не знайдено'))
    }

    await device.destroy()

    return res.json({ message: 'Пристрій видалено' })
  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}

```

Рисунок 3.12 – Контролер для видалення пристрою

Важливою складовою серверної логіки є контролери для роботи з користувачами. Саме вони реалізують механізми автентифікації та керування обліковими записами. У межах проєкту було реалізовано функціонал реєстрації нового користувача, входу до системи, зміни пароля та видалення користувача. Дані операції забезпечують повноцінне управління доступом до системи та

відповідають сучасним вимогам безпеки. Приклади контролерів наведено на рис. 3.13 – 3.16.

```
async changePassword(req, res, next) {
  try {
    const id = req.user.id
    const { oldPassword, newPassword } = req.body

    const user = await User.findOne({where: {id}})

    if (!user) {
      return next(ApiError.internal('Операція неможлива!'))
    }

    const isValidPassword = await bcrypt.compare(oldPassword, user.password)

    if (!isValidPassword) {
      return next(ApiError.badRequest('Невірний поточний пароль!'))
    }

    const newHashPassword = await bcrypt.hash(newPassword, 5)

    await user.update({ password: newHashPassword })

    const token = generateJwt(user.id, user.email, user.role, user.username)
    return res.json({ message: "Пароль змінено", token })
  } catch(e) {
    return next(ApiError.badRequest(e.message))
  }
}
```

Рисунок 3.13 – Контролер для зміни паролю користувача

```
async deleteUser(req, res, next) {
  try {
    const id = req.user.id
    const { password } = req.body

    const user = await User.findOne({where: {id}})

    if (!user) {
      return next(ApiError.internal('Операція неможлива!'))
    }

    const isValidPassword = await bcrypt.compare(password, user.password)

    if (!isValidPassword) {
      return next(ApiError.badRequest('Невірний пароль!'))
    }

    await user.destroy()
    return res.json({message: "Обліковий запис видалено!"})
  } catch(e) {
    return next(ApiError.badRequest(e.message))
  }
}
```

Рисунок 3.14 – Контролер видалення користувача

```

async registration(req, res, next) {
  try {
    const { username, email, password } = req.body

    if (!username) {
      return next(ApiError.badRequest('Введіть логін'))
    }

    if (!email || !password) {
      return next(ApiError.badRequest('Некоректний email або пароль'))
    }

    const candidateEmail = await User.findOne({ where: { email } })
    if (candidateEmail) {
      return next(ApiError.badRequest('Користувач з таким email вже існує'))
    }

    const candidateUsername = await User.findOne({ where: { username } })
    if (candidateUsername) {
      return next(ApiError.badRequest('Користувач з таким логіном вже існує'))
    }

    const hashPassword = await bcrypt.hash(password, 5)

    const user = await User.create({
      username,
      email,
      role: 'USER',
      password: hashPassword
    })

    await Basket.create({ UserId: user.id })

    const token = generateJwt(user.id, user.email, user.role, user.username)

    return res.json({ token })
  } catch (e) {
    next(ApiError.internal(e.message))
  }
}

```

Рисунок 3.15 – Контролер для реєстрації нового користувача

```

async login(req, res, next) {
  const {email, password} = req.body
  const user = await User.findOne({where: {email}})
  if(!user) {
    return next(ApiError.badRequest('Користувач з таким email не існує!'))
  }
  let comparePassword = await bcrypt.compareSync(password, user.password)
  if(!comparePassword) {
    return next(ApiError.badRequest('Вказано невірний пароль'))
  }
  const token = generateJwt(user.id, user.email, user.role, user.username)
  return res.json({token})
}

async check(req, res, next) {
  const token = generateJwt(req.user.id, req.user.email, req.user.role, req.user.username)
  return res.json({token})
}

```

Рисунок 3.16 – Контролер для автризації користувача

Оскільки процес реєстрації та авторизації користувача проходить через генерацію JWT-ключа, надається рис. 3.17 з прикладом його реалізації в програмі.

```
const generateJwt = (id, email, role, username) => {
  return jwt.sign(
    {id, email, role, username},
    process.env.SECRET_KEY,
    {expiresIn: '24h'}
  )
}
```

Рисунок 3.17 – Реалізація JWT

Як можна побачити з рис. 3.17 – функція приймає в себе: ідентифікаційний номер користувача, його пошту, роль та логін. Ключ оновлюється кожні 24-години для безпеки даних користувача.

Згенерований JWT-токен передається клієнту після успішної авторизації та використовується для доступу до захищених маршрутів. Кожен запит до таких маршрутів містить токен у заголовку Authorization. На сервері реалізовано middleware-компонент, який перевіряє валідність токена та декодує дані користувача. У разі відсутності або некоректності токена доступ до ресурсу обмежується.

Взаємодія з базою даних PostgreSQL реалізована через моделі, що відповідають окремим сутностям системи. Це дозволяє структуровано працювати з даними та виконувати основні операції створення, читання, оновлення та видалення записів. Приклад реалізації моделей представлено на рис. 3.18 – 3.21.

```
const Rating = sequelize.define('Rating', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  rate: {
    type: DataTypes.INTEGER,
    primaryKey: false,
    autoIncrement: false,
  },
})
```

Рисунок 3.18 – Опис таблиці рейтингу

```
const Device = sequelize.define('Device', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  name: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
  },
  price: {
    type: DataTypes.INTEGER,
  },
  rating: {
    type: DataTypes.FLOAT,
    defaultValue: 0,
  },
  img: {
    type: DataTypes.STRING,
    allowNull: false,
  },
})
```

Рисунок 3.19 – Опис таблиці пристроїв

```
const Type = sequelize.define('Type', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  name: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
  },
})
```

Рисунок 3.20 – Опис таблиці типів

```
const Brand = sequelize.define('Brand', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  name: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
  },
})
```

Рисунок 3.21 – Опис таблиці брендів

Реалізація моделі користувачів зображена на рис. 3.22.

```
const User = sequelize.define('User', {  
  id: {  
    type: DataTypes.INTEGER,  
    primaryKey: true,  
    autoIncrement: true,  
  },  
  username: {  
    type: DataTypes.STRING,  
    unique: true,  
    allowNull: false,  
  },  
  email: {  
    type: DataTypes.STRING,  
    unique: true,  
    allowNull: false,  
  },  
  password: {  
    type: DataTypes.STRING,  
    allowNull: false,  
  },  
  role: {  
    type: DataTypes.STRING,  
    allowNull: false,  
    defaultValue: "USER",  
  },  
})
```

Рисунок 3.22 – Опис таблиці користувачів

Важливим аспектом проєктування бази даних є встановлення зв'язків між основними таблицями. У реалізованій системі використано зв'язки типу «один до багатьох» та «багато до одного». Наприклад, кожен товар належить до певного бренду та типу, що реалізується через зовнішні ключі `brandId` та `typeId` у таблиці товарів.

Таким чином, один запис у таблиці брендів може бути пов'язаний із багатьма записами у таблиці товарів. Аналогічний підхід використовується для типів товарів. Це дозволяє уникнути дублювання даних та забезпечує цілісність бази даних.

Крім того, таблиця рейтингів пов'язана з таблицями користувачів і товарів, що дозволяє зберігати оцінки, залишені конкретними користувачами для конкретних товарів. На рис. 3.23 – 3.25 представлені приклади зв'язків між представленими таблицями.

