БАКАЛАВРСЬКА РОБОТА

БР.КІ-09.00.000 ПЗ

Група КІ-21-1

Маковійчук Артем

2025

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій

Кафедра комп'ютерних систем і мереж

Маковійчук Артем Ігорович

УДК 004.4

БАКАЛАВРСЬКА РОБОТА

Розробка web-сервісу для організації процесів конфігурування, складання та продажу комп'ютерів засобами PostgreeSQL,Vite, RestApi

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня *Маковійчук А.І.*
(підпис, ініціали та прізвище здобувача)

Науковий керівник *Бузоверя Н.Г., доцент*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,проф. *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Інститут *Інформаційних технологій*

Кафедра *Комп'ютерних систем та мереж*

Спеціальність 123 - *Комп'ютерна інженерія*

Освітній рівень *– бакалавр*

ЗАТВЕРДЖУЮ

Завідувач кафедри КСМ

(С.І. Мельничук)

*« 25» квітня 2025*року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Маковійчуку Артему Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема дипломної роботи *Розробка web-сервісу для організації процесів конфігурування, складання та продажів комп'ютерів засобами PostgreeSQL, Vite, RespApi*

Керівник проекту (роботи) *Бузоверя Надія Генадіївна, доцент*

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

затверджені наказом вищого навчального закладу від *05.05.2025 № 275/7*

2. Термін здачі студентом закінченої роботи *12 червня 2025 р*

3. Вихідні дані до проекту (роботи) *Методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз існуючих веб-сервісів для конфігурування та продажу комп'ютерної техніки. 2. Розробка архітектури клієнт-серверного web-додатку для збирання ПК. 3. Реалізація backend-частини веб-сервісу на Node.js з використанням PostgreSQL та REST API. 4. Реалізація frontend-інтерфейсу за допомогою React + Vite з підтримкою ролей користувачів (user/admin). 5. Опис механізму керування компонентами, конфігураціями ПК та замовленнями. 6. Забезпечення захисту та авторизації користувачів у системі. 7.Висновки щодо результатів розробки веб-сервісу.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): –

6. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата

6. Дата видачі завдання *29 січня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	*Збір інформації, вивчення літератури та пошук додаткової інформації*	*Лютий, 2025*	
2	*Аналіз предметної області та технічних засобів*	*Березень, 2025*	
3	*Проєктування web-сервісу*	*Квітень, 2025*	
4	*Реалізація системи*	*Травень, 2025*	
5	*Оформлення пояснювальної записки*	*Травень, 2025*	

Студент _____ *Маковійчук А.І.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Бузоверя Н.Г.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою даної бакалаврської роботи є розробка web-сервісу для автоматизації процесів збирання та продажу комп’ютерів на базі технологій PostgreSQL, REST API та Vite.

У першому розділі проведено аналіз предметної області, розглянуто існуючі програмні рішення у сфері продажу комп’ютерної техніки, а також обґрунтовано вибір технологічного стеку для реалізації системи.

У другому розділі розроблено архітектуру майбутнього web-сервісу, побудовано інформаційну модель бази даних, спроєктовано REST API та описано користувацький інтерфейс.

У третьому розділі реалізовано серверну та клієнтську частини проєкту, протестовано функціональність системи, перевірено її працездатність та виконано оцінку продуктивності.

Ключові слова: web-сервіс, PostgreSQL, REST API, Vite, збирання комп’ютерів, автоматизація.

ANNOTATION

The aim of this bachelor's thesis is to develop a web service to automate the processes of assembling and selling computers using PostgreSQL, REST API, and Vite technologies.

The first chapter analyzes the subject area, examines existing software solutions in the field of computer hardware sales, and justifies the choice of the technology stack for the system.

The second chapter focuses on the architecture of the future web service, presenting the information model of the database, the design of the REST API, and the user interface layout.

The third chapter covers the implementation of both the server and client parts of the project, testing of the system functionality, verification of its operability, and performance evaluation.

Keywords: web service, PostgreSQL, REST API, Vite, computer assembly, automation.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

API — Application Programming Interface (програмний інтерфейс взаємодії між компонентами програми)

БД — база даних

CRUD — Create, Read, Update, Delete (основні операції з даними в системі)

ER-діаграма — Entity-Relationship діаграма, модель зв’язків сутностей у базі даних

HTTPS — HyperText Transfer Protocol Secure (захищений протокол передавання даних)

JWT — JSON Web Token (формат передачі токенів автентифікації)

UI — User Interface (користувацький інтерфейс)

UX — User Experience (досвід користувача)

REST — Representational State Transfer (архітектурний стиль для побудови API)

Node.js — середовище виконання JavaScript-коду на стороні сервера

Express — фреймворк для Node.js для створення серверів і маршрутизації

React — бібліотека JavaScript для побудови інтерфейсів користувача

Vite — сучасний інструмент збірки та розробки фронтенду

JWT — токен для аутентифікації користувача, що зберігає інформацію у форматі JSON

SQL — Structured Query Language (мова запитів до бази даних)

JSON — JavaScript Object Notation (формат обміну даними між клієнтом і сервером)

.env — конфігураційний файл для зберігання змінних середовища

CRUD-операції — набір базових дій у базі даних: створення, читання, оновлення, видалення записів

ВСТУП

Актуальність. В умовах цифровізації бізнесу та стрімкого розвитку IT-сектору актуальним залишається завдання автоматизації процесів збирання та реалізації комп'ютерної техніки. Сучасні компанії стикаються з потребою в швидкому підборі конфігурацій ПК, точному управлінні складськими залишками, а також у зручному оформленні замовлень онлайн. У цьому контексті розробка інтуїтивно зрозумілих і технологічно гнучких web-сервісів є ефективним інструментом для підвищення продуктивності, мінімізації людського фактору та покращення взаємодії з клієнтами.

Об'єктом дослідження є процес розробки інформаційної системи у вигляді веб-сервісу для збирання та продажу комп'ютерів.

Предметом дослідження є методи та засоби розробки інформаційної системи у вигляді веб-сервісу для збирання та продажу комп'ютерів з використанням PostgreSQL, REST API та Vite.

Мета роботи. Розробка web-сервісу, що забезпечить автоматизацію процесів управління комп'ютерними компонентами, створення конфігурацій ПК, оформлення замовлень та управління користувацькими ролями у системі. Для досягнення мети необхідно виконати такі завдання:

1. Здійснити аналіз існуючих рішень у сфері продажу комп'ютерної техніки.
2. Проектування архітектури інформаційної системи.
3. Створення бази даних.
4. Реалізація серверної частини засобами Node.js, Express та PostgreSQL.
5. Реалізація клієнтської частини засобами Vite та React.
6. Провести тестування функціональності.

Методи дослідження включають аналіз предметної області, моделювання архітектури системи, проектування бази даних на основі PostgreSQL, та реалізацію клієнт-серверної взаємодії через REST API з використанням Vite. Результати дослідження дозволяють створити маштабовану, гнучку та швидкодіючу платформу для підприємств, що займаються індивідуальним збиранням комп’ютерної техніки.

Практична цінність розробки веб-сервісу для організації процесів конфігурування, складання та продажу комп’ютерів полягає у створенні ефективного інструменту для автоматизації бізнес-процесів, що забезпечує зручність для користувачів та підвищує продуктивність працівників. Сервіс дозволяє інтегрувати інтерфейс конфігурування комплектуючих, керування замовленнями та відображення актуальної інформації про складські запаси.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ

1.1 Аналіз існуючих рішень для збирання та продажу комп’ютерної техніки

У сучасному цифровому середовищі існує безліч компаній та онлайн магазинів, які пропонують послуги з індивідуального збирання комп'ютерів. Серед популярних платформ, що реалізують такі функції, можна виділити міжнародні сервіси, такі як PCPartPicker, Logical Increments, CyberPowerPC, а також локальні українські компанії, такі як Brain, Itbox та CompX. Основні функції цих систем включають інтерактивний підбір комплектуючих з перевіркою їх сумісності, розрахунок вартості, візуалізацію конфігурації та оформлення замовлення. Часто вони також надають можливість перегляду відгуків, рейтинги компонентів, підтримку користувацького кабінету, а також можливість збереження збірок і їх подальшої модифікації.

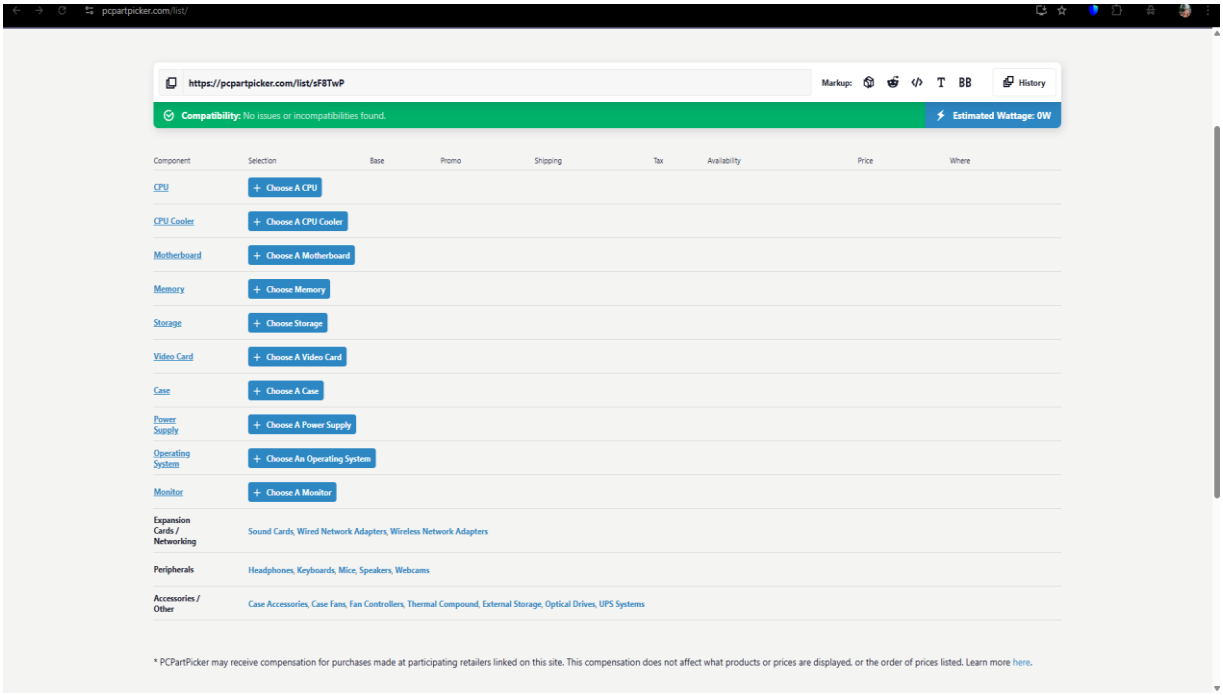

Рисунок 1.1 – Інтерфейс конструктора збірки комп’ютера на PCPartPicker

Як видно з інтерфейсу конструктора PCPartPicker, він орієнтований на технічно підготовленого користувача та забезпечує детальний вибір комплектуючих. Проте, для звичайного споживача інтерфейс може здатися перевантаженим через велику кількість параметрів і технічних характеристик. Це створює необхідність у більш інтуїтивному рішенні для українського ринку.

