МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ГУСКА Андрій Андрійович

Вебзастосунок інтернет-магазину цифрової техніки з пошуком і фільтрацією товарів за технічними характеристиками / Web application of a digital electronics online store with search and filtering of products by technical characteristics

спеціальність: 122 – Комп'ютерні науки

освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42

А. А. Гуска

Науковий керівник

к.т.н., доцент Д. І. Загородня

Кваліфікаційну роботу допущено до

захисту «___»_____ 20___ р.

Завідувач кафедри

_____ Н.В. Дзюбановська

Тернопіль - 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГУСКА Андрій Андрійович
(прізвище, ім'я, по батькові)

1. Вебзастосунок інтернет-магазину цифрової техніки з пошуком і фільтрацією товарів за технічними характеристиками / Web application of a digital electronics online store with search and filtering of products by technical characteristics
керівник роботи Д. І. Загородня к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 20 грудня 2025 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:

– здійснити аналіз предметної області електронної комерції цифрової техніки та огляд існуючих інтернет-магазинів і платформ (Rozetka, Foxtrot, Amazon, Magento, Shopify, WooCommerce), визначити їх функціональні можливості;

– розробити архітектурне, алгоритмічне та інформаційне забезпечення вебзастосунку, спроєктувати алгоритми пошуку та багатокритеріальної фільтрації товарів за технічними характеристиками;

– реалізувати клієнт-серверне програмне забезпечення вебзастосунку, створити інтерфейс користувача та провести тестування вебзастосунку.

5. Перелік графічного матеріалу в роботі:

– структурна схема (архітектура) вебзастосунку інтернет-магазину цифрової техніки;

– діаграма послідовності виконання пошукового запиту;

– результати тестування вебзастосунку (таблиці функціонального тестування, метрики швидкодії, скріншоти інтерфейсу).

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 1 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ А.А. Гуска_____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Д. І. Загородня
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Вебзастосунок інтернет-магазину цифрової техніки з пошуком і фільтрацією товарів за технічними характеристиками» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом 71 сторінок і містить 23 ілюстрації, 4 додатки та 26 використаних джерел.

Метою кваліфікаційної роботи є проєктування та реалізація вебзастосунку інтернет-магазину цифрової техніки з ефективною підсистемою пошуку та багатокритеріальної фільтрації товарів за технічними характеристиками, що забезпечує гнучкість моделі даних для різних категорій товарів і високу швидкодію пошукових запитів.

Об’єктом дослідження є процеси отримання, зберігання, обробки та використання інформації про товари в інтернет-магазині цифрової техніки.

Предметом дослідження є методи, моделі та програмні засоби реалізації багатокритеріального пошуку, зберігання динамічних технічних атрибутів і оптимізації продуктивності запитів фільтрації.

Результатом роботи є повнофункціональний вебзастосунок Volt, побудований за клієнт-серверною архітектурою, що поєднує багатокритеріальний пошук і фільтрацію за технічними характеристиками, нечіткий пошук із стійкістю до помилок, керування кошиком і замовленнями, систему відгуків, бонусну програму, повернення товарів, інтеграцію з платіжним сервісом LiqPay та адміністративну панель із керуванням каталогом і аналітикою пошукових запитів.

Ключові слова: ВЕБЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, БАГАТОКРИТЕРІАЛЬНИЙ ПОШУК, ФІЛЬТРАЦІЯ, ТЕХНІЧНІ ХАРАКТЕРИСТИКИ, POSTGRESQL, JSONB, GIN-ІНДЕКС, REST API, REACT.

ANNOTATION

Qualification work on the topic «Web application of a digital electronics online store with search and filtering of products by technical characteristics» for obtaining a bachelor's degree in the specialty 122 «Computer Science» of the educational program «Computer Science» is written on 71 pages and contains 23 illustrations, 4 appendices, and 26 references.

The aim of the qualification work is the design and implementation of a web application of a digital electronics online store with an effective subsystem for searching and faceted filtering of products by technical characteristics, providing flexibility of the data model for different product categories and high performance of search queries.

The object of research is the processes of acquisition, storage, processing, and use of product information in an online store of digital electronics.

The subject of research is the methods, models, and software tools for implementing faceted search, storing dynamic technical attributes, and optimizing the performance of filtering queries.

The result of the work is a fully functional web application named Volt, built on a client-server architecture, that combines faceted search and filtering by technical characteristics, typo-tolerant fuzzy search, cart and order management, a reviews system, a bonus program, product returns, integration with the LiqPay payment service, and an administrative panel with dynamic catalog management and search query analytics.

Keywords: WEB APPLICATION, ONLINE STORE, FACETED SEARCH, FILTERING, TECHNICAL CHARACTERISTICS, POSTGRESQL, JSONB, GIN INDEX, REST API, REACT.

ЗМІСТ

Перелік умовних позначень 7

Вступ 8

1 Аналіз предметної області та існуючих рішень інтернет-магазинів цифрової техніки 10

1.1 Аналіз предметної області та функціональної структури інтернет-магазину цифрової техніки 10

1.2 Огляд і аналіз існуючих програмних аналогів 14

1.3 Постановка задачі та вимоги до проєктованої системи 20

2 Проєктування алгоритмічного та інформаційного забезпечення вебзастосунку 25

2.1 Архітектура та функціональна структура вебзастосунку 25

2.2 Алгоритмічне забезпечення підсистеми пошуку та багатокритеріальної фільтрації 30

2.3 Інформаційне забезпечення вебзастосунку 35

3 Реалізація та тестування вебзастосунку 40

3.1 Програмна реалізація серверної та клієнтської частин 40

3.2 Інтерфейс користувача та сценарії взаємодії 45

3.3 Тестування системи та оцінка результатів 52

Висновки 56

Список використаних джерел 59

Додаток А Копія публікації 62

Додаток Б Порівняльна характеристика існуючих рішень 67

Додаток В Лістинги ключових фрагментів програмного коду 68

Додаток Г Результати аудиту якості інтерфейсу інструментом lighthouse 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface – програмний інтерфейс прикладного програмування

CRUD – Create, Read, Update, Delete – стандартні операції з даними

CSS – Cascading Style Sheets – каскадні таблиці стилів

EAV – Entity-Attribute-Value – модель «сутність-атрибут-значення»

FTS – Full-Text Search – повнотекстовий пошук

GIN – Generalized Inverted Index – узагальнений інвертований індекс PostgreSQL

HTML – HyperText Markup Language – мова розмітки гіпертексту

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту

JSON – JavaScript Object Notation – текстовий формат обміну даними

JSONB – Binary JSON – бінарний формат зберігання JSON у PostgreSQL

JWT – JSON Web Token – підписаний токен для автентифікації

KPI – Key Performance Indicator – ключовий показник ефективності

NPM – Node Package Manager – менеджер пакетів для Node.js

REST – Representational State Transfer – архітектурний стиль програмного інтерфейсу

SaaS – Software as a Service – програмне забезпечення як послуга

SKU – Stock Keeping Unit – обліковий номер товарної позиції

SMTP – Simple Mail Transfer Protocol – протокол передачі електронної пошти

SPA – Single Page Application – односторінковий вебзастосунок

SQL – Structured Query Language – мова структурованих запитів

СУБД – Система управління базами даних

UI – User Interface – інтерфейс користувача

UX – User Experience – досвід взаємодії користувача із системою

XSS – Cross-Site Scripting – міжсайтове виконання сценаріїв

ВСТУП

У сучасних умовах стрімкого розвитку електронної комерції онлайн-продаж цифрової техніки став одним із найбільш динамічних сегментів роздрібної торгівлі. Категорія «електроніка та техніка» стабільно входить до трійки лідерів за обсягом продажів у переважній більшості великих онлайн-майданчиків, а в Україні протягом останніх років неухильно зростає. Покупці дедалі частіше обирають смартфони, ноутбуки і телевізори через інтернет, очікуючи від онлайн-магазинів зручних інструментів для швидкого добору товару, що відповідає їхнім потребам.