```
User.hasMany(Rating)
Rating.belongsTo(User)
```

Рисунок 3.23 – Зв’язок між користувачем та рейтингом

```
Device.hasMany(Rating)
Rating.belongsTo(Device)
```

Рисунок 3.24 – Зв’язок між пристроєм та рейтингом

```
Type.hasMany(Device)
Device.belongsTo(Type)

Brand.hasMany(Device)
Device.belongsTo(Brand)

Type.belongsToMany(Brand, { through: TypeBrand })
Brand.belongsToMany(Type, { through: TypeBrand })
```

Рисунок 3.25 – Зв’язки між типами, брендами та пристроями

Окрему увагу під час реалізації серверної частини було приділено безпеці та коректності обробки даних. Для цього використовуються механізми валідації вхідних параметрів, перевірка прав доступу до адміністративних функцій, а також централізована обробка помилок. Це дозволяє уникнути некоректного введення, знизити ризик збоїв та забезпечити стабільну роботу системи навіть у випадку помилкових запитів.

Для перевірки коректності роботи серверної частини було виконано тестування API-запитів за допомогою середовища Postman. У процесі тестування перевірялися CRUD-операції, коректність статус-кодів HTTP-відповідей, структура JSON-об’єктів, а також обробка некоректних або неповних запитів. Такий підхід дозволив підтвердити стабільність реалізованого API та його готовність до використання у реальному середовищі. На рис. 3.26 – 3.28 представлені приклади успішних виконань запитів до серверу.

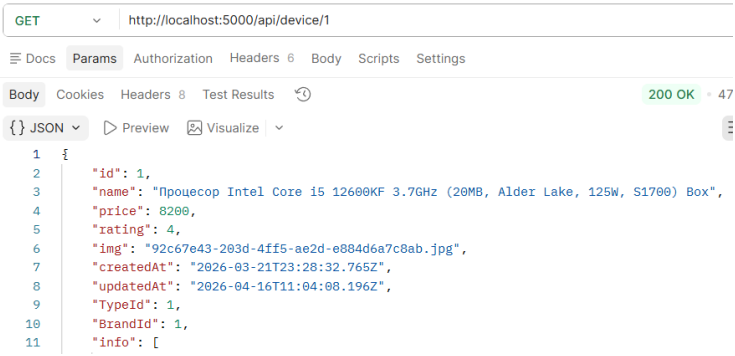


Рисунок 3.26 – Отримання певного пристрою

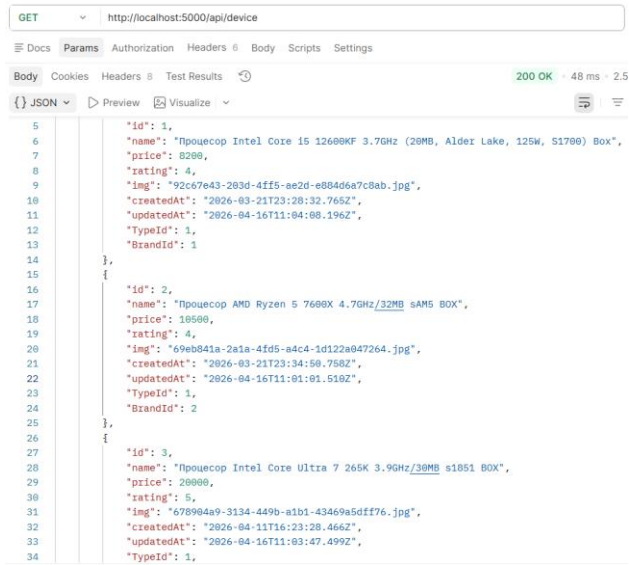


Рисунок 3.27 – Отримання усіх пристроїв

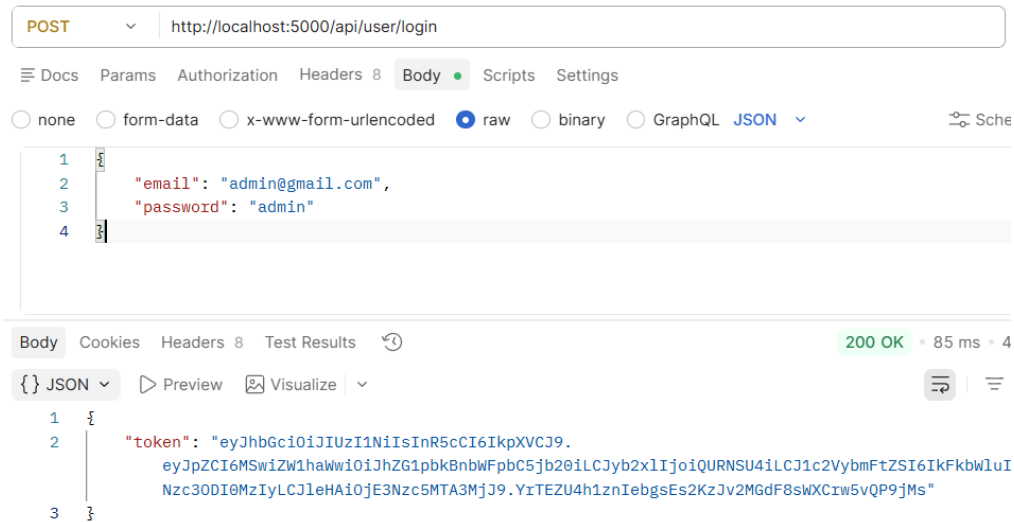


Рисунок 3.28 – Авторизація користувача

На рис. 3.29 – 3.31 представлено успішне виконання запитів до серверу.

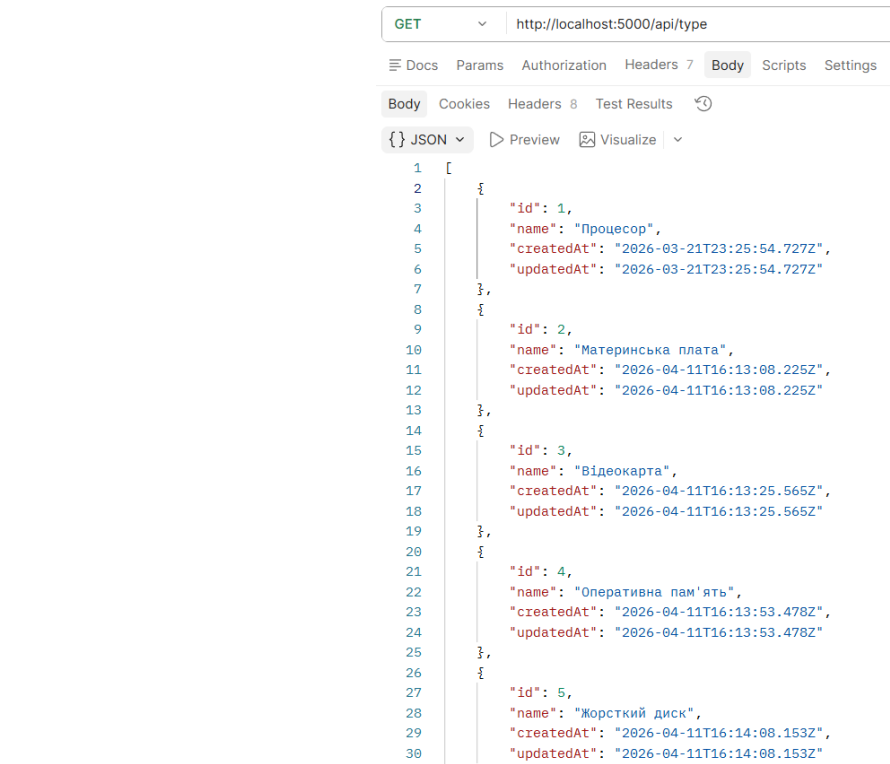


Рисунок 3.29 – Список усіх типів

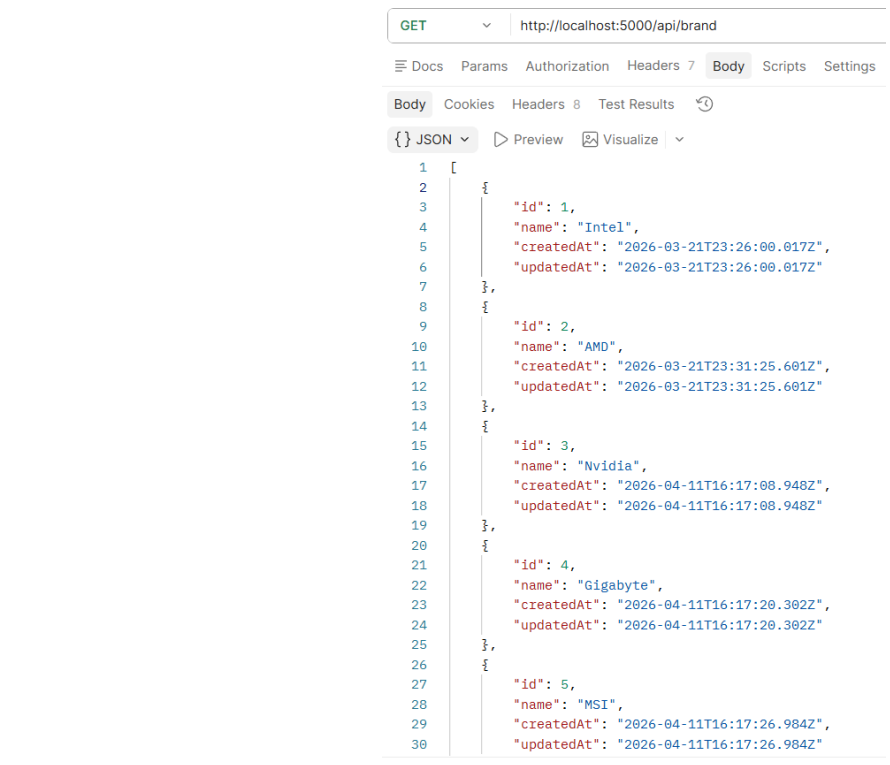


Рисунок 3.30 – Список усіх брендів

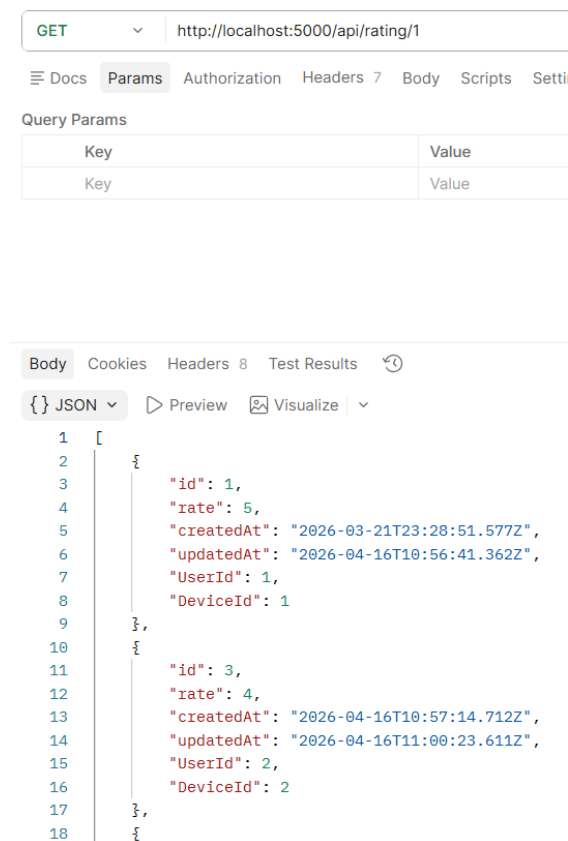


Рисунок 3.31 – Список усіх рейтингів

Таким чином, серверна частина інтернет-магазину комп'ютерних комплектуючих реалізована з урахуванням сучасних технологічних стандартів та принципів масштабованості. Обраний стек Node.js + Express.js + PostgreSQL забезпечує високу продуктивність, надійність та ефективну обробку даних, що створює стійку основу для подальшого розвитку програмної системи та впровадження додаткового функціоналу [12].

3.3 Алгоритм роботи програмної системи

У даному підрозділі розглядається алгоритм роботи програмної системи, який відображає повну послідовність дій користувача, а також внутрішні процеси, що відбуваються на стороні серверної частини інтернет-магазину комп'ютерних комплектуючих. Формалізація алгоритму є важливим етапом проектування, оскільки дозволяє систематизувати логіку взаємодії між клієнтським інтерфейсом, сервером та базою даних.

Розроблений алгоритм охоплює як основний сценарій використання системи – придбання товару, так і альтернативні сценарії, зокрема пов’язані з відсутністю авторизації користувача. Це дозволяє врахувати різні варіанти поведінки користувача та забезпечити коректну реакцію системи на кожному етапі взаємодії.

Особливу увагу приділено послідовності обробки запитів: від моменту ініціювання дії користувачем до формування відповіді сервером. Кожен крок алгоритму супроводжується виконанням певних операцій, таких як перевірка прав доступу, валідація даних, обробка бізнес-логіки та взаємодія з базою даних. Це дозволяє не лише відобразити зовнішню поведінку системи, але й частково розкрити її внутрішню архітектуру.

Для наочного представлення алгоритму було розроблено блок-схему (рис. 3.32), яка демонструє логіку виконання операцій під час придбання товару. Використання графічного представлення дозволяє краще зрозуміти взаємозв’язки між окремими етапами, умовні переходи та альтернативні гілки виконання процесу.

Таким чином, розроблений алгоритм роботи програмної системи відображає повну послідовність дій користувача під час взаємодії з інтернет-магазином комп’ютерних комплектуючих – від перегляду товару до завершення оформлення замовлення. Побудована блок-схема дозволила наочно представити логіку функціонування системи, визначити основні етапи обробки даних та врахувати альтернативні сценарії, пов’язані з авторизацією користувача. Реалізація такого алгоритму забезпечує цілісність бізнес-процесів, підвищує зручність використання системи та сприяє її ефективному функціонуванню.

Крім того, реалізація описаного алгоритму створює основу для подальшого масштабування та вдосконалення програмної системи. Чітко структурована логіка взаємодії між компонентами дозволяє легко інтегрувати додаткові функціональні модулі, зокрема систему онлайн-оплати, механізми рекомендацій товарів, автоматизоване керування складськими залишками та

засоби аналітики користувацької активності. Завдяки цьому забезпечується гнучкість архітектури програмного забезпечення, спрощується підтримка системи та підвищується її адаптивність до змін вимог користувачів і бізнес-процесів інтернет-магазину.

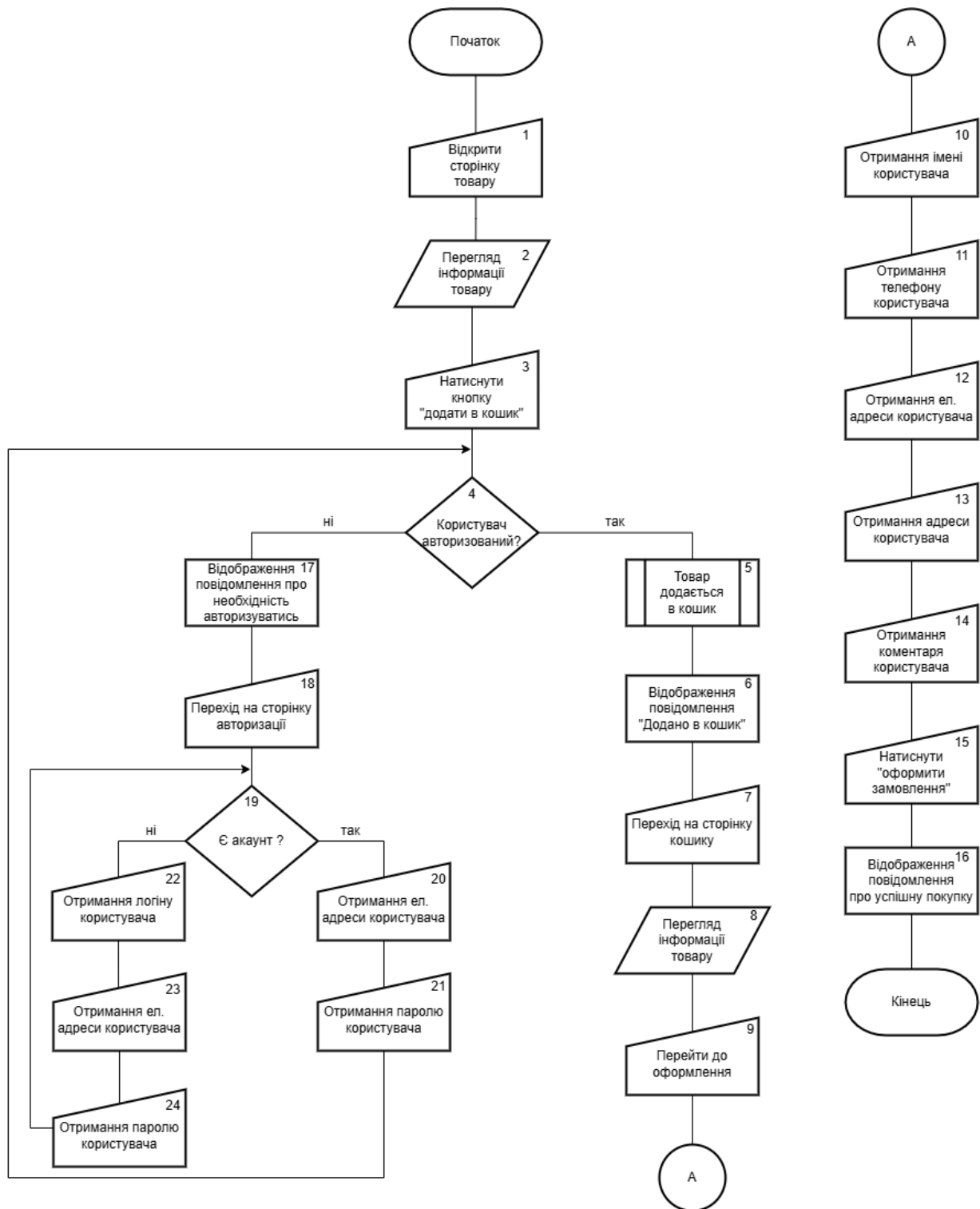


Рисунок 3.32 – Блок-схема процесу придбання товару

Етап 1. На початковому етапі користувач відкриває сторінку обраного товару в каталозі інтернет-магазину. Саме з цього моменту розпочинається процес подальшої взаємодії із системою.

Етап 2. Користувач переглядає детальну інформацію про товар, включаючи технічні характеристики, опис, вартість та доступність.

Етап 3. Після ознайомлення з інформацією користувач натискає кнопку «Додати в кошик». Ця дія ініціює перевірку можливості виконання операції.

Етап 4. Система здійснює перевірку статусу користувача та визначає, чи виконано авторизацію в системі. Від результату цієї перевірки залежить подальший сценарій роботи.

Етап 5. Якщо користувач авторизований, система додає обраний товар до кошика для подальшого оформлення.