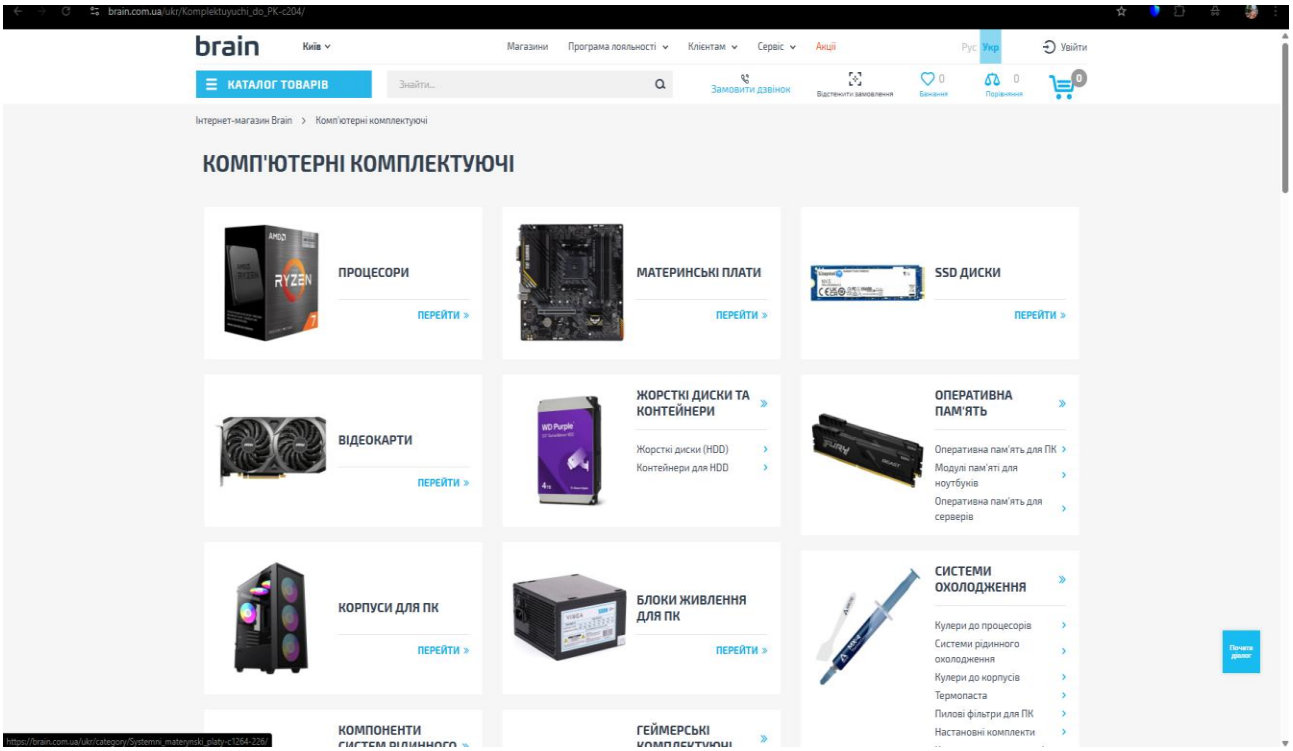

Рисунок 1.2 – Інтерфейс підбору комплектуючих у Brain.com.ua

Інтерфейс Brain.com.ua виглядає простішим і зручнішим для пересічного користувача. Однак, він має обмежений функціонал у частині створення власної конфігурації та перевірки сумісності компонентів у реальному часі. Це обмежує його застосування у складніших сценаріях збирання ПК.

Переглянемо ще один Український конфігуратор під назвою CompX, щоб побачити які можливісто для створення різних конфігурацій ПК він надає, та побачити різницю попередніми та наступники конфігураторами.

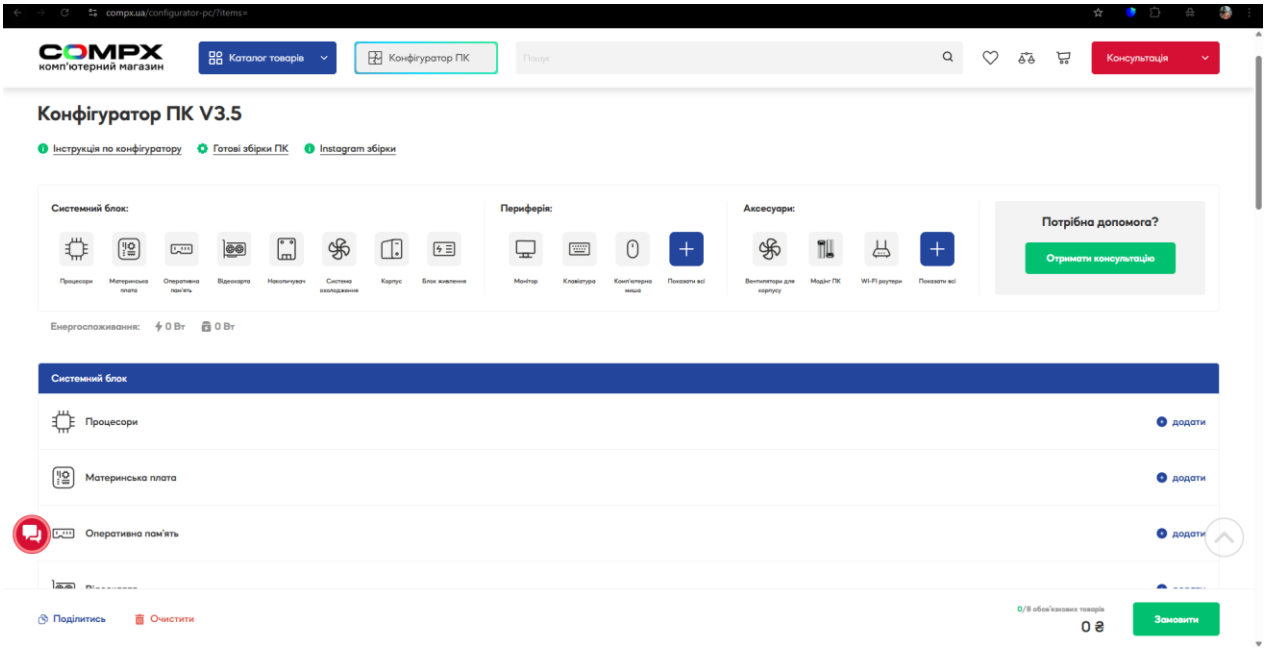

Рисунок 1.3 – Інтерфейс генератора конфігурацій CompX.ua

Розглянутий інтерфейс онлайн-сервісу CompX є прикладом спрощеного конструктора ПК, орієнтованого на українського споживача. Незважаючи на функціональність, яка дозволяє обрати компоненти та переглянути сумісність, такі сервіси рідко надають можливість гнучкого адміністрування, зміни ролей або повноцінної авторизації. Це і визначає актуальність створення власного рішення, адаптованого до специфіки конкретного бізнес-процесу.

Таблиця 1.1 – Порівняння функціональності онлайн-сервісів збирання ПК

Назва сервісу	Перевірка сумісності	Калькуляція ціни	Замовлення збірки	Рекомендації по збірці	Авторизація
PCPartPicker	+	+	-	+	+
Brain.com.ua	+	+	+	-	+
CompX	+	+	+	-	+
Прототип дипломного	+	+	+	+	+

Таблиця 1.1 узагальнює основні функції, які підтримуються популярними сервісами збирання ПК. Вона дає змогу наочно порівняти рівень інтерактивності, наявність авторизації, підтримку рекомендацій та можливість замовлення

конфігурацій. Таке зіставлення дозволяє зробити висновок про доцільність розробки власного рішення, яке б поєднувало переваги існуючих систем.

Однак більшість існуючих рішень є або комерційними платформами з закритим кодом, або складними ERP-системами, які важко адаптувати під індивідуальні потреби малого бізнесу. У таких умовах створення власного веб-сервісу дозволяє отримати гнучке рішення, яке враховує специфіку конкретної організації. Таким чином, розробка індивідуального веб-сервісу є доцільною як у навчальному контексті, так і для впровадження в практичну діяльність суб'єктів малого підприємництва, що займаються збиранням та реалізацією ПК.

1.2 Особливості предметної області

У процесі дослідження предметної області було проаналізовано низку нормативних, науково-технічних та законодавчих джерел, що регламентують створення та експлуатацію веб-систем для електронної торгіволі та автоматизації бізнес-процесів.

До ключових нормативних джерел, що забезпечують законодавчу та методологічну основу, належить:

- закон України «Про захист персональних даних» (№2297-VI від 01.06.2010), який регламентує правила зберігання, обробки і передачі персональної інформації користувачів;

- закон України «Про електронну комерцію» (№675-VII від 03.09.2015), що

визначає правові засади електронних правочинів, ідентифікація суб’єктів та захисту споживача;

- ДСТУ ISO/IEC 27001:2015 (Інформаційні технології – Методи забезпечення безпеки – Системи управління інформаційної безпекою), як рекомендація для формування вимог до безпеки системи;

- посібники та технічна документація по REST API, PostgreeSQL, Vite, React.js, а також best practices, оприлюднені в офіційній

- документаціїрозробників (MDN, React Docs, PostgreeSQL Doc, OWASP, тощо).

У контексті досліджуваного веб-сервісу для збирання та продажу комп’ютерів, до інформативних параметрів, що підлягли обліку та опрацювання під час розробки, належить:

- ідентифікаційні дані користувача (email,ID, роль – user/admin);
- характеристика компонентів (назва, тип, опис, ціна, кількість на складі, зображення);
- конфігурації ПК (назва, опис, список компонентів, загальна ціна);
- інформація про замовлення (ідентифікатор замовлення, дата створення, статус, ID користувача, ID конфігурації;
- адміністративні параметри (роль доступу, зміна статусів, редагування бази компонентів та замовлень);
- атрибути безпеки (JWT-токени, автентифікація, перевірка прав доступу через middleware).

Сфера збирання та продажу комп'ютерів характеризується високою динамікою та постійними змінами в актуальних компонентах. Постачальники часто змінюють асортимент, а замовники очікують значної гнучкості у виборі конфігурацій. Цей тип діяльності вимагає точного обліку наявних деталей, зручного механізму їх підбору та швидкої обробки замовлень. Особливу роль відіграє сумісність компонентів: не кожен процесор підходить до кожної материнської плати, і не кожна відеокарта може функціонувати в слабкій системі живлення.

Бізнеси, що працюють у цій галузі, повинні інтегрувати в робочий процес інформаційну систему, яка дозволяє зберігати повні дані про кожен елемент (тип, технічні характеристики, вартість, залишок на складі), забезпечувати швидкий пошук, перевірку сумісності та оформлення замовлень із можливістю адаптації під індивідуальні вимоги клієнтів. Важливим чинником у цій предметній області є оновлюваність даних та потреба в синхронізації між базою комплектуючих і реальним станом на складі. Також необхідно забезпечити

надійність системи та її доступність для співробітників і клієнтів. Ці особливості мають бути враховані під час розробки архітектури веб-сервісу.

1.3 Проблеми автоматизації бізнес-процесів

Незважаючи на широке впровадження цифрових технологій, багато підприємств, що займаються реалізацією комп'ютерної техніки, досі використовують напівавтоматизовані або навіть ручні процеси для ведення обліку товарів і обслуговування клієнтів. Це створює ризики, пов'язані з помилками в замовленнях, затримками в обробці даних, неактуальною інформацією про наявність товарів, а також ускладнює масштабування бізнесу.

Однією з основних проблем є відсутність інтеграції між обліковими системами, інтерфейсами замовлення та базами даних комплектуючих. У результаті співробітники змушені дублювати дії, вводячи дані в кілька систем або вручну звіряючи інформацію про наявність деталей. Такий підхід уповільнює роботу й підвищує ймовірність людських помилок.

Ще однією проблемою є обмежена гнучкість типових програмних рішень, які не дозволяють адаптувати логіку під конкретні потреби малого або середнього підприємства. Наприклад, у системах часто відсутні механізми перевірки сумісності компонентів або динамічного підбору конфігурацій. Також поширеною є відсутність єдиного доступу до інформації з боку клієнтів, співробітників і менеджерів, що ускладнює обслуговування, аналітику та планування закупівель. Відсутність веб-доступу до функціоналу призводить до того, що робота системи обмежується лише локальним середовищем офісу чи складу.

Автоматизація процесів, включаючи централізоване управління складом, динамічне оновлення каталогу комплектуючих, автоматичний підбір сумісних деталей, а також інтеграцію з формами замовлень, дозволяє суттєво підвищити ефективність і зменшити операційні витрати. Ці цілі ставляться перед розробкою власного веб-сервісу.

1.4 Обґрунтування вибору технологічного стеку

Для реалізації веб-сервісу, що відповідає сучасним вимогам до продуктивності, масштабованості та зручності використання, був обраний технологічний стек, який включає PostgreSQL як систему керування базами даних, REST API як механізм взаємодії між клієнтською та серверною частинами, а також Vite як інструмент для побудови фронтенду.

PostgreSQL була обрана через її відкритий код, високу стабільність, підтримку складних SQL-запитів, а також широкі можливості роботи з транзакціями та індексами. Ця СКБД є однією з найнадійніших і найефективніших для реляційного зберігання інформації, що ідеально підходить для систем, де важливе швидке виконання запитів до великого обсягу даних, як у випадку з товарами, замовленнями та користувачами.

REST API (Representational State Transfer) забезпечує стандартизований спосіб обміну інформацією між клієнтом і сервером через HTTP-запити. Його використання дозволяє створювати масштабовані сервіси з чітко визначеними ендпоінтами, що спрощує інтеграцію з іншими системами та клієнтськими додатками. REST API легко тестується та розширюється, що важливо для підтримки та модернізації проєкту в майбутньому.