Актуальність теми зумовлена тим, що сегмент цифрової техніки характеризується високою розмірністю простору технічних характеристик: кожен товар описується десятками параметрів – діагоналлю та типом матриці екрана, обсягом оперативної та вбудованої пам’яті, ємністю акумулятора, продуктивністю процесора, наявністю підтримки сучасних бездротових стандартів тощо. Набір значущих характеристик принципово відрізняється від категорії до категорії: те, що є визначальним для смартфона, не має сенсу для телевізора чи навушників. За таких умов класичний текстовий пошук та проста ієрархічна навігація за категоріями виявляються недостатніми. Ключовим інструментом, що визначає зручність і конкурентоспроможність інтернет-магазину, стає підсистема багатокритеріального пошуку та фільтрації, яка дозволяє користувачеві поступово звужувати вибірку за довільним набором характеристик і миттєво бачити кількість доступних товарів для кожного значення.

Реалізація ефективної підсистеми пошуку потребує вирішення низки технічних задач: вибору моделі зберігання слабоструктурованих технічних атрибутів, забезпечення високої швидкодії запитів фільтрації, коректного підрахунку лічильників критеріїв, а також стійкості пошуку до орфографічних помилок і синонімів. Саме ці питання визначають практичну цінність і науково-технічну складову роботи.

Метою роботи є проєктування та реалізація вебзастосунку інтернет-магазину цифрової техніки з ефективною підсистемою пошуку та багатокритеріальної фільтрації товарів за технічними характеристиками.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати предметну область і виконати порівняльний аналіз існуючих рішень;

– сформувати функціональні та нефункціональні вимоги до проєктованої системи;

– спроєктувати архітектуру, алгоритмічне та інформаційне забезпечення системи;

– розробити алгоритми багатокритеріального пошуку, фільтрації та виправлення помилок написання запиту;

– реалізувати клієнт-серверний вебзастосунок і провести функціональне тестування системи.

Об’єктом дослідження є процеси отримання, зберігання, обробки та використання інформації про товари в інтернет-магазині цифрової техніки.

Предметом дослідження є методи, моделі та програмні засоби реалізації багатокритеріального пошуку, зберігання динамічних технічних атрибутів та оптимізації продуктивності запитів фільтрації.

Практичне значення одержаних результатів полягає в тому, що розроблений вебзастосунок є завершеним програмним продуктом, придатним до використання інтернет-магазинами цифрової техніки, а запропоновані архітектурні та алгоритмічні рішення можуть бути застосовані під час побудови подібних систем електронної комерції з розвиненою підсистемою пошуку.

Результати роботи апробовано та опубліковано в матеріалах IV Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп’ютерні системи та мережі» (м. Тернопіль, 2026 р.) (додаток А).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ ІНТЕРНЕТ-МАГАЗИНІВ ЦИФРОВОЇ ТЕХНІКИ

1.1 Аналіз предметної області та функціональної структури інтернет-магазину цифрової техніки

Електронна комерція – це сфера діяльності, що охоплює здійснення торговельних операцій, обмін товарами та послугами і пов’язані з ними інформаційні взаємодії за допомогою інформаційно-комунікаційних технологій та мережі Інтернет [1]. Інтернет-магазин є програмним засобом, який реалізує повний цикл онлайн-торгівлі: представлення каталогу товарів, пошук і добір потрібного товару, оформлення та оплату замовлення, а також супровід покупця після продажу – повернення, відгуки, бонусні програми, повторні покупки. Серед усіх категорій товарів сегмент цифрової техніки – смартфонів, ноутбуків, телевізорів, аудіотехніки, портативної електроніки та аксесуарів – належить до найбільш затребуваних і водночас найскладніших для організації каталогу [2].

Складність зумовлена природою товарів цієї галузі. На відміну від багатьох інших категорій, кожен пристрій описується великою кількістю технічних характеристик, що мають різний тип і призначення. Для смартфона істотними є діагональ і тип матриці екрана, частота оновлення, обсяг оперативної та вбудованої пам’яті, ємність акумулятора, наявність підтримки мереж 5G; для ноутбука – модель і покоління процесора, обсяг пам’яті, тип накопичувача та його ємність, тип і роздільна здатність екрана, дискретна графіка; для телевізора – діагональ, технологія матриці (LED, QLED, OLED), роздільна здатність, частота оновлення, наявність функцій Smart TV. Набір характеристик принципово відрізняється від категорії до категорії, тому модель даних повинна бути гнучкою і допускати зберігання різнорідних атрибутів без потреби змінювати схему бази даних при додаванні нової категорії [3].

Особливістю сегмента цифрової техніки є також короткий життєвий цикл товарних позицій: моделі оновлюються швидко, нові надходження з’являються щомісячно, а застарілі моделі потребують своєчасного виведення з каталогу або переведення до архіву. Це означає, що адміністративна частина системи повинна

забезпечувати зручне і швидке додавання нових товарів, динамічне керування набором характеристик у межах категорії та масове редагування каталогу.

Поведінка покупця цифрової техніки також має свою специфіку. На відміну від магазинів, порівняння декількох варіантів, ознайомлення з відгуками, додавання до кошика та оформлення замовлення. Завдання інтернет-магазину – на кожному з цих етапів надати ефективний інструмент, що мінімізує когнітивне навантаження користувача.

Центральним інструментом такого магазину є підсистема пошуку та фільтрації. Класична фільтрація обмежує вибірку набором значень одного-двох параметрів і не показує, що залишиться доступним при застосуванні додаткових критеріїв. Натомість сучасні маркетплейси використовують багатокритеріальний пошук (англ. faceted search) – методику навігації, за якої вибірка товарів послідовно уточнюється за множинними незалежними критеріями, а система в режимі реального часу показує, скільки товарів відповідає кожному значенню критерію [4]. На відміну від простої фільтрації, багатокритеріальний пошук обчислює лічильники для всіх можливих значень характеристик з урахуванням уже застосованих умов (окрім фільтра за тією самою характеристикою), завдяки чому користувач завжди бачить релевантні варіанти подальшого уточнення запиту і не потрапляє у глухий кут із порожньою видачею.

Окрім багатокритеріальної фільтрації, важливою складовою є текстовий пошук за назвою товару, брендом чи категорією. Сучасні користувачі очікують, що пошук буде стійким до помилок у написанні, розумітиме скорочення та синоніми (наприклад, що запит «телефон» стосується смартфонів, а «акумулятор» – портативних зарядних пристроїв), а також пропонуватиме підказки під час набору [5]. Поєднання якісного текстового пошуку та багатокритеріальної фільтрації формує основну споживчу цінність інтернет-магазину цифрової техніки і є головним предметом дослідження цієї роботи.

З погляду теорії інформаційних систем у функціональній структурі інтернет-магазину доцільно виокремити дві взаємопов’язані складові: операційну (транзакційну) та аналітичну. Операційна складова забезпечує обслуговування повсякденних взаємодій із користувачем: каталогізацію товарів, пошук і

фільтрацію, оформлення замовлень, обробку платежів, надсилання сповіщень. Аналітична складова забезпечує підтримку прийняття управлінських рішень: збір даних про популярні пошукові запити, аналіз нульових результатів пошуку, оцінку конверсії за категоріями, оцінку ефективності бонусних програм. Розмежування цих двох складових дозволяє чітко спланувати функціональність майбутньої системи і визначити пріоритети розробки.

Для систематизації функцій інтернет-магазину цифрової техніки доцільно побудувати дерево функцій, яке відображає основні групи можливостей системи та підпорядковані їм підфункції (рисунок 1.1).