Етап 6. Після успішного додавання товару система формує повідомлення про успішне виконання операції.

Етап 7. Користувач переходить до сторінки кошика, де зібрано всі обрані позиції. Це дає змогу перевірити склад замовлення перед оформленням.

Етап 8. На сторінці кошика користувач переглядає інформацію про товари, їх кількість та загальну вартість.

Етап 9. Після підтвердження коректності даних користувач переходить до етапу оформлення покупки.

Етап 10. На першому етапі оформлення система отримує ім'я користувача.

Етап 11. Вводиться контактний телефон користувача.

Етап 12. Система отримує електронну адресу користувача.

Етап 13. Користувач зазначає адресу доставки товару.

Етап 14. За потреби користувач може залишити додатковий коментар до замовлення.

Етап 15. Після заповнення всіх необхідних полів користувач натискає кнопку «Оформити замовлення». Система розпочинає процедуру обробки заявки.

Етап 16. Після успішного завершення оформлення система відображає повідомлення про успішну покупку. Це свідчить про завершення процесу створення замовлення.

Етап 17. Якщо авторизація відсутня, відображається повідомлення про необхідність входу до облікового запису.

Етап 18. Якщо користувач не авторизований, він перенаправляється на сторінку входу до системи. Це необхідно для подальшого продовження покупки.

Етап 19. На сторінці авторизації система перевіряє, чи має користувач вже створений обліковий запис. Залежно від цього обирається сценарій входу або реєстрації.

Етап 20/22. Якщо акаунт існує, система отримує електронну адресу для входу. Якщо ж обліковий запис відсутній, користувач вводить бажаний логін для створення нового профілю.

Етап 21/23. Наступним кроком є введення пароля або електронної адреси відповідно до обраного сценарію.

Етап 24. Завершальним етапом є введення пароля користувача для підтвердження дій. Після цього система створює або активує обліковий запис, що дозволяє продовжити оформлення замовлення.

4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОЇ СИСТЕМИ

Тестування програмної системи є невід'ємним етапом життєвого циклу розробки програмного забезпечення, оскільки воно дозволяє перевірити відповідність реалізованого продукту функціональним вимогам, виявити можливі дефекти та забезпечити стабільність його роботи. За сучасними підходами до розробки програмного забезпечення, тестування спрямоване не лише на пошук помилок, а й на підтвердження якості, безпеки та продуктивності системи перед її впровадженням у реальне середовище. Дослідження у сфері програмної інженерії підкреслюють, що своєчасне тестування дозволяє суттєво зменшити ризики відмови системи та скоротити витрати на виправлення помилок на пізніх етапах розробки [13].

Для інтернет-магазину комп'ютерних комплектуючих тестування має особливе значення, оскільки система працює з критично важливими бізнес-процесами: обробкою замовлень, збереженням даних користувачів, керуванням товарами та взаємодією між клієнтською і серверною частинами. Надійність таких процесів безпосередньо впливає на ефективність роботи системи та рівень довіри кінцевих користувачів. Саме тому у межах даного розділу виконується оцінка працездатності програмної системи шляхом перевірки її функціональних модулів, аналізу стабільності виконання основних сценаріїв використання та підтвердження відповідності встановленим технічним вимогам [14].

4.1 Методика тестування програмної системи

Для оцінки якості розробленого інтернет-магазину комп'ютерних комплектуючих було обрано комплексний підхід до тестування, що охоплює ключові аспекти користувацького досвіду та технічної реалізації системи.

Основна увага приділялася перевірці верстки, функціональності та адаптивності вебзастосунку на різних типах пристроїв.

Першим етапом стало тестування верстки, метою якого була перевірка коректного відображення елементів інтерфейсу на сторінках системи. У процесі оцінювалася цілісність структури сторінок, правильність розташування компонентів, відповідність стилістики, читабельність тексту та відсутність візуальних дефектів. Особливу увагу було приділено узгодженості інтерфейсних елементів, що безпосередньо впливає на зручність користування системою.

Другим етапом виконувалося функціональне тестування, яке передбачало перевірку працездатності основних модулів системи. Було протестовано процеси реєстрації та авторизації користувачів, перегляду каталогу, пошуку товарів, роботи кошика, оформлення замовлення та використання адміністративної панелі. Для кожного сценарію визначався очікуваний результат, після чого проводилася перевірка фактичної поведінки системи.

Окремим напрямом стало тестування адаптивності під мобільні пристрої, що є критично важливим для сучасних вебзастосунків. Перевірка здійснювалася шляхом емуляції різних розмірів екранів та використання мобільного режиму браузера. Оцінювалася коректність масштабування елементів, зручність навігації, доступність функціоналу та швидкість взаємодії користувача із системою на смартфонах і планшетах.

Такий підхід дозволив комплексно оцінити якість реалізованого програмного продукту та підтвердити його готовність до використання в реальних умовах експлуатації.

4.2 Тестування верстки програмної системи

Тестування верстки програмної системи є важливим етапом оцінки якості вебзастосунку, оскільки саме коректне відображення інтерфейсу

безпосередньо впливає на зручність користування та загальне сприйняття продукту.. Результати тестування верстки наведено у таблиці 4.1 у вигляді чек-листа, що відображає перевірені критерії та стан їх виконання.

Таблиця 4.1 – Чек-лист тестування верстки

Критерій	Google Chrome (v.147.0.7721.116)	Opera One (v.130.0.5847.41)	Microsoft Edge (v.142.0.3595.94)
Перевірити коректне відображення сторінок згідно файлів дизайну	Пройдено	Пройдено	Пройдено
Перевірити наявність підвалу сайту	Пройдено	Пройдено	Пройдено
Перевірити наявність рядка пошуку	Пройдено	Пройдено	Пройдено