Для фронтенд-розробки був обраний Vite — сучасний збирач і розробницьке середовище для JavaScript-проєктів. Він забезпечує миттєве оновлення під час розробки (hot module replacement), оптимізацію ресурсів для продакшн-збірки та просту інтеграцію з React або іншими сучасними фреймворками. Завдяки швидкості роботи та простоті налаштування Vite є ефективним рішенням для створення інтерфейсу користувача.

Сукупність цих технологій дозволяє забезпечити високу надійність, модульність і зручність підтримки програмного продукту, а також закладає основу для подальшого масштабування системи — як функціонального, так і за кількістю користувачів.

1.5 Постановка завдання

Метою цієї бакалаврської роботи є розробка сучасного веб-сервісу, який забезпечує зручну платформу для збирання комп'ютерів на основі обраних комплектуючих, їх продажу та обробки замовлень. Такий сервіс має оптимізувати процес взаємодії між кінцевим користувачем та системою, надаючи можливість перегляду доступних компонентів, створення індивідуальних конфігурацій, оформлення замовлень та адміністрування вмісту сайту.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати предметну область та дослідити особливості функціонування подібних онлайн-сервісів для збору ПК;
- визначити функціональні та нефункціональні вимоги до програмного забезпечення;
- обґрунтувати вибір архітектури та стеку технологій, зокрема PostgreSQL, Vite, React, Node.js, Express;
- розробити структуру бази даних, яка забезпечує зберігання інформації про компоненти, замовлення, користувачів і конфігурації;
- створити модулі серверної частини, які реалізують REST API для взаємодії з клієнтом;
- розробити клієнтську частину веб-додатку з адаптивним інтерфейсом та логікою взаємодії з API;
- забезпечити автентифікацію та авторизацію користувачів із поділом на ролі (адміністратор / звичайний користувач);
- реалізувати функції адміністрування: керування компонентами, конфігураціями, замовленнями та користувачами;
- провести тестування функціональності веб-сервісу на основі стандартних кейсів (реєстрація, вхід, створення замовлення, тощо);
- оцінити ефективність і переваги розробленого рішення, а також можливості подальшого розширення функціоналу.

Реалізація поставлених завдань дозволить створити ефективний інструмент для підтримки процесу підбору комп'ютерних комплектуючих, що може використовуватися як у комерційних цілях, так і для навчальних або демонстраційних потреб.

У процесі реалізації програмного продукту було використано низку сучасних засобів, які забезпечили ефективну розробку, тестування та супровід веб-сервісу для організації зі збирання та продажу комп'ютерів. Вибір інструментів обумовлювався як функціональними потребами проєкту, так і вимогами щодо зручності, продуктивності та підтримки сучасних веб-технологій.

Середовище розробки.

Основною інтегрованою платформою для створення коду виступила Visual Studio Code – легкий, кросплатформений текстовий редактор із розширеною підтримкою JavaScript, Node.js та React. Завдяки широкому вибору розширень, таких як Prettier, ESLint, PostgreSQL, Thunder Client, середовище розробки було адаптовано під потреби як серверної, так і клієнтської частини проєкту. Використання вбудованого терміналу значно пришвидшило процес запуску, дебагу та контролю над локальним сервером.

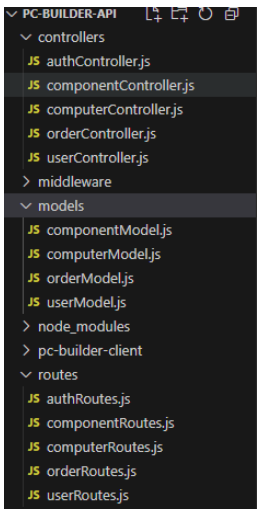

Рисунок 1.4 - Робоче середовище VS Code, серверна частина

На рисунку зображено середовище розробки Visual Studio Code під час роботи над серверною частиною проєкту. Структура файлів поділена за

логічними модулями, зокрема: routes, controllers, models тощо. Такий підхід забезпечує зручну навігацію та підтримку чистої архітектури.

Для налаштування зручної структури проєкту використовувалися шаблонізатори типу npm create vite@latest, що забезпечує швидкий старт для фронтенд-проєктів на React. Крім того, завдяки Vite був забезпечений швидкий hot-reload та оптимізована збірка у продакшен.

Серверна частина:

Для реалізації серверної логіки був обраний Node.js у поєднанні з Express.js – фреймворком, що дозволяє швидко створити REST API з чіткою маршрутизацією та структурованим розподілом по контролерах і моделях. Express забезпечив обробку HTTP-запитів, захист маршрутів, обробку JWT-токенів, роботу з middleware, а також взаємодію з базою даних PostgreSQL.

Для зберігання даних обрано PostgreSQL – потужну реляційну СУБД, яка підтримує транзакції, зв'язки між таблицями, індекси, агрегатні функції та складні запити. Зв'язок між Node.js і PostgreSQL було реалізовано за допомогою офіційного пакету pg, який дозволяє надсилати параметризовані запити до бази, захищаючи систему від SQL-ін'єкцій.

Клієнтська частина.

Фронтенд частина була реалізована за допомогою React – бібліотеки, яка дозволяє створювати багаторазові компоненти, управляти станами та маршрутизацією сторінок. Для побудови інтерфейсу використовувався React Router, який дозволяє реалізувати SPA (Single Page Application) із динамічною зміною URL без перезавантаження сторінки.

У процесі стилізації використовувався Bootstrap 5, що дозволило швидко створити адаптивний, акуратний інтерфейс з готовими класами оформлення, а також покращити юзабіліті без додаткової ручної верстки.

Тестування:

Для перевірки API та аутентифікації використовувався Thunder Client – вбудований REST-клієнт у VS Code, який дозволив легко формувати запити типу

POST, PUT, DELETE та GET до серверної частини проєкту, передавати JWT-токени в заголовках і спостерігати за відповідями у зручному форматі.

Управління версіями:

Усі файли проєкту контролювалися за допомогою Git, а для зберігання репозиторію використовувався GitHub. Це дозволило створювати резервні копії, відстежувати зміни, керувати гілками розробки та працювати з pull-запитами, навіть у локальному режимі.

У першому розділі було здійснено аналіз існуючих веб-сервісів для збирання комп'ютерів. Оцінено переваги та недоліки аналогічних рішень, розглянуто приклади інтерфейсів та принципи їх функціонування. Проведений огляд підтвердив актуальність розробки власного адаптивного рішення, орієнтованого як на користувача, так і на адміністратора, з гнучкою логікою формування замовлення, захистом даних та сучасним дизайном.

2 ПРОЄКТУВАННЯ WEB-СЕРВІСУ

2.1 Архітектура інформаційної системи

Розробка ефективного веб-сервісу неможлива без попереднього проєктування його архітектури, яка визначає загальну організацію взаємодії між компонентами системи, їх функціональні ролі та способи обміну даними. Для досягнення високої продуктивності, гнучкості й можливості подальшого масштабування в рамках цього проєкту була обрана клієнт-серверна архітектура з поділом на три основні рівні: презентаційний (інтерфейс користувача), прикладний (серверна логіка) та рівень роботи з базою даних.

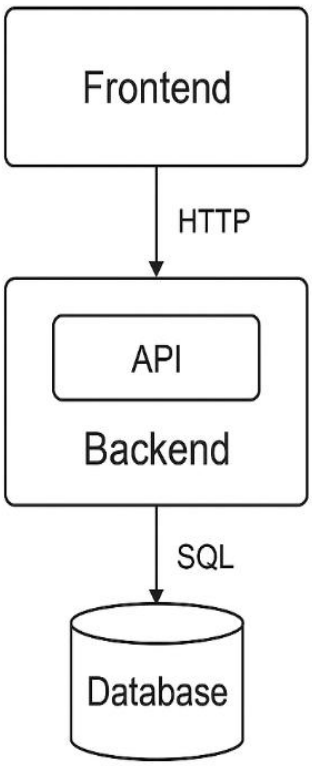

Рисунок 2.1 – Загальна архітектура інформаційної системи web-сервісу

Зображена архітектура забезпечує чітке розмежування відповідальностей між клієнтською, серверною та базою даних. Це дозволяє досягнути високої

масштабованості, забезпечити безпеку та легкість супроводу системи. Кожен модуль виконує ізольовані функції, що сприяє модульному тестуванню.

На рівні презентації передбачено використання сучасних веб-технологій, зокрема інструменту Vite у поєднанні з React. Це рішення забезпечує високу швидкість завантаження сторінок, адаптивний інтерфейс і можливість подальшого розширення функціоналу без кардинальних змін у структурі клієнтського додатка.

Прикладний рівень представлений серверною частиною, реалізованою за допомогою Node.js та фреймворку Express. Цей рівень відповідає за обробку запитів від клієнта, виконання бізнес-логіки та формування відповідей у форматі JSON. Комунікація між клієнтом і сервером реалізована через REST API, що дозволяє спростити інтеграцію з іншими системами та забезпечує незалежність клієнтської частини від серверної.

Рівень роботи з даними базується на реляційній системі керування базами даних PostgreSQL, яка є оптимальним вибором для обробки структурованої інформації про користувачів, замовлення, комплектуючі та інші об'єкти предметної області. Завдяки використанню механізмів індексування, транзакцій і складних SQL-запитів забезпечується висока швидкість доступу до даних і їх цілісність.

Архітектурна модель проєкту дозволяє гнучко змінювати та розширювати окремі компоненти системи без необхідності повної перебудови. Наприклад, у разі зростання кількості користувачів або необхідності додавання нових функцій можливо без проблем масштабувати як серверну частину, так і базу даних. Крім того, така архітектура сприяє підвищенню безпеки, оскільки дозволяє ізолювати компоненти системи, обмежити прямий доступ до бази даних і впроваджувати додаткові засоби захисту, такі як токенна аутентифікація, шифрування даних та контроль доступу на рівні API.

2.2 Функціональні вимоги до системи

При розробці веб-сервісу для організації збирання та продажу комп'ютерів важливо врахувати набір функціональних вимог, які забезпечують реалізацію основних бізнес-процесів, потреб клієнтів та ефективну роботу персоналу.

Вимоги формуються з урахуванням особливостей предметної області та необхідності забезпечення високого рівня зручності, продуктивності та безпеки системи.

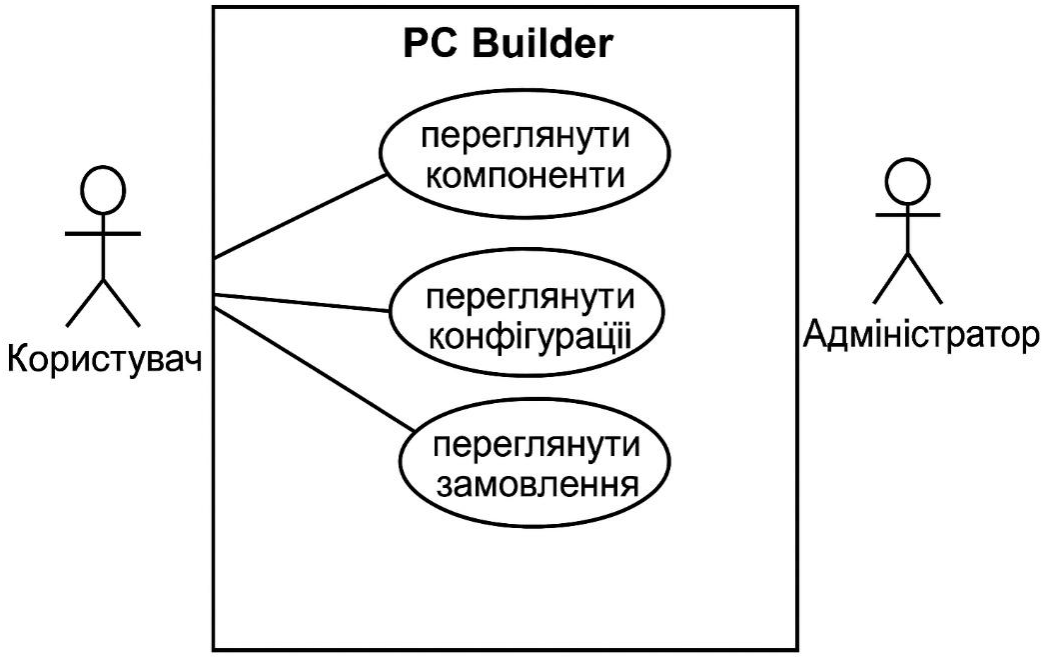

Рисунок 2.2 – Діаграма варіантів використання системи

Дана діаграма чітко окреслює функціональні можливості для кожної категорії користувачів. Розподіл ролей між адміністратором і звичайним користувачем надає системі гнучкості та безпечної взаємодії з бізнес-логікою. Усі дії формують повний життєвий цикл взаємодії з ПК-конфігурацією.