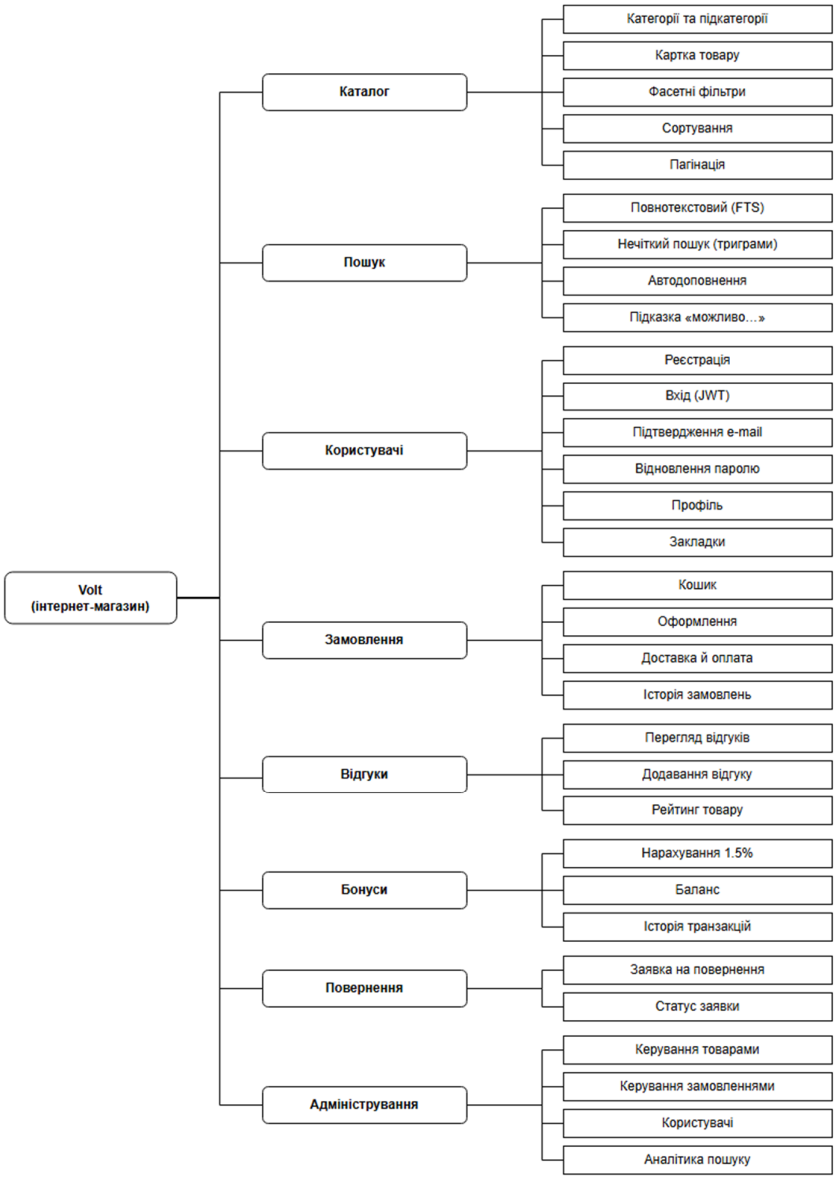

Рисунок 1.1 – Дерево функцій інтернет-магазину цифрової техніки

Наведена структура дерева функцій інтернет-магазину (представлена на рисунку 1.1) включає низку взаємопов’язаних функціональних блоків:

– робота з каталогом: ієрархічна організація категорій, керування брендами, динамічна схема технічних характеристик для кожної категорії, додавання та редагування товарів, керування зображеннями та наявністю на складі;

– пошук і навігація: повнотекстовий пошук за назвою товару, брендом і категорією; багатокритеріальна фільтрація за технічними характеристиками з підрахунком лічильників; нечіткий пошук зі стійкістю до помилок; автодоповнення під час набору; пропозиція виправлень за низькорезультативних запитів;

– робота з користувачами: реєстрація з підтвердженням електронної пошти, автентифікація, керування профілем, відновлення пароля, керування адресами доставки;

– оформлення замовлень: кошик з можливістю редагування кількості, вибір способу доставки та оплати, розрахунок підсумкової суми, інтеграція з платіжним сервісом, надсилання підтвердження поштою;

– відгуки та оцінки: залишення відгуків з оцінками від 1 до 5 зірок, обчислення середньої оцінки товару, обмеження «один відгук від одного користувача на товар», модерація відгуків адміністратором;

– бонусна програма: автоматичне нарахування бонусних балів за замовлення (наприклад, 3% від суми), облік історії бонусних транзакцій, можливість перегляду балансу в особистому кабінеті;

– повернення товарів: подання заявки на повернення із зазначенням причини, відстеження статусу заявки, інтерфейс адміністратора для обробки повернень;

– адміністрування та аналітика: керування всім каталогом (товари, категорії, бренди, атрибути), керування замовленнями та зміна їх статусів, керування користувачами і призначення прав адміністратора, модерація відгуків і повернень, аналітика пошукових запитів із виокремленням популярних і безрезультатних запитів.

Така деталізація функціональної структури дозволяє чітко визначити обсяг роботи та сформувати вимоги до окремих компонентів системи. Особливу увагу слід приділити підсистемі пошуку, оскільки саме вона формує основну споживчу цінність інтернет-магазину цифрової техніки.

Ефективність роботи інтернет-магазину прийнято оцінювати за низкою ключових показників (англ. Key Performance Indicators, KPI). До найбільш важливих з них належать: коефіцієнт конверсії (англ. Conversion Rate, CR) – відношення кількості оформлених замовлень до кількості відвідувань; середній чек (англ. Average Order Value, AOV); коефіцієнт використання пошуку – частка сесій, у яких користувач застосовував пошук або фільтри; коефіцієнт безрезультатних пошукових запитів – частка пошуків, що не повернули жодного товару; глибина перегляду каталогу – середня кількість сторінок товарів, переглянутих за сесію. Аналіз цих показників і відповідне коригування каталогу та підсистеми пошуку безпосередньо впливають на комерційну ефективність системи [6].

Таким чином, інтернет-магазин цифрової техніки можна розглядати як комплексну інформаційну систему, у структурі якої центральне місце посідає підсистема пошуку та багатокритеріальної фільтрації, а додатковими обов'язковими складовими є підсистеми обліку користувачів, оформлення замовлень, оплати, відгуків, бонусної програми та адміністративного керування з аналітичним модулем. Подальше проєктування системи має враховувати взаємозв'язки між цими підсистемами та забезпечити узгодженість їх роботи в межах єдиного програмного продукту.

1.2 Огляд і аналіз існуючих програмних аналогів

Для визначення вимог до проєктованої системи доцільно розглянути наявні рішення. Існуючі програмні продукти зручно поділити на дві групи: діючі інтернет-магазини цифрової техніки та платформи (рушії) для побудови інтернет-магазинів. Аналіз першої групи дозволяє оцінити очікувану функціональність з погляду користувача та виявити сильні сторони лідерів галузі; аналіз другої групи – типові архітектурні підходи, моделі даних та їхні обмеження, що допомагає сформувати

обґрунтовані технічні рішення. Розгляньмо ключові аналоги кожної групи за низкою порівнюваних критеріїв.

Rozetka [7] – найбільший український онлайн-маркетплейс, який охоплює широкий асортимент товарів, у тому числі великий розділ цифрової техніки. Платформа є фактичним еталоном функціональності у сегменті цифрової техніки в Україні. Розгляньмо її ключові особливості:

– багатокритеріальна фільтрація – реалізована з підтримкою великої кількості характеристик для кожної категорії, лічильниками товарів біля кожного значення фільтра, груповим керуванням за категоріями та брендами; інтерфейс адаптовано до тривалого набору фільтрів із підтримкою згортання;

– текстовий пошук – підтримує автодоповнення під час набору, пропозиції з товарів, категорій і брендів, пропозицію виправлень для помилкових запитів;

– ієрархічна структура каталогу – категорії організовано у дерево з кількома рівнями вкладеності; користувач може переходити від загальних розділів до конкретних підкатегорій;

– сторінка товару – містить розгорнуту таблицю характеристик, галерею зображень, відгуки з можливістю сортування за корисністю, блок «з цим товаром купують», порівняння товарів;

– інтеграція з гаманцем, програмою лояльності, оплатою, відстеженням замовлень – формує екосистемний продукт з утриманням клієнта;

– відкритість програмної реалізації – закрита; повторне використання архітектурних рішень неможливе; сторонні розробники не мають доступу до внутрішнього API.

Сильною стороною Rozetka є зрілий механізм пошуку та зручний інтерфейс, недоліком з погляду розробника – закритість, що унеможливлює застосування цього рішення як технологічної основи. На рисунку 1.2 наведено інтерфейс каталогу Rozetka з панеллю багатокритеріальних фільтрів.