Перевірити наявність повідомлень при	Пройдено	Пройдено	Пройдено
Перевірити наявність повідомлень додаванні до кошика	Пройдено	Пройдено	Пройдено
Перевірити стиль та коректне відображення шрифтів тексту	Пройдено	Пройдено	Пройдено
Перевірити коректне відображення кнопок, блоків меню та ін.	Пройдено	Пройдено	Пройдено
Перевірити коректне вирівнювання елементів	Пройдено	Пройдено	Пройдено

За результатами проведеного тестування верстки встановлено, що програмна система коректно відображається у всіх перевірених браузерах: Google Chrome, Opera One та Microsoft Edge. Усі визначені критерії успішно пройдено, що свідчить про стабільність реалізованого інтерфейсу та його сумісність із сучасними вебплатформами.

4.3 Функціональне тестування системи

Функціональне тестування є одним із ключових етапів перевірки якості програмного забезпечення, оскільки воно дозволяє оцінити відповідність реалізованого функціоналу поставленим вимогам та очікуваним сценаріям використання. Результати тестування функціоналу наведено у таблиці 4.2.

Таблиця 4.2 – Чек-лист тестування функціоналу

Критерій	Google Chrome (v.147.0.7727.116)	Opera One (v.130.0.5847.41)	Microsoft Edge (v.142.0.3595.94)
1	2	3	4
Перевірити реєстрацію на сайті	Пройдено	Пройдено	Пройдено
Перевірити зміну паролю в особистому кабінеті	Пройдено	Пройдено	Пройдено
Валідація на порожні поля	Пройдено	Пройдено	Пройдено
Перевірити роботу пошуку за назвами товарів	Пройдено	Пройдено	Пройдено
Додати товар в кошик	Пройдено	Пройдено	Пройдено

Продовження таблиці 4.2

1	2	3	4
Перевірити перехід у випадяючому списку поля пошуку після введення ключового слова	Пройдено	Пройдено	Пройдено
Перевірити фільтрування за категоріями	Пройдено	Пройдено	Пройдено
Правильне вирахування кількості товару в кошику після додавання	Пройдено	Пройдено	Пройдено

За результатами проведеного тестування функціоналу встановлено, що функції системи виконуються коректно у всіх перевірених браузерах: Google Chrome, Opera One та Microsoft Edge. Отримані результати підтверджують кросбраузерну сумісність продукту та його готовність до використання в різних середовищах без необхідності додаткових до опрацювань.

4.4 Тестування адаптивності вебзастосунку

Тестування адаптивності вебзастосунку є важливим етапом перевірки якості програмного продукту, спрямованим на оцінювання коректності відображення та функціонування інтерфейсу на пристроях із різними розмірами екранів і роздільною здатністю. Його основною метою є забезпечення зручного та стабільного користувацького досвіду незалежно від типу пристрою – настільного комп'ютера, ноутбука, планшета чи смартфона. Під час тестування особлива увага приділялася перевірці коректного масштабування елементів

інтерфейсу, адаптивності навігаційного меню та збереженню функціональності основних компонентів вебзастосунку на різних типах пристроїв. Результати тестування наведені в таблиці 4.3.

Таблиця 4.3 – Чек-лист тестування адаптивності вебзастосунку

Критерій	Google Chrome (v.147.0.7727.116)	Opera One (v.130.0.5847.41)	Microsoft Edge (v.142.0.3595.94)
Перевірити коректне відображення сторінок	Пройдено	Пройдено	Пройдено
Перевірити коректне відображення кнопок, блоків меню та ін.	Пройдено	Пройдено	Пройдено
Перевірка переходу на сторінки після натискання на елементи сайту	Пройдено	Пройдено	Пройдено
Перевірити чи всі елементи доступні (адаптуються до мобільної версії та не обрізаються)	Пройдено	Пройдено	Пройдено
Перевірка коректного відображення елементів при зміні орієнтації девайсу	Пройдено	Пройдено	Пройдено

За результатами тестування адаптивності вебзастосунку у браузерх Google Chrome, Opera One та Microsoft Edge встановлено, що всі перевірені критерії виконуються коректно. Інтерфейс системи стабільно відображається на різних пристроях і роздільних здатностях, елементи сторінок, зокрема кнопки та блоки меню, коректно масштабуються та не виходять за межі екрану. Також

забезпечено правильне функціонування навігації між сторінками та повну доступність усіх елементів у мобільній версії. Додатково підтверджено коректне відображення інтерфейсу при зміні орієнтації пристрою, що свідчить про якісну реалізацію адаптивного дизайну вебзастосунку та його готовність до використання на різних типах пристроїв без втрати функціональності.

4.5 Аналіз результатів тестування

Аналіз результатів тестування є завершальним етапом оцінки якості програмного забезпечення, що дозволяє узагальнити отримані дані та визначити рівень готовності системи до практичного використання. Згідно з сучасними підходами до забезпечення якості програмного забезпечення, ефективність системи визначається не лише її функціональністю, а й стабільністю, продуктивністю, сумісністю та зручністю використання. Саме комплексна перевірка цих характеристик дозволяє сформувати об'єктивну оцінку програмного продукту.

У межах виконаного тестування було перевірено три основні напрями: коректність верстки, функціональність та адаптивність системи для мобільних пристроїв. Результати свідчать про те, що вебзастосунок стабільно працює у сучасних браузерях, забезпечує правильне відображення елементів інтерфейсу та підтримує однаковий рівень зручності використання на різних платформах. Це відповідає загальним принципам user-centered design, відповідно до яких адаптивність та передбачуваність інтерфейсу є критичними показниками успішності цифрових продуктів.

Функціональне тестування підтвердило коректну реалізацію ключових бізнес-процесів: авторизації користувачів, роботи з каталогом товарів, формування кошика, оформлення замовлень та адміністрування системи. Усі тестові сценарії були виконані без критичних збоїв, що свідчить про цілісність логіки програмної системи та правильність взаємодії між клієнтською, серверною частинами та базою даних.