До основних функціональних вимог належать:

- управління каталогом комплектуючих. Система повинна забезпечувати можливість додавання, редагування, видалення та перегляду інформації про комп'ютерні компоненти, включаючи їх технічні характеристики, вартість, кількість на складі та зображення;

- формування готових конфігурацій ПК. Передбачається можливість створення заздалегідь визначених конфігурацій комп'ютерів з можливістю подальшого редагування складу комплектуючих або вибору з каталогу готових рішень;

- реалізація конструктора комп'ютерів. Сервіс повинен містити інструмент для формування індивідуальної конфігурації ПК із перевіркою сумісності обраних компонентів;

- оформлення та обробка замовлень. Користувачі повинні мати змогу оформляти замовлення, переглядати статус їх виконання, а адміністратори — обробляти ці замовлення, змінювати їх статуси та вести облік клієнтів;

- користувацький кабінет. Повинна бути реалізована система особистого кабінету, де користувачі можуть переглядати історію своїх замовлень, змінювати персональні дані та налаштовувати параметри профілю;

- адміністративна панель. Адміністративна частина системи повинна надавати можливість керування каталогами, перегляду аналітичної інформації про продажі, управління користувачами та доступами;

- аутентифікація та авторизація. Необхідно забезпечити надійну систему реєстрації та входу до системи з використанням сучасних методів захисту, таких як хешування паролів та застосування токенів доступу;

- фільтрація та пошук даних. Система має підтримувати гнучкий пошук по каталогу товарів за різними параметрами та фільтрами для швидкого підбору потрібних комплектуючих;

- звіти та аналітика. Повинна бути реалізована можливість формування звітів для аналізу обсягів продажів, популярності окремих конфігурацій та прогнозування попиту.

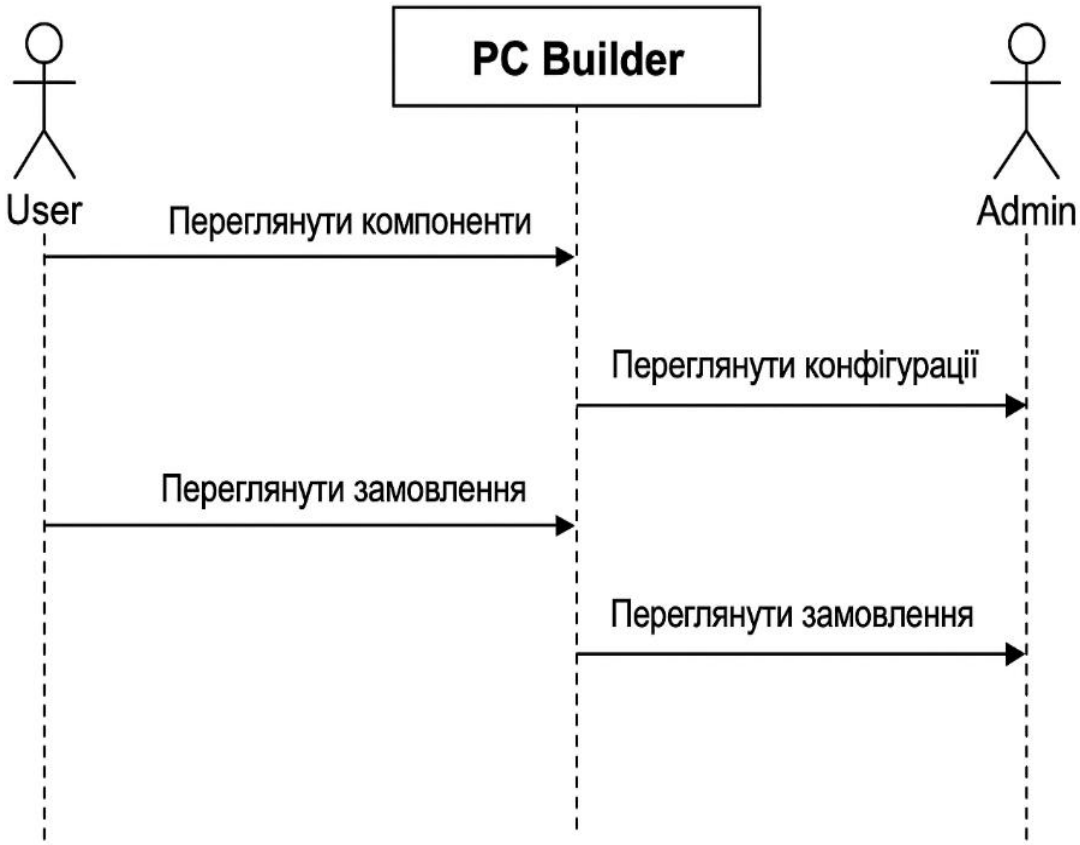

Рисунок 2.3 – Діаграма взаємодії

Макет інтерфейсу створений із урахуванням простоти навігації, інтуїтивності та доступності для широкого кола користувачів. Важливим критерієм було дотримання адаптивності до різних типів пристроїв.

2.3 Інформаційна модель бази даних

Для ефективного зберігання, обробки та отримання інформації в межах web-сервісу спроєктовано реляційну базу даних, що реалізована за допомогою системи керування базами даних PostgreSQL. Структура бази даних була побудована з урахуванням нормалізації даних, що дозволяє уникнути дублювання інформації та забезпечити цілісність даних.

Основними таблицями в базі даних є:

- users — зберігає інформацію про зареєстрованих користувачів. Основні поля: ідентифікатор користувача, ім’я, електронна пошта, хеш пароля, роль користувача (адміністратор або клієнт), дата реєстрації;

- components — містить відомості про доступні комплектуючі: їх назву, тип (процесор, відеокарта, пам’ять тощо), опис, вартість, кількість на складі та посилання на зображення;

Таблиця 2.1 – Опис таблиці Users

Поле	Розмір поля	Тип даних	Значення
Id	4	Int	Первинний ключ
Username	255	Varchar	Імя юзера
Email	255	Varchar	Електронна пошта
Password	255	Varchar	Хеш паролю user
Role	20	Varchar	Роль користувача

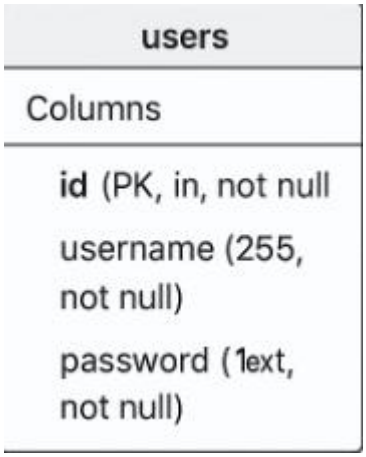

Рисунок 2.3 – Таблиця Users

- computers — таблиця для збереження готових конфігурацій комп’ютерів. Включає назву конфігурації, опис, загальну вартість та дату створення;

- computer_components — проміжна таблиця, що реалізує зв’язок між комп’ютерами та комплектуючими за принципом «багато-до-багатьох». Дозволяє фіксувати, які саме комплектуючі входять до складу конкретної конфігурації ПК;

- orders — містить інформацію про оформлені замовлення. Основні поля: ідентифікатор замовлення, ідентифікатор користувача, дата оформлення, загальна вартість замовлення, статус виконання;

order_items — таблиця для збереження інформації про товари в межах конкретного замовлення, включаючи кількість одиниць кожного товару та посилання на відповідну конфігурацію комп'ютера.

Таблиця Users зберігає дані зареєстрованих користувачів. Вона є основою для реалізації авторизації, розподілу прав доступу, а також для формування історії замовлень конкретного користувача.

Перейдемо до наступної таблиці.

Таблиця 2.2 – Опис таблиці Components

Поле	Розмір поля	Тип даних	Значення
Id	4	Int	Первинний ключ
Name	255	Varchar	Назва комлектуючого
Type	100	Varchar	Тип комплектуючого
Descrisption	-	Text	Опис
Price	8	Float	Ціна
Stock	4	Int	Кількість
Image_url	255	Varchar	Посилання на зображення

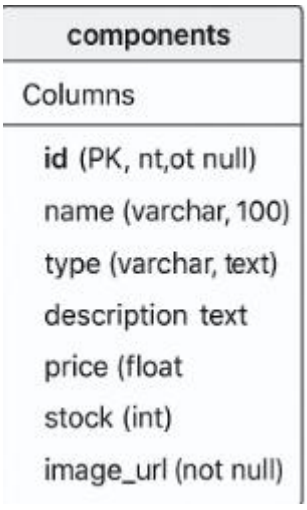

Рисунок 2.4 – Таблиця Components

Ця таблиця призначена для зберігання повної інформації про наявні комплектуючі. Вона дозволяє реалізувати функції фільтрації, перегляду та обліку складу на сайті.

Перейдемо до наступної таблиці.

Таблиця 2.3 – Опис таблиці Computers

Поле	Розмір поля	Тип даних	Значення
Id	4	Int	Первинний ключ
Name	255	Varchar	Назва конфігурації
Description	-	Text	Опис
Price	8	Float	Ціна

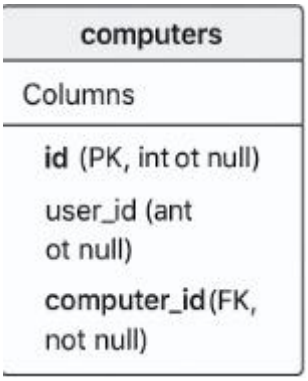

Рисунок 2.5 – Таблиця Computers

На рисунку 2.5 зображено схему таблиці бази даних з назвою computers. Вона містить такі стовпці:

- id — первинний ключ (PK), типу int, не може бути null;
- user_id — типу int, також не може бути null (є помилка у написанні "ant ot null", імовірно мало бути "int not null");
- computer_id — зовнішній ключ (FK), типу int, не може бути null.

Таблиця Computers використовується для збереження даних про зібрані конфігурації ПК. У ній зберігається унікальна назва збірки, опис і загальна вартість, що полегшує створення шаблонів для клієнтів.

Перейдемо до наступної таблиці.

Таблиця 2.4 – Опис таблиці Orders

Поле	Розмір поля	Тип даних	Значення
Id	4	Int	Первинний ключ
User_id	4	Int	Зовнішній ключ з таблицею users
Computer_id	4	Int	Зовнішній ключ з таблицею computers
Total_price	8	Float	Загальна сума
Status	50	Varchar	Статус замовлення
Created_at	-	Timestamp	Дата створення

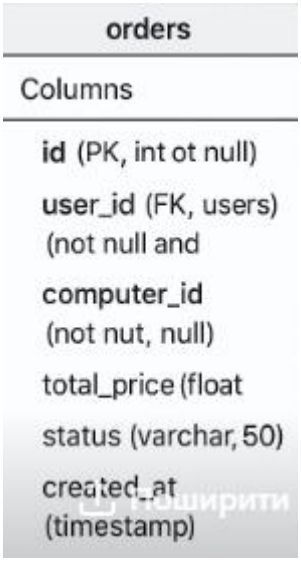

Рисунок 2.6 – Таблиця Orders

Таблиця Orders відображає оформлені користувачами замовлення. Вона є основним джерелом інформації для адміністративної панелі, обліку замовлень і контролю їхнього виконання.

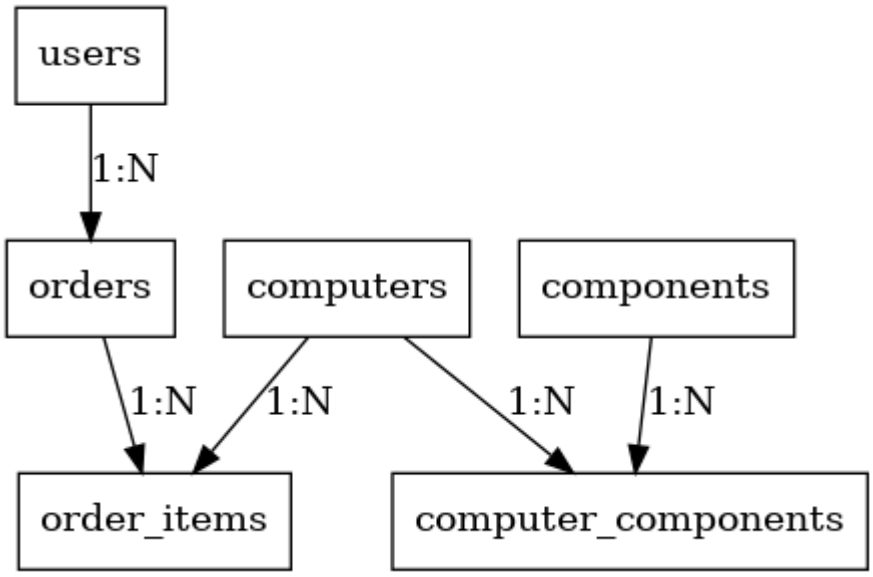

Рисунок 2.7 – ER-діаграма до інформаційної бази даних

ER-діаграма є ключовим етапом у проєктуванні бази даних, оскільки вона дозволяє наочно представити зв'язки між сутностями. Саме завдяки такій

структурі забезпечується цілісність даних, ефективність запитів та можливість гнучкого масштабування системи без порушення логіки даних.

