

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджментуКафедра ІнформатикиРівень вищої освіти перший (бакалаврський)Спеціальність 122 Комп'ютерні науки
(код і повна назва)Тип програми освітньо-професійнаОсвітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« _____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУздобувачеві Барану Данилу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебсайту для продажу комп'ютерної техніки з використанням інтелектуального помічника по підбору комплектуючих

затверджена наказом університету від 26 травня 2026 року № 435Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 22 червня 2026 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література з питань розроблення вебзастосунків, рекомендаційних систем і перевірки сумісності комп'ютерних комплектуючих; матеріали конференцій; дані інтернет-мережі; документація React, Node.js, Express, Prisma ORM та PostgreSQL.

4. Перелік питань, що потрібно опрацювати в роботі _____

1. Аналіз предметної області та постановка задачі.

2. Проєктування вебсайту та інтелектуального помічника.

3. Програмна реалізація та тестування вебсайту для продажу комп'ютерної техніки з інтелектуальним помічником-конфігуратором

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність розроблення вебсайту для продажу комп'ютерної техніки; постановка задачі; схема архітектури клієнт-серверного вебзастосування; ER-діаграма бази даних; алгоритм роботи інтелектуального помічника-конфігуратора; приклади інтерфейсу каталогу, конфігуратора, кошика, особистого кабінету та адміністративної панелі; результати тестування вебзастосування.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих рішень для підбору комп'ютерних комплектуючих	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-08.05.26	
6	Програмна реалізація	26.04.26-23.05.26	
7	Аналіз отриманих результатів	24.05.26-08.06.26	
8	Оформлення пояснювальної записки	08.06.26-13.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тітова О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 82 с., 2 табл., 22 рис., 30 джерел.

ІНТЕЛЕКТУАЛЬНИЙ ПОМІЧНИК, КОНФІГУРАТОР ПК, ПЕРЕВІРКА СУМІСНОСТІ, POSTGRESQL, REACT, NODE.JS.

Об'єктом роботи є процес розроблення вебсайту для продажу комп'ютерної техніки з інтегрованим інтелектуальним помічником для підбору комплектуючих.

Метою роботи є створення вебзастосунку для продажу комп'ютерної техніки, який дозволяє користувачам підбирати комплектуючі з урахуванням їх сумісності, продуктивності та бюджету.

Під час роботи використано: методи аналізу предметної області, проєктування клієнт-серверних вебзастосунків, моделювання бази даних, алгоритми перевірки сумісності комплектуючих.

Практична цінність вебзастосунку полягає у спрощенні процесу підбору комп'ютерних комплектуючих та підвищенні зручності користування.

Розроблено вебзастосунок, що забезпечує перегляд каталогу комп'ютерних комплектуючих, покроковий підбір компонентів, перевірку їх сумісності та формування збалансованої конфігурації ПК.

Область застосування: інтернет-магазини комп'ютерної техніки, онлайн-конфігуратори ПК, сервіси підбору комплектуючих.

POSTGRESQL, REACT, NODE.JS, INTELLIGENT ASSISTANT, PC CONFIGURATOR, COMPATIBILITY CHECKING.

The object of the work is the process of developing a website for selling computer equipment with an intelligent assistant for selecting components.

The aim of the work is to create a web application for selling computer equipment that allows users to select components with regard to compatibility, performance, and budget.

During the work, methods of subject area analysis, client-server web application design, database modeling, and component compatibility checking algorithms were used.

The practical value of the web application lies in simplifying the selection of computer components and improving the usability of an online store.

A web application has been developed that provides catalog browsing, step-by-step component selection, compatibility checking, and formation of a balanced PC configuration.

Application area: online computer equipment stores, PC configurators, component selection services.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	6
Вступ.....	8
1 Аналіз предметної області та постановка задачі	10
1.1 Огляд ринку електронної комерції у сфері продажу комп'ютерної техніки	10
1.2 Аналіз існуючих рішень та платформ з продажу комп'ютерних комплектуючих.....	12
1.3 Роль інтелектуальних систем рекомендацій в електронній комерції	15
1.4 Особливості перевірки сумісності комп'ютерних компонентів....	19
1.5 Постановка задачі.....	21
2 Проєктування вебсайту та інтелектуального помічника	23
2.1 Визначення функціональних вимог до системи.....	23
2.2 Проєктування архітектури клієнт-серверного вебзастосунку	26
2.3 Розроблення структури бази даних для зберігання інформації про товари та користувачів.....	30
2.4 Моделювання логіки роботи інтелектуального помічника.....	36
2.5 Розроблення алгоритмів перевірки сумісності комплектуючих ...	41
3 Програмна реалізація вебсайту для продажу комп'ютерної техніки	45
3.1 Обґрунтування вибору середовища та технологій програмної реалізації.....	45
3.2 Структура програмного проєкту.....	47
3.3 Реалізація клієнтської частини вебзастосунку	51
3.4 Реалізація серверної частини та бази даних	65
3.5 Реалізація інтелектуального помічника-конфігуратора	69
3.6 Тестування роботи вебзастосунку	74
Висновки.....	77
Перелік джерел посилання	79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

БЖ – блок живлення

Інтелектуальний помічник – програмний модуль, який аналізує вибір користувача, перевіряє сумісність комплектуючих та пропонує відповідні варіанти компонентів

Конфігуратор ПК – програмний інструмент, який дозволяє користувачу послідовно обирати комплектуючі для персонального комп'ютера з урахуванням їх сумісності

ПК – персональний комп'ютер

СУБД – система управління базами даних

ATX, Micro-ATX, Mini-ITX – формфактори материнських плат, які визначають їх розміри та сумісність із корпусом

ATX, SFX, TFX – формфактори блоків живлення, які визначають їх фізичні розміри та сумісність із корпусом

API – Application Programming Interface (інтерфейс програмного застосунку, який використовується для обміну даними між клієнтською та серверною частинами системи)

Backend – серверна частина вебзастосунку, яка відповідає за оброблення запитів, роботу з базою даних, користувачами, товарами, кошиком і замовленнями

CORS – Cross-Origin Resource Sharing (механізм, який дозволяє клієнтській частині вебзастосунку надсилати запити до серверної частини з іншого домену або порту)

DDR4, DDR5 – покоління оперативної пам'яті, які відрізняються технічними характеристиками та вимогами до сумісності з материнською платою

Express – фреймворк для Node.js, який використовується для створення REST API

Frontend – клієнтська частина вебзастосунку, яка відповідає за відображення інтерфейсу користувача та взаємодію з ним

HTTP – HyperText Transfer Protocol (протокол передавання даних між клієнтською та серверною частинами вебзастосунку)

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

JSONB – формат зберігання JSON-даних у PostgreSQL у бінарному вигляді, який дозволяє ефективно зберігати та обробляти структуровані характеристики товарів

NVMe – Non-Volatile Memory Express (протокол доступу до швидкісних SSD-накопичувачів)

ORM – Object-Relational Mapping (підхід до роботи з базою даних через об'єктні моделі замість написання SQL-запитів вручну)

PCIe – Peripheral Component Interconnect Express, (інтерфейс підключення відеокарт, накопичувачів та інших пристроїв до материнської плати)

REST API – архітектурний підхід до організації взаємодії між клієнтською та серверною частинами вебзастосунку через HTTP-запити

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

SQL – Structured Query Language (мова структурованих запитів для роботи з реляційними базами даних)

SSD – Solid-State Drive (твердотільний накопичувач)

TDP – Thermal Design Power (показник теплової потужності компонента, який використовується для приблизної оцінки енергоспоживання)

ВСТУП

Продаж комп'ютерної техніки через інтернет-магазини має свої особливості, оскільки користувач обирає не просто окремий товар, а часто цілий набір компонентів для майбутнього персонального комп'ютера. Для звичайного каталогу достатньо показати назву, ціну, опис і характеристики товару, але у випадку з комп'ютерними комплектуючими цього не завжди вистачає. Покупцю потрібно розуміти, чи зможуть вибрані компоненти працювати разом і чи буде така збірка відповідати його потребам.

Під час формування конфігурації ПК важливо враховувати залежності між основними компонентами. Процесор повинен підходити до материнської плати за сокетом, оперативна пам'ять має відповідати підтримуваному типу, відеокарта та система охолодження повинні поміщатися в корпус, а блок живлення має забезпечувати достатню потужність із запасом. Якщо хоча б один із цих параметрів буде проігноровано, користувач може отримати конфігурацію, яка не працюватиме коректно або буде незручною для подальшого використання.

У багатьох інтернет-магазинах комп'ютерної техніки користувачу доступні каталог, пошук і фільтри за окремими характеристиками. Такий набір інструментів допомагає знайти потрібний товар, але не завжди пояснює, як цей товар буде працювати разом з іншими компонентами. Досвідчений покупець може самостійно перевірити сумісність, а для новачка цей процес часто перетворюється на пошук інформації на форумах, у технічних оглядах або через консультації з іншими людьми.

Навіть наявність конфігуратора ПК не завжди гарантує правильний результат. Частина таких сервісів перевіряє тільки найпростіші параметри, наприклад сокет процесора або тип оперативної пам'яті. Водночас під час реального складання також важливі габарити корпусу, довжина відеокарти, висота кулера, формфактор блока живлення та запас потужності. Через це користувач може зібрати конфігурацію, яка на перший погляд виглядає

правильною, але має обмеження або незручності під час практичного використання.

Для спрощення такого вибору доцільно поєднати інтернет-магазин із помічником-конфігуратором. Його задача полягає не лише в тому, щоб показати користувачу список доступних комплектуючих, а й у тому, щоб врахувати вже зроблений вибір. Після додавання одного компонента система повинна перевіряти, які наступні товари можуть бути використані разом із ним, попереджати про несумісність і допомагати поступово сформувати повну збірку ПК.

Актуальність роботи пов'язана з потребою у вебсайті, який не обмежується звичайним каталогом комп'ютерної техніки, а підтримує користувача під час підбору комплектуючих. Розроблення такого рішення дозволяє автоматизувати частину перевірок, зменшити ризик неправильного вибору та зробити процес формування конфігурації ПК більш зрозумілим для користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд ринку електронної комерції у сфері продажу комп'ютерної техніки

Сьогодні купівля товарів через інтернет-магазини стала звичною для багатьох користувачів. Покупець може переглянути асортимент, порівняти ціни, ознайомитися з характеристиками та оформити замовлення без відвідування фізичного магазину. Ринок електронної комерції продовжує розвиватися як у світі, так і в Україні, що підтверджується сучасними аналітичними матеріалами щодо розвитку e-commerce [1–3]. Водночас продаж комп'ютерної техніки має свою специфіку, оскільки такі товари швидко оновлюються, мають багато технічних параметрів і потребують уважного підходу під час вибору.

Особливість вибору комп'ютерних компонентів полягає в тому, що кожна деталь майбутнього ПК пов'язана з іншими елементами збірки. На відміну від багатьох інших товарів, процесор, материнську плату, оперативну пам'ять або відеокарту не можна обирати повністю окремо. Під час підбору потрібно враховувати сокет, підтримуваний тип пам'яті, інтерфейси підключення, габарити компонентів, потужність блока живлення та інші характеристики. Через це звичайна покупка може перетворитися на складний процес, у якому користувачу доводиться перевіряти не один параметр, а сумісність усієї майбутньої конфігурації. Саме тому для цієї категорії товарів важливо мати не лише каталог, а й допоміжний механізм, який підказує, які компоненти можуть працювати разом.

Користувачів інтернет-магазинів комп'ютерної техніки можна умовно поділити на дві групи. До першої належать досвідчені покупці, які добре розуміють характеристики комплектуючих і можуть самостійно оцінити їх сумісність. Для них важливі детальні фільтри, порівняння товарів, повні технічні характеристики та можливість швидко знайти потрібну модель. До

другої групи належать користувачі, які хочуть оновити комп'ютер або зібрати новий ПК, але не мають достатніх знань для самостійного підбору всіх компонентів. Саме для них процес вибору найчастіше є складним і потребує додаткових підказок.

Більшість торговельних платформ надає користувачу каталог товарів, пошук і базові фільтри. Такий підхід зручний для перегляду асортименту, але не завжди допомагає правильно сформувати конфігурацію ПК. Наприклад, користувач може знайти потрібний процесор або відеокарту, але не одразу зрозуміти, чи підходять вони до інших вибраних компонентів. Навіть якщо на сайті є конфігуратор, його можливості можуть обмежуватися простими перевітками та не враховувати всі деталі майбутньої збірки.

Додаткову складність створює постійне оновлення асортименту комп'ютерної техніки. На ринку регулярно з'являються нові моделі процесорів, відеокарт, материнських плат та інших комплектуючих, а ціни змінюються залежно від попиту, постачання та загальної ситуації на ринку [2, 4]. Через це користувачу складно орієнтуватися лише за каталогом. Потрібно не тільки знайти товар за ціною або брендом, а й зрозуміти, чи буде він доречним у конкретній збірці.

У розробленому вебсайті ця проблема вирішується за допомогою покрокового конфігуратора ПК. Такий інструмент враховує вже вибрані комплектуючі, бюджет і призначення майбутнього комп'ютера. Після кожного етапу система може відображати ті варіанти, які підходять до поточної конфігурації, або попереджати користувача про можливу несумісність. Це робить процес підбору більш послідовним і зменшує ймовірність помилок під час формування збірки.

Отже, для сфери продажу комп'ютерної техніки недостатньо мати лише каталог товарів і стандартні фільтри. Користувачу потрібен інструмент, який допомагає не просто переглядати комплектуючі, а поступово формувати сумісну конфігурацію ПК відповідно до бюджету та власних потреб. Саме тому в роботі розглядається створення вебсайту з інтелектуальним

помічником, який поєднує функції інтернет-магазину та конфігуратора комплектуючих.

1.2 Аналіз існуючих рішень та платформ з продажу комп'ютерних комплектуючих

Ринок комп'ютерної техніки сьогодні представлений різними типами онлайн-платформ. До них належать великі маркетплейси, спеціалізовані інтернет-магазини, сервіси з продажу готових ПК та окремі конфігуратори комплектуючих. Кожен із цих варіантів може бути зручним для певного сценарію покупки, однак не всі вони однаково добре підходять для користувача, який хоче самостійно сформувати конфігурацію персонального комп'ютера.

Основна складність такого вибору полягає в необхідності перевіряти сумісність комплектуючих між собою. Більшість платформ показує характеристики товарів, надає пошук і фільтри, але користувач часто самостійно вирішує, чи можуть вибрані компоненти працювати разом. Для конфігурованих продуктів цього недостатньо, оскільки покупець формує не окрему покупку, а набір елементів, які мають відповідати певним умовам сумісності [5].

Деякі спеціалізовані магазини мають конфігуратори ПК, за допомогою яких користувач може поступово додавати компоненти до збірки. Проте такі інструменти часто обмежуються базовими перевітками. Наприклад, система може підібрати материнські плати за сокетом процесора або приховати оперативну пам'ять неподходящого типу. Це корисно, але під час реального складання також потрібно враховувати довжину відеокарти, висоту системи охолодження, формфактор корпусу, запас потужності блока живлення та загальну збалансованість конфігурації. Якщо ці параметри не перевіряються,

користувач може отримати збірку, яка формально виглядає правильною, але має практичні обмеження.

Ще одна проблема пов'язана з оцінкою збалансованості майбутнього ПК. Користувач може вибрати дорогий процесор і слабку відеокарту або підібрати блок живлення без достатнього запасу потужності. У результаті частина ресурсів системи буде використовуватися неефективно або можуть виникнути проблеми зі стабільністю роботи. Тому під час підбору комплектуючих важливо не тільки перевіряти окремі параметри, а й розглядати конфігурацію як цілісну систему. Питання моделювання процесу підбору комплектуючих і його автоматизації розглядається в роботах, пов'язаних із конфігураторами комп'ютерних систем [6].

Окремим варіантом є продаж уже зібраних ПК. Такий підхід зручний для користувачів без технічного досвіду, оскільки їм не потрібно самостійно перевіряти сумісність компонентів. Водночас готові ПК часто мають обмежений вибір конфігурацій, вищу вартість і меншу гнучкість під конкретні потреби користувача. Через це доцільним є проміжний підхід, коли користувач сам обирає комплектуючі, але система допомагає уникати несумісних або невдалих поєднань.

Важливу роль у таких системах відіграє інтерфейс. Якщо конфігуратор має складну структуру, перевантажені сторінки або змушує користувача постійно переходити між різними частинами сайту, процес складання стає незручним. Для комп'ютерних комплектуючих це особливо важливо, оскільки сторінки товарів містять багато технічної інформації. Тому більш зручним є покроковий підхід, за якого користувач бачить поточний етап вибору, доступні варіанти та пояснення можливих проблем сумісності.

Під час розроблення таких систем потрібно враховувати не лише перелік функцій, а й якість взаємодії користувача з вебінтерфейсом. Саме через інтерфейс користувач працює з каталогом, конфігуратором, кошиком і результатами підбору. Логічна структура сторінок, послідовне подання

інформації та зрозуміла навігація впливають на зручність роботи із системою [7].

З технічної точки зору, багато платформ не підтримують повну перевірку конфігурації в реальному часі. Пошук і фільтри допомагають звузити перелік товарів, але не завжди враховують уже вибрані компоненти, бюджет або призначення майбутнього ПК. Через це користувач може бачити товари, які не підходять до поточної збірки або роблять її незбалансованою. Водночас існуючі онлайн-конфігуратори ПК показують, що автоматизований підбір комплектуючих є корисним напрямом розвитку таких систем [6, 8, 9].

У таблиці 1.1 наведено порівняльний аналіз функціональних можливостей типових платформ та очікуваних характеристик розроблюваної системи.

Таблиця 1.1 – Порівняння існуючих рішень та вимог до інтелектуального помічника

Критерій	Глобальні маркетплейси	Спеціалізовані магазини з конфігураторами	Готові ПК	Розроблений вебсайт з помічником
1	2	3	4	5
Перевірка базової сумісності	Частково	Так	Так	Так
Перевірка габаритів компонентів	Ні	Частково	Так	Так
Розрахунок потужності блоку живлення	Ні	Частково	Так	Так, із запасом потужності

Продовження таблиці 1.1

1	2	3	4	5
Урахування бюджету користувача	Частково	Частково	Частково	Так
Покрокове формування конфігурації	Ні	Так	Ні	Так
Збереження готової конфігурації	Ні	Частково	Ні	Так
Адаптивний UI / SPA	Частково	Ні	Ні	Так

Порівняння існуючих рішень показує, що більшість платформ добре підходить для перегляду асортименту, але не завжди достатньо допомагає під час формування повної конфігурації ПК. Основними обмеженнями залишаються неповна перевірка сумісності, слабе врахування бюджету, відсутність пояснень щодо несумісних компонентів і обмежені можливості збереження готової збірки. Це підтверджує доцільність розроблення вебсайту, у якому каталог товарів поєднується з інтелектуальним помічником для покрокового підбору комплектуючих.

1.3 Роль інтелектуальних систем рекомендацій в електронній комерції

У звичайному інтернет-магазині користувач часто самостійно переглядає велику кількість товарів, порівнює характеристики та поступово звужує вибір. Для комп'ютерних комплектуючих такий процес є складнішим, оскільки товари потрібно оцінювати не лише окремо, а й у зв'язку з іншими

компонентами майбутньої збірки. Через це в електронній комерції важливе місце займають рекомендаційні системи, які допомагають користувачу швидше знайти товари, що відповідають його запиту, потребам або вже зробленому вибору.