З технічної точки зору, обраний стек технологій React + Node.js + PostgreSQL забезпечив високу швидкодію, гнучкість масштабування та ефективну обробку запитів. За даними галузевих досліджень, використання JavaScript-екосистеми для повного циклу веброзробки сприяє прискоренню створення застосунків та спрощує підтримку коду завдяки єдиній технологічній платформі.

Таким чином, результати тестування підтверджують, що розроблений інтернет-магазин комп'ютерних комплектуючих відповідає поставленим вимогам, є технічно стабільним, зручним у використанні та готовим до подальшого впровадження. Отримані показники дозволяють оцінити програмну систему як ефективне рішення для організації електронної комерції у сфері продажу комп'ютерних комплектуючих.

ВИСНОВКИ

З метою розробки сучасного та функціонального вебзастосунку для продажу комп'ютерних комплектуючих у роботі вирішено актуальне завдання створення інтернет-магазину, який забезпечує ефективну взаємодію між користувачем, серверною частиною та базою даних. Проведений аналіз предметної області показав високу актуальність електронної комерції у сфері продажу технічної продукції, а також підтвердив необхідність створення спеціалізованих платформ, орієнтованих на зручність вибору, швидкість оформлення замовлення та доступність товарної інформації.

У межах дослідження виконано аналіз існуючих інтернет-магазинів комп'ютерних комплектуючих, що дозволило визначити їх сильні та слабкі сторони, а також сформулювати перелік функціональних та нефункціональних вимог до розроблюваної системи. На основі отриманих результатів було сформульовано постановку задачі, яка передбачала створення адаптивного, продуктивного та масштабованого програмного рішення для онлайн-продажу комплектуючих.

У процесі проєктування програмної системи розроблено архітектуру інтернет-магазину, визначено структуру основних модулів та спроектовано базу даних із використанням PostgreSQL. Це дозволило забезпечити логічну організацію інформації про товари, користувачів, кошики та замовлення, а також створити основу для ефективної роботи серверної частини.

Реалізацію програмного продукту виконано із застосуванням сучасного стеку технологій: React для клієнтської частини, Node.js та Express.js для серверної логіки, PostgreSQL для зберігання даних. Використання React-Bootstrap дозволило сформувати зручний та адаптивний інтерфейс користувача, а модульний підхід до побудови серверної частини забезпечив структурованість, підтримуваність та можливість подальшого розвитку системи.

Проведене тестування підтвердило коректність роботи програмної системи за трьома основними напрямками: верстка, функціональність та адаптивність. Усі перевірені сценарії були виконані успішно, що свідчить про стабільність роботи вебзастосунку, сумісність із сучасними браузером та коректне відображення на мобільних пристроях.

Практична значущість отриманих результатів полягає у можливості використання розробленої системи як основи для впровадження повноцінного комерційного вебресурсу у сфері продажу комп'ютерної техніки. Запропоноване рішення може бути адаптоване до інших напрямів електронної торгівлі, а також використане як навчальний приклад у процесі вивчення веброзробки.

Подальший розвиток роботи доцільно спрямувати на інтеграцію платіжних систем, автоматизацію складського обліку, розширення аналітичного функціоналу та впровадження рекомендаційних механізмів на основі поведінки користувачів. Це дозволить підвищити конкурентоспроможність системи та розширити її практичне застосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rozetka для бізнесу | Торгсофт. URL: <https://torgsoft.ua/articles/stati/business-info/rozetka/> (дата звернення: 23.04.2026).
2. Правила користування інтернет-магазином | Kashemirmoda.com. URL: <https://www.kashemirmoda.com/stranky/pravila-koristyvannja-internet-magazinom> (дата звернення: 23.04.2026).
3. Інтернет-магазин під ключ. URL: <https://e-lavka.com.ua/internet-magazin> (дата звернення: 23.04.2026).
4. Інтернет-магазин телефонів і електроніки «Територія Твоїєї Техніки» - продаж мобільної техніки в Києві, Харкові, Дніпрі, Одесі, Львові – ТТТ. URL: <https://www.ttt.ua/ua> (дата звернення: 24.04.2026).
5. Інтернет-магазин ехе.ua: комп'ютери, комплектуючі, ноутбуки, мережеве обладнання, програмне забезпечення, сервери | Опт, роздріб, ПДВ, ТОВ. URL: <https://exe.ua/> (дата звернення: 24.04.2026).
6. "Пешка" - Магазин комп'ютерної техніки. URL: <https://peshka.com.ua/> (дата звернення: 24.04.2026).
7. Key UI-UX Elements for E-commerce | Blog | Trustline Digital. URL: <https://www.trustlinedigital.com/blogs/web-design-development/5-essential-ui-ux-principles-for-e-commerce> (date of access: 25.04.2026).
8. Nayyar S. Accessible Web Design: UX Tips for Better Usability. URL: <https://www.webability.io/blog/accessible-web-design-ux-tips-for-better-usability> (date of access: 25.04.2026).
9. React Component Based Architecture – GeeksforGeeks. URL: <https://www.geeksforgeeks.org/reactjs/react-component-based-architecture/> (date of access: 25.04.2026).
10. Index | Node.js v25.9.0 Documentation. URL: <https://nodejs.org/docs/latest/api/> (date of access: 26.04.2026).
11. PostgreSQL: PostgreSQL 9.4 Increases Flexibility, Scalability and Performance. URL: <https://www.postgresql.org/about/news/postgresql-94-increases->

flexibility-scalability-and-performance-1557/ (date of access: 26.04.2026).

12. API Design Best Practices | Meerako. URL: <https://www.meerako.com/blogs/api-design-best-practices-nodejs-rest-graphql> (date of access: 27.04.2026).

13. Importance of Software Testing – GeeksforGeeks. URL: <https://www.geeksforgeeks.org/software-testing/why-software-testing-is-important/> (date of access: 27.04.2026).

14. The Importance of Software Testing. URL: <https://www.computer.org/resources/importance-of-software-testing?source=resources> (date of access: 28.04.2026).

ДОДАТОК А
Текст програми

A.1 Текст файла userRouter.js