Завдяки такій структурі реалізується ефективне управління інформацією про товари, користувачів і замовлення, що спрощує реалізацію необхідної бізнес-логіки системи.

Додатково в базі даних передбачено створення індексів для прискорення пошуку по найбільш використовуваних полях, таких як ідентифікатори замовлень, адреси електронної пошти користувачів та назви комплектуючих. Це забезпечує високий рівень продуктивності при збільшенні обсягів даних.

Запропонована модель бази даних є гнучкою та може бути розширена у разі потреби додавання нових функціональних можливостей системи.

2.4 Розробка REST API

Для забезпечення ефективної взаємодії між клієнтською частиною та серверною частиною системи було спроєктовано REST API. Архітектурний стиль REST дозволяє стандартизувати комунікацію через протокол HTTP, використовуючи стандартні методи: GET, POST, PUT, DELETE. Усі дані обмінюються у форматі JSON.

Таблиця 2.5 – Таблиця модулів та їх специфікацій

№	Назва модуля	Призначення	Вхідні дані	Вихідні дані
1	Модуль автентифікації	Забезпечення входу та реєстрації користувачів	Email, пароль	Токен доступу, повідомлення про статус
2	Модуль управління ПК	Додавання, редагування, перегляд конфігурацій ПК	Дані ПК, список компонентів	Підтвердження, оновлена конфігурація
3	Модуль управління замовленнями	Створення та перегляд замовлень	ID комп’ютера та користувача	Інформація про замовлення
4	Модуль компонентів	Виведення та адміністрування списку комплектуючих	Тип компоненту, фільтри	Список компонентів
5	Адмінстративна панель	Керування користувачами, ролями статусами замовлень	Дані користувача, ID замовлень	Оновлені дані, статуси

Наведена таблиця демонструє модульність розробленого API, що забезпечує зрозумілу й стандартизовану взаємодію між клієнтською та серверною частинами. Такий підхід полегшує підтримку та розширення системи, а також дозволяє легко інтегрувати нові функції.

Нижче наведено приклади основних ендпоінтів API:

Користувачі (Users):

- POST /api/auth/register — реєстрація нового користувача;
- POST /api/auth/login — вхід у систему, отримання токена доступу;
- GET /api/users — отримання списку всіх користувачів (доступно лише адміністраторам).
- Комплектуючі (Components):
- GET /api/components — отримання списку доступних комплектуючих;
- POST /api/components — додавання нового компоненту (адмін-панель);
- PUT /api/components/:id — редагування інформації про компонент;
- DELETE /api/components/:id — видалення компоненту.
- Комп’ютери (Computers):
- GET /api/computers — отримання списку готових конфігурацій ПК;
- POST /api/computers — додавання нової конфігурації (адміністратор);
- GET /api/computers/:id — отримання інформації про конкретну конфігурацію.

Замовлення (Orders):

- POST /api/orders — створення нового замовлення;
- GET /api/orders/:id — перегляд деталей замовлення;
- GET /api/orders/user/:userId — отримання всіх замовлень конкретного користувача;
- PUT /api/orders/:id — зміна статусу замовлення (наприклад, з «очікується» на «виконано»).

Кожен ендпоінт супроводжується перевіркою прав доступу. Для захищених ресурсів використовується аутентифікація через JWT-токени, які передаються в заголовках запитів. Це забезпечує безпечну роботу з

персональними даними користувачів і конфіденційною інформацією про замовлення.

Дизайн API побудований таким чином, щоб бути максимально розширюваним і підтримувати майбутнє додавання нових функцій без необхідності порушення існуючої логіки взаємодії.

2.5 Проєктування користувацького інтерфейсу

Одним із ключових аспектів при створенні web-сервісу є розробка інтуїтивно зрозумілого та привабливого інтерфейсу для кінцевих користувачів. Якість інтерфейсу безпосередньо впливає на зручність використання системи, швидкість взаємодії з сервісом і загальне враження користувачів від роботи з платформою.

При проєктуванні інтерфейсу було прийнято рішення використовувати бібліотеку React у поєднанні з Vite як ефективний інструмент для збірки й оптимізації фронтенду. Це рішення забезпечує високу швидкодію, адаптивність під різні типи пристроїв та простоту розширення функціональності в майбутньому.

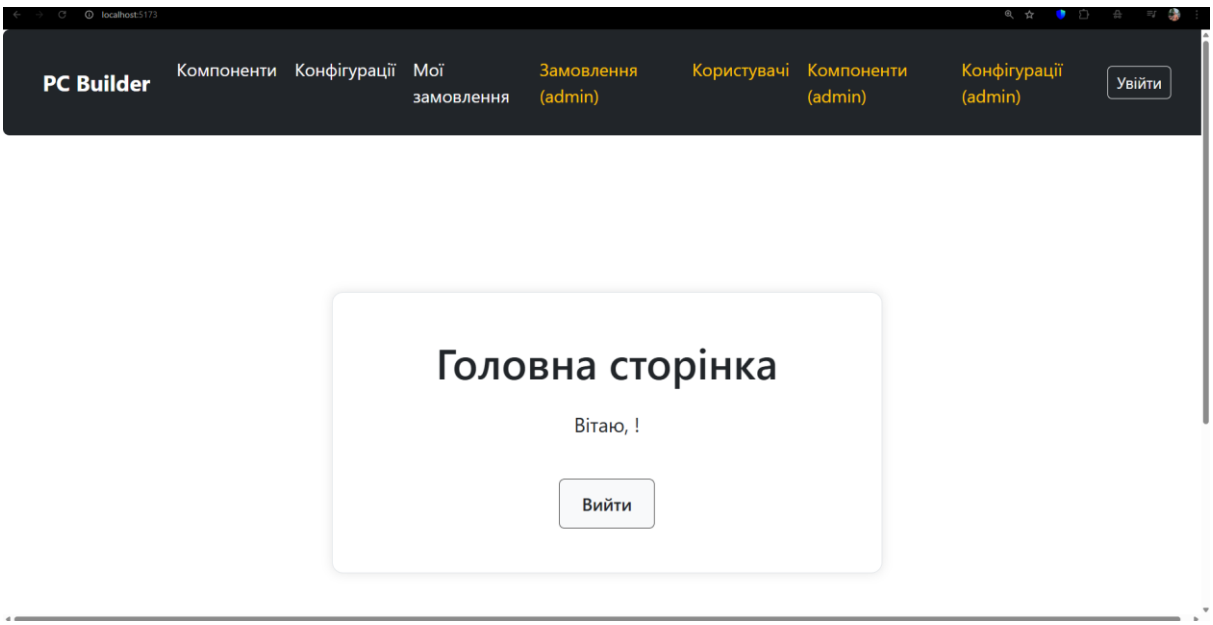

Рисунок 2.8 – Головна сторінка web-сервісу

Головна сторінка веб-сервісу розроблена таким чином, щоб забезпечити перше знайомство користувача з функціоналом сайту. Вона містить елементи навігації, авторизації, а також надає загальну інформацію про сервіс. Простота і зрозумілість інтерфейсу дозволяє новим користувачам інтуїтивно взаємодіяти із застосунком без потреби у додатковій допомозі. Основна мета цієї сторінки — залучити користувача до подальшої взаємодії з іншими модулями системи.

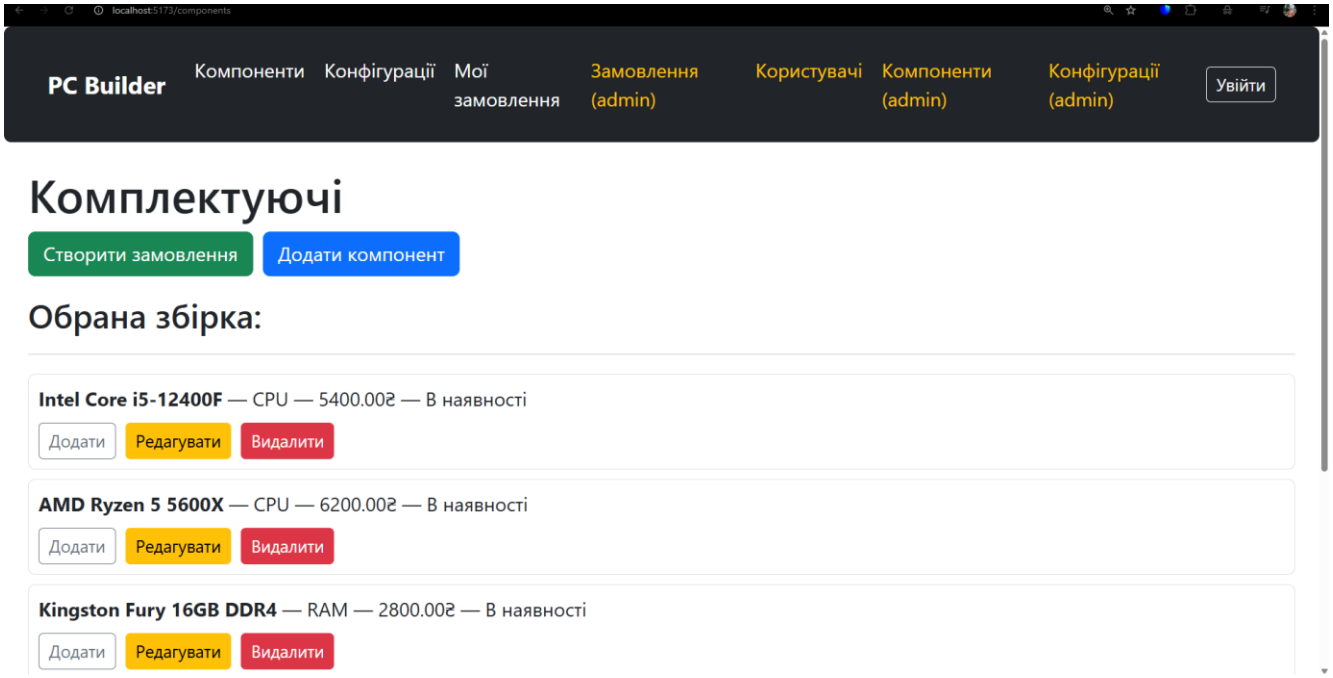

Рисунок 2.9 – Сторінка з каталогом комплектуючих web-сервісу

Каталог комплектуючих відображає усі доступні компоненти, які можна обрати для формування індивідуальної конфігурації комп’ютера. Кожен компонент супроводжується назвою, категорією, вартістю та наявністю на складі. Цей модуль дозволяє як пересічному користувачу, так і адміністратору оперативно переглядати актуальні пропозиції й здійснювати вибір комплектуючих за визначеними параметрами.

Тепер перейдемо до списку конфігурацій.