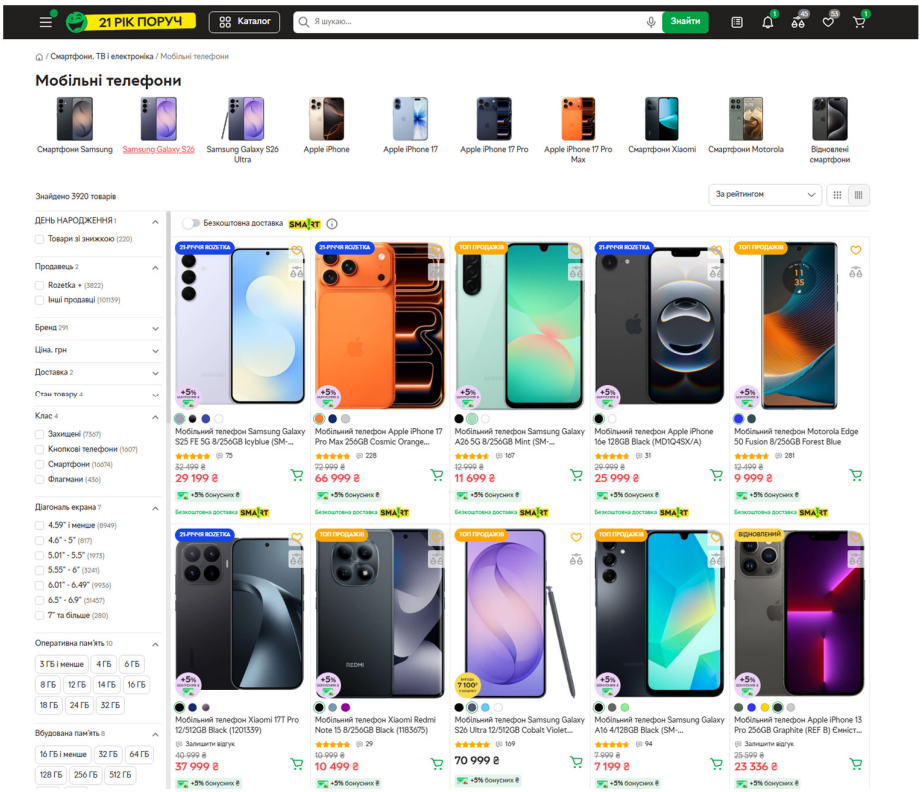

Рисунок 1.2 – Інтерфейс каталогу інтернет-магазину Rozetka з панеллю багатокритеріальних фільтрів

Foxtrot [8] – мережа магазинів побутової техніки та електроніки з добре розвиненим онлайн-каналом. На відміну від маркетплейсного підходу Rozetka, Foxtrot орієнтується саме на сегмент цифрової техніки. Розгляньмо особливості його реалізації:

– багатокритеріальна фільтрація – близька за функціональністю до Rozetka, проте з помітно меншою глибиною характеристик у деяких категоріях;

– сторінка товару – містить детальну таблицю характеристик, наочно демонструє формування коду товару (числового артикула), що є зручним для пошуку конкретної моделі;

– розрахунок вартості з акціями – реалізовано прозоро, з відображенням знижок і програм розстрочки;

– переваги – глибока спеціалізація на цифровій техніці, велика база актуальних товарів, фізична присутність у регіонах;

– недоліки – закрита платформа, обмежений набір додаткових сервісів порівняно з маркетплейсами.

На рисунку 1.3 наведено інтерфейс каталогу Foxtrot з панеллю багатокритеріальних фільтрів.

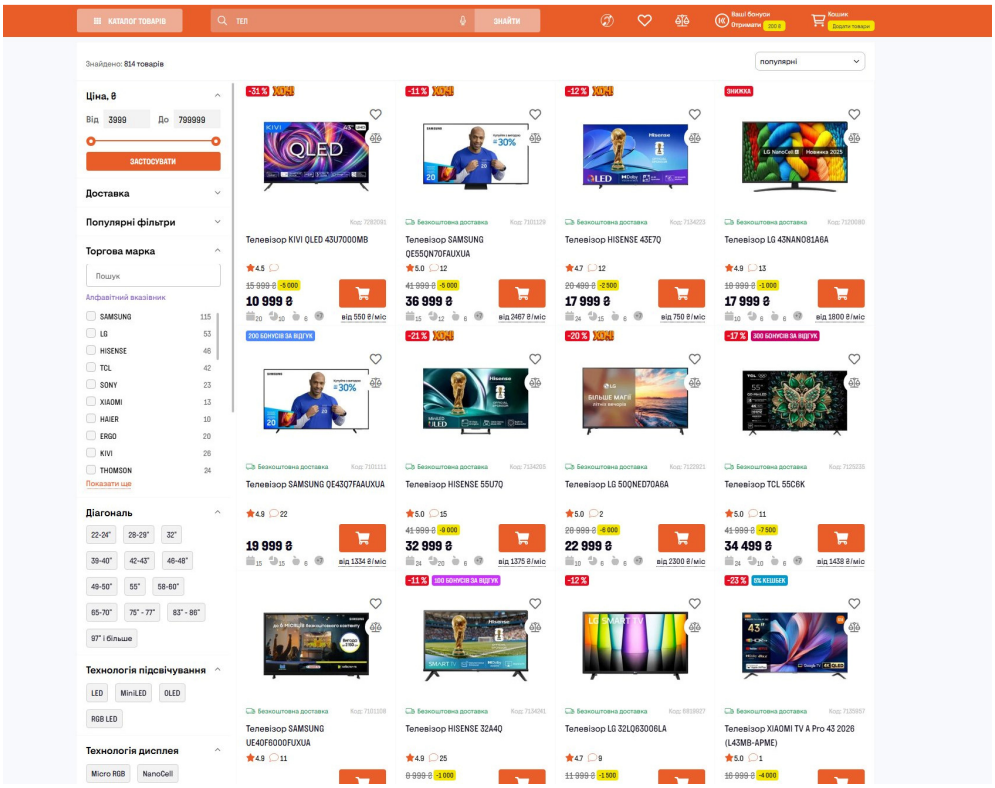

Рисунок 1.3 – Інтерфейс каталогу інтернет-магазину Foxtrot з панеллю багатокритеріальних фільтрів

Amazon [9] – світовий лідер електронної комерції, що задає міжнародні стандарти у сегменті. У контексті цифрової техніки Amazon демонструє декілька важливих особливостей: повнотекстовий пошук з ранжуванням за релевантністю та персоналізованими рекомендаціями; розвинений механізм відгуків з верифікованими покупцями; персоналізовані пропозиції на основі історії перегляду; добре розроблений механізм «питання-відповіді» біля кожного товару, що формує співтовариство покупців навколо продукту. Водночас масштаб і складність такого рішення значно перевищують потреби типового національного магазину цифрової техніки, а ліцензійні обмеження не дозволяють використовувати архітектуру Amazon у власних проєктах.

На рисунку 1.4 наведено інтерфейс каталогу Amazon з панеллю багатокритеріальних фільтрів.

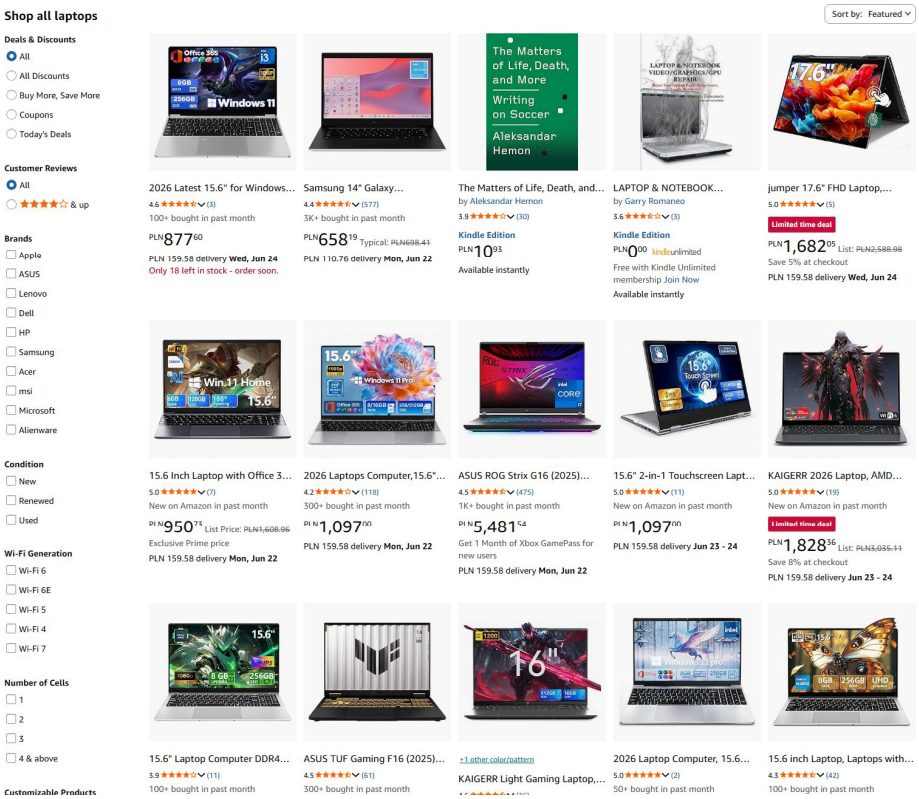

Рисунок 1.4 – Інтерфейс каталогу інтернет-магазину Amazon з панеллю багатокритеріальних фільтрів

До другої групи – платформ для побудови інтернет-магазинів – належать Magento (Adobe Commerce), Shopify та WooCommerce. Magento [10] є потужною системою з підтримкою складних каталогів, проте зберігання атрибутів товарів реалізовано в ній за моделлю «сутність-атрибут-значення» (англ. Entity-Attribute-Value, EAV). Ця модель забезпечує гнучкість: можна додавати атрибути без зміни структури таблиць. Натомість фільтрація за кількома атрибутами вимагає численних операцій з’єднання (JOIN) таблиць, що в каталогах із сотнями тисяч товарів призводить до суттєвої деградації продуктивності. Це широко відомий «антипатерн» проєктування баз даних [11]. На рисунку 1.5 схематично показано структуру даних за моделлю EAV.