```
const Router = require('express')
const router = new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration', userController.registration)
router.post('/login', userController.login)
router.get('/auth', authMiddleware, userController.check)
router.delete('/account', authMiddleware, userController.deleteUser)
router.post('/account', authMiddleware, userController.changePassword)

module.exports = router
```

A.2 Текст файла deviceRouter.js

```
const Router = require('express')
const router = new Router()
const deviceController = require('../controllers/deviceController')
const authMiddleware = require('../middleware/authMiddleware')
const checkRole = require('../middleware/checkRoleMiddleware')

router.post('/', authMiddleware, checkRole('ADMIN'), deviceController.create)
router.get('/filters', deviceController.getFilters)
router.get('/', deviceController.getAll)
router.get('/:id', deviceController.getOne)
router.delete('/:id', authMiddleware, checkRole('ADMIN'),
deviceController.deleteDevice)

module.exports = router
```

A.3 Текст файла userController.js

```
const {User, Basket} = require('../models/models')
const ApiError = require('../error/ApiError')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
require('dotenv').config()
const generateJwt = (id, email, role, username) => {
  return jwt.sign(
```

```

    {id, email, role, username},
    process.env.SECRET_KEY,
    {expiresIn: '24h'}
  )
}

class UserController {
  async registration(req, res, next) {
    try {
      const { username, email, password } = req.body

      if (!username) {
        return next(ApiError.badRequest('Введіть логін'))
      }

      if (!email || !password) {
        return next(ApiError.badRequest('Некоректний email або пароль'))
      }

      const candidateEmail = await User.findOne({ where: { email } })
      if (candidateEmail) {
        return next(ApiError.badRequest('Користувач з таким email вже існує'))
      }

      const candidateUsername = await User.findOne({ where: { username } })
      if (candidateUsername) {
        return next(ApiError.badRequest('Користувач з таким логіном вже існує'))
      }

      const hashPassword = await bcrypt.hash(password, 5)

      const user = await User.create({
        username,
        email,
        role: 'USER',
        password: hashPassword
      })

      await Basket.create({ UserId: user.id })

      const token = generateJwt(user.id, user.email, user.role, user.username)

      return res.json({ token })
    } catch (e) {

```

```

    next(ApiError.internal(e.message))
  }
}

async login(req, res, next) {
  const {email, password} = req.body
  const user = await User.findOne({where: {email}})
  if(!user) {
    return next(ApiError.badRequest('Користувач з таким email не існує!'))
  }
  let comparePassword = await bcrypt.compareSync(password, user.password)
  if(!comparePassword) {
    return next(ApiError.badRequest('Вказано невірний пароль'))
  }
  const token = generateJwt(user.id, user.email, user.role, user.username)
  return res.json({token})
}

async check(req, res, next) {
  const token = generateJwt(req.user.id, req.user.email, req.user.role,
req.user.username)
  return res.json({token})
}

async deleteUser(req, res, next) {
  try {
    const id = req.user.id
    const { password } = req.body

    const user = await User.findOne({where: {id}})

    if(!user) {
      return next(ApiError.internal('Операція неможлива!'))
    }

    const isValidPassword = await bcrypt.compare(password, user.password)

    if(!isValidPassword) {
      return next(ApiError.badRequest('Невірний пароль!'))
    }

    await user.destroy()
    return res.json({message: "Обліковий запис видалено!"})
  }
}

```

```

    } catch(e) {
      return next(ApiError.badRequest(e.message))
    }
  }

  async changePassword(req, res, next) {
    try {
      const id = req.user.id
      const { oldPassword, newPassword } = req.body

      const user = await User.findOne({where: {id}})

      if (!user) {
        return next(ApiError.internal('Операція неможлива!'))
      }

      const isValidPassword = await bcrypt.compare(oldPassword, user.password)

      if(!isValidPassword) {
        return next(ApiError.badRequest('Невірний поточний пароль!'))
      }

      const newHashPassword = await bcrypt.hash(newPassword, 5)

      await user.update({ password: newHashPassword })

      const token = generateJwt(user.id, user.email, user.role, user.username)
      return res.json({ message: "Пароль змінено", token })
    } catch(e) {
      return next(ApiError.badRequest(e.message))
    }
  }
}

module.exports = new UserController()

```

A.4 Текст файлу deviceController.js

```

const { Device, DeviceInfo, FilterType, FilterValue } = require('../models/models')
const { Op } = require('sequelize')
const ApiError = require('../error/ApiError')
const uuid = require('uuid')

```