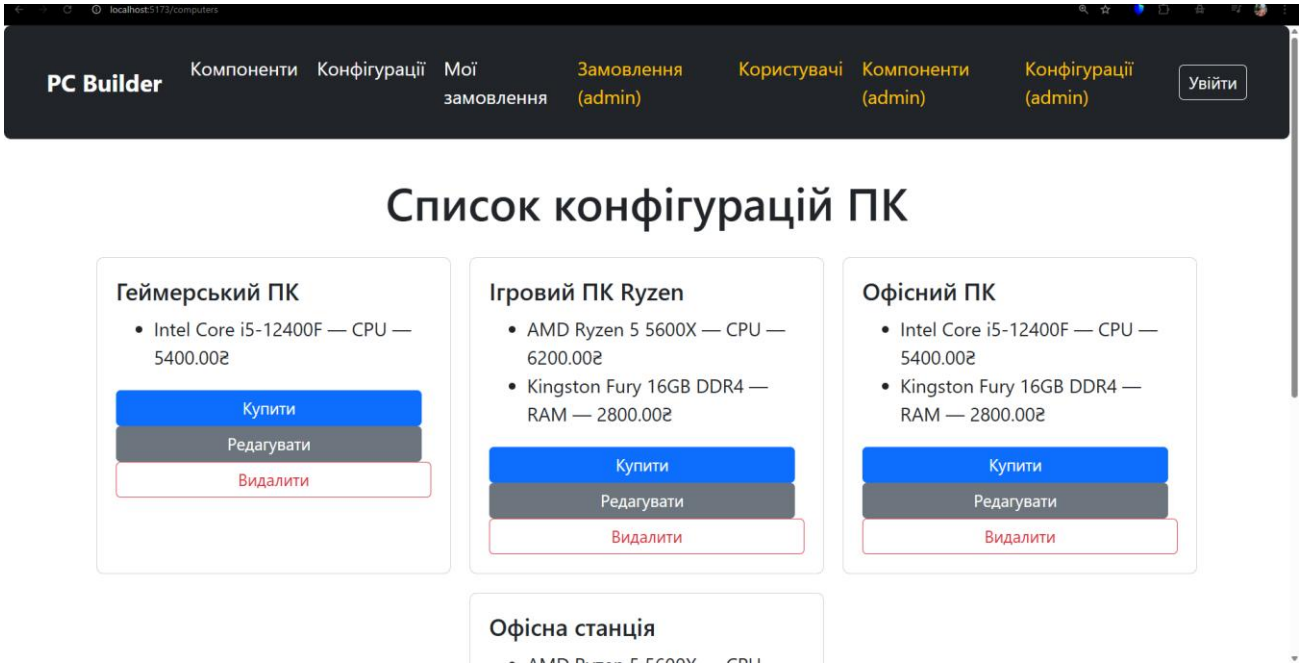

Рисунок 2.10 – Сторінка конструктора ПК web-сервісу

Конструктор ПК є центральною частиною функціональності сервісу. Завдяки інтерактивному підходу, користувач має змогу поступово обирати деталі збірки, зважаючи на сумісність та вартість компонентів. Система автоматично оновлює підсумкову ціну, що дозволяє контролювати бюджет користувача. Реалізація цього модуля значно спрощує процес підбору техніки для користувачів без глибоких технічних знань.

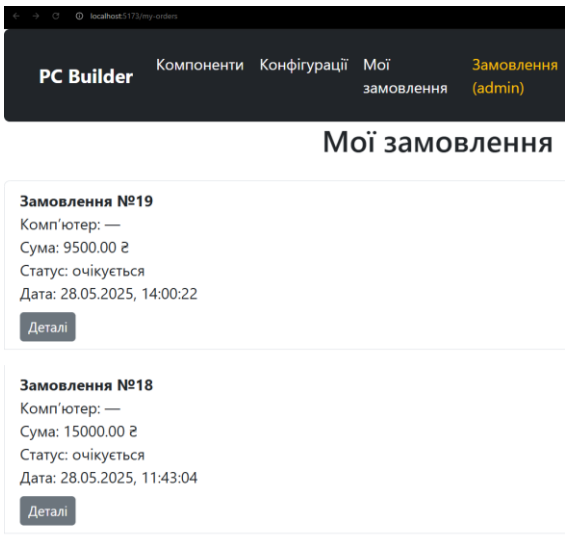

Рисунок 2.11 – Кабінет користувача з інформацією про зроблені замовлення

Особистий кабінет є простором, де користувач може переглядати свої попередні замовлення, відстежувати їх статус, а також управляти персональними даними. Такий функціонал покликаний підвищити зручність використання системи, забезпечити прозорість взаємодії з сервісом і дозволити оперативно орієнтуватися в історії покупок.Тепер перейдемо до адмін панелі:

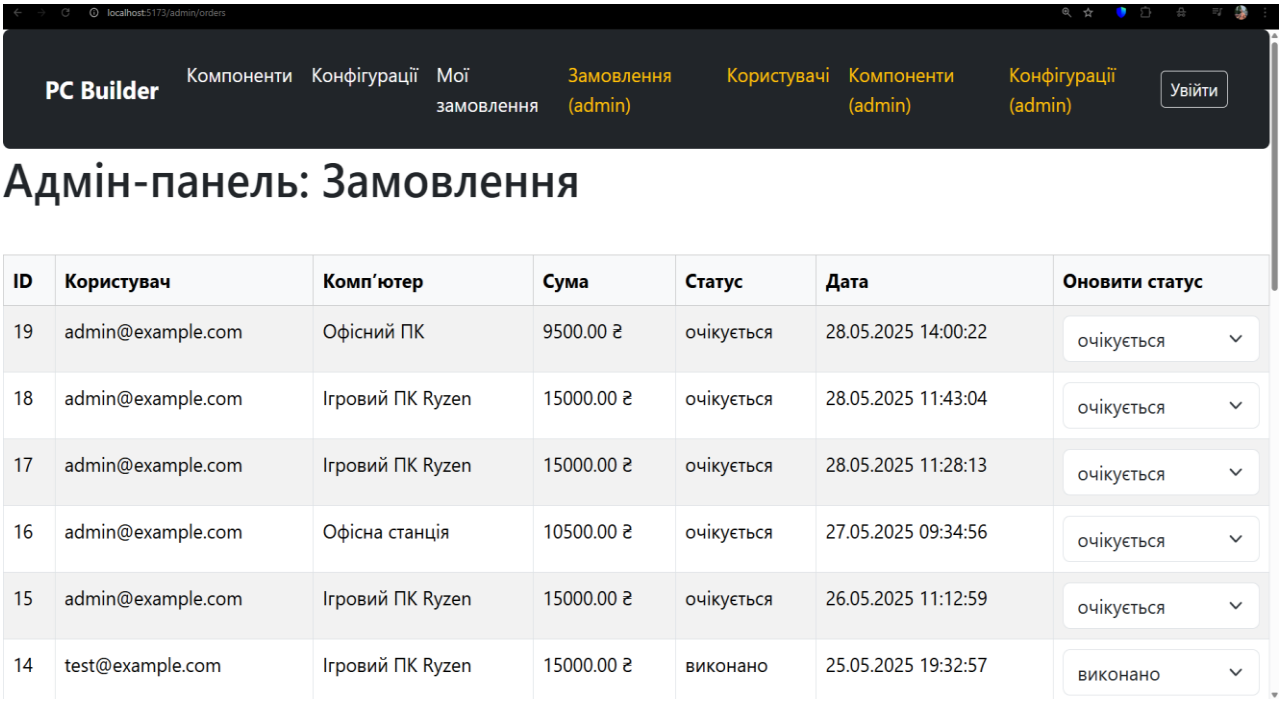

Рисунок 2.12 – Адмін панель для відстежиння замовлень користувачів та змін їхніх статусів

Адміністративна панель надає змогу співробітникам обробляти замовлення, змінювати їх статуси, переглядати деталі замовлень та історію дій користувачів. Завдяки цьому модулю забезпечується централізований контроль над ключовими операціями сервісу, що є важливим для стабільного функціонування системи.

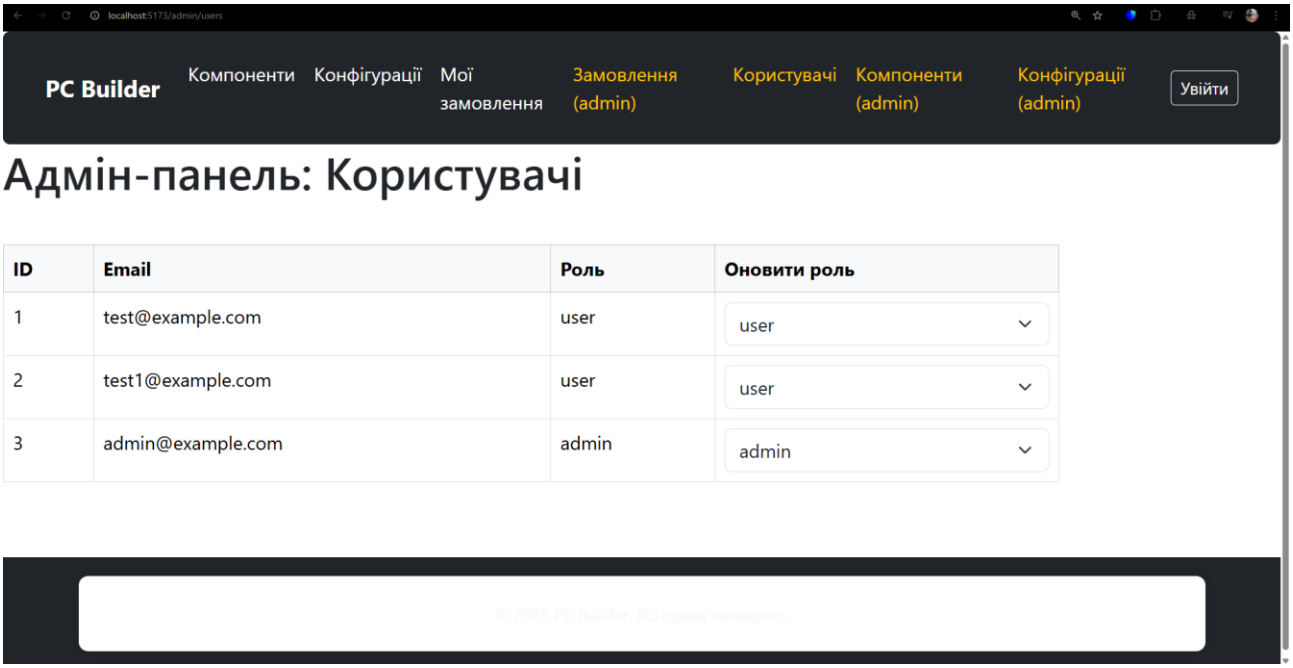

Рисунок 2.13 – Адмін панель з керуванням прав доступа користувачів

Цей інтерфейс дозволяє адміністратору змінювати ролі користувачів — від звичайного користувача до адміністратора. Реалізація механізму керування доступом є важливим аспектом безпеки системи, що унеможливлює несанкціонований доступ до критичних частин сервісу та гарантує відповідальність дій кожного користувача.

Головні сторінки, передбачені в системі:

- головна сторінка ("/") — містить загальний огляд сервісу, акційні пропозиції, популярні комп’ютерні збірки та кнопки швидкого переходу до каталогу товарів;
- каталог комплектуючих ("/components") — надає можливість перегляду списку доступних комплектуючих з детальними фільтрами за категоріями, характеристиками та ціною. Реалізовано сортування за популярністю, новизною та вартістю;
- конструктор ПК ("/builder") — дозволяє користувачам самостійно формувати конфігурацію комп’ютера, обираючи комплектуючі з перевіркою їх сумісності у режимі реального часу. Після завершення підбору система автоматично розраховує підсумкову вартість збірки;

- оформлення замовлення ("/checkout") — реалізує процес оформлення покупки, включаючи введення контактної інформації, вибір способу доставки та оплати;
- кабінет користувача ("/profile") — надає можливість переглядати історію замовлень, редагувати персональні дані та змінювати налаштування профілю;
- адміністративна панель ("/admin") — доступна лише адміністраторам. Містить інтерфейс для управління каталогами комплектуючих, перегляду статистики продажів, обробки замовлень і керування правами доступу користувачів.

Інтерфейс проєктувався з урахуванням принципів UI/UX-дизайну: простота та логічна структура навігації, мінімалістичний дизайн, акцент на ключові дії користувача та адаптивне відображення сторінок на мобільних пристроях.

Використання бібліотеки компонентів, таких як Material UI або shadcn/ui, дозволяє досягти візуальної цілісності всього інтерфейсу та прискорити процес розробки за рахунок готових елементів дизайну.

Результатом є сучасний, привабливий і функціональний інтерфейс, який забезпечує комфортну роботу користувачів із сервісом незалежно від типу пристрою, який вони використовують.

Отже, спроєктована система базується на сучасних принципах побудови інформаційних систем. Її архітектура дозволяє забезпечити масштабованість, підтримку ролей користувачів, ефективну взаємодію через REST API та зрозумілий інтерфейс. Таким чином, підготовлена технічна база є надійною основою для реалізації повноцінного веб-сервісу.

У другому розділі було представлено архітектуру веб-сервісу, детально описано модулі, сценарії взаємодії, структуру бази даних, механізми авторизації та маршрутизації. Особливу увагу приділено UI-макету, який забезпечує інтуїтивну взаємодію користувача з системою. Запропонована модель системи повністю відповідає сучасним стандартам розробки веб-застосунків, що дозволяє ефективно реалізувати поставлені задачі.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка серверної частини

Серверна частина системи виконує роль посередника між клієнтським інтерфейсом і базою даних. Основними її завданнями є обробка запитів користувачів, виконання бізнес-логіки, взаємодія з базою даних PostgreSQL, а також забезпечення безпеки та автентифікації.