У сфері продажу комп'ютерної техніки рекомендації не можуть обмежуватися лише популярністю товару або його ціною. Користувачу потрібно отримати не просто список процесорів, відеокарт чи материнських плат, а такі варіанти, які можна використати у конкретній конфігурації. Тому інтелектуальний помічник у такому вебсайті повинен враховувати характеристики товарів, бюджет, призначення ПК і сумісність між компонентами.

Для рекомендацій товарів можуть використовуватися різні підходи: підбір за характеристиками товару, аналіз попередньої поведінки користувачів, використання знань про предметну область або перевірка певних обмежень. У випадку з комп'ютерними комплектуючими найбільш важливим є підхід, за якого система не просто пропонує схожий або популярний товар, а перевіряє, чи може він працювати разом з іншими компонентами збірки [5, 10].

Звичайна рекомендація за популярністю для такої предметної області може бути недостатньою. Наприклад, процесор може мати хороші характеристики, але не підходити до вибраної материнської плати за сокетом. Аналогічно оперативна пам'ять DDR4 не може бути коректним варіантом для плати, яка підтримує лише DDR5. Тому під час підбору комплектуючих рекомендаційна логіка має поєднувати вибір товарів із перевіркою технічних залежностей між ними.

У зв'язку з цим для конфігураторів комп'ютерної техніки доцільно використовувати знання-орієнтовані системи (Knowledge-Based Recommender Systems) та системи на основі обмежень (Constraint-Based Systems). Такі системи працюють не лише з поведінкою користувача, а й із формалізованими правилами предметної області. У випадку підбору комплектуючих такими

правилами є відповідність сокетів, типів пам'яті, формфакторів, габаритів корпусу та вимог до живлення [10].

Система на основі обмежень перевіряє кожен вибір користувача й відкидає ті варіанти, які порушують встановлені правила. Це дозволяє зменшити кількість помилок і зробити процес підбору зрозумілішим навіть для недосвідченого користувача. Подібний підхід також відповідає ідеї конфігурованих продуктів, де кінцевий результат формується з окремих елементів, але кожен елемент має відповідати певним умовам сумісності [5, 10].

Крім сумісності, система має враховувати й потреби конкретного користувача. Користувачі можуть мати різні цілі: ігри, навчання, робота з графікою, програмування або офісні задачі. Тому інтелектуальний помічник повинен не лише перевіряти сумісність компонентів, а й враховувати бюджет і призначення майбутнього комп'ютера. Такий підхід дозволяє адаптувати процес підбору до конкретних потреб користувача та зробити роботу з конфігуратором більш практичною.

У розробленому вебсайті рекомендаційний підхід реалізується через покроковий конфігуратор ПК: користувач обирає призначення майбутнього комп'ютера, задає бюджет і поступово додає компоненти до збірки. Після кожного вибору система аналізує поточну конфігурацію, порівнює характеристики комплектуючих і відображає доступні варіанти з урахуванням сумісності, наявності товару, бюджету та загальної збалансованості системи. Такий підхід можна розглядати як поєднання правил і аналізу даних для підтримки прийняття рішень у змінних умовах [11].

Окрім базової перевірки сумісності, важливою функцією є автоматичний розрахунок потужності блоку живлення. Система підсумовує енергоспоживання всіх вибраних компонентів і пропонує відповідні варіанти блоків живлення з необхідним запасом потужності. Це прибирає потребу користувача самотійно виконувати технічні розрахунки та робить фінальний етап підбору простим і безпомилковим.

Для зручності користувача важливо забезпечити простий та зрозумілий інтерфейс. Система повинна приховувати технічну складність за простим покроковим інтерфейсом з розширеними можливостями. Користувач повинен мати можливість послідовно обирати компоненти, отримувати повідомлення про можливі проблеми сумісності та швидко змінювати вибрані позиції. У розробленому вебсайті це забезпечується за рахунок SPA-інтерфейсу, у якому список доступних комплектуючих оновлюється без повного перезавантаження сторінки.

Практичне використання інтелектуального помічника може бути корисним як для користувача, так і для інтернет-магазину. Користувач швидше отримує сумісну конфігурацію, менше помиляється під час вибору та не витрачає багато часу на пошук додаткової інформації. Для магазину це також має значення, оскільки автоматизована підтримка вибору, фільтрація несумісних позицій і формування рекомендацій допомагають скоротити шлях користувача від перегляду каталогу до формування конкретної конфігурації та можуть підвищити ефективність взаємодії з електронним сервісом [12].

Технічна реалізація такої системи потребує правильної організації даних. У базі даних мають зберігатися характеристики компонентів, їх ціни, наявність на складі та інформація про користувацькі конфігурації. Логіка підбору може бути реалізована як на серверній, так і на клієнтській частині. У розробленому проєкті клієнтська частина отримує товари з серверного API та виконує швидко перевірку сумісності на основі характеристик компонентів. Це дозволяє швидко оновлювати список доступних варіантів без додаткового запиту до сервера після кожної дії користувача.

Отже, для інтернет-магазину комп'ютерних комплектуючих рекомендаційна система має не просто показувати популярні товари, а допомагати користувачу сформувати сумісну та збалансовану конфігурацію ПК. Для цієї предметної області найбільш доцільним є поєднання рекомендаційного підходу з правилами сумісності, що дозволяє спростити вибір користувача та зменшити ймовірність технічних помилок.

1.4 Особливості перевірки сумісності комп'ютерних компонентів

Підбір комплектуючих для персонального комп'ютера потребує перевірки багатьох параметрів, оскільки кожен компонент впливає на роботу всієї системи. Перевірка сумісності не обмежується лише відповідністю інтерфейсів. Вона також охоплює типи пам'яті, формфактори, габарити комплектуючих, тепловиділення та вимоги до живлення. Саме тому підбір комплектуючих доцільно розглядати як окрему задачу, яку можна частково автоматизувати за допомогою конфігуратора.

Одним із ключових етапів є підбір процесора та материнської плати. Для цього потрібно перевірити відповідність сокету процесора сокету материнської плати. Якщо ці параметри не збігаються, компоненти не можуть бути використані разом. Також важливо враховувати підтримуваний тип оперативної пам'яті, оскільки різні платформи можуть працювати з DDR4 або DDR5. Помилка на цьому етапі призводить до того, що користувач обирає компоненти, які фізично не підходять один до одного.

При виборі оперативної пам'яті основними параметрами є її тип і частота. Наприклад, пам'ять DDR4 не може бути встановлена в плату, яка підтримує лише DDR5. Також важливо враховувати максимально підтримувану частоту, оскільки занадто високі значення можуть працювати лише після додаткового налаштування або розгону.

Вибір відеокарти також пов'язаний із рядом обмежень. Окрім сумісності через інтерфейс PCIe, важливу роль відіграють її габарити. Сучасні відеокарти можуть мати значну довжину та товщину, тому треба перевіряти, чи поміститься вона у вибраний корпус. Якщо корпус компактний, навіть технічно сумісна відеокарта може не підійти за фізичними розмірами.

Окрему увагу потрібно приділити накопичувачам. Для SATA-накопичувачів важливою є наявність відповідних портів на материнській

платі, а для NVMe SSD – наявність M.2-слотів. Якщо потрібний інтерфейс відсутній, накопичувач не зможе бути встановлений у систему. Тому перевірка накопичувачів також має бути частиною загального алгоритму сумісності.

Важливим компонентом є система охолодження процесора. Під час її вибору потрібно враховувати підтримувані сокети, тепловий пакет процесора та висоту кулера. Кулер повинен підходити до процесора за кріпленням, забезпечувати достатнє відведення тепла та фізично поміщатися в обраний корпус. Якщо висота кулера перевищує допустимий розмір корпусу, така конфігурація не може вважатися коректною.

Корпус також впливає на сумісність усієї системи. Він має підтримувати формфактор вибраної материнської плати, наприклад ATX, Micro-ATX або Mini-ITX. Крім того, корпус визначає максимальну довжину відеокарти, допустиму висоту процесорного кулера та формфактор блоку живлення. Тому корпус не можна розглядати лише як зовнішній елемент, оскільки він задає важливі фізичні обмеження для всієї збірки.

Завершальним етапом є підбір блоку живлення. Він повинен забезпечувати достатню потужність для всіх компонентів і мати певний запас для стабільної роботи системи. Для цього можна підсумувати приблизне енергоспоживання вибраних компонентів і додати рекомендований запас потужності. Крім потужності, потрібно також враховувати формфактор блоку живлення, щоб він підходив до вибраного корпусу.

У сучасних конфігураторах ПК частина цих перевірок може виконуватися автоматично. Наприклад, система може фільтрувати материнські плати за сокетом процесора, перевіряти тип оперативної пам'яті, оцінювати габарити відеокарти, перевіряти висоту кулера, розраховувати рекомендовану потужність блоку живлення та приховувати несумісні варіанти. Такі підходи використовуються у роботах, присвячених алгоритмам сумісності та вебконфігураторам персональних комп'ютерів [6, 8]. Практичні онлайн-рішення також демонструють доцільність автоматизованого підбору компонентів із урахуванням їх сумісності [9].

Отже, сумісність комп'ютерних комплектуючих залежить не від одного параметра, а від сукупності технічних і фізичних обмежень. Для її реалізації необхідно мати структурований опис компонентів, набір правил сумісності та механізми швидкої перевірки вибраних позицій. Це безпосередньо впливає на подальше проектування бази даних, архітектури вебзастосунку та алгоритмів роботи інтелектуального помічника.

1.5 Постановка задачі

Проведений аналіз предметної області, існуючих рішень та особливостей перевірки сумісності комп'ютерних компонентів показав, що процес самостійного підбору комплектуючих є складним для багатьох користувачів. Основна проблема полягає в тому, що покупцю потрібно одночасно враховувати технічні характеристики, сумісність між компонентами, бюджет, призначення майбутнього комп'ютера, потужність блоку живлення, габарити корпусу та загальну збалансованість системи. Більшість інтернет-магазинів не надає достатньої допомоги у цьому процесі, тому частина відповідальності за правильність вибору залишається на користувачеві.

У зв'язку з цим виникає потреба у створенні вебсайту для продажу комп'ютерної техніки з інтелектуальним помічником, який буде підтримувати користувача під час прийняття рішення. Такий підхід відповідає загальним принципам побудови інформаційних систем підтримки прийняття рішень, де програмний засіб допомагає користувачу обрати раціональний варіант на основі наявних даних та заданих обмежень [13].

Об'єктом роботи є процес розроблення вебсайту для продажу комп'ютерної техніки з інтегрованим інтелектуальним помічником для підбору комплектуючих.

Метою роботи є створення вебзастосунку для продажу комп'ютерної техніки, який дозволяє користувачам підбирати комплектуючі з урахуванням їх сумісності, продуктивності та бюджету.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасні платформи електронної комерції у сегменті комп'ютерних комплектуючих та визначити їхні обмеження;
- сформулювати функціональні та нефункціональні вимоги до вебсайту та інтелектуального помічника;
- розробити архітектуру клієнт-серверного вебзастосунку з використанням сучасних frontend- та backend-технологій;
- спроектувати структуру бази даних для зберігання інформації про товари, користувачів, кошик, замовлення та збережені конфігурації;
- розробити алгоритми автоматизованої перевірки сумісності комплектуючих на основі їх технічних характеристик;
- реалізувати механізм покрокового підбору компонентів із фільтрацією несумісних позицій;
- реалізувати врахування бюджету користувача та призначення майбутнього комп'ютера під час формування конфігурації;
- реалізувати базові адміністративні функції для керування товарами, замовленнями та активністю системи;
- створити адаптивний та зрозумілий інтерфейс користувача для перегляду каталогу, роботи з конфігуратором, кошиком і особистим кабінетом;
- провести тестування системи та оцінити коректність роботи основних функцій вебзастосунку.

2 ПРОЄКТУВАННЯ ВЕБСАЙТУ ТА ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА

2.1 Визначення функціональних вимог до системи

Перед початком проєктування архітектури, бази даних та алгоритмів потрібно чітко визначити, які функції система повинна виконувати та яким вимогам вона має відповідати. Для цього вимоги поділяють на функціональні, тобто ті, що описують можливості системи, та нефункціональні, які визначають якість її роботи. Оскільки розроблюваний вебсайт містить інтелектуальний помічник для підбору комплектуючих, його можна розглядати як систему підтримки прийняття рішень у межах конкретної предметної області [13].

Розроблений вебсайт має забезпечувати два основні сценарії використання. Перший сценарій пов'язаний зі стандартною роботою інтернет-магазину: переглядом каталогу, пошуком товарів, фільтрацією, додаванням позицій до кошика та оформленням замовлення. Другий сценарій пов'язаний із використанням інтелектуального помічника-конфігуратора, який допомагає користувачу покроково сформувати конфігурацію персонального комп'ютера.

До основних функціональних вимог належить робота з каталогом товарів. Система повинна відображати список доступних комплектуючих із можливістю фільтрації за категоріями, виробником, ціною та наявністю товару. Користувач має бачити назву товару, ціну, зображення, кількість на складі та основні характеристики. Для зручності також має бути реалізований пошук, який дозволяє швидко знайти потрібний товар у каталозі.

Окремою функціональною вимогою є перегляд детальної інформації про товар. Після вибору конкретної позиції користувач повинен мати можливість переглянути повнішу інформацію про компонент: опис, ціну, категорію, наявність і технічні характеристики. Це важливо для комп'ютерних

комплектуючих, оскільки саме характеристики товару використовуються під час перевірки сумісності.

Головною особливістю системи є покроковий конфігуратор ПК. Користувач послідовно обирає компоненти майбутньої збірки: процесор, материнську плату, оперативну пам'ять, відеокарту, накопичувач, систему охолодження, корпус і блок живлення. На кожному кроці система аналізує вже вибрані компоненти та фільтрує доступні позиції відповідно до правил сумісності. Якщо певний товар не підходить до поточної конфігурації, він може бути прихований або відображений як несумісний із поясненням причини.

Для підвищення зручності конфігуратор повинен враховувати не лише технічну сумісність, а й потреби користувача. Перед або під час формування збірки користувач може вказати бюджет та призначення майбутнього комп'ютера, наприклад ігри, робота, навчання або графіка. Це дозволяє системі оцінювати не тільки правильність підбору компонентів, а й загальну збалансованість конфігурації.

Система повинна перевіряти кілька типів сумісності компонентів. До базових перевірок належать відповідність сокету процесора і материнської плати, тип оперативної пам'яті, наявність потрібних інтерфейсів для накопичувачів, підтримка PCIe для відеокарти та формфактор материнської плати. До перевірок за габаритами належать довжина відеокарти, висота процесорного кулера та сумісність корпусу з вибраними компонентами. Також повинна виконуватися перевірка блоку живлення за потужністю і формфактором. Такий підхід спирається на роботу з формалізованими параметрами компонентів і правилами їх зіставлення [14, 15].

Важливою функцією є розрахунок рекомендованої потужності блоку живлення. Після вибору основних компонентів система розраховує їхнє загальне енергоспоживання та рекомендує лише ті блоки живлення, які мають запас потужності 20 %. Також виконується перевірка формфактора блоку живлення відповідно до розміру обраного корпусу. У результаті користувачу

пропонуються тільки ті моделі, які відповідають розрахованому енергоспоживанню та розміру корпусу.

Після завершення підбору користувач повинен мати можливість переглянути підсумкову конфігурацію, її загальну вартість і список вибраних комплектуючих. Готову збірку можна додати до кошика як окрему позицію або зберегти в особистому кабінеті для подальшого перегляду чи редагування. Це дозволяє користувачу не втрачати підібрані варіанти та повертатися до них пізніше.

Система також повинна підтримувати стандартний функціонал інтернет-магазину. До нього належать реєстрація користувача, авторизація, робота з кошиком, оформлення замовлення, перегляд історії замовлень та збережених конфігурацій. Після оформлення замовлення система повинна зберігати його склад, загальну суму, статус і дату створення.

Для адміністратора мають бути передбачені окремі функції керування системою. Адміністратор повинен мати можливість переглядати замовлення, змінювати їх статус, оновлювати залишки товарів на складі та переглядати основні події в журналі активності. Наявність такого функціоналу спрощує керування інтернет-магазином і дозволяє контролювати роботу основних процесів системи.

Щодо нефункціональних вимог, то насамперед важлива продуктивність роботи системи. Вебзастосунок повинен швидко реагувати на дії користувача, зокрема на фільтрацію товарів, вибір компонентів у конфігураторі, зміну бюджету або призначення ПК. Перевірка сумісності та оновлення списку доступних позицій мають виконуватися без повного перезавантаження сторінки.

З точки зору зручності використання, інтерфейс має бути інтуїтивно зрозумілим і послідовним. Користувач повинен бачити, на якому етапі підбору він перебуває, які компоненти вже вибрані, які товари є сумісними, а які не підходять до поточної конфігурації. Повідомлення про помилки та

попередження мають бути зрозумілими для користувача й пояснювати причину несумісності.

Вимоги до надійності та безпеки включають базовий захист даних користувачів. У межах прототипу реалізовано реєстрацію та авторизацію користувачів, збереження паролів у хешованому вигляді та розмежування доступу за ролями. Користувач має доступ до власного кошика, збережених конфігурацій та історії замовлень, а адміністративні функції доступні лише користувачу з відповідною роллю.

Архітектура системи має залишатися придатною для подальшого розширення. У майбутньому до вебсайту можна додавати нові категорії товарів, розширювати набір характеристик комплектуючих, доповнювати алгоритми перевірки сумісності та вдосконалювати рекомендації інтелектуального помічника. Це важливо, оскільки ринок комп'ютерної техніки швидко змінюється, а нові компоненти можуть вимагати додаткових правил перевірки.

Отже, сформульовані функціональні та нефункціональні вимоги визначають основу для подальшого проєктування архітектури, бази даних та алгоритмів роботи інтелектуального помічника. Саме ці вимоги враховуються під час розроблення структури вебзастосунку та його програмної реалізації.

2.2 Проєктування архітектури клієнт-серверного вебзастосунку

Під час проєктування архітектури вебзастосунку було враховано необхідність швидкої роботи інтерфейсу, зручного перегляду каталогу, оброблення замовлень та покрокового підбору комп'ютерних комплектуючих. Оскільки користувач повинен отримувати результат після кожної дії без повного перезавантаження сторінки, для клієнтської частини було обрано підхід односторінкового вебзастосунку.

Загальна архітектура системи побудована за клієнт-серверним принципом і складається з трьох основних рівнів: клієнтського рівня (frontend), серверного рівня (backend) та рівня даних (база даних). Клієнтський рівень відповідає за відображення інтерфейсу користувача та взаємодію з ним. Серверний рівень забезпечує оброблення HTTP-запитів, маршрутизацію, роботу з користувачами, товарами, кошиком, замовленнями та адміністративними функціями. Рівень даних відповідає за зберігання інформації про товари, категорії, користувачів, кошик, замовлення, збережені конфігурації та журнал активності.

Клієнтська частина реалізована у вигляді SPA-застосунку з використанням бібліотеки React. Такий підхід дозволяє створювати інтерфейс із незалежних компонентів, повторно використовувати їх у різних частинах сторінки та оновлювати лише потрібні елементи без повного перезавантаження сайту [16]. Основними частинами клієнтського інтерфейсу є каталог товарів, сторінка категорій, картка товару, конфігуратор ПК, кошик, особистий кабінет користувача та адміністративна панель.

Клієнтська частина взаємодіє із сервером через асинхронні HTTP-запити. Дані передаються у форматі JSON, що є зручним для обміну інформацією між frontend- та backend-частинами. Наприклад, клієнтська частина може отримувати список товарів і категорій, надсилати дані для авторизації користувача, оновлювати кошик, зберігати конфігурацію або створювати замовлення. Завдяки цьому інтерфейс залишається динамічним і швидко реагує на дії користувача.