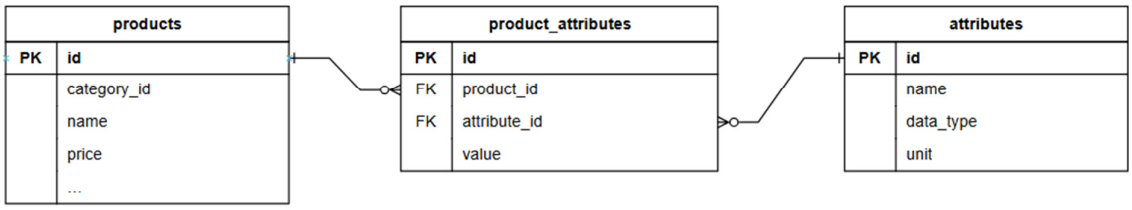

Рисунок 1.5 – Схематичне представлення моделі «сутність-атрибут-значення» (EAV)

Shopify [12] як хмарне SaaS-рішення обмежує свободу проєктування бази даних та логіки пошуку: розробник працює в межах накладених платформою обмежень, не має прямого доступу до бази даних, а розширення функціональності здійснюється через систему додатків і вебхуків. Перевагою є швидкий старт і вбудована підтримка платежів та доставки, недоліком – постійні платежі, обмежена гнучкість та залежність від постачальника послуг.

WooCommerce [13] – модуль для системи керування контентом WordPress. Він безкоштовний, відкритий і має велику спільноту розробників, однак успадковує обмеження WordPress: значне навантаження на сервер при великих обсягах даних, відносно повільне виконання складних запитів фільтрації та залежність від загального стану WordPress-середовища. WooCommerce підходить для невеликих і середніх магазинів, але виявляється субоптимальним для каталогів із гнучкою структурою характеристик і високим навантаженням.

Узагальнення розглянутих аналогів за ключовими критеріями наведено в таблиці функціональних можливостей та подано у додатку Б.

Проведений аналіз показує, що готові рішення або є закритими і не придатними для повторного використання, або накладають суттєві обмеження на модель даних і продуктивність пошуку. Це обґрунтовує доцільність розробки власного вебзастосунку з продуманою моделлю зберігання технічних характеристик та оптимізованою підсистемою багатокритеріального пошуку. Зокрема, як альтернативу моделі EAV у роботі пропонується застосувати сучасний підхід – зберігання характеристик товару в одному полі типу JSONB (бінарний

JSON) у PostgreSQL разом з індексом GIN (узагальнений інвертований індекс), що поєднує гнучкість схеми з високою продуктивністю запитів фільтрації [14, 15]. Обґрунтування цього вибору буде наведено у розділі 2 під час проєктування інформаційного забезпечення.

1.3 Постановка задачі та вимоги до проєктованої системи

Аналіз предметної області та наявних рішень дозволяє сформулювати постановку задачі. Метою роботи є проєктування та реалізація вебзастосунку інтернет-магазину цифрової техніки з ефективною підсистемою пошуку та багатокритеріальної фільтрації товарів за технічними характеристиками. Розроблюваний вебзастосунок має ім’я Volt і призначений для забезпечення повного циклу онлайн-торгівлі цифровою технікою – від першого відвідування каталогу до повторних замовлень постійним клієнтом.

Функціональні вимоги визначають перелік можливостей, які має надавати вебзастосунок:

1) забезпечити перегляд каталогу товарів за категоріями та брендами з пагінацією та сортуванням за ціною, оцінкою, новизною та популярністю;

2) реалізувати багатокритеріальну фільтрацію товарів за технічними характеристиками з підрахунком кількості товарів для кожного значення критерію за принципом виключення власного виміру;

3) реалізувати текстовий пошук за назвою товару, брендом і категорією зі стійкістю до орфографічних помилок та підтримкою синонімів, з автодоповненням під час набору і пропозицією виправлень для безрезультатних запитів;

4) забезпечити перегляд детальної сторінки товару з галереєю зображень, повним переліком технічних характеристик та блоком відгуків з оцінками;

5) реалізувати реєстрацію та автентифікацію користувачів з підтвердженням електронної пошти, відновлення пароля за токеном, керування особистим кабінетом і списком закладок;

6) реалізувати функціонал кошика з можливістю редагування кількості, оформлення замовлення з вибором способу доставки та оплати, у тому числі онлайн-оплати через сервіс LiqPay;

7) забезпечити можливість залишення відгуків про товари з оцінкою від 1 до 5 зірок, обчислення середньої оцінки товару та автоматичне нарахування бонусних балів за замовлення;

8) реалізувати функціонал повернення товарів – подання заявки користувачем і опрацювання її адміністратором;

9) розробити адміністративну панель для керування товарами, категоріями, брендами, атрибутами, замовленнями, користувачами, відгуками, поверненнями та перегляду аналітики пошукових запитів;

10) забезпечити надсилання поштових сповіщень – підтвердження пошти, відновлення пароля, підтвердження замовлення.

Нефункціональні вимоги визначають якісні характеристики системи:

– швидкодія – час відгуку на запити пошуку та фільтрації не повинен перевищувати кількох сотень мілісекунд у каталогах із десятками тисяч товарів;

– гнучкість моделі даних – можливість зберігати різні набори характеристик для різних категорій без зміни структури таблиць; додавання нової характеристики до категорії має відбуватися без правок у програмному коді;

– безпека – захист персональних даних користувачів, безпечне зберігання паролів із застосуванням криптографічного хешування, розмежування прав доступу між звичайними користувачами та адміністраторами, захист від типових веб-загроз (SQL-ін’єкції, XSS);

– зручність використання – інтуїтивна навігація, адаптивність інтерфейсу до екранів різного розміру, наочне представлення результатів пошуку та фільтрації, мінімізація кроків для виконання типових сценаріїв;

– розширюваність та супроводжуваність – модульна структура коду з чітким розподілом відповідальності, документований програмний інтерфейс, відстеження змін структури бази даних через нумеровані сценарії міграцій.

Користувачів системи поділено на три категорії з різними наборами прав і функціональних можливостей. Гість (неавторизований відвідувач) може

переглядати каталог, виконувати пошук і фільтрацію, переглядати сторінки товарів і відгуки. Зареєстрований користувач отримує доступ до особистого кабінету, історії замовлень, бонусного рахунку, списку закладок, може залишати відгуки та подавати заявки на повернення. Адміністратор має повний доступ до керування каталогом, замовленнями, користувачами, відгуками, поверненнями та переглядає аналітику пошукових запитів. Узагальнену схему варіантів використання системи наведено на рисунку 1.6.