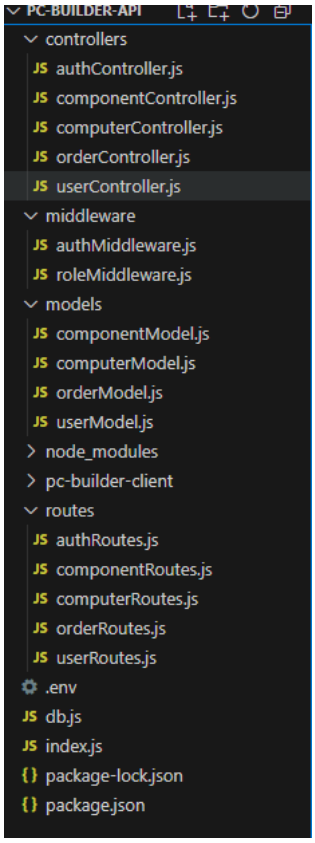

Рисунок 3.1 – Структура проєкту серверної частини

Для реалізації серверної частини було обрано середовище Node.js у поєднанні з фреймворком Express — одним із найпопулярніших засобів для створення REST API. Такий вибір забезпечив високу швидкодію, просту масштабованість і велику кількість готових рішень та пакетів, що пришвидшили процес розробки.

Серверна частина проєкту реалізована з дотриманням принципів MVC (Model-View-Controller), що дозволяє ефективно розділити логіку обробки запитів, бізнес-логіку та взаємодію з БД. Такий підхід полегшує розширення та супровід проєкту.

Базова структура серверного застосунку включає такі ключові елементи:

- маршрутизатори (routes) — містять логіку обробки запитів до різних частин API (користувачі, компоненти, комп'ютери, замовлення);
- контролери (controllers) — реалізують функціональну частину кожного ендпоінта, включаючи запити до бази даних, обробку помилок та формування відповідей;
- моделі (models) — описують структуру даних і зв'язки між сутностями в базі;
- сервіси (services) — містять додаткову логіку, наприклад, обробку токенів, перевірку прав доступу або перевірку сумісності комплектуючих.

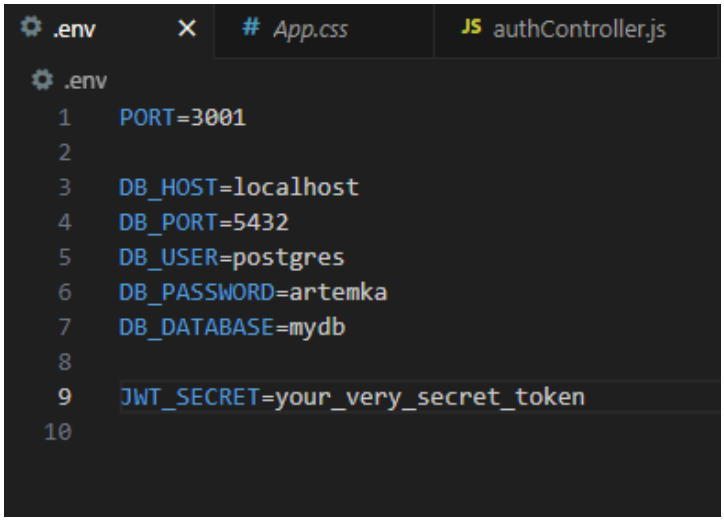

Рисунок 3.2 – Вміст файлу .env

Для взаємодії з PostgreSQL використовується бібліотека pg, яка дозволяє виконувати параметризовані SQL-запити, що знижує ризики SQL-ін'єкцій. Всі конфігураційні параметри (дані підключення до БД, секретні ключі)

зберігаються у файлі .env, що не потрапляє в публічний репозиторій і не розголошує чутливу інформацію.

Важливою частиною серверної логіки є реалізація JWT-аутентифікації. Після успішної авторизації користувачу видається токен доступу, який передається в заголовках запитів. Це дозволяє безпечно перевіряти особу користувача при зверненні до захищених ендпоінтів.

middleware > JS authMiddleware.js > ...

```
const jwt = require("jsonwebtoken");

function verifyToken(req, res, next) {
  const authHeader = req.headers.authorization;
  if (!authHeader || typeof authHeader !== "string") {
    return res.status(403).json({ error: "No token provided or token format is incorrect" });
  }

  const token = authHeader.startsWith("Bearer ") ? authHeader.slice(7) : authHeader;

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    return res.status(401).json({ error: "Invalid token" });
  }
}

module.exports = { verifyToken };
```

Рисунок 3.3 – Фрагмент коду authmiddlware для перевірки автентичності користувача через JWT

JWT токен дозволяє реалізувати безпечну авторизацію в системі, передаючи дані про роль користувача, термін дії сесії та унікальний ідентифікатор. Обробка токена реалізована відповідно до сучасних вимог безпеки.

Для обробки помилок використовується централізований middleware, який дозволяє уніфіковано реагувати на всі типи винятків — як внутрішні помилки сервера, так і некоректні запити з боку клієнта.

У результаті розробки серверної частини отримано стабільний та розширюваний backend-додаток, який забезпечує повноцінну взаємодію між

інтерфейсом і базою даних, гарантуючи коректну обробку бізнес-логіки та високий рівень захищеності даних.

3.2 Розробка клієнтської частини

Клієнтська частина є тією складовою системи, з якою безпосередньо взаємодіє користувач. Саме вона відповідає за відображення даних, отриманих від сервера, формування запитів та надання зрозумілого й приємного інтерфейсу для навігації, вибору товарів і оформлення замовлень.

Для реалізації фронтенду було використано сучасний інструмент Vite у зв'язці з бібліотекою React. Такий підхід дозволив досягти високої швидкості завантаження сторінок, зручного компонування елементів інтерфейсу та ефективної роботи з динамічними даними. React-компоненти структуровано за принципом розділення відповідальностей: кожна сторінка та окремий блок інтерфейсу мають власну логіку та стиль.

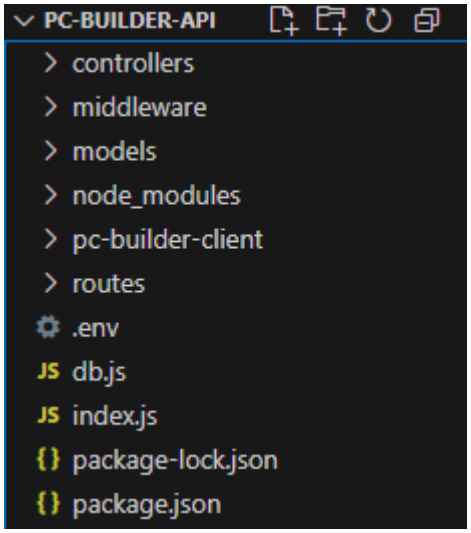

Рисунок 3.4 – Структура проєкту клієнтської частини

Ключові технічні аспекти реалізації:

- реалізація маршрутизації — здійснюється за допомогою бібліотеки react-router-dom. Це дозволяє перемикатись між сторінками без перезавантаження сайту, забезпечуючи плавний перехід і швидкий доступ до контенту;

- взаємодія з API — організована через fetch або бібліотеку axios, залежно від задачі. Для захищених запитів додаються JWT-токени до заголовків. Отримані з сервера дані зберігаються в локальному стані (через useState, useEffect), або у глобальному (через контекст чи Redux, за потреби);

- формування форм і валідація — при створенні форм авторизації, реєстрації чи оформлення замовлення використовується ручна або бібліотечна валідація введених даних. Це дозволяє уникнути помилок при відправленні запитів на сервер;

- адаптивна верстка — інтерфейс побудовано таким чином, щоб він коректно відображався на різних розмірах екранів: від смартфонів до широкоформатних моніторів. Для стилізації використовуються утиліти Boostrap CSS або готові бібліотеки компонентів (наприклад, shadcn/ui);

- обробка помилок — реалізовано обробку помилок запитів: при невдалому зверненні до сервера користувач отримає сповіщення з відповідною інформацією (наприклад, про відсутність з'єднання чи неправильні дані).

Загалом, розроблена клієнтська частина дозволяє швидко взаємодіяти з інформацією, представлена в зручному вигляді, та надає весь набір функцій для повноцінної роботи користувача з сервісом — від перегляду каталогу до оформлення замовлень. Простота у використанні поєднується з технічною гнучкістю та можливістю швидкої модифікації або розширення функціоналу.

3.3 Тестування функціональності

Після реалізації серверної та клієнтської частин системи важливим етапом стала перевірка її працездатності. Метою тестування було виявлення можливих помилок у логіці роботи, перевірка коректності взаємодії між компонентами, а

також оцінка зручності користування веб-сервісом з точки зору кінцевого користувача.

Тестування проводилося на рівнях:

- модульне тестування — перевірялися окремі функції, зокрема ті, що пов’язані з обробкою даних, валідацією форм, перевіркою автентичності користувачів. Усі функції контролерів були протестовані вручну або за допомогою тестових викликів з Postman;

- інтеграційне тестування — здійснювалося перевіркою взаємодії між клієнтською частиною та сервером. У цьому випадку особливу увагу було приділено обміну даними по REST API. Наприклад, запит POST /api/orders повинен успішно створювати нове замовлення, збережене у базі даних, а GET /api/components — повертати актуальний список компонентів;

- тестування інтерфейсу (UI) — перевірялася коректність відображення елементів, робота навігації, адаптивність інтерфейсу. Також проводилась імітація реальних сценаріїв користування: реєстрація, створення замовлення, редагування профілю, пошук по каталогу;

- тестування помилок і нестандартних ситуацій — симулювалися дії, які могли викликати помилки: спроба авторизації з неправильними даними, оформлення замовлення без вибраних компонентів, введення некоректних значень у форму. У таких випадках система мала повертати інформативні повідомлення про помилки.

Результати тестування показали стабільну роботу основного функціоналу. Інтерфейс працює швидко й без збоїв, API коректно обробляє запити та відповіді, а всі критичні помилки обробляються належним чином. Виявлені дрібні недоліки були усунені в процесі оптимізації.

Таким чином, система пройшла первинну перевірку працездатності та готова до подальших етапів розгортання й можливого впровадження.

3.4 Засоби безпеки та захисту даних

Питання безпеки під час розробки web-сервісів має критичне значення, особливо коли система оперує персональними даними користувачів, інформацією про замовлення, а також взаємодіє з базою даних через API. Саме тому при створенні даного сервісу особливу увагу було приділено впровадженню надійних засобів захисту.

Першочергово реалізовано механізм аутентифікації та авторизації користувачів. Для цього застосовується JSON Web Token (JWT) — після успішного входу система генерує унікальний токен, який зберігається на клієнті та передається в заголовках усіх захищених запитів. Таким чином, сервер перевіряє дійсність токена та визначає рівень доступу користувача.

Паролі зберігаються не в чистому вигляді, а у вигляді хешів, що створюються з використанням криптографічного алгоритму bcrypt. Це гарантує, що навіть у разі компрометації бази даних зловмисник не отримає справжні паролі.

Для захисту запитів між клієнтом і сервером використовується HTTPS-протокол, який шифрує весь трафік і унеможливлює перехоплення або зміну даних сторонніми особами. Сервер також налаштований на обробку CORS-запитів лише з дозволених доменів, що зменшує ризик несанкціонованого доступу.

На рівні REST API реалізовано перевірку прав доступу: деякі маршрути доступні лише для адміністраторів (наприклад, додавання товарів, перегляд усіх замовлень), тоді як звичайні користувачі можуть працювати лише в межах своїх прав. Усі спроби доступу до заборонених ресурсів супроводжуються відповіддю про помилку з поясненням.

Крім того, серверна частина має централізований механізм обробки помилок та валідації вхідних даних. Усі введені користувачем значення перевіряються перед збереженням у базу, що дозволяє уникнути ін’єкцій, спроб XSS-атаки або введення некоректної інформації.

У сукупності ці заходи забезпечують базовий, але надійний рівень безпеки, якого достатньо для захисту персональних даних і стабільної роботи web-сервісу у відкритому доступі.

3.5 Приклади роботи системи

Для підтвердження працездатності створеного web-сервісу було здійснено демонстрацію ключових функцій із використанням тестових даних. У ході перевірки система успішно обробляла запити, відображала відповідні дані та виконувала базову бізнес-логіку.

1. Головна сторінка — при завантаженні користувач бачить перелік актуальних конфігурацій комп’ютерів із можливістю перегляду детальної інформації, переходу до конструктора або оформлення замовлення. Також доступні кнопки входу/реєстрації.