Серверна частина реалізована на основі Node.js з використанням фреймворку Express. Node.js використовується як середовище виконання серверного JavaScript-коду, а Express забезпечує створення REST API, маршрутизацію запитів, оброблення даних від клієнтської частини та формування відповідей у форматі JSON [17, 18]. Такий підхід дозволяє відокремити клієнтську логіку від серверної та зробити взаємодію між частинами системи більш зрозумілою.

REST API у розробленому вебзастосунку використовується для отримання каталогу товарів, списку категорій, реєстрації та авторизації користувачів, роботи з кошиком, оформлення замовлень, керування збереженими конфігураціями та виконання адміністративних дій. Кожна група операцій реалізується через окремі маршрути API, що спрощує підтримку серверної частини та дозволяє надалі розширювати функціональність системи.

Для взаємодії серверної частини з базою даних використовується Prisma ORM. Prisma дозволяє описувати структуру даних у вигляді моделей, виконувати запити до бази даних через об'єктний інтерфейс і зменшує потребу в написанні SQL-запитів вручну [19]. У межах проєкту Prisma використовується для роботи з товарами, категоріями, користувачами, кошиком, замовленнями, збереженими конфігураціями та журналом активності.

Рівень даних, у свою чергу, представлений реляційною базою даних PostgreSQL. У ній зберігається інформація, необхідна для роботи інтернет-магазину та конфігуратора: товари, їх технічні характеристики, категорії, користувачі, кошик, історія замовлень, збережені конфігурації та події активності. Технічні характеристики комплектуючих зберігаються у гнучкому форматі, що дозволяє використовувати різні набори параметрів для різних категорій товарів.

Окремою особливістю архітектури є організація роботи інтелектуального помічника. Серверна частина надає клієнтській частині дані про товари та їх характеристики, а перевірка сумісності виконується на клієнтському рівні в модулі конфігуратора. Такий підхід дозволяє швидко оновлювати список доступних компонентів після кожного вибору користувача без додаткового запиту до сервера. Клієнтська частина аналізує вже вибрані комплектуючі, перевіряє їхні характеристики та відображає сумісні або несумісні варіанти.

Взаємодія між рівнями системи відбувається за такою схемою: користувач виконує дію в інтерфейсі React, клієнтська частина за потреби надсилає HTTP-запит до Express API, сервер обробляє запит і звертається до PostgreSQL через Prisma ORM. Після цього сервер повертає результат у форматі JSON, а клієнтська частина оновлює відповідний елемент інтерфейсу. Для конфігуратора частина перевірок виконується безпосередньо на клієнтській стороні після отримання даних про товари.

На рисунку 2.1 наведено загальну схему архітектури системи, яка ілюструє основні компоненти та зв'язки між ними.

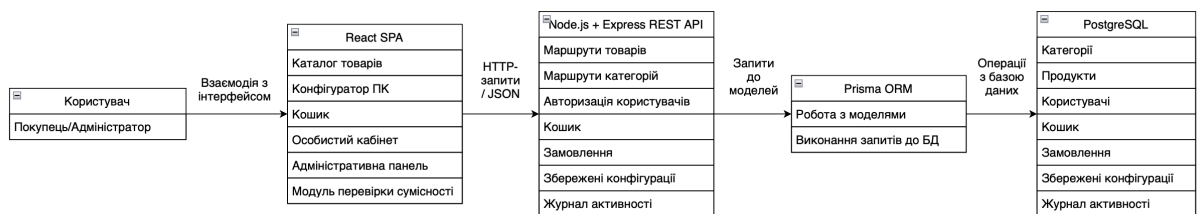


Рисунок 2.1 – Схема архітектури клієнт-серверного вебзастосунку

Запропонована архітектура забезпечує розділення відповідальності між клієнтською частиною, серверною частиною та базою даних. React відповідає за інтерфейс користувача, Express API – за оброблення запитів і взаємодію з бізнес-даними, Prisma ORM – за доступ до PostgreSQL, а база даних – за зберігання інформації. Така структура є зручною для подальшого розширення вебсайту, додавання нових категорій товарів, удосконалення конфігуратора та розвитку адміністративних функцій.

2.3 Розроблення структури бази даних для зберігання інформації про товари та користувачів

Для функціонування вебсайту з продажу комп'ютерної техніки та роботи інтелектуального помічника-конфігуратора потрібна база даних для зберігання каталогу товарів, профілів користувачів, кошиків, замовлень, збережених конфігурацій та журналу активності. Для проєкту використовується реляційна база даних PostgreSQL, оскільки вона добре підходить для зберігання структурованих даних і підтримує роботу з JSON-полями. Це важливо для предметної області комп'ютерних комплектуючих, оскільки різні категорії товарів мають різні набори технічних характеристик [20].

Для роботи з базою даних використовується Prisma ORM. Вона дозволяє описувати структуру даних у вигляді моделей, виконувати операції з базою через об'єктний інтерфейс та зменшує потребу в написанні SQL-запитів вручну [19]. У межах проєкту Prisma використовується як проміжний рівень між Express API та PostgreSQL.

Основні сутності предметної області та їхні атрибути наведено в таблиці 2.1.

У програмній реалізації наведені сутності відповідають моделям Prisma, описаним у файлі `schema.prisma`. Зв'язок між категоріями та товарами реалізовано як «один-до-багатьох», оскільки одна категорія може містити багато товарів. Користувач може мати один кошик, декілька замовлень, декілька збережених конфігурацій та записи в журналі активності. Для зберігання технічних характеристик товарів, вмісту кошика, складу замовлення та конфігурації використовуються JSON-поля, що дозволяє гнучко зберігати дані різної структури.

Таблиця 2.1 – Сутності та атрибути бази даних

Сутність	Призначення	Основні атрибути
1	2	3
Категорія	Зберігання категорій товарів	№Категорії, назва категорії, назва одного елемента категорії, зображення, порядок відображення
Товар	Зберігання інформації про комп'ютерні комплектуючі	№Товару, №Категорії, назва товару, виробник, ціна, кількість на складі, зображення, опис, технічні характеристики, порядок відображення, дата створення, дата оновлення
Користувач	Зберігання облікових даних покупців та адміністраторів	№Користувача, ім'я користувача, електронна пошта, хешований пароль, роль, дата створення облікового запису
Кошик	Зберігання товарів і готових конфігурацій, доданих користувачем до кошика	№Користувача, вміст кошика, дата останнього оновлення

Продовження таблиці 2.1

1	2	3
Замовлення	Зберігання інформації про оформлені покупки	№Замовлення, номер замовлення, №користувача, дані покупця, вміст замовлення, загальна сума, статус, дата створення
Збережена конфігурація	Зберігання конфігурацій ПК, створених користувачем у конфігураторі	№Конфігурації, №Користувача, назва конфігурації, склад конфігурації, загальна вартість, дата створення
Журнал активності	Зберігання основних подій системи для подальшого перегляду адміністратором	№Запису, №Користувача, повідомлення, дата створення

Далі коротко описано призначення кожної сутності:

– таблиця «Категорії» призначена для зберігання категорій товарів. У системі категорії використовуються для поділу комплектуючих на окремі групи: процесори, материнські плати, оперативна пам'ять, відеокарти, накопичувачі, системи охолодження, корпуси та блоки живлення. Поле «назва категорії» містить назву категорії, «назва одного елемента категорії» використовується для відображення назви одного елемента цієї категорії, «зображення» зберігає посилання на зображення категорії, а «порядок відображення» визначає порядок відображення категорій в інтерфейсі;

– таблиця «Товар» є центральною в базі даних, оскільки саме в ній зберігається інформація про товари. Вона містить назву товару, бренд, опис,

ціну, зображення, кількість на складі та зв'язок із категорією. Для зберігання технічних характеристик використовується поле «технічні характеристики» типу JSON. Такий підхід є зручним, оскільки різні типи комплектуючих мають різні характеристики. Наприклад, для процесора важливими є сокет, кількість ядер, частота та TDP, для відеокарти – обсяг пам'яті, довжина та енергоспоживання, а для блоку живлення – потужність і формфактор. Використання JSON-поля для характеристик дозволяє не створювати окрему таблицю для кожного типу комплектуючих. Це спрощує структуру бази даних і дає змогу додавати нові параметри товарів без повної зміни схеми. Саме ці характеристики надалі використовуються клієнтським модулем конфігуратора для перевірки сумісності компонентів;

- таблиця «Користувачі» призначена для зберігання даних користувачів. Вона містить ім'я користувача, електронну пошту, хешований пароль, роль і дату створення облікового запису. Поле «роль» використовується для розмежування доступу між звичайним покупцем та адміністратором. Звичайний користувач може працювати з каталогом, кошиком, замовленнями та збереженими конфігураціями, а адміністратор має доступ до функцій керування товарами, замовленнями та журналом активності;

- таблиця «Кошик» використовується для зберігання кошика користувача. Вона має зв'язок із таблицею «Користувачі» через поле «№Користувача». Склад кошика зберігається у полі «вміст кошика» у форматі JSON. Це дозволяє зберігати як окремі товари, так і сформовані конфігурації ПК, які користувач додає до кошика після завершення роботи з інтелектуальним помічником. Поле «дата останнього оновлення» фіксує час останнього оновлення кошика;

- таблиця «Замовлення» призначена для зберігання оформлених замовлень. Вона містить ідентифікатор користувача, перелік позицій замовлення, загальну суму, статус і дату створення. Поле «вміст замовлення» зберігає склад замовлення у форматі JSON. Такий підхід дозволяє зафіксувати

товари, їх кількість і ціну на момент оформлення замовлення. Це важливо, оскільки ціни або залишки товарів можуть змінюватися пізніше, але історія замовлень користувача має залишатися коректною;

– таблиця «Збережені конфігурації» використовується для зберігання конфігурацій ПК, створених користувачем у конфігураторі. Кожна конфігурація має назву, зв'язок із користувачем, список вибраних компонентів, загальну вартість і дату створення. Склад конфігурації зберігається у полі «склад конфігурації» у форматі JSON. Завдяки цьому користувач може зберегти підібрану збірку, переглянути її пізніше або додати до кошика;

– таблиця «Журнал активності» призначена для зберігання основних подій системи. У ній можуть фіксуватися дії, пов'язані з оформленням замовлень, оновленням залишків товарів, зміною статусів замовлень або іншими важливими подіями. Наявність журналу активності є корисною для адміністратора, оскільки дозволяє відстежувати основні зміни в системі та контролювати роботу вебсайту.

Особливістю спроектованої бази даних є те, що правила сумісності не зберігаються в окремій таблиці. Вони реалізовані алгоритмічно на рівні модуля конфігуратора. База даних зберігає технічні характеристики комплектуючих у полі «технічні характеристики», а логіка перевірки сумісності використовує ці характеристики для порівняння компонентів між собою. Наприклад, під час вибору материнської плати перевіряється відповідність сокета процесора, під час вибору оперативної пам'яті – тип пам'яті, а під час вибору корпусу – габарити відеокарти та висота кулера.

Зв'язки між основними сутностями бази даних можна описати таким чином. Одна категорія може містити багато товарів, тому між «Категорія» і «Товар» існує зв'язок «один-до-багатьох». Один користувач може мати один кошик, багато замовлень і багато збережених конфігурацій. Тому «Користувач» пов'язаний із «Кошик», «Замовлення» та «Збережені

конфігурації». Журнал активності зберігає окремі події системи та може використовуватися для контролю адміністративних дій.

Зв'язки між описаними сутностями наведено на ER-діаграмі (рис. 2.2).

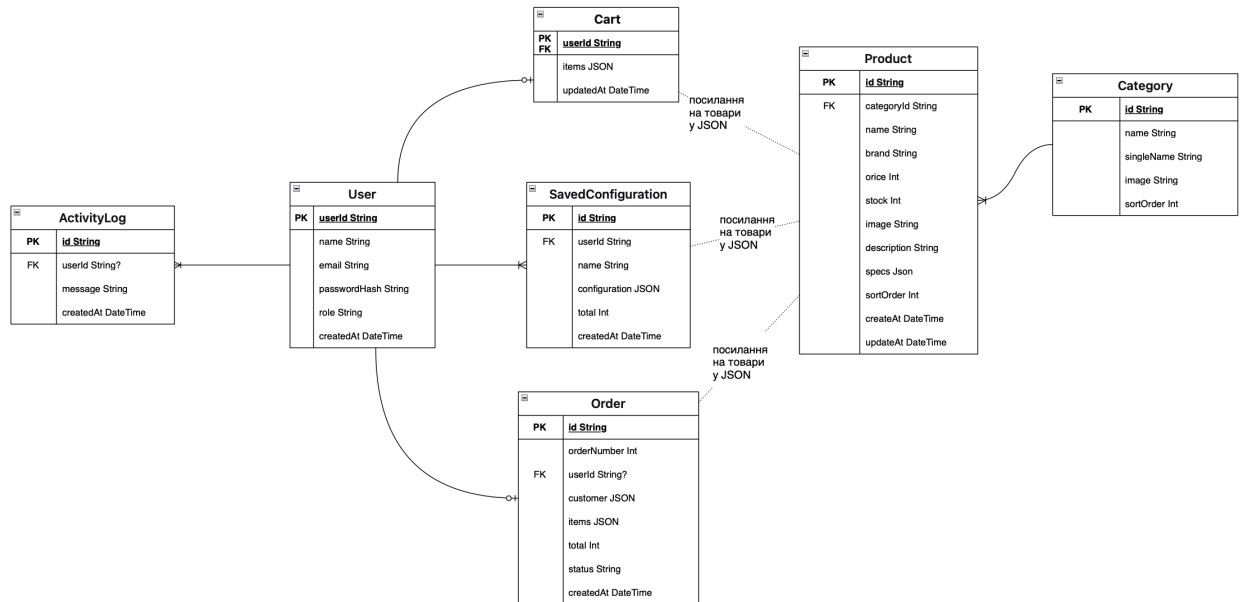


Рисунок 2.2 – ER-діаграма бази даних

Окремі позиції кошика, замовлення та збереженої конфігурації зберігаються у JSON-полях. Тому зв'язок цих сутностей із товарами є логічним: у JSON-структурах міститься інформація про вибрані товари та їх кількість, але окрема таблиця зв'язку для позицій кошика або замовлення в межах даної реалізації не використовується.

Спроектowana база даних відповідає потребам розробленого вебзастосунку. Вона дозволяє зберігати товари та їх характеристики, облікові записи користувачів, кошик, замовлення, збережені конфігурації та журнал активності. Використання Prisma ORM спрощує доступ до даних із серверної частини, а JSON-поля дають змогу гнучко зберігати характеристики комплектуючих і склад користувацьких конфігурацій.

2.4 Моделювання логіки роботи інтелектуального помічника

Інтелектуальний помічник є центральним елементом вебсайту, який відрізняє його від звичайних інтернет-магазинів комп'ютерної техніки. Його основне завдання полягає в тому, щоб супроводжувати користувача під час формування конфігурації персонального комп'ютера, перевіряти сумісність вибраних компонентів і допомагати підібрати збалансовану збірку відповідно до потреб користувача.

Логіка роботи помічника побудована на покроковому підході. Користувач не обирає всі комплектуючі одночасно, а поступово переходить від одного компонента до іншого. Це спрощує процес підбору, оскільки на кожному етапі система враховує вже вибрані позиції та відображає лише ті товари, які можуть бути використані в поточній конфігурації. Такий підхід відповідає принципам систем на основі обмежень, у яких кожен новий вибір перевіряється відповідно до заданих правил предметної області [11].

Перед початком підбору користувач може вказати бюджет і призначення майбутнього комп'ютера. У системі передбачено кілька сценаріїв використання: ігри, робота, навчання та графіка. Такі параметри допомагають не лише перевіряти технічну сумісність компонентів, а й оцінювати загальну відповідність конфігурації потребам користувача. Наприклад, для ігрового ПК важливу роль відіграє відеокарта, а для робочої або навчальної конфігурації більшу увагу можна приділити загальній збалансованості та вартості системи.

Загальний алгоритм роботи помічника складається з кількох послідовних кроків.

Крок 1. Початок роботи з конфігуратором. Користувач відкриває сторінку конфігуратора ПК. На цьому етапі система відображає доступні категорії комплектуючих, поточний стан конфігурації, загальну вартість збірки та основні елементи керування. Користувач може перейти до вибору компонентів, скористатися пошуком або переглянути інформацію про

доступні товари. Також у конфігураторі передбачено можливість роботи з бюджетом та призначенням майбутнього комп'ютера.

Крок 2. Вибір бюджету та призначення ПК. Перед формуванням конфігурації користувач може вказати орієнтовний бюджет і вибрати призначення майбутнього комп'ютера. У системі передбачено кілька сценаріїв використання: ігри, робота, навчання та графіка. Ці параметри використовуються для подальшої оцінки збірки, перевірки її загальної збалансованості та відповідності потребам користувача.

Крок 3. Вибір процесора. Першим основним компонентом є процесор, оскільки від нього залежить подальший підбір материнської плати, оперативної пам'яті та системи охолодження. Для процесора враховуються такі характеристики, як сокет, кількість ядер, частота та тепловий пакет. Після вибору конкретної моделі система зберігає процесор у поточній конфігурації та використовує його характеристики під час наступних перевірок.

Крок 4. Вибір материнської плати. На цьому етапі система порівнює сокет вибраного процесора із сокетом материнської плати. Якщо ці параметри не збігаються, така плата вважається несумісною з поточною конфігурацією. Також враховується підтримуваний тип оперативної пам'яті та формфактор материнської плати, який надалі використовується під час вибору корпусу. У результаті користувач бачить лише відповідні варіанти або отримує пояснення, чому певна позиція не підходить.

Крок 5. Вибір оперативної пам'яті. Після вибору процесора та материнської плати користувач переходить до підбору оперативної пам'яті. Система перевіряє відповідність типу пам'яті, наприклад DDR4 або DDR5, до характеристик вибраної платформи. Також може враховуватися підтримувана частота пам'яті. Завдяки цьому користувач отримує тільки ті модулі оперативної пам'яті, які можуть працювати з уже вибраними компонентами.

Крок 6. Вибір відеокарти. Далі користувач обирає відеокарту. Для цього компонента важливими є інтерфейс підключення, орієнтовне енергоспоживання та фізичні розміри. Якщо корпус ще не вибрано, перевірка

довжини відеокарти виконується пізніше. Після вибору корпусу система порівнює довжину відеокарти з максимально допустимою довжиною, яку підтримує корпус.

Крок 7. Вибір накопичувача. На цьому етапі користувач обирає накопичувач для зберігання даних. Система враховує тип підключення накопичувача: SATA або NVMe. Для NVMe-накопичувача перевіряється наявність відповідного M.2-слота на материнській платі, а для SATA-накопичувача – наявність потрібних портів. Це дозволяє уникнути вибору накопичувача, який неможливо підключити до вибраної плати.

Крок 8. Вибір системи охолодження. Окремим етапом є вибір системи охолодження процесора. Система перевіряє, чи підтримує кулер сокет вибраного процесора, чи достатній його тепловий запас для відведення тепла, а також чи поміститься він у вибраний корпус. Якщо корпус ще не вибрано, перевірка висоти кулера виконується після його вибору.

Крок 9. Вибір корпусу. Після вибору основних компонентів користувач переходить до вибору корпусу. Система перевіряє, чи підтримує корпус формфактор вибраної материнської плати, чи достатньо місця для відеокарти, чи підходить висота процесорного кулера та чи підтримується потрібний формфактор блоку живлення. Корпус є важливим елементом конфігурації, оскільки саме він визначає фізичні обмеження для встановлення інших комплектуючих.