Рисунок 1.6 – Діаграма варіантів використання системи Volt

Технологічні вимоги до реалізації передбачають побудову системи за клієнт-серверною архітектурою з використанням сучасного вебстеку. Обґрунтуємо вибір конкретних компонентів стеку:

– клієнтська частина – бібліотека React 18 із зборкою інструментом Vite та стилізацією через утилітарну бібліотеку Tailwind CSS. React забезпечує компонентний підхід, високу швидкодію оновлень інтерфейсу за рахунок віртуального DOM та широку екосистему; Vite дає швидку розробку з гарячим перезавантаженням модулів; Tailwind CSS дозволяє створювати інтерфейс безпосередньо в розмітці компонентів без потреби в окремих файлах стилів;

– серверна частина – платформа Node.js з фреймворком Express. Node.js забезпечує асинхронну неблокуючу модель обробки запитів, що особливо ефективно для I/O-операцій із базою даних; Express є мінімалістичним фреймворком, що дозволяє побудувати REST API з чіткою структурою маршрутів;

– система управління базами даних – PostgreSQL версії 14 або вище. Цей вибір зумовлений нативною підтримкою типу JSONB з GIN-індексами, що дозволяє ефективно реалізувати гнучку модель характеристик товарів; повнотекстовим пошуком засобами вбудованого типу tsvector; розширенням pg_trgm для нечіткого пошуку на основі триграм; а також можливістю реалізації тригерів і послідовностей для автоматизації;

– автентифікація – на основі веб-токенів JSON (JWT), що дозволяє побудувати безстанову (stateless) автентифікацію без потреби у серверних сесіях; паролі зберігаються виключно у вигляді криптографічних хешів, обчислених бібліотекою bcrypt;

– платіжний сервіс – LiqPay від ПриватБанку як один із найпоширеніших платіжних сервісів в Україні, з підтримкою тестового sandbox-режиму, що зручно для розробки;

– поштові сповіщення – бібліотека nodemailer з підтримкою SMTP, у тому числі через сервіс Gmail; за відсутності налаштованого поштового сервера передбачено резервне виведення листів у консоль для потреб розробки.

Технічними обмеженнями системи вважаються: робота у браузерах останніх версій (Chrome, Firefox, Edge, Safari) із підтримкою сучасного JavaScript (ES2020+); орієнтація переважно на українську та англійську мовні аудиторії з відповідними лінгвістичними особливостями пошуку; забезпечення адаптивності інтерфейсу до екранів від 360 до 1920 пікселів за шириною.

Сформульовані вимоги і обраний технологічний стек створюють основу для подальшого проєктування архітектури, алгоритмічного та інформаційного забезпечення системи, що розглядається у наступному розділі.

2 ПРОЄКТУВАННЯ АЛГОРИТМІЧНОГО ТА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ВЕБЗАСТОСУНКУ

2.1 Архітектура та функціональна структура вебзастосунку

Архітектурне рішення вебзастосунку Volt побудоване за класичною трирівневою моделлю «клієнт – сервер застосунку – база даних» (англ. three-tier architecture), яка є де-факто стандартом для сучасних вебзастосунків електронної комерції [16]. Така модель забезпечує чіткий розподіл обов'язків між рівнями, спрощує супровід коду, дозволяє масштабувати кожен рівень незалежно і – що особливо важливо – формує природний контур безпеки, у якому база даних взагалі недоступна напряму з боку клієнтського пристрою.

Перший рівень – рівень подання. Його реалізовано як односторінковий застосунок (SPA, англ. single-page application) на основі бібліотеки React. Він виконується у браузері користувача, відповідає за побудову інтерфейсу, маршрутизацію в межах застосунку (без перезавантаження сторінок), збереження клієнтського стану (вміст кошика, обраного, активних фільтрів) та обмін даними з сервером через виклики REST API. Сам рівень подання не містить бізнес-логіки, що стосувалася б цілісності даних: усі бізнес-правила (формування ціни, контроль залишків, нарахування бонусів, перевірка дозволів) винесено на сервер.

Другий рівень – рівень логіки застосунку. Його реалізовано як HTTP-сервер на платформі Node.js з фреймворком Express. Він приймає запити від клієнта, виконує автентифікацію та авторизацію, перетворює параметри запиту у SQL-вирази, звертається до бази даних, формує відповіді у форматі JSON і повертає їх клієнту. Сервер є безстановим (англ. stateless): сесія користувача ідентифікується не серверним кешем, а JWT-токеном, який клієнт передає у заголовку Authorization при кожному захищеному запиті. Такий підхід полегшує горизонтальне масштабування – за зростання навантаження достатньо запустити декілька екземплярів сервера за балансувальником, без потреби синхронізації сесій між ними.

Третій рівень – рівень даних. Він забезпечується СУБД PostgreSQL. У ній зберігається все, що потребує тривалого збереження: каталог товарів і

характеристик, користувацькі акаунти, замовлення, відгуки, аналітика пошукових запитів. На цьому рівні також реалізовано частину бізнес-логіки, що тісно прив'язана до даних: тригер автоматичного формування пошукового вектора, послідовність для генерації артикулів, обмеження цілісності даних, налаштування індексів. Логічне розділення на рівні дозволяє замінити будь-який з них без зміни решти – наприклад, перевести клієнтську частину на іншу JavaScript-бібліотеку чи замінити поштовий сервіс – за умови збереження інтерфейсу взаємодії.

Окрім трьох основних рівнів, система взаємодіє з декількома зовнішніми сервісами. Це сервіс приймання платежів LiqPay, через який обробляються онлайн-оплати банківськими картками; SMTP-сервер для надсилання користувачам листів підтвердження реєстрації, відновлення паролю та сповіщень про статус замовлення; а також локальна файлова система для зберігання завантажених адміністратором зображень товарів. Структуру взаємодії всіх компонентів унаочнено на рисунку 2.1.

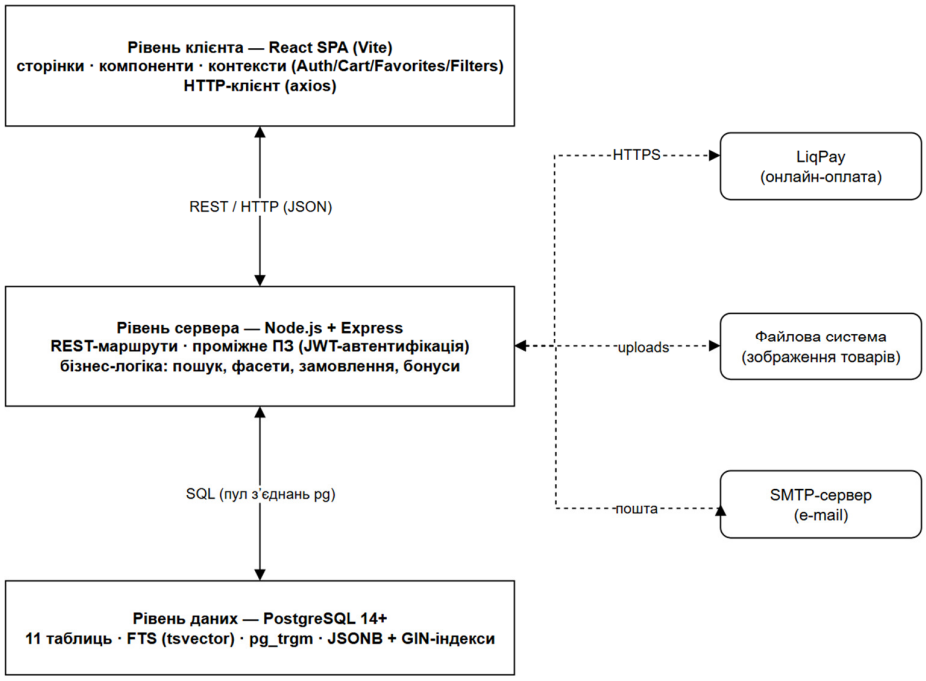

Рисунок 2.1 – Загальна архітектура вебзастосунку Volt

На рівні API сервер реалізує єдиний контракт обміну даними у вигляді REST-маршрутів, кожен з яких відповідає окремому ресурсу системи. Маршрути логічно

об’єднано у функціональні групи: каталог (категорії, бренди, товари), користувачі (реєстрація, автентифікація, профіль), замовлення (створення, перегляд, статуси), відгуки, повернення, бонусні нарахування, платежі та адміністрування. Перелік основних маршрутів подано у таблиці 2.1.