```

const path = require('path')

class DeviceController {

  async create(req, res, next) {
    try {
      let { name, price, BrandId, TypeId, info } = req.body
      const { img } = req.files

      const fileName = uuid.v4() + ".jpg"
      img.mv(path.resolve(__dirname, '..', 'static', fileName))

      const device = await Device.create({
        name,
        price,
        BrandId,
        TypeId,
        img: fileName
      })

      if (info) {
        info = JSON.parse(info)

        await Promise.all(
          info.map(i =>
            DeviceInfo.create({
              title: i.title,
              description: i.description,
              deviceId: device.id
            })
          )
        )
      }

      return res.json(device)

    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async getAll(req, res, next) {
    try {
      let { BrandId, TypeId, page, limit, sort, filters } = req.query

```

```

page = page || 1
limit = limit || 9
let offset = page * limit - limit

let where = {}

if (BrandId) {
  where.BrandId = BrandId
}

if (TypeId) {
  where.TypeId = TypeId
}

let order = []

switch (sort) {
  case 'price_asc':
    order = [['price', 'ASC']]
    break

  case 'price_desc':
    order = [['price', 'DESC']]
    break

  case 'rating':
    order = [['rating', 'DESC']]
    break

  default:
    order = [['id', 'ASC']]
    break
}

let include = []

if (filters) {
  let filterIds = []

  if (typeof filters === 'string') {
    filterIds = filters.split(',').map(Number)
  } else if (Array.isArray(filters)) {
    filterIds = filters
  }
}

```

```

        include.push({
          model: FilterValue,
          where: {
            id: {
              [Op.in]: filterIds
            }
          },
          through: { attributes: [] }
        })
      }

      const devices = await Device.findAndCountAll({
        where,
        include,
        limit,
        offset,
        order,
        distinct: true
      })

      return res.json(devices)
    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async getOne(req, res, next) {
    try {
      const { id } = req.params

      const device = await Device.findOne({
        where: { id },
        include: [{ model: DeviceInfo, as: 'info' }]
      })

      return res.json(device)
    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async deleteDevice(req, res, next) {
    try {

```

```

const { id } = req.params

const device = await Device.findOne({
  where: { id }
})

if (!device) {
  return next(ApiError.badRequest('Пристрій не знайдено'))
}

await device.destroy()

return res.json({ message: 'Пристрій видалено' })

} catch (e) {
  next(ApiError.badRequest(e.message))
}
}

async getFilters(req, res, next) {
  try {
    const { typeId } = req.query

    const filters = await FilterType.findAll({
      where: { TypeId: typeId },
      include: [{
        model: FilterValue,
        as: 'FilterValues'
      }]
    })

    return res.json(filters)

  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}

module.exports = new DeviceController()

```

A.5 Текст файла deviceController.js

```

import React, { useContext } from "react";
import { Context } from "..";
import { ADMIN_ROUTE, BASKET_ROUTE, LOGIN_ROUTE, SHOP_ROUTE,
USER_ACCOUNT } from "../utils/consts";

import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import Button from 'react-bootstrap/Button';
import { observer } from "mobx-react-lite";
import { NavLink, useNavigate } from "react-router-dom";
import SearchBar from "../SearchBar";
import NavDropdown from 'react-bootstrap/NavDropdown';

import basketIcon from '../assets/basket.png';
import adminIcon from '../assets/admin.png';
import userIcon from '../assets/user.png';
import logoutIcon from '../assets/logout.png';

const NavBar = observer(() => {
  const {user} = useContext(Context);
  const navigate = useNavigate();

  return(
    <Navbar bg="dark" data-bs-theme="dark" className="py-2 shadow-sm">
      <Container>

        <NavLink
          to={SHOP_ROUTE}
          style={{
            color: "white",
            textDecoration: "none",
            fontWeight: "bold",
            fontSize: "18px"
          }}
        >
          Головна
        </NavLink>

        <div style={{width: '40%'}}>
          <SearchBar/>
        </div>

```

```

{user.isAuth ?
<Nav className="d-flex align-items-center gap-2">

  <Button
    variant="dark"
    className="d-flex align-items-center gap-2"
    onClick={() => navigate(BASKET_ROUTE)}
  >
    <img src={basketIcon} alt="" width={20} height={20}/>
    Кошик
  </Button>

  {user.isAdmin && (
    <Button
      variant="dark"
      className="d-flex align-items-center gap-2"
      onClick={() => navigate(ADMIN_ROUTE)}
    >
      <img src={adminIcon} alt="" width={20} height={20}/>
      Адмін панель
    </Button>
  )}

  <NavDropdown
    title={<span className="text-white">{user.user.username}</span>}
    align="end"
  >
    <NavDropdown.Item onClick={() => navigate(USER_ACCOUNT)}>
      <div className="d-flex align-items-center gap-2">
        <img src={userIcon} alt="" width={20} height={20}/>
        Обліковий запис
      </div>
    </NavDropdown.Item>
    <NavDropdown.Divider />
    <NavDropdown.Item onClick={() => user.logout()}>
      <div className="d-flex align-items-center gap-2">
        <img src={logoutIcon} alt="" width={20} height={20}/>
        Вийти
      </div>
    </NavDropdown.Item>
  </NavDropdown>

</Nav>
:

```

```

        <Nav>
            <Button          variant="primary"          onClick={() =>
navigate(LOGIN_ROUTE)}}>
                Авторизуватись
            </Button>
        </Nav>
    }

    </Container>
</Navbar>
)
});