Крок 10. Вибір блоку живлення. Останнім етапом є вибір блоку живлення. Система підсумовує приблизне енергоспоживання вибраних компонентів і додає рекомендований запас потужності. Після цього користувачу пропонуються блоки живлення, які відповідають розрахованій потужності та підходять до вибраного корпусу за формфактором. Це дозволяє зменшити ризик вибору недостатньо потужного або фізично несумісного блоку живлення.

Крок 11. Пошук і відображення несумісних компонентів. Під час роботи з конфігуратором користувач може скористатися пошуком для швидкого

знаходження потрібного компонента. Також у системі передбачено можливість показувати несумісні товари. У такому разі користувач бачить не лише доступні позиції, а й ті товари, які не підходять до поточної конфігурації, разом із поясненням причини несумісності.

Крок 12. Діагностика конфігурації. Після вибору компонентів система формує діагностику поточної збірки. Вона містить інформацію про загальну вартість конфігурації, відповідність бюджету, орієнтовне енергоспоживання, рекомендовану потужність блоку живлення та загальну збалансованість системи. Якщо конфігурація має проблеми, користувач отримує відповідні попередження.

Крок 13. Завершення роботи з конфігуратором. Після завершення підбору користувач бачить підсумковий список вибраних комплектуючих, їхню загальну вартість та основні повідомлення системи. Готову конфігурацію можна додати до кошика як окрему позицію або зберегти в особистому кабінеті для подальшого перегляду чи редагування.

Описана логіка роботи інтелектуального помічника дозволяє спростити процес підбору комплектуючих і зменшити кількість помилок під час формування конфігурації ПК. Користувач отримує не просто каталог товарів, а покроковий інструмент, який враховує сумісність компонентів, бюджет, призначення комп'ютера та загальну збалансованість збірки.

Для наочного подання логіки роботи інтелектуального помічника було побудовано блок-схему, наведену на рисунку 2.3. Вона відображає послідовність дій користувача під час формування конфігурації ПК.

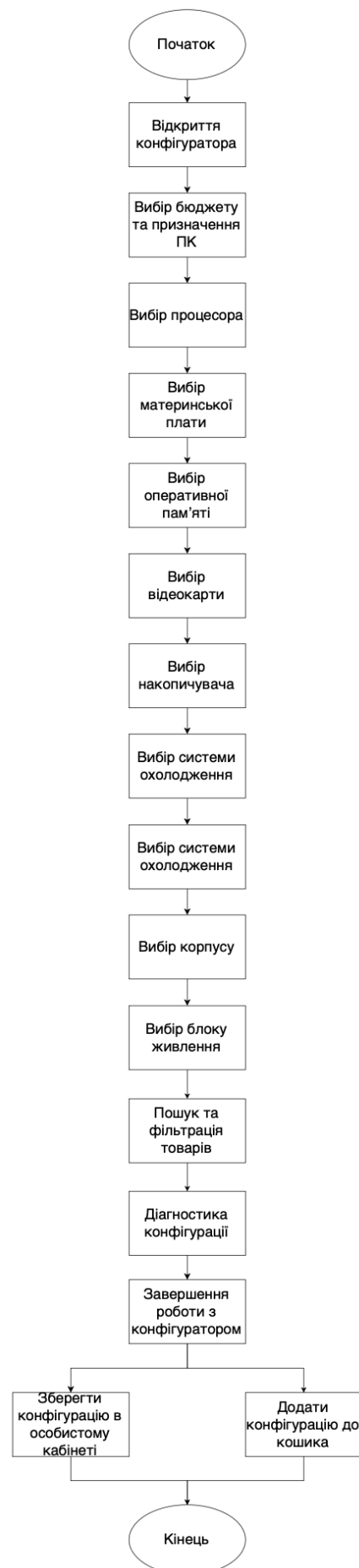


Рисунок 2.3 – Блок-схема алгоритму роботи інтелектуального помічника

2.5 Розроблення алгоритмів перевірки сумісності комплектуючих

Алгоритми перевірки сумісності комплектуючих є основою роботи інтелектуального помічника-конфігуратора. Їхнє завдання полягає в тому, щоб аналізувати вже вибрані користувачем компоненти та визначати, які товари можуть бути додані до поточної конфігурації без порушення технічних вимог. На відміну від звичайного каталогу, де користувач самостійно аналізує характеристики товарів, конфігуратор автоматично виконує частину перевірок і допомагає уникнути типових помилок під час складання ПК.

У розробленому вебзастосунку правила сумісності реалізовані алгоритмічно на клієнтській частині. Серверна частина через REST API надає список товарів та їх технічні характеристики, а модуль конфігуратора виконує перевірку на основі отриманих даних. Такий підхід дозволяє швидко оновлювати список доступних компонентів після кожного вибору користувача без додаткового звернення до сервера. Для зберігання технічних параметрів товарів використовується поле «технічні характеристики», у якому містяться характеристики, необхідні для перевірки сумісності.

Загальна логіка перевірки полягає в тому, що для кожного компонента система порівнює його характеристики з уже вибраними елементами конфігурації. Якщо товар відповідає вимогам поточної збірки, він залишається доступним для вибору. Якщо хоча б одне важливе правило порушується, товар позначається як несумісний або не відображається серед рекомендованих варіантів. При цьому користувач може переглянути причину несумісності, що робить роботу конфігуратора більш зрозумілою.

Першою важливою перевіркою є сумісність процесора та материнської плати. Для цього система порівнює сокет процесора із сокетом материнської плати. Якщо значення не збігаються, материнська плата не може бути використана в поточній конфігурації. Це одне з базових правил, оскільки фізична несумісність сокетів повністю унеможливорює спільне використання цих компонентів.

Наступна група перевірок пов'язана з оперативною пам'яттю. Система аналізує тип пам'яті, який підтримується материнською платою та процесором, і порівнює його з типом вибраного модуля пам'яті. Наприклад, модуль DDR4 не може бути використаний із материнською платою, яка підтримує лише DDR5. Також може враховуватися максимально підтримувана частота пам'яті. Якщо частота модуля перевищує допустимі значення, система може повідомити користувача про потенційне обмеження або несумісність.

Для накопичувачів перевіряється тип підключення. Якщо користувач обирає NVMe-накопичувач, система перевіряє наявність M.2-слота на материнській платі. Для SATA-накопичувачів враховується наявність відповідних SATA-портів. Це дозволяє уникнути ситуації, коли вибраний накопичувач неможливо фізично підключити до материнської плати.

Під час вибору відеокарти система враховує наявність відповідного інтерфейсу підключення та фізичні розміри компонента. Якщо корпус уже вибрано, довжина відеокарти порівнюється з максимально допустимою довжиною, яку підтримує корпус. Якщо відеокарта перевищує це значення, вона позначається як несумісна. Крім того, її енергоспоживання враховується під час подальшого розрахунку рекомендованої потужності блоку живлення.

Окремо перевіряється сумісність системи охолодження. Кулер повинен підтримувати сокет вибраного процесора та мати достатній тепловий запас для його охолодження. Для цього система порівнює TDP процесора з тепловими можливостями кулера. Також перевіряється висота кулера, якщо корпус уже вибрано. Якщо висота системи охолодження перевищує допустиме значення для корпусу, такий кулер не може бути використаний у поточній збірці.

Корпус перевіряється за кількома параметрами. Насамперед система визначає, чи підтримує корпус формфактор вибраної материнської плати, наприклад ATX, Micro-ATX або Mini-ITX. Також враховується максимальна довжина відеокарти, допустима висота процесорного кулера та підтримуваний формфактор блоку живлення. Через це корпус є одним із компонентів, який впливає одразу на кілька інших елементів конфігурації.

Окремим етапом є розрахунок блоку живлення. Алгоритм підсумовує значення TDP усіх вибраних компонентів: процесора, відеокарти, накопичувачів та інших пристроїв. До отриманої суми додається рекомендований запас потужності, який враховує можливі пікові навантаження. У межах роботи використовується запас 20 %, тому рекомендована потужність блоку живлення обчислюється за формулою:

$$P_{recommended} = (\sum P_{components}) * 1,2, \quad (2.1)$$

де $P_{recommended}$ – рекомендована потужність блоку живлення;

$\sum P_{components}$ – сумарне енергоспоживання всіх вибраних компонентів.

Після розрахунку система порівнює отримане значення з потужністю доступних блоків живлення. До списку сумісних потрапляють лише ті моделі, потужність яких не менша за розраховане рекомендоване значення. Додатково перевіряється формфактор блоку живлення, щоб він відповідав корпусу.

Крім перевірки технічної сумісності, у конфігураторі враховується загальна вартість збірки. Система підсумовує ціни вибраних компонентів і порівнює результат із бюджетом користувача. Якщо вартість конфігурації перевищує бюджет, користувач отримує відповідне попередження. Це не завжди означає технічну помилку, але допомагає користувачу контролювати витрати під час складання ПК.

Також виконується оцінка збалансованості конфігурації. Вона потрібна для того, щоб попередити користувача про можливі невідповідності між рівнем процесора та відеокарти або про загальну нерівномірність збірки. Наприклад, якщо користувач обирає дуже потужний процесор і слабку відеокарту для ігрового сценарію, система може повідомити, що конфігурація не є оптимальною для обраного призначення. Така перевірка не блокує вибір, але допомагає користувачу прийняти більш обґрунтоване рішення.

Загальний алгоритм перевірки сумісності можна описати так:

- отримання списку товарів та їх характеристик із серверної частини;

- збереження вибраного користувачем компонента в поточній конфігурації;
- визначення категорії компонента, який потрібно підібрати наступним;
- порівняння характеристик доступних товарів із уже вибраними компонентами;
- формування списку сумісних і несумісних позицій;
- відображення користувачу доступних товарів та пояснення причин несумісності;
- оновлення загальної вартості, енергоспоживання та діагностики конфігурації.

Розроблені алгоритми дозволяють автоматизувати основні перевірки, які зазвичай користувач виконує вручну. Завдяки цьому інтелектуальний помічник допомагає поступово формувати конфігурацію ПК, враховуючи сумісність компонентів, бюджет, призначення комп'ютера, габаритні обмеження та вимоги до живлення. Це підвищує зручність користування вебсайтом і зменшує ймовірність помилок під час підбору комплектуючих.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБСАЙТУ ДЛЯ ПРОДАЖУ КОМП'ЮТЕРНОЇ ТЕХНІКИ

3.1 Обґрунтування вибору середовища та технологій програмної реалізації

Після проєктування архітектури, структури бази даних та логіки роботи інтелектуального помічника було виконано програмну реалізацію вебсайту для продажу комп'ютерної техніки. Основним завданням цього етапу було створення клієнт-серверного вебзастосунку, який поєднує функції інтернет-магазину та конфігуратора ПК. У межах реалізації потрібно було забезпечити перегляд каталогу товарів, пошук і фільтрацію комплектуючих, роботу з кошиком, оформлення замовлень, збереження готових конфігурацій, адміністративне керування товарами та перевірку сумісності компонентів.

Для реалізації вебсайту було обрано набір технологій, який дозволяє розділити клієнтську частину, серверну частину та базу даних. Такий підхід є зручним для розроблення вебзастосунків, оскільки кожен рівень системи виконує окремі функції. Клієнтська частина відповідає за інтерфейс користувача, серверна частина обробляє запити та взаємодіє з базою даних, а база даних зберігає інформацію про товари, користувачів, кошики, замовлення та збережені конфігурації.

Клієнтську частину реалізовано з використанням бібліотеки React. Її було обрано через зручну компонентну структуру, яка дозволяє розділяти інтерфейс на окремі логічні частини: каталог, картку товару, кошик, особистий кабінет, адміністративну панель і конфігуратор ПК. Це спрощує підтримку коду та дозволяє змінювати окремі елементи інтерфейсу без повної переробки всієї клієнтської частини.

Для серверної частини використано Node.js та Express. Node.js забезпечує виконання серверного JavaScript-коду, а Express використовується для створення REST API. Через API клієнтська частина отримує список

товарів і категорій, надсилає дані авторизації, оновлює кошик, створює замовлення та зберігає конфігурації ПК. Обмін даними між клієнтом і сервером виконується у форматі JSON, що є зручним для вебзастосунків такого типу.

Для зберігання даних використано PostgreSQL. У базі даних зберігаються товари, категорії, користувачі, кошики, замовлення, збережені конфігурації та журнал активності. Оскільки різні категорії комплектуючих мають різні набори характеристик, у структурі даних використано JSON-поля. Це дозволяє зберігати характеристики процесорів, материнських плат, відеокарт, блоків живлення та інших компонентів без створення великої кількості окремих таблиць.

Для взаємодії серверної частини з базою даних використано Prisma ORM. Цей інструмент дозволяє описувати структуру даних у вигляді моделей і виконувати запити до бази даних через об'єктний інтерфейс. Завдяки цьому серверний код стає зрозумілішим, а робота з товарами, користувачами, замовленнями та збереженими конфігураціями спрощується.

Розроблення проєкту виконувалося у середовищі Visual Studio Code. Це середовище було використано для роботи з frontend- та backend-кодом, файлами стилів, структурою бази даних, налаштуваннями Prisma та допоміжними файлами проєкту. Для запуску й перевірки роботи застосунку використовувалися npm-скрипти, локальний сервер і підключення до бази даних PostgreSQL.

Окремою особливістю обраного стеку є те, що він добре підходить для реалізації SPA-застосунку з динамічним інтерфейсом. Користувач може переглядати каталог, переходити між розділами, працювати з кошиком і конфігуратором без повного перезавантаження сторінки. Це важливо для вебсайту з підбором комплектуючих, оскільки список доступних компонентів має швидко змінюватися після кожної дії користувача.

Отже, обрані технології відповідають завданням програмної реалізації вебсайту для продажу комп'ютерної техніки. React забезпечує побудову

клієнтського інтерфейсу, Node.js та Express відповідають за серверну логіку й API, PostgreSQL використовується для зберігання даних, а Prisma ORM спрощує взаємодію з базою даних. Такий стек дозволяє реалізувати основні функції інтернет-магазину, роботу з користувачами та збереження даних, необхідних для функціонування інтелектуального помічника-конфігуратора.

3.2 Структура програмного проєкту

Програмний проєкт побудовано відповідно до клієнт-серверної архітектури, описаної у другому розділі. Його структура розділена на кілька основних частин: клієнтську частину, серверну частину, модулі для роботи з базою даних, службові файли та ресурси інтерфейсу. Такий підхід дозволяє розділити відповідальність між різними частинами системи та спростити подальшу підтримку вебзастосунку.

Клієнтська частина розміщена в каталозі `src`. У ньому знаходяться файли React-застосунку, компоненти інтерфейсу, стилі, допоміжні функції та модулі для взаємодії із серверним API. Основним файлом клієнтської частини є `App.js`, який відповідає за запуск основного інтерфейсу та підключення головного компонента сторінки. Основна логіка відображення розділів сайту зосереджена у `SearchPage.js`, де реалізовано перемикання між каталогом, конфігуратором, кошиком, особистим кабінетом та адміністративною частиною.

На рисунку 3.1 наведено загальну структуру програмного проєкту у середовищі Visual Studio Code.

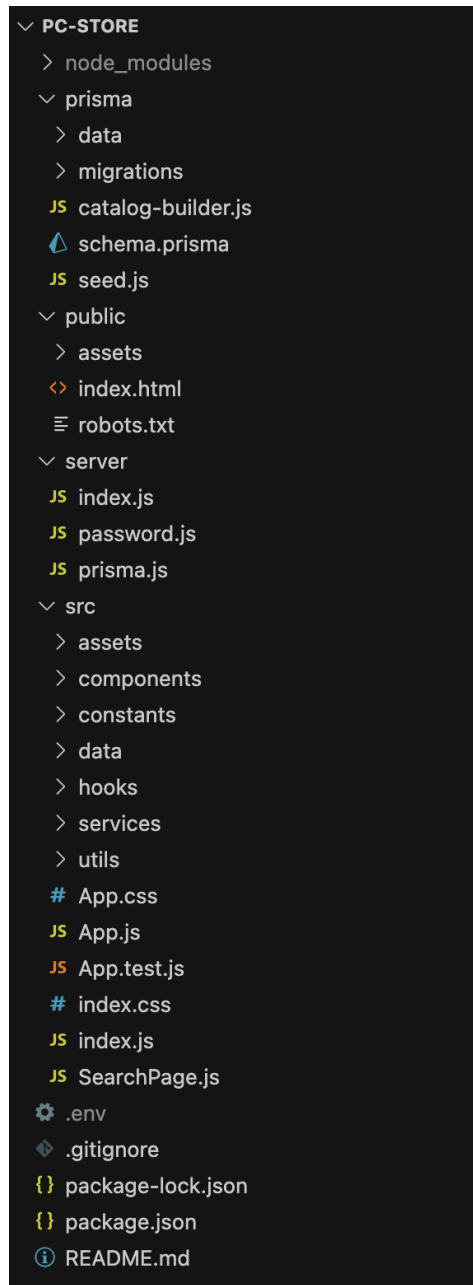


Рисунок 3.1 – Структура програмного проєкту у середовищі Visual Studio Code

Основні елементи клієнтського інтерфейсу розміщені в каталозі `components`. У ньому знаходяться компоненти для відображення товарів, каталогу, кошика, особистого кабінету, адміністративної панелі та конфігуратора ПК. Наприклад, компонент `ProductCard` використовується для короткого відображення товару в каталозі, `ProductModal` – для перегляду детальної інформації про товар, `CartView` – для роботи з кошиком,

AccountView – для особистого кабінету користувача, а ConfiguratorView – для покрокового підбору компонентів ПК.

Окремо в клієнтській частині реалізовано компоненти загальної структури сторінки. Верхня частина сайту містить логотип, навігаційні елементи та доступ до основних розділів вебзастосунку. Вона дозволяє користувачу швидко перейти до каталогу, конфігуратора, кошика або особистого кабінету.

На рисунку 3.2 наведено приклад відображення верхньої частини вебсайту, яка реалізована за допомогою компонента MainHeader.js.



Рисунок 3.2 – Верхня частина вебсайту з елементами навігації

Нижня частина сторінки використовується для відображення допоміжної інформації про вебсайт. Вона завершує структуру сторінки та робить інтерфейс більш цілісним. Приклад нижньої частини сторінки інтернет-магазину наведено на рисунку 3.3.

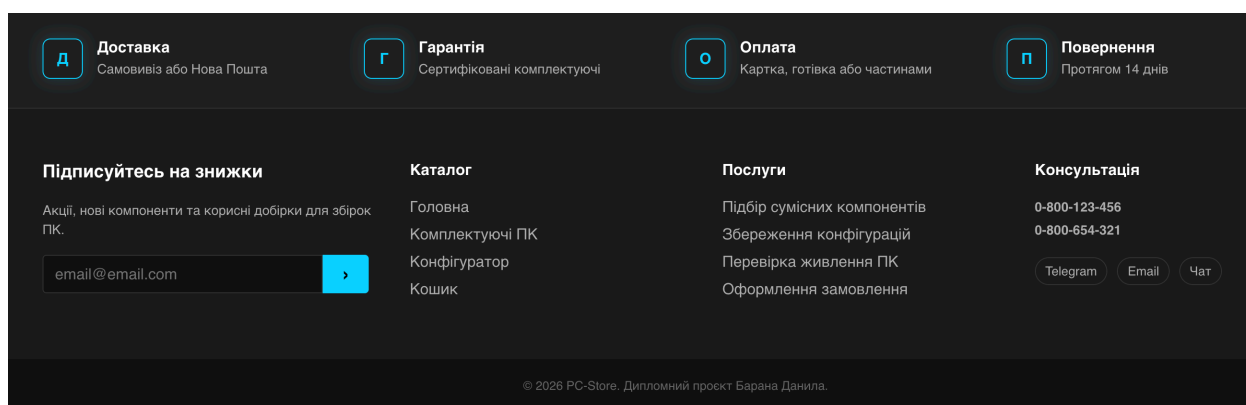


Рисунок 3.3 – Нижня частина вебсайту

Для взаємодії клієнтської частини із сервером використовується файл `api.js`, який розміщено в каталозі `services`. У ньому зосереджено функції для надсилання HTTP-запитів до REST API. Завдяки цьому компоненти

інтерфейсу не формують запити самостійно, а використовують готові функції для отримання товарів, авторизації користувача, оновлення кошика, створення замовлення та збереження конфігурацій.