Таблиця 2.1 – Основні маршрути REST API

Маршрут	Метод	Призначення
/api/categories	GET	Перелік усіх категорій каталогу
/api/categories/:slug	GET	Детальна інформація про категорію
/api/brands	GET	Перелік брендів
/api/products	GET	Пошук, фільтрація і пагінація товарів
/api/products/suggest	GET	Підказки автодоповнення під час введення запиту
/api/products/facets/:cat	GET	Доступні значення критеріїв для категорії
/api/products/filters	GET	Універсальні критерії (без прив’язки до категорії)
/api/products/:slug	GET	Детальна сторінка товару
/api/auth/register	POST	Реєстрація нового користувача
/api/auth/login	POST	Автентифікація і видача JWT-токена
/api/auth/me	GET/PUT	Перегляд і редагування власного профілю
/api/auth/forgot-password	POST	Запит на відновлення паролю
/api/orders	POST/GET	Створення замовлення; історія замовлень користувача
/api/reviews/:slug	GET/POST	Відгуки про товар; додавання власного
/api/bonuses	GET	Поточний баланс і історія бонусних транзакцій
/api/payments/liqpay/checkout	POST	Ініціація онлайн-оплати
/api/admin/*	ANY	Адміністративні операції (доступ лише для is_admin)

Усі маршрути обробляються через єдиний ланцюжок проміжного програмного забезпечення (англ. middleware). На вході запит проходить через модуль безпеки helmet, що додає заголовки захисту від найпоширеніших атак XSS

і clickjacking; модуль cors, який обмежує перелік доменів, уповноважених звертатися до API; парсер JSON-тіла запиту; систему журналювання morgan, що фіксує всі звернення. Для захищених маршрутів – додатково проходить через middleware автентифікації, який перевіряє наявність та валідність JWT і завантажує дані поточного користувача. На виході відпрацьовує централізований обробник помилок, який гарантує, що клієнт у будь-якому випадку отримає коректну JSON-відповідь з відповідним HTTP-статусом, навіть якщо у глибині запиту виникла непередбачена помилка.

Контекстну діаграму процесу обробки клієнтського запиту в нотації IDEF0 подано на рисунку 2.2. Вона унаочнює інформаційні потоки між основними процесами – формуванням запиту в інтерфейсі користувача, обробкою на сервері, виконанням SQL-запиту в БД та формуванням відповіді – а також механізми керування (правила автентифікації, схеми атрибутів) і виконавчі механізми (БД, Node.js, файлова система).

Рисунок 2.2 – Контекстна IDEF0-діаграма процесу обробки запиту користувача

Для відображення динамічної взаємодії компонентів під час виконання типового пошукового запиту побудовано діаграму послідовності (sequence diagram), наведену на рисунку 2.3. На ній показано хронологічну послідовність повідомлень між клієнтським компонентом фільтрів, REST-сервером, СУБД і модулем формування критеріїв – від моменту вибору значення фільтра користувачем до оновлення вмісту каталогу і лічильників критеріїв у бічній панелі.

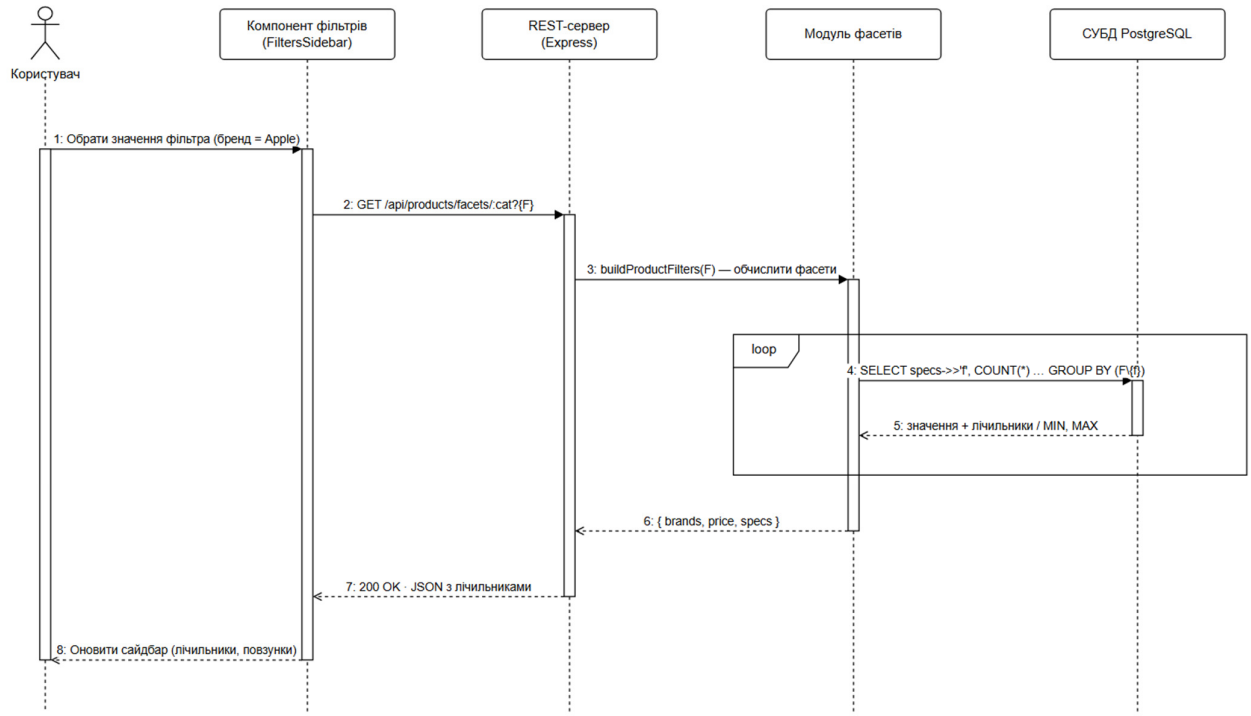

Рисунок 2.3 – Діаграма послідовності виконання пошукового запиту

Описана архітектура є основою для проєктування алгоритмічного та інформаційного забезпечення, що розглядаються у наступних підрозділах. У підрозділі 2.2 деталізовано ключові алгоритми – текстового пошуку, багатокритеріальної фільтрації та виправлення помилок написання, – а у підрозділі 2.3 проаналізовано модель даних і обґрунтовано вибір способу зберігання технічних характеристик.

2.2 Алгоритмічне забезпечення підсистеми пошуку та багатокритеріальної фільтрації

Центральною функціональною можливістю вебзастосунку Volt, що формує її основну споживчу цінність, є поєднана підсистема пошуку та багатокритеріальної фільтрації. У цьому підрозділі розглянуто алгоритмічну основу її роботи: алгоритм текстового пошуку з виправленням помилок, алгоритм обчислення лічильників критеріїв і алгоритм нечіткого зіставлення пошукового запиту з категоріями каталогу.

Текстовий пошук у системі реалізовано трирівневим: послідовно застосовуються методи повнотекстового пошуку, нечіткого зіставлення та підрядкового збігу, а результати об’єднуються через логічне АБО. Такий підхід забезпечує одночасно і високу точність (за наявності точної відповідності), і толерантність до помилок написання, і коректну роботу за коротких часткових запитів. Загальна послідовність обробки текстового пошукового запиту наведена на рисунку 2.4.

На першому рівні застосовується повнотекстовий пошук (англ. full-text search, FTS) засобами вбудованої функціональності PostgreSQL. Для кожного товару під час його створення або редагування автоматично обчислюється спеціальний пошуковий вектор типу tsvector. Цей вектор будується із трьох джерел: назви товару, короткого опису і повного опису, причому слова з різних джерел отримують різну вагу – назва маркується вагою A, короткий опис вагою B, повний опис вагою C. При виконанні запиту користувацький текст перетворюється функцією plainto_tsquery у tsquery-структуру, після чого здійснюється пошук операцією @@. Релевантність кожного знайденого товару оцінюється функцією ts_rank, яка для кожного входження слова враховує його вагу і обчислюється за такою спрощеною формулою (2.1) [10]:

$$rank(d, q) = \Sigma_i w_i \cdot cnt(lex_i, d) \quad (2.1)$$

де d – документ (товар);

q – пошуковий запит;

lex_i – і-та лексема (нормалізована форма слова) пошукового запиту;

w_i – вага лексеми, яка визначається маркером її входження в документ: лексеми з частини «назва товару» маркуються вагою A (найвищий коефіцієнт, за замовчуванням 1,0), з короткого опису – вагою B (0,4), з повного опису – вагою C (0,2);

$cnt(lex_i, d)$ – кількість входжень і-тої лексеми в документ d.