export default NavBar;

```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Слайд 1

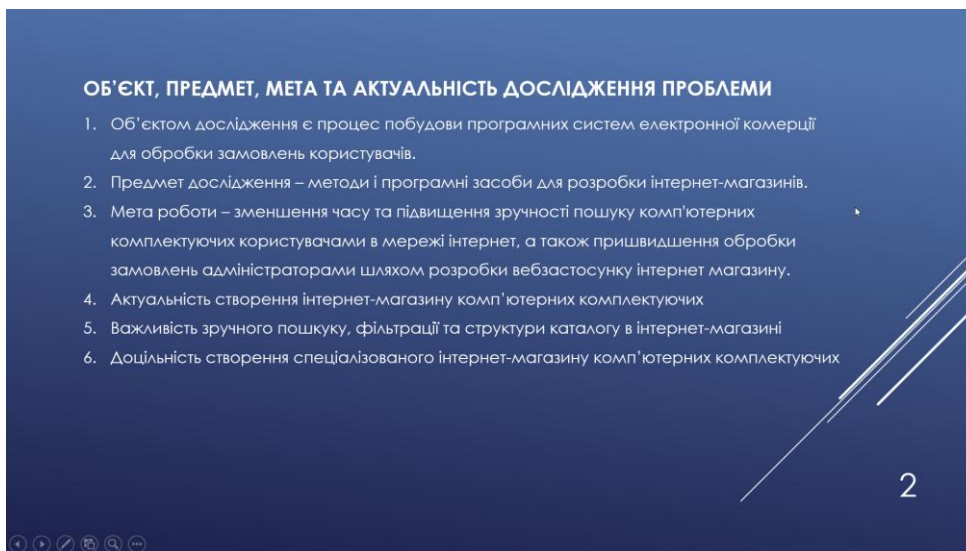


Рисунок Б.2 – Слайд 2



Рисунок Б.3 – Слайд 3

ПОРІВНЯННЯ СТЕКІВ РОЗРОБКИ

Таблиця 2 – Порівняння стеків розробки

Критерій	React + Node.js + PostgreSQL	Angular + Java + Spring + MySQL	Vue.js + Laravel + MySQL
Швидкість розробки	Висока	Середня	Висока
Продуктивність	Висока	Висока	Середня
Масштабованість	Висока	Дуже висока	Середня
Зручність підтримки	Висока	Складніша структура	Висока
Єдина мова програмування	Так (JavaScript)	Ні	Часткова
Підходить для SPA	Так	Так	Так
Робота з великим навантаженням	Висока	Висока	Середня
Гнучкість архітектури	Висока	Середня	Висока

4

Рисунок Б.4 – Слайд 4

ПРОЄКТУВАННЯ СТРУКТУРИ ПРОГРАМНОЇ СИСТЕМИ

Обрана клієнт-серверна архітектура

Система поділяється на:

- клієнтську частину
- серверну логіку
- базу даних

Переваги:

- гнучкість
- масштабованість
- зручність супроводу

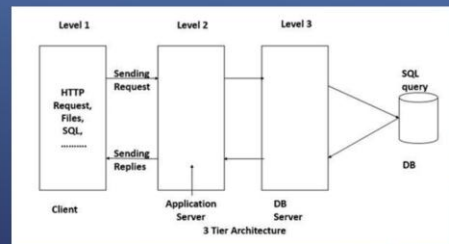


Рисунок 1 – Клієнт-серверна архітектура

5

Рисунок Б.5 – Слайд 5

СТРУКТУРА ПРОЄКТУ

- Backend: маршрути, контролери, моделі, middleware;
- Database: міграції;
- Конфігурація та змінні середовища
- Frontend: компоненти, сторінки, store
- HTTP-запити та утиліти
- Статичні ресурси й assets

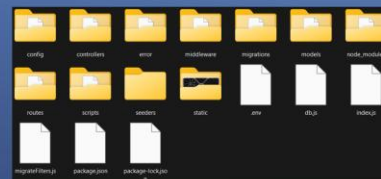


Рисунок 2 – структура серверу

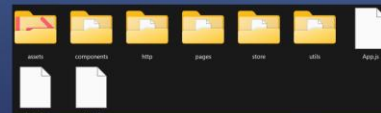


Рисунок 3 – структура клієнту

6

Рисунок Б.6 – Слайд 6

ОСНОВНІ МОДУЛІ СИСТЕМИ

Головна сторінка – точка входу

Каталог товарів – список комплектуючих

Сторінка товару – опис, характеристика, ціна

Пошук – швидке знаходження продукції

Фільтри – категорії, бренди, ціна

Кошик – збереження вибраних товарів

Особистий кабінет – персоналізований доступ



Рисунок 4 – Структурна схема

7

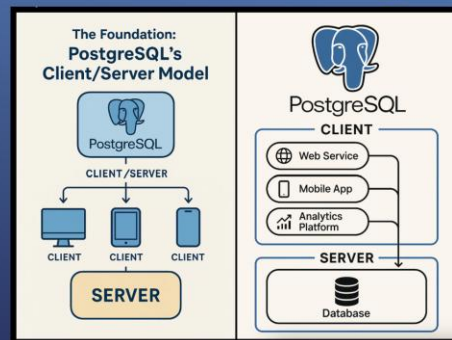
Рисунок Б.7 – Слайд 7

ПРОЄКТУВАННЯ БАЗИ ДАНИХ

Використано PostgreSQL

Причини вибору:

- ▶ висока продуктивність
- ▶ надійність
- ▶ масштабованість
- ▶ підтримка складних SQL-запитів
- ▶ забезпечення цілісності даних



8

Рисунок Б.8 – Слайд 8

ОСНОВНІ ТАБЛИЦІ БД

Таблиця	Призначення
Devices	Товари
DeviceInfos	Характеристики товарів
Types	Типи товарів
Brands	Бренди товарів
FilterTypes	Системи фільтрації
FilterValues	
DeviceFilters	
Users	Користувачі
Baskets	Кошик товарів
BasketDevices	
Ratings	Рейтинг товару

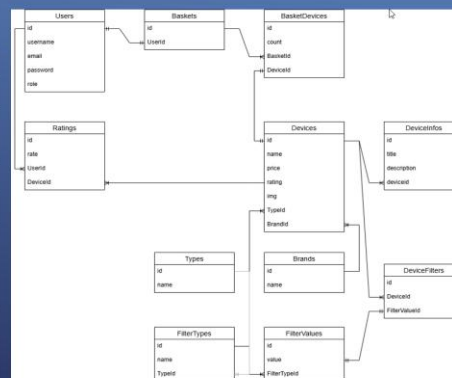


Рисунок 5 – Схема БД

9

Рисунок Б.9 – Слайд 9

МОЖЛИВОСТІ КОРИСТУВАЧІВ

Для користувача:

- перегляд каталогу
- пошук і фільтрація
- реєстрація та авторизація
- особистий кабінет
- оцінювання товарів
- додавання до кошика
- оформлення замовлення

Для адміністратора:

- управління товарами
- управління категоріями
- Брендами
- характеристиками

Рисунок 6 – Use Case діаграма

10

Рисунок Б.10 – Слайд 10

АЛГОРИТМ РОБОТИ ПРОГРАМНОЇ СИСТЕМИ

Алгоритм роботи покупки товару представлено у вигляді блок-схеми, яка наочно демонструє послідовність виконання основних дій.

Рисунок 7 – схема алгоритму системи

11

Рисунок Б.11 – Слайд 11

РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ

Рисунок 8 – Головна сторінка

Рисунок 9 – Особистий кабінет

Рисунок 10 – Сторінка товару

Рисунок 11 – Адмін панель

12

Рисунок Б.12 – Слайд 12

ТЕСТУВАННЯ СИСТЕМИ

- ▶ Тестування верстки – Таблиця 1
- ▶ Тестування функціоналу – Таблиця 2
- ▶ Тестування адаптивності – Таблиця 3

Примітки:

Google Chrome (v.147.0.7727.116)

Opera One (v.130.0.5847.41)

Microsoft Edge (v.142.0.3595.94)

Таблиця 3 – Тестування верстки

Критерій	Google Chrome	Opera One	Microsoft Edge
Перевірити наявність підказки сайту	Пройдено	Пройдено	Пройдено
Перевірити наявність підказки сайту	Пройдено	Пройдено	Пройдено
Перевірити наявність рядка пошуку	Пройдено	Пройдено	Пройдено

Таблиця 4 – Функціональне тестування

Критерій	Google Chrome	Opera One	Microsoft Edge
Перевірити реєстрацію на сайті	Пройдено	Пройдено	Пройдено
Валідація на порожні поля	Пройдено	Пройдено	Пройдено
Додати товар в кошик	Пройдено	Пройдено	Пройдено

Таблиця 5 – Тестування адаптивності

Критерій	Google Chrome	Opera One	Microsoft Edge
Перевірити коректне відображення сторінок	Пройдено	Пройдено	Пройдено
Перевірити коректне відображення кнопок блоку меню та ін.	Пройдено	Пройдено	Пройдено
Перевірка переходу на сторінку після натискання на елементи сайту	Пройдено	Пройдено	Пройдено

13

Рисунок Б.13 – Слайд 13

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи було розроблено програмну систему у вигляді інтернет-магазину комп'ютерних комплектуючих, яка відповідає сучасним вимогам електронної комерції та забезпечує зручний процес вибору, пошуку й придбання товарів.

У ході роботи було проведено аналіз предметної області, визначено основні вимоги до системи, спроектовано її архітектуру, структуру бази даних та функціональні модулі. Для реалізації програмного продукту було обрано сучасний технологічний стек, що забезпечує високу продуктивність, масштабованість і зручність подальшого супроводу.

Розроблена система включає основний функціонал для користувачів та адміністраторів: каталог товарів, систему пошуку та фільтрації, кошик, особистий кабінет, оцінювання продукції, а також засоби керування асортиментом. Це дозволяє ефективно організувати процес онлайн-продажу комп'ютерних комплектуючих.

Практична цінність роботи полягає у можливості використання створеної системи як готового рішення для ведення електронного бізнесу в сфері продажу комп'ютерної техніки. Результати дослідження та реалізації можуть бути основою для подальшого розширення функціоналу, інтеграції платіжних сервісів, аналітики продажів та мобільної адаптації.

Таким чином, поставлена мета роботи була досягнута, а всі визначені завдання — успішно виконані.

14

Рисунок Б.14 – Слайд 14