Каталог `utils` містить допоміжну логіку, яка використовується в різних частинах клієнтського застосунку. Найважливішим для теми роботи є модуль `configurator.js`, у якому реалізовано функції перевірки сумісності комплектуючих, фільтрації доступних товарів, розрахунку рекомендованої потужності блоку живлення та формування підказок для користувача. Саме цей модуль забезпечує роботу інтелектуального помічника на клієнтському рівні.

Серверна частина проєкту розміщена в каталозі `server`. Основним файлом є `index.js`, у якому налаштовано Express-сервер, маршрути REST API та оброблення запитів від клієнтської частини. Допоміжні файли серверної частини відповідають за підключення Prisma Client і роботу з паролями користувачів. Через серверну частину виконуються операції з товарами, користувачами, кошиком, замовленнями, збереженими конфігураціями та адміністративними функціями.

Файли, пов'язані з базою даних, розміщені в каталозі `prisma`. У файлі `schema.prisma` описано моделі бази даних, які відповідають основним сутностям вебсайту: категоріям, товарам, користувачам, кошику, замовленням, збереженим конфігураціям і журналу активності. Також у цьому каталозі містяться допоміжні файли для наповнення бази даних початковими товарами та категоріями.

Статичні ресурси, зображення та стилі використовуються для оформлення інтерфейсу вебзастосунку. Вони забезпечують візуальне відображення категорій, товарів, елементів навігації та інших частин сайту. Завдяки цьому інтерфейс має завершений вигляд і відповідає тематиці інтернет-магазину комп'ютерної техніки.

Отже, структура програмного проєкту відповідає клієнт-серверній архітектурі та дозволяє логічно розділити основні частини вебзастосунку.

Клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, серверна частина за оброблення запитів і роботу з даними, а каталог prism за опис структури бази даних. Така організація спрощує підтримку проєкту та дозволяє надалі розширювати функціональність вебсайту.

3.3 Реалізація клієнтської частини вебзастосунку

Клієнтська частина вебзастосунку реалізує основну взаємодію користувача із системою. Через інтерфейс користувач може переглядати каталог комп'ютерних комплектуючих, відкривати детальну інформацію про товари, додавати позиції до кошика, оформлювати замовлення, реєструватися, авторизуватися, працювати з особистим кабінетом та використовувати конфігуратор ПК.

Основна логіка відображення розділів сайту зосереджена у React-компонентах. Інтерфейс побудовано так, щоб користувач міг переходити між основними частинами вебсайту без повного перезавантаження сторінки. Це важливо для інтернет-магазину з конфігуратором, оскільки під час вибору комплектуючих інтерфейс має швидко реагувати на дії користувача.

Головна сторінка вебсайту виконує роль початкової точки взаємодії користувача із системою (рис. 3.4). На ній розміщено загальну інформацію про магазин, візуальний блок із назвою вебсайту, навігацію до основних розділів та елементи, які дозволяють швидко перейти до каталогу або конфігуратора ПК. Такий підхід дозволяє користувачу одразу зрозуміти призначення вебсайту та вибрати потрібний сценарій роботи: перегляд окремих товарів або створення повної конфігурації комп'ютера.

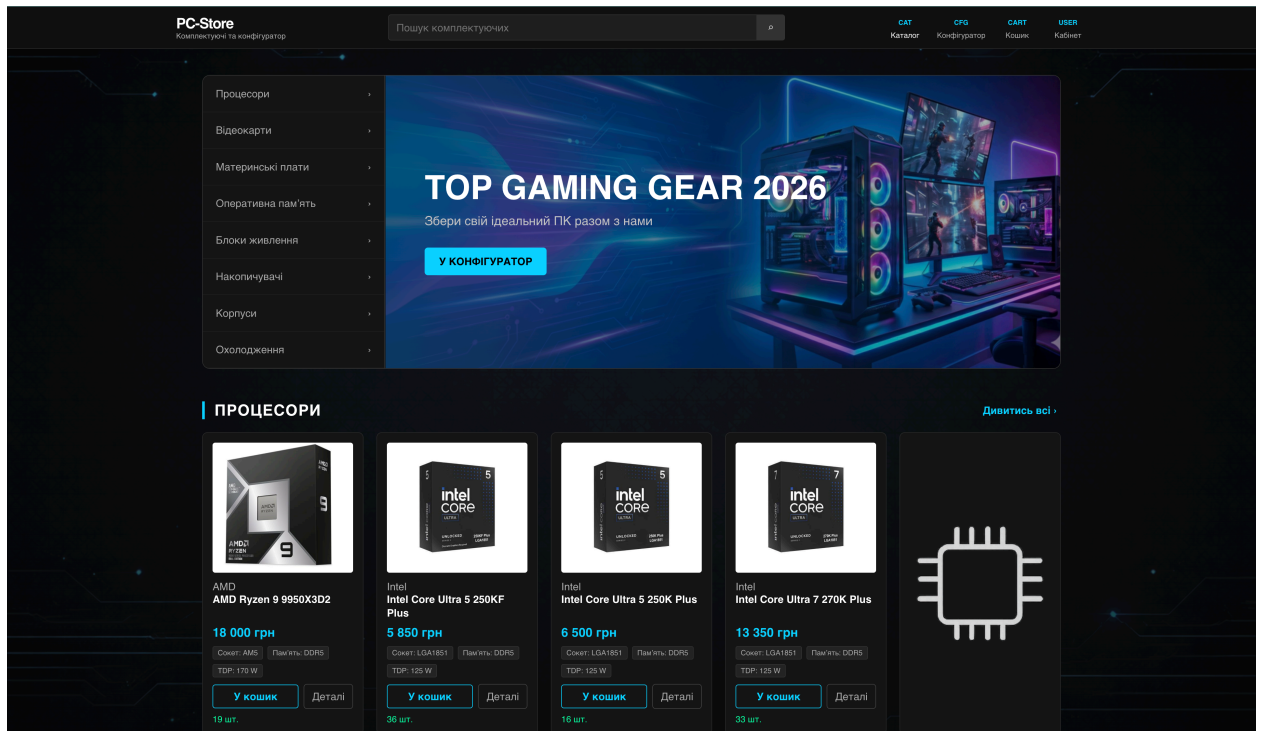


Рисунок 3.4 – Головна сторінка вебсайту

Верхня частина сайту реалізована у вигляді окремого компонента MainHeader.js. Вона містить логотип вебсайту, навігаційні елементи, кнопку переходу до кошика та доступ до особистого кабінету. Такий компонент використовується на різних сторінках застосунку, тому користувач постійно має доступ до основних розділів. Це спрощує навігацію та робить структуру сайту зрозумілішою.

Основним розділом вебсайту є каталог товарів. Він реалізований таким чином, щоб користувач міг спочатку переглянути категорії комплектуючих на головній сторінці, а потім перейти до товарів конкретної категорії натиснувши на відповідну іконку або напис. У каталозі передбачено окремі категорії для процесорів, материнських плат, оперативної пам'яті, відеокарт, накопичувачів, систем охолодження, корпусів і блоків живлення. Також є окрема сторінка каталогу саме категорій, де при натисканні можна перейти до пошуку потрібної категорії (рис. 3.5). Такий поділ відповідає логіці підбору комплектуючих для ПК і полегшує пошук потрібного товару.

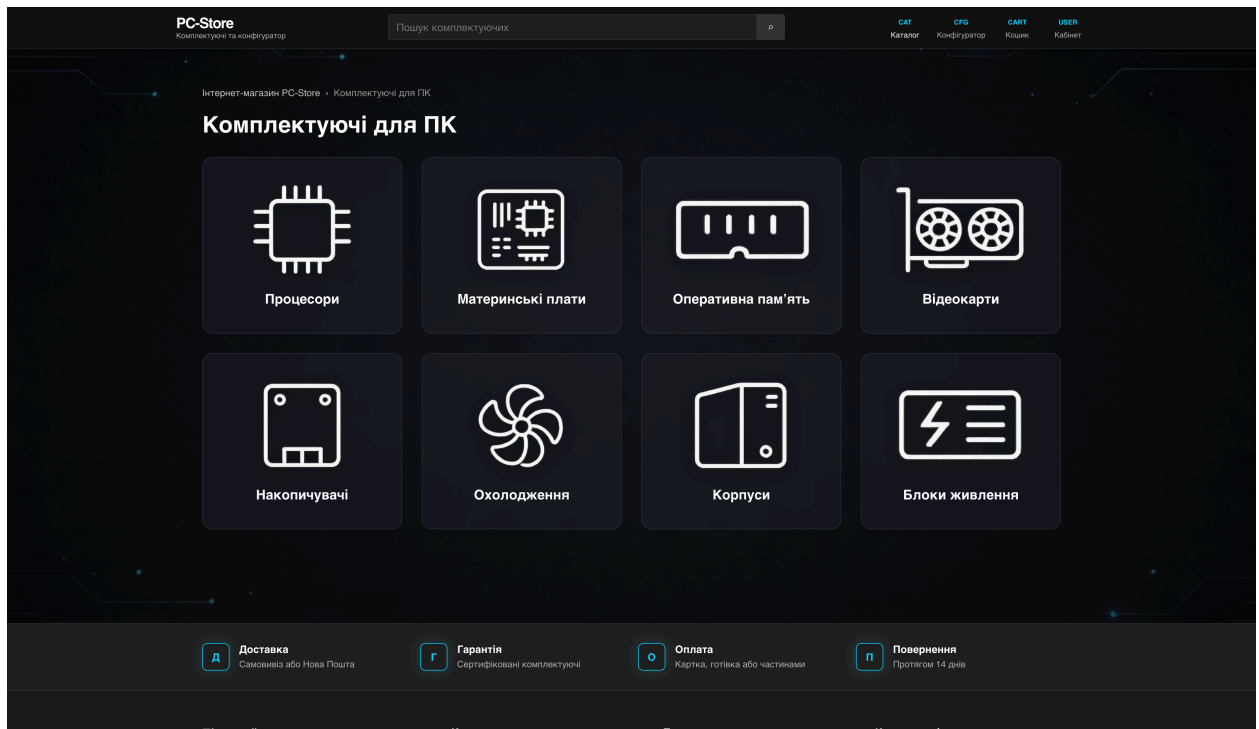


Рисунок 3.5 – Відображення каталогу категорій

Після вибору категорії користувачу відображається список товарів, які до неї належать. Кожен товар подається у вигляді картки. Картка містить зображення товару, назву, бренд, ціну, інформацію про наявність на складі та кнопки для перегляду детальнішої інформації або додавання товару до кошика. Використання карток дозволяє компактно подати велику кількість товарів і водночас залишити інтерфейс зручним для перегляду. На рисунку 3.6 зображено сторінку каталогу з картками для вибраної категорії «Процесори».

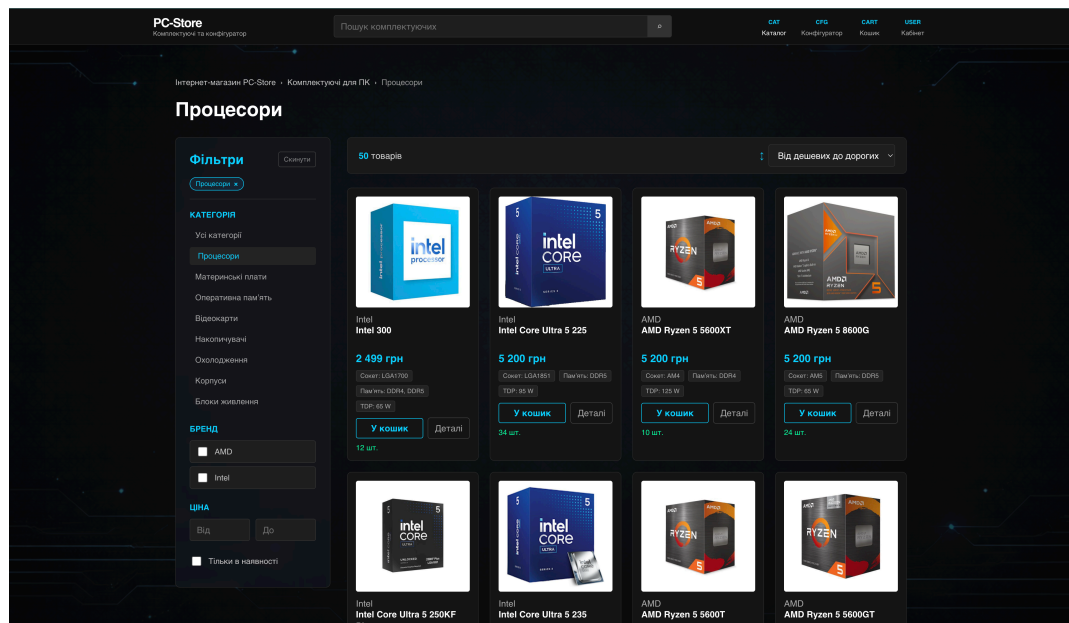


Рисунок 3.6 – Сторінка каталогу товарів для категорії «Процесори»

Для відображення окремого товару використовується компонент ProductCard.js. Він застосовується як у звичайному каталозі, так і в конфігураторі ПК. Це дозволяє повторно використовувати один і той самий компонент у різних частинах вебзастосунку. Завдяки цьому картки товарів мають однаковий вигляд, а користувач не стикається з різними варіантами відображення однієї й тієї самої інформації.

Окремо реалізовано модальне вікно перегляду товару (рис. 3.7). Воно відкривається після вибору відповідної кнопки на картці товару. У модальному вікні користувач може переглянути детальнішу інформацію про комплектуючий: назву, бренд, опис, ціну, наявність і технічні характеристики. Для комп'ютерної техніки це важливо, оскільки саме характеристики визначають можливість використання компонента в конкретній конфігурації.

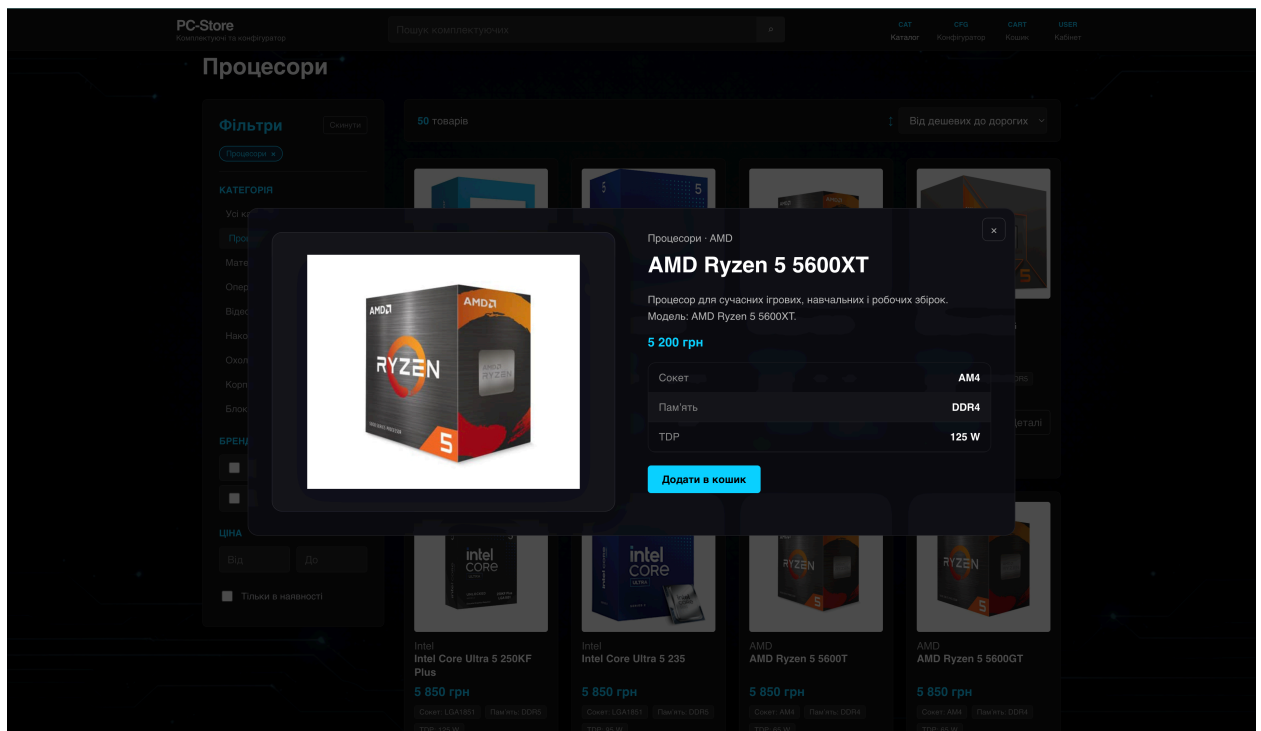


Рисунок 3.7 – Модальне вікно з детальною інформацією про товар

Для зручності користувача в каталозі реалізовано пошук і фільтрацію товарів. Пошук дозволяє швидко знайти потрібну модель за назвою або брендом. Фільтрація допомагає звужити список товарів за окремими параметрами, наприклад категорією, виробником, ціною або наявністю. Це особливо важливо для інтернет-магазину комп'ютерних комплектуючих, оскільки каталог може містити багато позицій, а ручний перегляд усіх товарів займає багато часу.

Під час реалізації пошуку було враховано, що користувач може вводити лише частину назви товару або шукати компонент за брендом. Тому логіка пошуку виконує перевірку введеного запиту та порівнює його з даними товарів. Після цього список товарів на сторінці оновлюється відповідно до результатів пошуку. Це дозволяє швидко знаходити потрібні комплектуючі без переходу на окрему сторінку. Приклад роботи пошуку та фільтрів наведено на рисунку 3.8.