2. Каталог компонентів — реалізовано зручну таблицю з фільтрами за типом комплектуючих (процесори, материнські плати тощо). При натисканні на позицію відкривається вікно з детальним описом, технічними характеристиками та кнопкою "Додати до збірки".

3. Конструктор ПК — користувач може обирати компоненти покроково, а система в режимі реального часу перевіряє сумісність між ними. При повному заповненні конфігурації виводиться підсумкова вартість і кнопка для оформлення.

4. Оформлення замовлення — після створення збірки користувач переходить до сторінки введення контактних даних і способу доставки. Після підтвердження замовлення формується запис у базі даних, і користувач бачить відповідне повідомлення.

5. Кабінет користувача — доступний після авторизації. Містить історію замовлень, дані профілю, можливість змінити пароль або оновити контактну інформацію.

6. Адмін-панель — адміністратор має доступ до сторінки управління компонентами, списку замовлень користувачів, статистики продажів. Також реалізована можливість вручну змінювати статуси замовлень.

7. Приклади API-запитів:

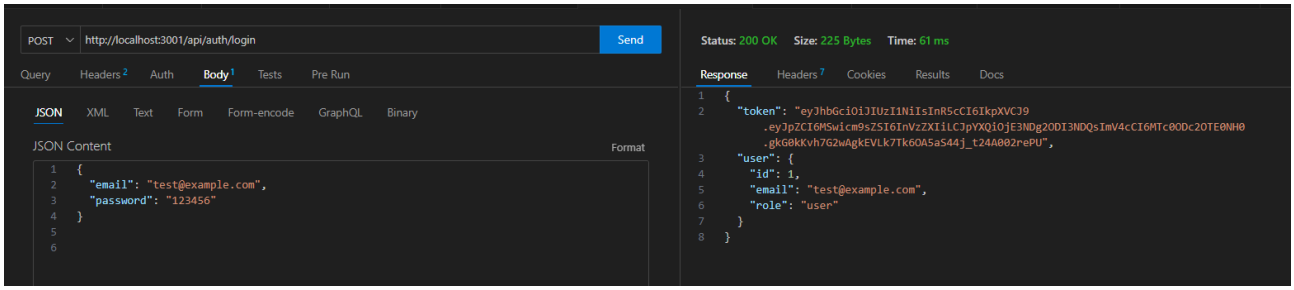

Рисунок 3.5 - POST /api/auth/login повертає токен зареєстрованого користувача

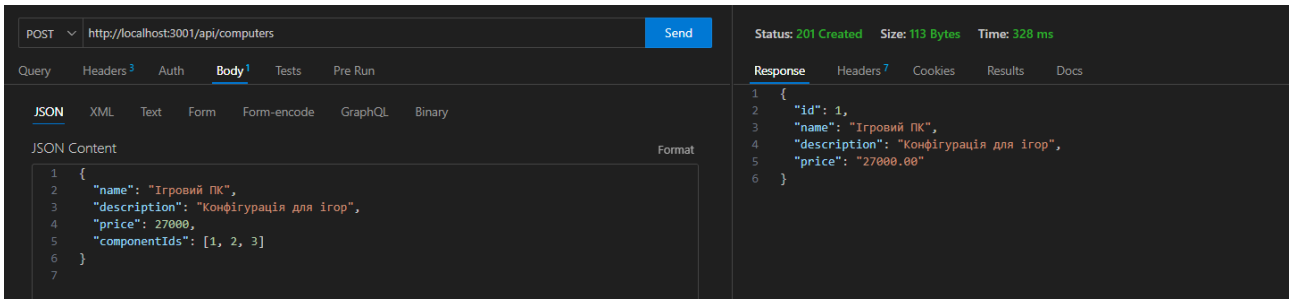

Рисунок 3.6 - POST /api/computers створює нову конфігурацію ПК

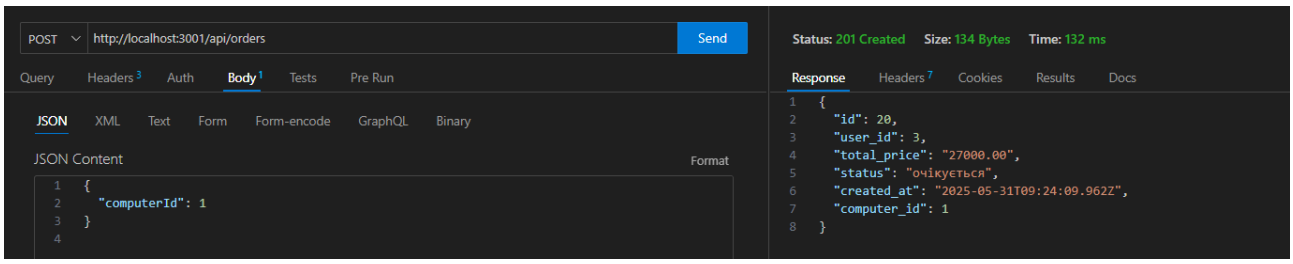

Рисунок 3.7 - POST /api/orders виводить усі замовлення користувача з ID = 1

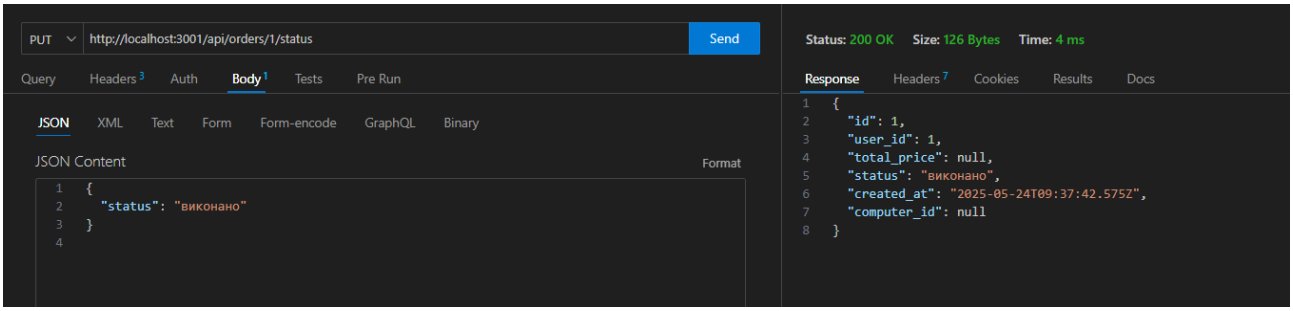

Рисунок 3.8 – PUT /api/orders/status оновлене замовлення зі статусом «виконано»

Демонстрація всіх функціональних частин дозволила впевнено оцінити повну працездатність системи в умовах тестового використання, а також показала потенціал до масштабування в рамках реального бізнес-процесу.

У цьому розділі описано реалізацію основних модулів розробленого веб-сервісу. Наведено фрагменти коду, REST-запити, приклади обробки даних, структуру серверної частини та JWT-авторизацію. Проведено тестування ключових функцій, що підтвердило коректність роботи системи. В результаті створено надійний та функціональний веб-продукт, готовий до реального використання або розгортання в продакшн-середовищі.

3.6 Підготовка програмного продукту до впровадження (продакшену)

Після завершення реалізації функціональної частини веб-сервісу необхідним етапом стала його підготовка до розгортання у робочому середовищі, або так званий «перехід у продакшен». Цей процес передбачає оптимізацію коду, налаштування конфігурацій, забезпечення безпеки даних, тестування продуктивності та готовності системи до реальної експлуатації.

У ході підготовки було виконано низку обов’язкових технічних заходів:

- Налаштування конфігурацій середовища. Було створено файл .env, в якому зберігаються критично важливі змінні середовища, такі як секретні ключі для JWT, порти, URI до бази даних тощо. Це дозволяє легко масштабувати систему, не змінюючи код вручну. Додатково сформовано файл .env.example — шаблон, який полегшує розгортання проєкту на іншому сервері або у середовищі розробника.

- Збірка проєкту. Для мінімізації розміру та оптимізації роботи клієнтської частини виконано фінальну збірку за допомогою інструменту Vite. Команда npm run build дозволила сформувати папку dist, яка містить вже скомпільовані ресурси, готові до розміщення на хостингу або в хмарному сховищі. Усі залежності та маршрути були перевірені на працездатність у зібраному вигляді.

- Очищення коду. Після завершення основних етапів розробки було видалено зайві логування (console.log), тимчасові компоненти, тестові елементи інтерфейсу, а також перевірено відповідність структур API REST-принципам. Це значно покращило читабельність і підтримуваність коду.

- Налаштування безпеки. Серверна частина доповнена middleware-компонентами для контролю заголовків та дозволених доменів, зокрема helmet та cors, які дозволяють захистити застосунок від низки відомих вразливостей (наприклад, XSS або CSRF-атак). Аутентифікація реалізована з використанням JWT, токени зберігаються у локальному сховищі клієнта з обмеженим терміном дії.

- Обробка помилок. На сервері реалізована централізована обробка помилок з поверненням статус-кодів (400, 401, 403, 500 тощо), що дозволяє точно інформувати користувача про результат дій. На клієнті також реалізовано механізми відображення помилок та повторних спроб.

- Перевірка на стійкість до некоректних дій. У рамках тестування проведено перевірки на реакцію системи на відсутність токена, спробу доступу без авторизації, введення некоректних даних при вході, реєстрації та створенні замовлень. Усі механізми відповіли стабільно й відповідно до очікуваної поведінки.

Всі ці заходи дозволили привести web-сервіс до повноцінного готового до запуску стану, що відповідає вимогам розгортання в публічному середовищі (наприклад, на хостингу або VPS). Завдяки грамотному структуруванню коду, налаштуванню конфігурацій та врахуванню питань безпеки, система може бути використана як базовий шаблон для розробки інших подібних комерційних застосунків.

ВИСНОВКИ

У межах виконання бакалаврської роботи було реалізовано повноцінний web-сервіс для організації, що займається збиранням та продажем комп'ютерної техніки. Результатом стала система, яка поєднує в собі зручний інтерфейс користувача, ефективну обробку бізнес-логіки та надійне зберігання даних у реляційній базі.

У теоретичній частині проєкту було проведено аналіз сучасного стану ринку аналогічних програмних рішень, визначено специфіку предметної області, окреслено основні проблеми автоматизації й обґрунтовано вибір технологій. На підставі аналізу було спроєктовано архітектуру системи, визначено функціональні вимоги, розроблено модель бази даних, описано REST API та елементи користувацького інтерфейсу.

Практична частина включала реалізацію серверної логіки на платформі Node.js з використанням Express, побудову фронтенду на основі Vite і React, підключення до бази даних PostgreSQL, налаштування аутентифікації користувачів, захист REST API та тестування роботи всіх ключових модулів. Було здійснено демонстрацію основних сценаріїв взаємодії користувача з системою, що підтвердило її працездатність, зручність та потенціал для реального впровадження.

Створений сервіс відповідає актуальним вимогам до сучасних інформаційних систем малого та середнього бізнесу, є масштабованим і розширюваним. Його функціональність може бути адаптована або розширена у майбутньому, зокрема за рахунок інтеграції платіжних систем, логістичних модулів, систем аналітики тощо.

У підсумку можна стверджувати, що поставлені завдання дипломної роботи повністю виконані, а мета — створення web-сервісу для автоматизації процесів збирання і продажу комп'ютерів — досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 15.04.2025).
2. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 17.04.2025).
3. React Documentation. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 17.04.2025).
4. Vite Official Guide. URL: https://vitejs.dev/guide/ (дата звернення: 18.05.2025).
5. OpenJS Foundation. Node.js Documentation. URL: https://nodejs.org/en/docs/ (дата звернення: 18.04.2025).
6. Mozilla Developer Network. Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 18.05.2025).
7. JWT — JSON Web Tokens Introduction. URL: https://jwt.io/introduction (дата звернення: 20.05.2025).
8. bcrypt.js Documentation. URL: https://github.com/kelektiv/node.bcrypt.js (дата звернення: 22.05.2025).
9. REST API Design Guidelines. URL: https://restfulapi.net/ (дата звернення: 22.05.2025).
10. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 26.05.2025).
11. Shadcn UI Components. URL: https://ui.shadcn.com/ (дата звернення: 26.05.2025).
12. Postman API Platform. URL: https://www.postman.com/ (дата звернення: 28.05.2025).
13. ISO/IEC 27001:2022 — Інформаційна безпека, технічний стандарт. (дата звернення: 28.05.2025).

14. В.Ю. Воронков. Основи проектування інформаційних систем: навч. посіб. — К.: Наукова думка, 2021. (дата звернення: 28.05.2025).

15. Офіційна документація npm. URL: https://docs.npmjs.com/ (дата звернення: 29.05.2025).