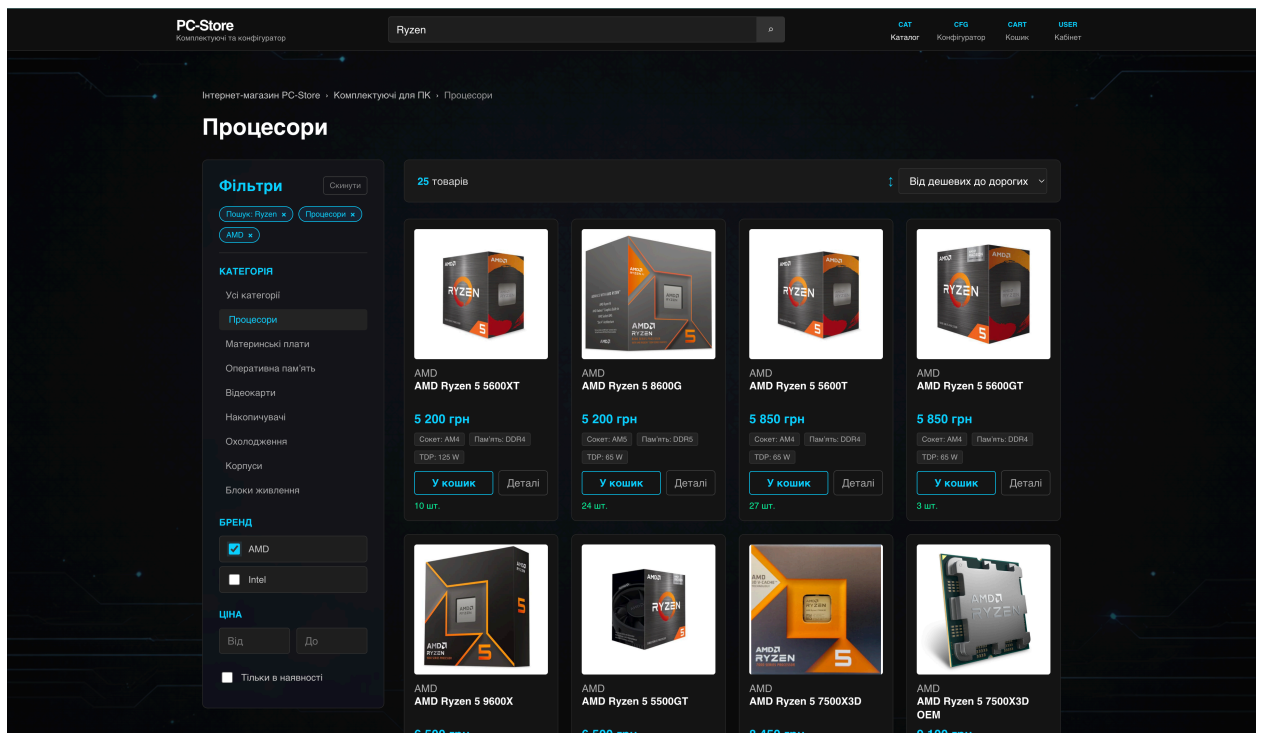


Рисунок 3.8 – Приклад роботи пошуку та фільтрації товарів

Для роботи з кошиком реалізовано окремий компонент CartView.js. Кошик дозволяє користувачу переглядати всі товари, які були додані під час роботи з каталогом або конфігуратором (рис. 3.9). У кошику відображається назва товару, кількість, ціна, загальна сума та елементи керування. Користувач може змінювати кількість товарів, видаляти позиції або перейти до оформлення замовлення.

Особливістю реалізації кошика є підтримка не лише окремих товарів, а й готових конфігурацій ПК. Після завершення роботи з інтелектуальним помічником користувач може додати всю сформовану збірку до кошика як окрему позицію. При цьому в кошику відображається не тільки назва конфігурації, а й список комплектуючих, які входять до її складу. Це дозволяє користувачу перевірити готову збірку перед оформленням замовлення.

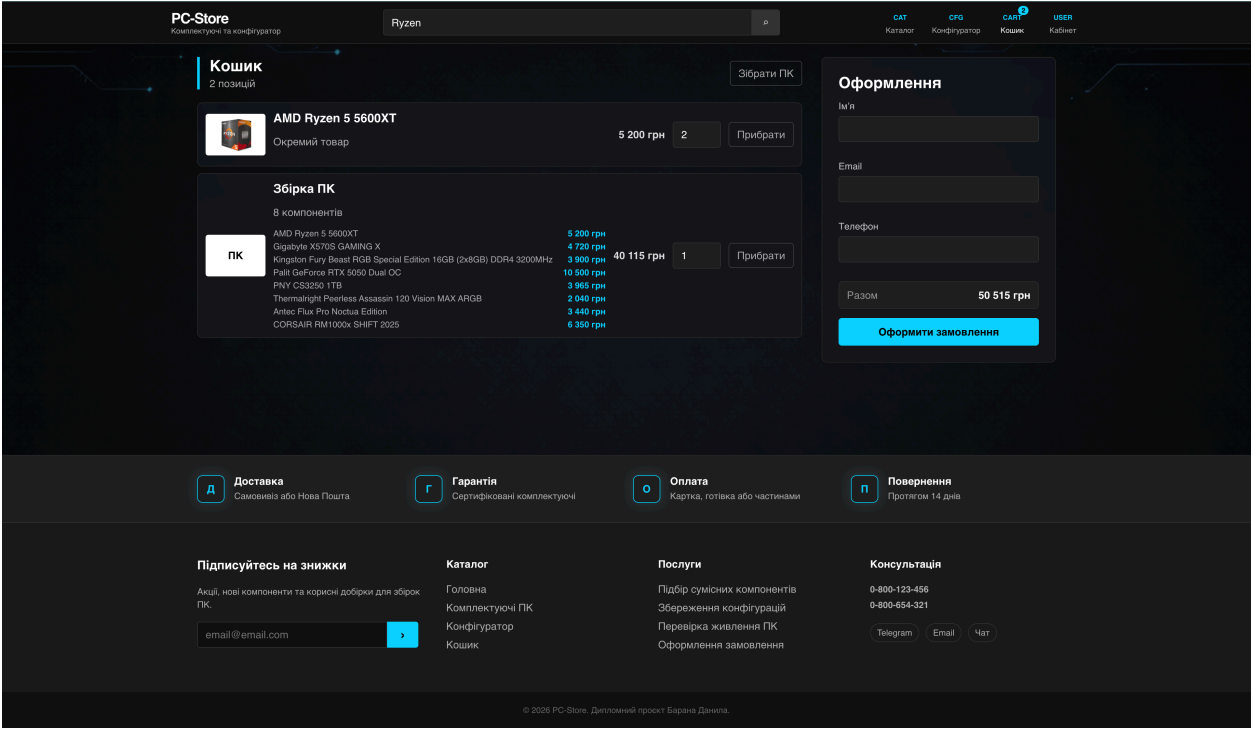


Рисунок 3.9 – Кошик користувача з доданим товаром та конфігурацією ПК

Після перегляду кошика користувач може перейти до оформлення замовлення. На цьому етапі система відображає підсумкову інформацію про вибрані товари або конфігурацію, загальну суму та форму для введення контактних даних покупця. Після підтвердження замовлення дані передаються на серверну частину, де зберігаються в базі даних. Після успішного оформлення замовлення користувач отримає повідомлення у верхньому куті екрана, як це показано на рисунку 3.10.

Рисунок 3.10 – Успішне оформлення замовлення

Окремою частиною клієнтської реалізації є робота з обліковими записами користувачів. У вебзастосунку передбачено можливість реєстрації нового користувача та авторизації вже створеного облікового запису. Це потрібно для того, щоб користувач міг не лише переглядати каталог і додавати товари до кошика, а й зберігати власні конфігурації ПК, переглядати історію замовлень та повертатися до створених раніше збірок.

Під час реєстрації користувач вводить ім'я, електронну пошту та пароль. Після надсилання форми дані передаються на серверну частину, де виконується перевірка коректності введеної інформації та створення нового облікового запису. Якщо користувач із такою електронною поштою вже існує, система повинна повідомити про помилку. У разі успішної реєстрації користувач може увійти до системи та отримати доступ до особистого кабінету. Форма реєстрації показана на рисунку 3.11.

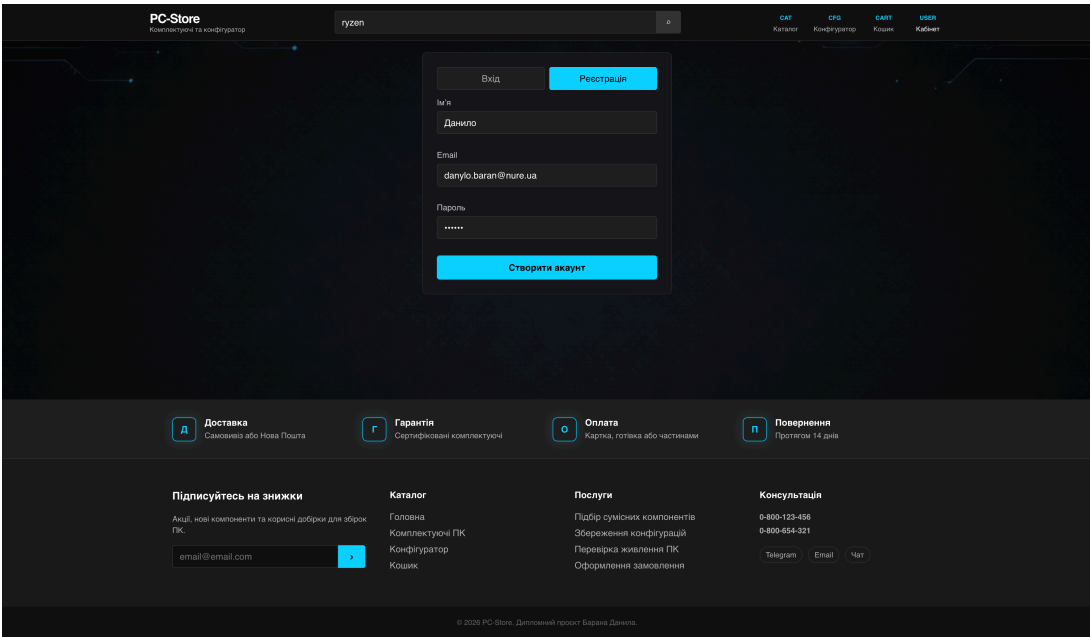


Рисунок 3.11 – Форма реєстрації користувача

Авторизація користувача виконується через окрему форму входу (рис. 3.12). Користувач вводить електронну пошту та пароль, після чого клієнтська частина надсилає ці дані до серверного API. Сервер перевіряє наявність користувача в базі даних і правильність введеного пароля. Якщо дані введено правильно, клієнтська частина зберігає інформацію про поточну сесію користувача та відкриває доступ до функцій, які потребують авторизації.

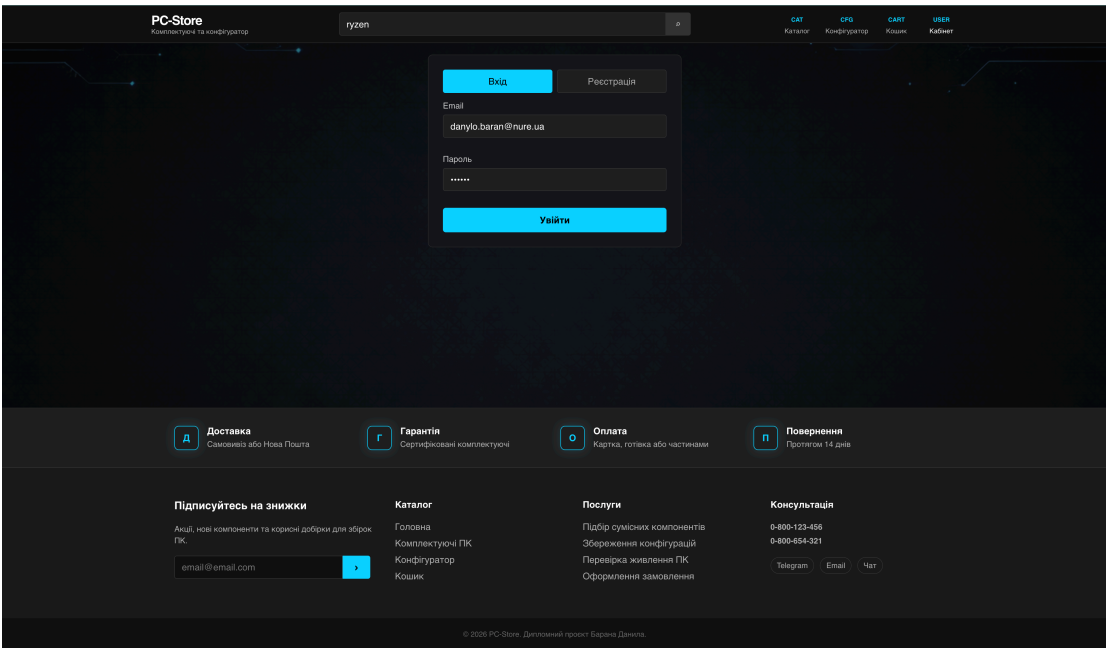


Рисунок 3.12 – Форма авторизації користувача

Після входу до системи користувачу стає доступний особистий кабінет (рис. 3.13). У межах особистого кабінету реалізовано перегляд раніше оформлених замовлень. Користувач може бачити склад замовлення, загальну суму, дату створення та поточний статус. Це робить роботу із сайтом зручнішою, оскільки покупець може контролювати власну історію покупок і перевіряти, які товари або конфігурації вже були оформлені.

Окремо в особистому кабінеті відображаються збережені конфігурації ПК. Такий функціонал є важливим саме для вебсайту з інтелектуальним помічником, оскільки процес підбору комплектуючих може займати певний час. Користувач може сформувати збірку, зберегти її, а потім повернутися до неї пізніше для перегляду або додавання до кошика.

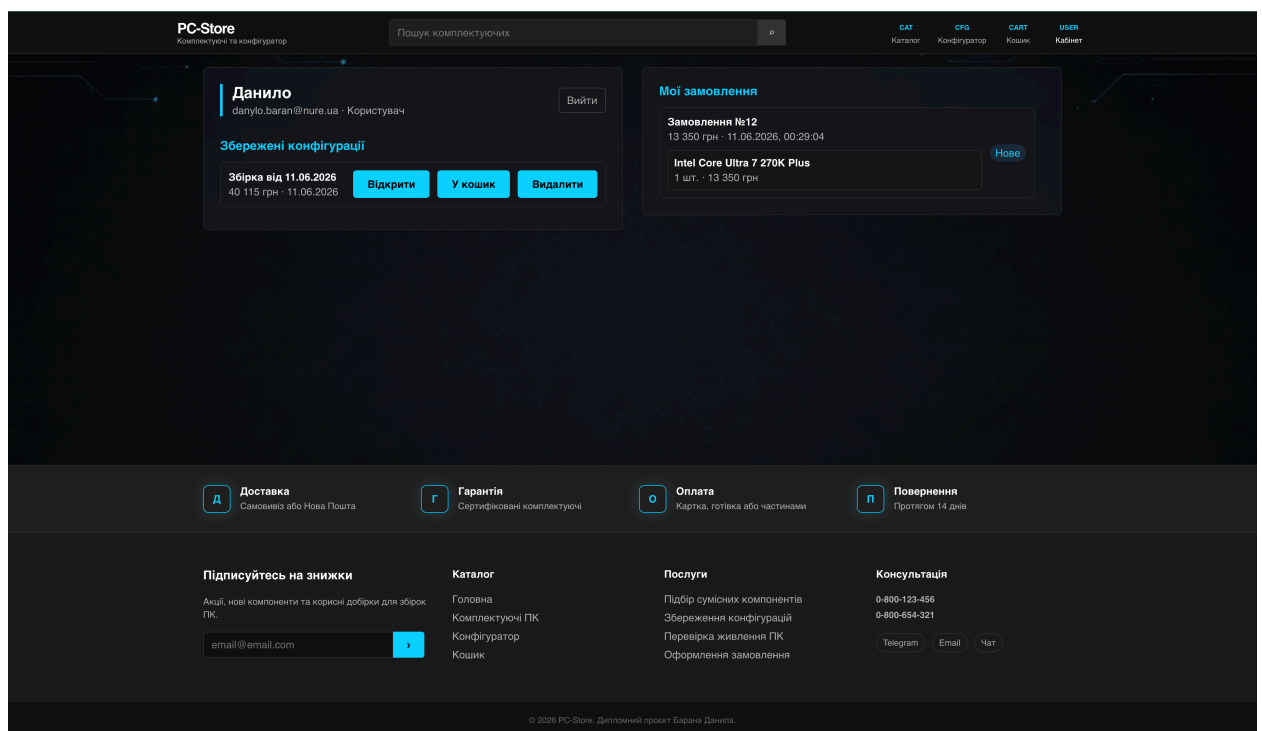


Рисунок 3.13 – Особистий кабінет користувача

У клієнтській частині також реалізовано відображення адміністративних функцій. Якщо користувач має роль адміністратора, йому доступні додаткові можливості для керування системою. Адміністратор може переглядати замовлення, змінювати їх статус, оновлювати залишки товарів на складі та

переглядати журнал активності. Для відкриття складу передбачена окрема кнопка, яка відкриває або закриває блок зі складом залежно від потреби адміністратора. Таке розмежування інтерфейсу дозволяє використовувати один вебзастосунок як для покупців, так і для адміністратора магазину. Приклад особистого кабінету адміністратора та роботи зі складом продемонстровано на рисунках 3.14 та 3.15.

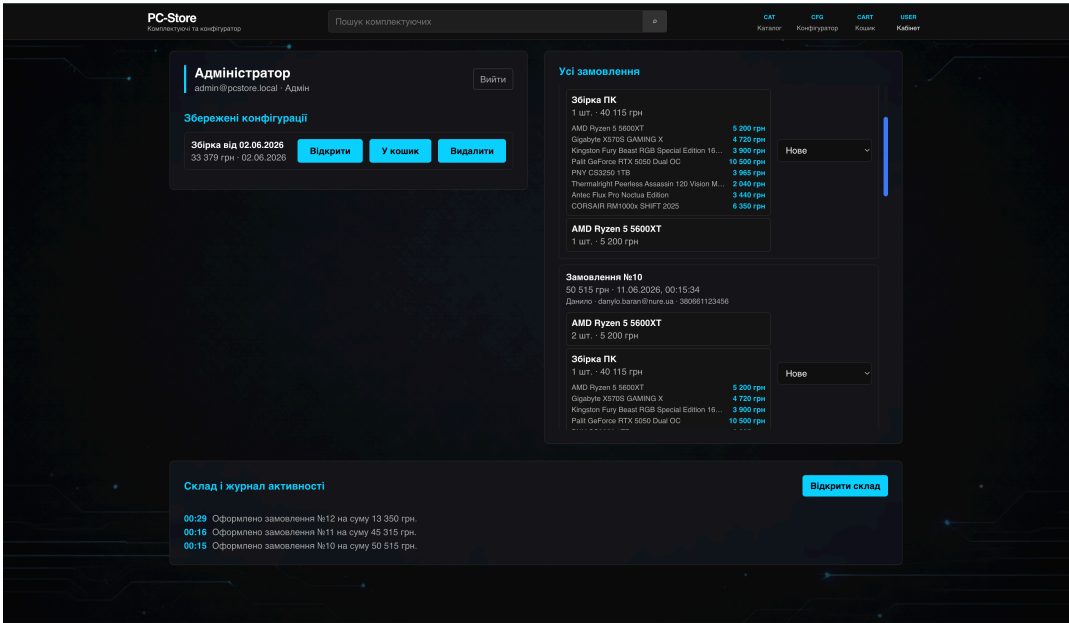


Рисунок 3.14 – Адміністративна панель вебсайту

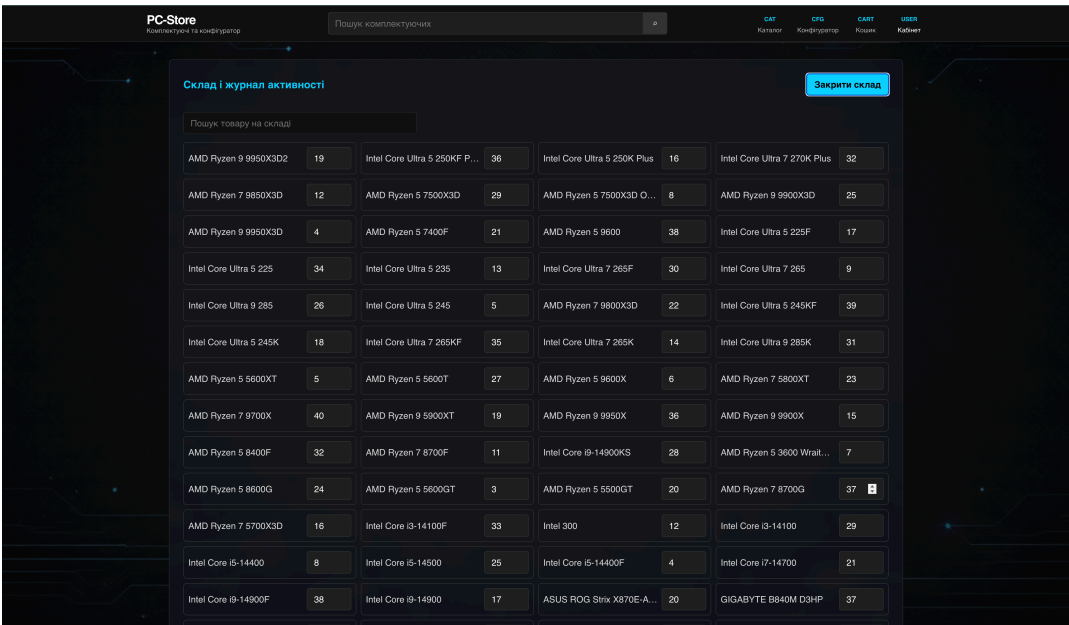


Рисунок 3.15 – Приклад роботи зі складом

Під час реалізації клієнтської частини було враховано необхідність збереження частини даних між оновленнями сторінки. Для цього використовується локальне сховище браузера. У ньому можуть зберігатися дані сесії користувача, стан кошика, вибраний розділ сайту, фільтри та поточна конфігурація ПК. Це дозволяє уникнути втрати даних після випадкового оновлення сторінки та робить роботу з вебсайтом зручнішою.

Для роботи з локальним сховищем у клієнтській частині використано окремий користувацький React-хук `useLocalStorage`, який наведено у лістингу 3.1. Він дозволяє отримувати початкове значення зі сховища браузера, оновлювати його під час зміни стану та використовувати ці дані в різних компонентах вебзастосунку.

Лістинг 3.1 Фрагмент реалізації збереження даних у локальному сховищі браузера:

```
export function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    try {
      const stored = window.localStorage.getItem(key);
      return stored ? JSON.parse(stored) : initialValue;
    } catch {
      return initialValue;
    }
  });
  const updateValue = useCallback((nextValue) => {
    setValue((current) => {
      const resolved = typeof nextValue === 'function' ? nextValue(current) :
nextValue;
      window.localStorage.setItem(key, JSON.stringify(resolved));
      return resolved;
    });
  });
```

```

    }, [key]);
    return [value, updateValue];
}

```

Взаємодія клієнтської частини із сервером виконується через файл `api.js`, у якому зосереджено функції для надсилання HTTP-запитів. Завдяки цьому компоненти інтерфейсу не містять дубльованого коду для звернення до API, а лише викликають готові функції. Такий підхід спрощує підтримку клієнтської частини та дозволяє швидше змінювати логіку взаємодії із сервером у разі потреби.

У файлі `api.js` реалізовано спільну функцію `request`, яка формує HTTP-запит, додає заголовок `Content-Type`, перетворює тіло запиту у формат JSON, обробляє відповідь сервера та повертає результат клієнтському компоненту. У разі помилки функція створює об'єкт помилки з відповідним статусом, що дозволяє обробляти невдалі запити в інтерфейсі користувача. У лістингу 3.2 наведено структуру файлу `api.js`.

Лістинг 3.2 Фрагмент реалізації взаємодії клієнтської частини із серверним API:

```

const API_BASE_URL = process.env.REACT_APP_API_URL ||
'http://localhost:4000/api';

async function request(path, options = {}) {
  const response = await fetch(`${API_BASE_URL}${path}`, {
    method: options.method || 'GET',
    headers: {
      'Content-Type': 'application/json',
      ...(options.headers || {}),
    },
    body: options.body ? JSON.stringify(options.body) : undefined,
  });
}

```

```

const text = await response.text();
const payload = text ? JSON.parse(text) : null;
if (!response.ok) {
  const error = new Error(payload?.error || 'API request failed');
  error.status = response.status;
  throw error;
}
return payload;
}

```

Після створення спільної функції `request` у цьому ж файлі описано окремі методи для роботи з основними сутностями вебзастосунку. Наприклад, `getProducts` використовується для отримання списку товарів, `login` та `register` – для авторизації й реєстрації користувачів, `updateUserCart` – для оновлення кошика, `createOrder` – для створення замовлення, `createSavedConfiguration` – для збереження конфігурації ПК.

Винесення запитів до окремого сервісного файлу дозволяє зробити компоненти клієнтської частини простішими. Наприклад, компонент кошика не формує HTTP-запит самостійно, а викликає готовий метод `updateUserCart`. Компонент авторизації використовує метод `login`, а під час оформлення замовлення викликається метод `createOrder`. Завдяки цьому логіка роботи з API відокремлена від логіки відображення інтерфейсу.

Реалізована клієнтська частина забезпечує основні сценарії роботи користувача з вебсайтом. Користувач може переглядати каталог, шукати потрібні комплектуючі, відкривати детальну інформацію про товар, додавати позиції до кошика, оформлювати замовлення, працювати з особистим кабінетом і створювати конфігурації ПК. Завдяки компонентній структурі React інтерфейс залишається гнучким і може бути розширений новими сторінками або функціями.

Отже, клієнтська частина вебзастосунку реалізує основну взаємодію користувача із системою. Вона поєднує стандартні функції інтернет-магазину з додатковими можливостями, необхідними для підбору комп'ютерних комплектуючих. Використання React дозволило створити динамічний інтерфейс, у якому користувач може швидко переходити між розділами, працювати з каталогом, кошиком, особистим кабінетом і конфігуратором без повного перезавантаження сторінки.

3.4 Реалізація серверної частини та бази даних

Серверна частина вебзастосунку реалізована з використанням Node.js та фреймворку Express. Вона забезпечує оброблення запитів від клієнтської частини, взаємодію з базою даних, роботу з користувачами, товарами, кошиком, замовленнями, збереженими конфігураціями та адміністративними функціями. Дані між клієнтом і сервером передаються у форматі JSON через REST API.

Backend-частина розміщена в каталозі `server`. Основним файлом є `index.js`, у якому налаштовано Express-сервер, `middleware` для оброблення JSON-запитів, CORS та маршрути API. Окремо виділено файл `prisma.js` для підключення Prisma Client і файл `password.js` для допоміжної роботи з паролями користувачів.

Для доступу до бази даних використовується Prisma ORM. Вона дозволяє працювати з таблицями PostgreSQL через об'єктні моделі, описані у файлі `schema.prisma`. Підключення Prisma Client винесено в окремий файл `prisma.js`, що дозволяє використовувати один екземпляр клієнта в різних маршрутах серверної частини. У лістингу 3.3 наведено зміст файлу підключення Prisma Client.

Лістинг 3.3 Підключення Prisma Client у серверній частині:

```
const { PrismaClient } = require('@prisma/client');
const prisma = new PrismaClient();
module.exports = prisma;
```

База даних описана у файлі `schema.prisma`. У межах проєкту використовуються моделі `Category`, `Product`, `User`, `Cart`, `Order`, `SavedConfiguration` та `ActivityLog`. Найважливішою для роботи каталогу та інтелектуального помічника є модель `Product`, оскільки саме вона зберігає інформацію про комп'ютерні комплектуючі. Фрагмент моделі `Product` наведено в лістингу 3.4.

Лістинг 3.4 Фрагмент моделі `Product` у файлі `schema.prisma`:

```
model Product {
  id          String @id
  categoryId  String
  category    Category @relation(fields: [categoryId], references: [id],
onDelete: Restrict)
  name        String
  brand        String
  price        Int
  stock        Int @default(0)
  image        String
  description  String
  specs        Json
  sortOrder    Int @default(0)
  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
  @@index([categoryId])
  @@index([brand])
}
```


Використання JSON-поля `specs` є важливим для предметної області комп'ютерних комплектуючих. Різні категорії товарів мають різні набори характеристик: для процесора важливими є сокет, кількість ядер, частота та TDP, для материнської плати – сокет, тип оперативної пам'яті та формфактор, для відеокарти – обсяг пам'яті, довжина та енергоспоживання. Зберігання характеристик у JSON-форматі дозволяє не створювати окрему таблицю для кожного типу комплектуючих і спрощує додавання нових параметрів товарів.

Інші моделі бази даних забезпечують роботу з користувачами, кошиком, замовленнями, збереженими конфігураціями та журналом активності. Модель `User` зберігає дані користувачів, хешований пароль і роль, що використовується для розмежування доступу між покупцем та адміністратором. Модель `Cart` зберігає вміст кошика у JSON-форматі, тому в ньому можуть міститися як окремі товари, так і готові конфігурації ПК. Модель `Order` використовується для зберігання оформлених замовлень, їх складу, загальної суми, статусу та дати створення. Модель `SavedConfiguration` відповідає за збереження сформованих збірок ПК, а `ActivityLog` – за фіксацію основних подій системи.

Серверна частина надає клієнтському застосунку маршрути REST API для роботи з основними функціями вебсайту. Через API клієнтська частина отримує список категорій і товарів, надсилає дані для реєстрації та авторизації, оновлює кошик, створює замовлення, зберігає конфігурації ПК та виконує адміністративні дії. Дані передаються у форматі JSON, після чого сервер обробляє запит, звертається до бази даних через Prisma ORM і повертає результат клієнтській частині.

Під час реєстрації сервер перевіряє наявність користувача з вказаною електронною поштою, після чого зберігає пароль у хешованому вигляді. Під час авторизації виконується перевірка електронної пошти та пароля, а у відповідь клієнтська частина отримує основні дані користувача: ідентифікатор, ім'я, електронну пошту та роль. Це дозволяє відображати різні можливості для звичайного користувача й адміністратора.

Окремі маршрути відповідають за роботу з кошиком, замовленнями та збереженими конфігураціями. Коли користувач додає товар або готову збірку до кошика, клієнтська частина надсилає оновлений список позицій на сервер. Під час оформлення покупки сервер створює запис замовлення, який потім доступний в особистому кабінеті користувача та адміністративній частині. Якщо користувач не хоче одразу оформлювати покупку, сформована конфігурація може бути збережена в особистому кабінеті для подальшого перегляду.

Адміністративні функції реалізовані через маршрути для оновлення залишків товарів, зміни статусів замовлень і перегляду журналу активності. Завдяки цьому адміністратор може керувати основними процесами магазину через інтерфейс вебзастосунку без прямої роботи з базою даних.

Важливою особливістю реалізації є те, що серверна частина не виконує основну перевірку сумісності комплектуючих. Вона надає клієнтській частині каталог товарів із технічними характеристиками, а логіка перевірки сумісності виконується в модулі конфігуратора на frontend-рівні. Це дозволяє швидко оновлювати доступні варіанти після кожного вибору користувача без додаткового звернення до сервера.

Отже, серверна частина вебзастосунку забезпечує оброблення основних запитів, роботу з базою даних, реєстрацію та авторизацію користувачів, збереження кошика, оформлення замовлень, збереження конфігурацій і виконання адміністративних операцій. Використання Express, Prisma ORM та PostgreSQL дозволило реалізувати backend-частину, яка взаємодіє з клієнтським інтерфейсом через REST API та забезпечує зберігання основних даних вебсайту.

3.5 Реалізація інтелектуального помічника-конфігуратора

Однією з основних функціональних частин розробленого вебзастосунку є інтелектуальний помічник-конфігуратор. Його призначення полягає у тому, щоб допомогти користувачу послідовно підібрати комплектуючі для персонального комп'ютера та зменшити ймовірність помилок під час вибору. На відміну від звичайного каталогу товарів, конфігуратор аналізує вже вибрані компоненти та показує користувачу варіанти з урахуванням сумісності, бюджету і призначення майбутнього ПК.

Логіка роботи конфігуратора реалізована на клієнтському рівні. Серверна частина надає список товарів і їх технічні характеристики, а клієнтська частина виконує перевірку сумісності та оновлює список доступних комплектуючих після кожного вибору користувача. Такий підхід дозволяє швидко реагувати на дії користувача без додаткового звернення до сервера після кожної зміни конфігурації.

Робота з конфігуратором починається з вибору призначення майбутнього комп'ютера та бюджету. Призначення ПК використовується для оцінки загальної збалансованості збірки, а бюджет дозволяє контролювати орієнтовну вартість обраних компонентів. Після цього користувач переходить до покрокового вибору комплектуючих. Сторінка інтелектуального помічника продемонстрована на рисунку 3.16.

Підбір компонентів виконується послідовно. Користувач обирає процесор, материнську плату, оперативну пам'ять, відеокарту, накопичувач, систему охолодження, корпус і блок живлення. Після вибору кожного компонента система оновлює поточну конфігурацію та перевіряє, які товари можуть бути використані на наступних етапах. Це дає змогу уникнути ситуації, коли користувач спочатку обирає несумісні компоненти, а помилку бачить лише в кінці. На рисунку 3.17 показано приклад покрокового вибору комплектуючих.

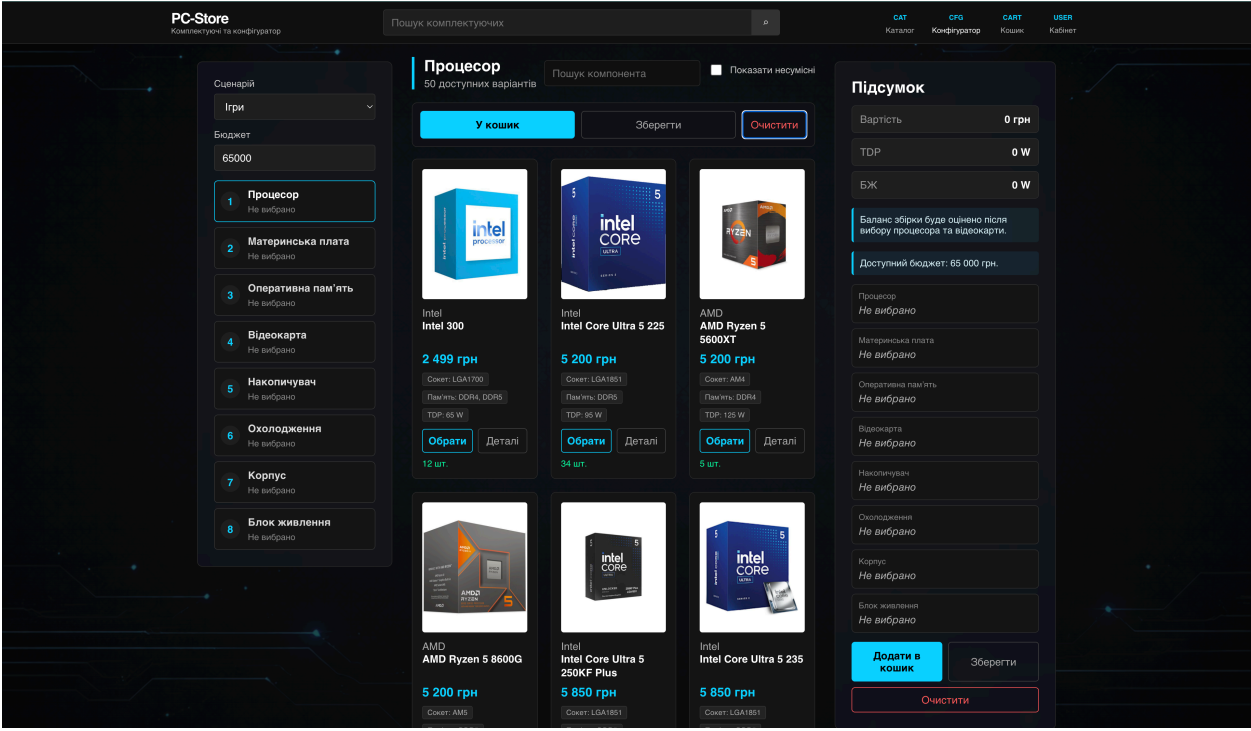


Рисунок 3.16 – Початкова сторінка інтелектуального помічника

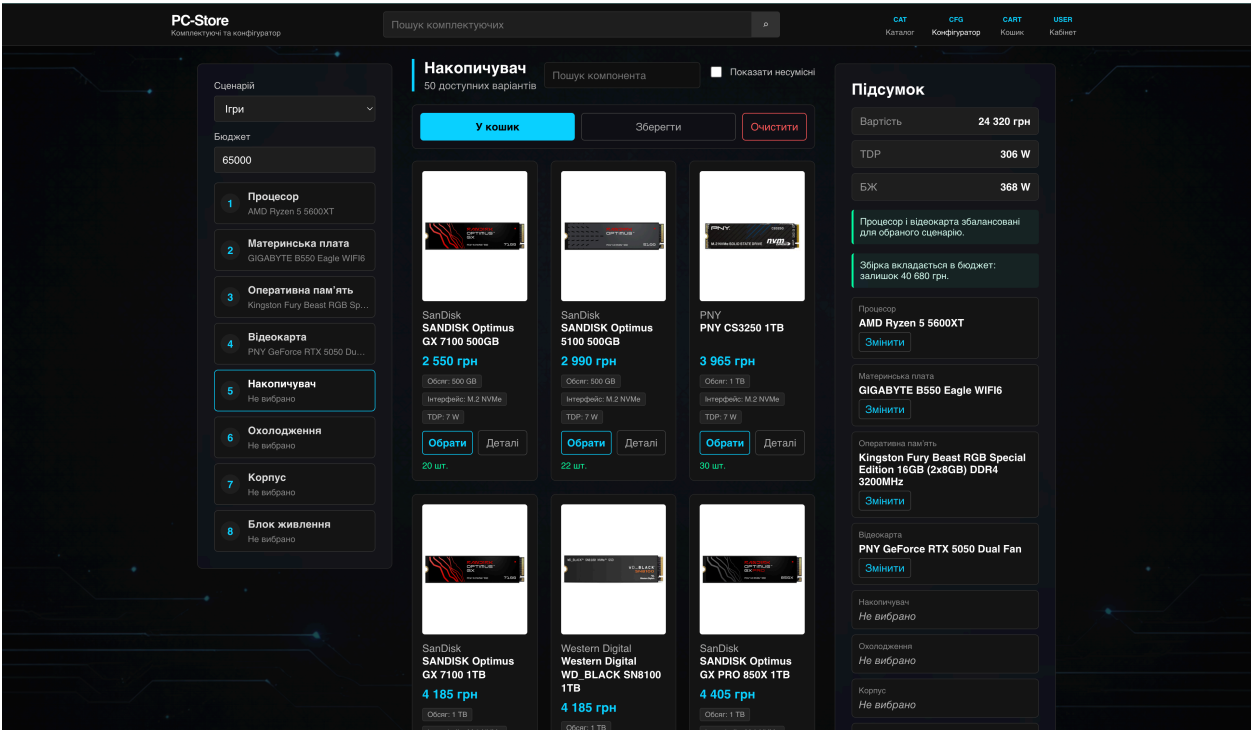


Рисунок 3.17 – Покроковий вибір комплектуючих у конфігураторі

Основою роботи інтелектуального помічника є набір правил сумісності. Для процесора та материнської плати перевіряється відповідність сокета. Для оперативної пам'яті враховується її тип, наприклад DDR4 або DDR5, а також

підтримка цього типу материнської плати. Для відеокарти перевіряється наявність відповідного інтерфейсу та її фізична сумісність із корпусом. Для системи охолодження враховуються підтримувані сокети, TDP процесора та висота кулера. Під час вибору блоку живлення система враховує приблизне енергоспоживання компонентів і додає запас потужності.

Якщо компонент не відповідає поточній конфігурації, він не повинен сприйматися користувачем як рекомендований варіант. У системі такі товари можуть бути приховані або позначені як несумісні з поясненням причини. Це робить роботу конфігуратора зрозумілішою, оскільки користувач бачить не лише факт несумісності, а й причину, через яку компонент не підходить. Приклад вигляду несумісних компонентів у конфігураторі зазначено на рисунку 3.18.

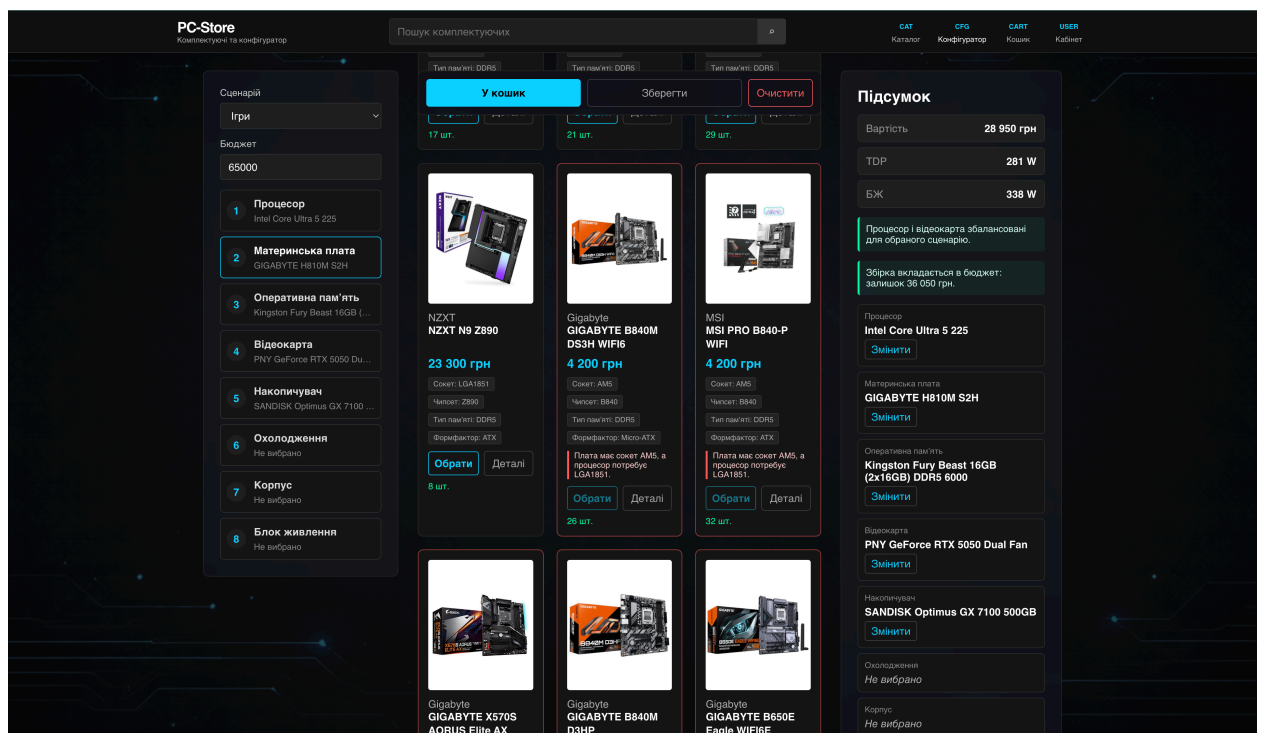


Рисунок 3.18 – Відображення несумісних комплектуючих у конфігураторі

Фрагмент перевірки сумісності комплектуючих наведено в лістингу 3.5. У цьому прикладі показано перевірку процесора відносно вже вибраної материнської плати та оперативної пам'яті. Якщо сокет процесора не

збігається із сокетом материнської плати, система формує повідомлення про помилку. Також перевіряється, чи підтримує процесор тип оперативної пам'яті, вибраний користувачем.

Лістинг 3.5 Фрагмент перевірки сумісності комплектуючих

```

if (category === 'cpu') {
    if (selected.motherboard && spec.socket !==
selected.motherboard.specs.socket) {
        pushError(`Сокет ${spec.socket} не підходить до плати
${selected.motherboard.specs.socket}.`);
    }
    if (selected.ram &&
!spec.memoryTypes?.includes(selected.ram.specs.memoryType)) {
        pushError(`Процесор не підтримує пам'ять
${selected.ram.specs.memoryType}.`);
    }
}
}

```

Наведений фрагмент демонструє принцип роботи правил сумісності в конфігураторі. Перша перевірка порівнює сокет процесора із сокетом уже вибраної материнської плати. Якщо ці значення не збігаються, компонент вважається несумісним із поточною конфігурацією. Друга перевірка стосується оперативної пам'яті: якщо процесор не підтримує тип пам'яті, який уже вибрав користувач, система також формує повідомлення про помилку.

Окрему роль у роботі конфігуратора відіграє розрахунок рекомендованої потужності блоку живлення. Для цього система враховує енергоспоживання вибраних компонентів і додає запас потужності. Такий підхід дозволяє не пропонувати користувачу блоки живлення, які можуть бути недостатніми для стабільної роботи системи. Крім потужності, також

перевіряється формфактор блоку живлення, щоб він відповідав обраному корпусу.

Після вибору комплектуючих система формує підсумкову конфігурацію. У ній користувач бачить список вибраних компонентів, загальну вартість, відповідність бюджету та можливі попередження. Якщо конфігурація має проблеми, наприклад недостатню потужність блоку живлення або несумісність окремих компонентів, система повідомляє про це користувача (рис. 3.19).

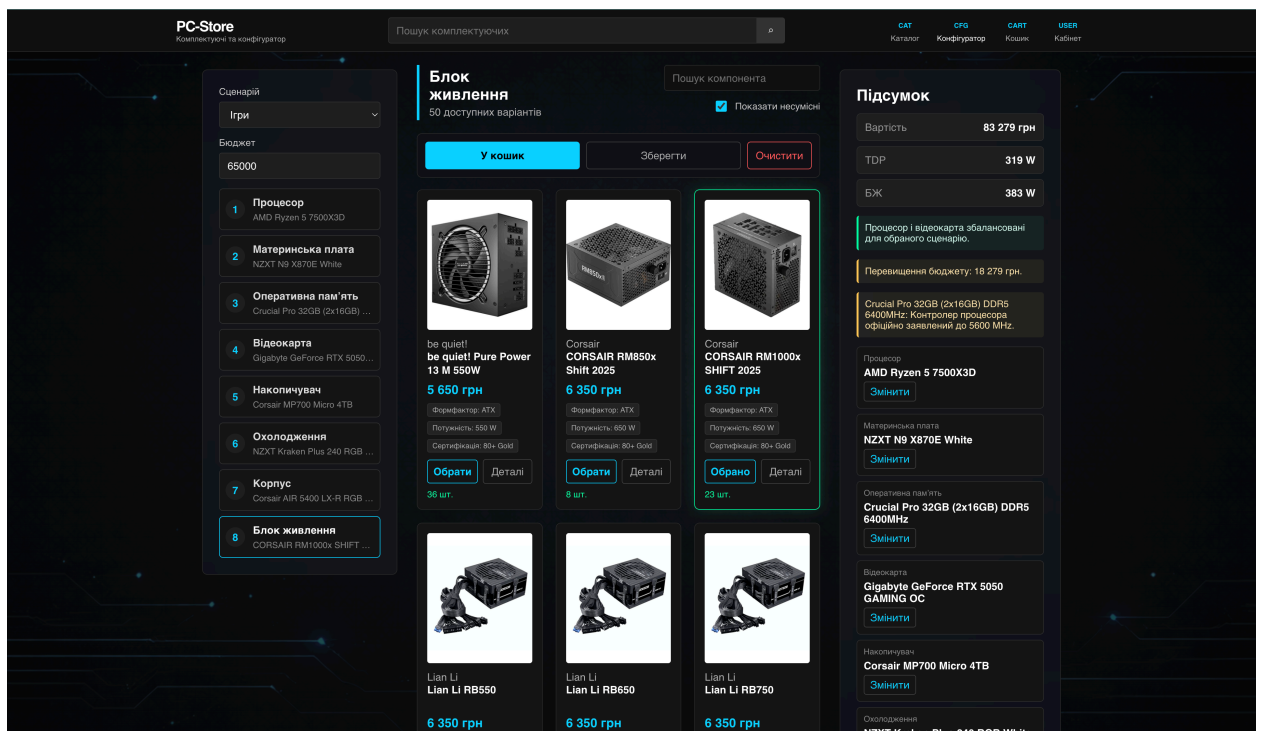


Рисунок 3.19 – Діагностика та підсумкова конфігурація ПК

Готову конфігурацію користувач може додати до кошика або зберегти в особистому кабінеті. Збереження збірки є важливим для користувачів, які не хочуть одразу оформлювати покупку або бажають повернутися до вибору пізніше. У такому випадку склад конфігурації та її загальна вартість передаються на сервер і зберігаються в базі даних.

Інтелектуальний помічник також спрощує взаємодію користувача з каталогом. Користувачу не потрібно самостійно перевіряти кожен

характеристику товару, шукати інформацію про сумісність або виконувати розрахунок потужності блоку живлення вручну. Система бере на себе частину технічної перевірки та дозволяє зосередитися на виборі потрібних комплектуючих.

Отже, інтелектуальний помічник-конфігуратор реалізує покроковий підбір комплектуючих, перевірку їх сумісності, врахування бюджету та формування підсумкової конфігурації ПК. Його використання дозволяє зробити процес складання комп'ютера зрозумілішим для користувача, зменшити кількість помилок під час вибору та поєднати функції інтернет-магазину з можливостями автоматизованого підбору компонентів.

3.6 Тестування роботи вебзастосунку

Після завершення програмної реалізації було проведено тестування основних функцій вебзастосунку. Метою тестування була перевірка коректності роботи клієнтської частини, серверного API, бази даних, кошика, замовлень, особистого кабінету, адміністративних функцій та інтелектуального помічника-конфігуратора.

Тестування виконувалося вручну шляхом послідовної перевірки основних сценаріїв використання системи. Такий підхід є доцільним для прототипу вебзастосунку, оскільки дозволяє перевірити не лише окремі функції, а й загальну логіку взаємодії користувача із сайтом. Під час перевірки враховувалася правильність відображення інтерфейсу, оброблення дій користувача, збереження даних і коректність роботи алгоритмів сумісності.

Окрему увагу під час тестування було приділено роботі інтелектуального помічника-конфігуратора. Перевірялося, чи правильно система реагує на вибір компонентів, чи враховує вже сформовану частину збірки та чи попереджає користувача про несумісність. Наприклад, якщо користувач обирає процесор

з одним сокетом, а потім намагається підібрати материнську плату з іншим сокетом, система повинна визначити таку комбінацію як некоректну.

Також було перевірено роботу з оперативною пам'яттю, корпусом, системою охолодження та блоком живлення. Для оперативної пам'яті система враховує її тип, для корпусу – формфактор материнської плати та габарити компонентів, для системи охолодження – підтримуваний сокет і висоту кулера, а для блоку живлення – потужність і формфактор. Це дозволяє перевірити не одну окрему умову, а комплексну логіку підбору комплектуючих.

Під час тестування кошика перевірялося додавання окремих товарів і готових конфігурацій ПК. Після додавання позицій система коректно оновлювала кількість товарів, загальну суму та список елементів у кошику. Також було перевірено оформлення замовлення: після введення даних покупця замовлення створювалося в системі та ставало доступним для перегляду в особистому кабінеті.

Окремо було перевірено адміністративну частину вебсайту. Адміністратор має доступ до перегляду замовлень, оновлення залишків товарів на складі, зміни статусу замовлення та перегляду журналу активності. Це дозволяє контролювати основні процеси інтернет-магазину без прямого звернення до бази даних.

Проведене тестування показало, що основні функції вебзастосунку працюють коректно. Користувач може переглядати каталог, шукати товари, працювати з кошиком, оформлювати замовлення, реєструватися, авторизуватися, зберігати конфігурації та використовувати інтелектуальний помічник для підбору комплектуючих. Адміністратор, у свою чергу, може керувати замовленнями, залишками товарів і переглядати основні події системи.

У процесі тестування було виявлено, що розроблений вебзастосунок виконує поставлені функціональні вимоги. Разом із тим система може бути розширена в майбутньому. До можливих напрямів удосконалення належать

додавання більшої кількості товарів, розширення правил сумісності, покращення рекомендацій для різних сценаріїв використання ПК, підключення онлайн-оплати та реалізація автоматичних сповіщень про зміну статусу замовлення.

У подальшому також можна розглянути розширення інтелектуального помічника за рахунок аналізу зображень комплектуючих. Наприклад, користувач міг би завантажити фотографію або зображення компонента, після чого система визначала б тип товару, порівнювала його з каталогом і пропонувала сумісні варіанти для збірки. У цьому напрямі можуть бути корисними наукові роботи, пов'язані з пошуком візуальних об'єктів, класифікацією зображень, формуванням ознак та аналізом структурних описів зображень [21–26]. Ще одним напрямом розвитку є додавання діалогового помічника, який зможе пояснювати вибір комплектуючих, відповідати на уточнення користувача та формувати рекомендації природною мовою з використанням сучасних мовних моделей [27, 28]. Для стабільної роботи системи з більшою кількістю користувачів і товарів також доцільно враховувати питання оптимізації серверної частини інформаційної системи [29].

Отже, тестування підтвердило працездатність основних модулів вебсайту та коректність взаємодії між клієнтською частиною, серверним API і базою даних. Розроблений вебзастосунок забезпечує базові функції інтернет-магазину комп'ютерної техніки та підтримує роботу інтелектуального помічника-конфігуратора, який допомагає користувачу сформувати сумісну конфігурацію ПК.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розглянуто процес розроблення вебсайту для продажу комп'ютерної техніки з інтегрованим інтелектуальним помічником для підбору комплектуючих. Метою роботи було створення вебзастосунку, який дозволяє користувачам підбирати комплектуючі з урахуванням їх сумісності, продуктивності та бюджету.

Під час виконання роботи було проведено аналіз предметної області, розглянуто особливості продажу комп'ютерної техніки в електронній комерції та визначено основні проблеми, які виникають під час самостійного підбору комплектуючих. Було встановлено, що користувачу потрібно враховувати сумісність процесора і материнської плати, тип оперативної пам'яті, габарити відеокарти, формфактор корпусу, потужність блоку живлення та інші технічні параметри.

Було сформульовано функціональні й нефункціональні вимоги до вебзастосунку, спроектовано клієнт-серверну архітектуру, структуру бази даних та логіку роботи інтелектуального помічника. Для програмної реалізації використано React, Node.js, Express, PostgreSQL та Prisma ORM. Клієнтська частина забезпечує взаємодію користувача із сайтом, серверна частина обробляє запити й працює з базою даних, а база даних зберігає інформацію про товари, користувачів, кошик, замовлення та збережені конфігурації.

У результаті роботи було реалізовано вебзастосунок із каталогом комп'ютерних комплектуючих, пошуком, кошиком, оформленням замовлень, реєстрацією та авторизацією користувачів, особистим кабінетом і адміністративною панеллю. Окремо реалізовано інтелектуальний помічник-конфігуратор, який забезпечує покроковий підбір компонентів, перевіряє їх сумісність, враховує бюджет і призначення майбутнього ПК та дозволяє сформувати готову конфігурацію.

Проведене тестування показало коректність роботи основних функцій вебзастосунку. Користувач може переглядати товари, додавати їх до кошика,

оформлювати замовлення, зберігати конфігурації та використовувати конфігуратор для підбору сумісних компонентів. Адміністратор має можливість переглядати замовлення, змінювати їх статус, оновлювати залишки товарів і переглядати журнал активності.

Отже, мету роботи досягнуто: створено вебзастосунок для продажу комп'ютерної техніки, який дозволяє підбирати комплектуючі з урахуванням їх сумісності, продуктивності та бюджету. Практична цінність розробленого рішення полягає у спрощенні процесу вибору комп'ютерних комплектуючих і зменшенні ймовірності помилок під час формування конфігурації ПК. У подальшому систему можна розширити шляхом додавання більшої кількості товарів, удосконалення правил сумісності, підключення онлайн-оплати та реалізації сповіщень про зміну статусу замовлення.

Результати роботи опубліковано у вигляді однієї тези доповіді під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [30].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Promodo. (2025). Дослідження українського eCommerce: тренди та прогнози на 2025. URL: <https://www.promodo.ua/ukrayinskiy-ecommerce-2024-2> (дата звернення 04.05.2026).
2. Horoshop. (2026). Яким був 2025 рік для українського e-commerce та чого чекати у 2026. URL: <https://horoshop.ua/ua/blog/ukrainian-ecommerce-resume-2025/> (дата звернення 04.05.2026).
3. Export Sales & Service. (2025). Світові тренди eCommerce у 2025 році: загальні та B2B. URL: <https://ess.business/svitovi-trendi-ecommerce-u-2025-roci-zagalni-ta-b2b/> (дата звернення 06.05.2026).
4. YouControl Market. (2026). Економіка e-commerce: виручка лідерів та статистика галузі. URL: <https://bdf.gov.ua/ekonomika-e-commerce-vyruchka-lideriv-ta-statystyka-haluzi/> (дата звернення 06.05.2026).
5. Falkner, A., Felfernig, A., & Haag, A. (2011). Recommendation technologies for configurable products. *AI Communications*, 24(2), 99–108.
6. Willför, B. (2025). Design and implementation of a compatibility algorithm for a computer configurator: Bachelor's thesis. Metropolia University of Applied Sciences. URL: <https://www.theseus.fi/handle/10024/895421> (дата звернення 15.05.2026).
7. Tvoroshenko, I., & Andrieieva, A. (2021). Development of web applications for remote learning of English.
8. Ткачук, В. Ю., & Рейда, О. М. (2025). Розробка інтерактивного вебдодатку конфігуратора персонального комп'ютера. Молодь в науці: дослідження, проблеми, перспективи (МН-2025). Вінниця: ВНТУ. URL: <https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/47740/25610.pdf> (дата звернення 18.05.2026).
9. PCPartPicker. (2026). PCPartPicker system builder. URL: <https://pcpartpicker.com/list/> (дата звернення 18.05.2026).

10. Felfernig, A., Friedrich, G., Jannach, D., & Zanker, M. (2015). Constraint-based recommender systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (pp. 161–191). Springer.
11. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.
12. Євтушенко Д., Творошенко І.С. (2024) Особливості застосування методів оптимізації бізнес-процесів, реалізованих засобами машинного навчання та нейронних мереж, *Abstracts of XI International Scientific and Practical Conference «Modern generation: current problems, experience, development prospects»*, (November 12 – 15, 2024). Seville, Spain, pp. 369-371.
13. Творошенко, І. С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ.
14. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник. Харків: ХНУРЕ.
15. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.
16. React. (2026). React Documentation. URL: <https://react.dev/docs> (дата звернення 20.05.2026).
17. REST API Tutorial. (2025). What is REST? URL: <https://restfulapi.net/> (дата звернення 20.05.2026).
18. Node.js. (2026). Node.js Documentation. URL: <https://nodejs.org/en/docs/> (дата звернення 21.05.2026).
19. Prisma. (2026). Prisma ORM Documentation. URL: <https://www.prisma.io/docs> (дата звернення 22.05.2026).
20. PostgreSQL. (2026). PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення 22.05.2026).

21. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.
22. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.
23. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.
24. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.
25. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.
26. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.
27. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.
28. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

29. Karakonyantyn D., Tvoroshenko I. (2024) About the issue of optimization the performance of the server part of the information system, *Abstracts of XII International Scientific and Practical Conference «Prospective directions of modern science and education in the world», (November 19 – 22, 2024). Rotterdam, Netherlands*, pp. 364-366.

30. Баран Д. С. (2026). Розробка інтелектуального вебсервісу з елементами мультимедіа для рекомендацій з підбору комп'ютерної техніки. Радіoeлектроніка та молодь у ХХІ столітті: 30 міжнародний форум. С. 31-32.