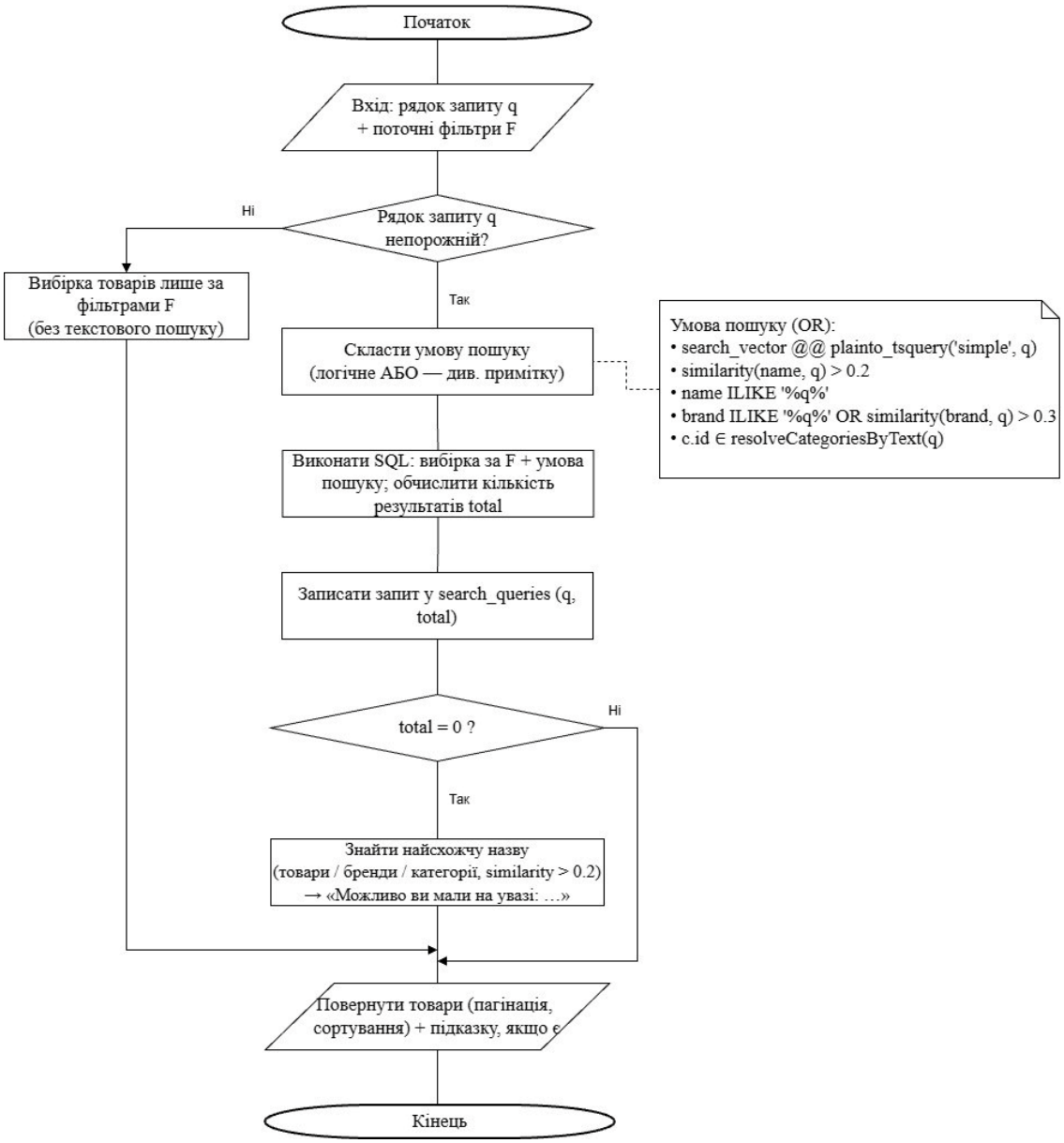

Рисунок 2.4 – Блок-схема алгоритму текстового пошуку з виправленням помилок

Такий розрахунок гарантує, що в перших позиціях видачі опиняються товари, у назві яких пошукові слова зустрічаються найчастіше, а збіги в описі мають менший, але ненульовий внесок у підсумкову оцінку релевантності.

На другому рівні застосовується нечіткий пошук на основі триграм. Триграма – це впорядкована послідовність із трьох послідовних символів рядка. Слово розкладається на множину всіх своїх триграм; наприклад, для слова «телефон» з огляду на додавання маркерів початку і кінця рядка множиною триграм буде { т, те, тел, еле, леф, ефо, фон, он }. Подібність двох рядків оцінюється коефіцієнтом Жаккара (2.2) [11], що обчислюється для множин їх триграм:

$$sim(A, B) = |T(A) \cap T(B)| / |T(A) \cup T(B) \quad (2.2)$$

де T(A) і T(B) – множини триграм рядків A і B.

У формулі 2.2 чисельник – потужність їх перетину; знаменник – потужність об'єднання. Значення коефіцієнта належить діапазону [0; 1]: 0 означає повну відсутність спільних триграм, 1 – повний збіг рядків. Експериментально встановлено [10], що значення sim > 0.2 для назв товарів і sim > 0.3 для назв брендів дають оптимальне співвідношення повноти та точності – система знаходить товари за запитами з 1–2 одруками, але не пропонує невідповідних результатів. У PostgreSQL цей механізм реалізовано розширенням pg_trgm, а для прискорення зіставлення створено GIN-індекси з оператором gin_trgm_ops над полями name таблиць products, brands і categories.

На третьому рівні застосовується простий підрядковий пошук операцією ILIKE з шаблоном «%запит%». Цей рівень забезпечує спрацьовування на коротких запитах (1–2 символи), для яких FTS не повертає результатів, та на запитах з нелатинськими/некириличними символами і цифрами.

Окремою складовою алгоритму є зіставлення пошукового запиту з категоріями каталогу. Якщо користувач вводить запит «телефон», очікувано отримати товари категорії «Смартфони», хоча самого слова «телефон» у назвах товарів може не бути. Для розв'язання цієї задачі у таблиці categories введено поле

keywords – текстовий рядок із ключовими словами та синонімами, що описують категорію. Алгоритм зіставлення працює у два етапи. На першому етапі (сильний збіг) шукається підрядкове входження запиту у назву категорії або в поле keywords; якщо такі категорії знайдено – результат використовується безпосередньо. На другому етапі (нечіткий fallback), який виконується лише за відсутності сильних збігів, застосовується триграмна подібність до назви категорії (поріг 0.35) або до окремих слів з keywords (поріг 0.45). Такий двоетапний підхід дозволяє обробити одруки типу «телевізер» → «телевізор», «акамулятор» → «акумулятор», але водночас не розширює видачу, коли точне зіставлення вже знайдено.

Алгоритм багатокритеріальної фільтрації обчислює лічильники товарів для кожного можливого значення кожної характеристики у поточній видачі. Ключова особливість його реалізації – правило «виключити власний вимір» (англ. exclude own dimension), за яким при підрахунку доступних значень конкретного критерію фільтр за цим самим критерієм тимчасово виключається з умов запиту. Інакше користувач не зміг би перемкнути значення фільтра: переключившись зі «512 ГБ» на «256 ГБ», він побачив би лічильник «512 ГБ» рівним нулю – бо обмеження «обсяг = 512 ГБ» виключило б усі товари з обсягом 256 ГБ. Алгоритм для конкретного критерію f та поточного набору фільтрів F формалізується як обчислення кардинальності множини за виразом (2.3):

$$count(f, v) = |\{\, p \in P : * \,(F \setminus \{f\}) \text{ виконуються для } p \;\wedge p.specs[f] = v\}| \quad (2.3)$$

де P – множина всіх активних товарів;
F – поточний набір застосованих фільтрів;
F \ {f} – набір фільтрів без фільтра за виміром f;
v – кандидатне значення критерію.

Результат обчислюється єдиним SQL-запитом виду SELECT specs->>'<f>', COUNT(*) FROM products WHERE … GROUP BY specs->>'<f>'. Для числових характеристик замість лічильника обчислюються мінімальне та максимальне значення в поточній вибірці – ці значення задають межі повзунка діапазону у інтерфейсі. Загальна схема обчислення критеріїв наведена на рисунку 2.5.

Окремою функціональністю є побудова підказок автодоповнення під час введення пошукового запиту. Алгоритм формування підказок виконує паралельно три запити: пошук серед назв товарів, пошук серед брендів і пошук серед категорій. Для кожного кандидата обчислюється зважена оцінка релевантності, яка враховує спосіб збігу (точний підрядок отримує вищу оцінку ніж нечіткий) і тип сутності (категорія отримує вищий пріоритет, ніж окремий товар). Об'єднаний список сортується за оцінкою у спадному порядку та обмежується десятьма позиціями, формуючи компактну і релевантну підказку.

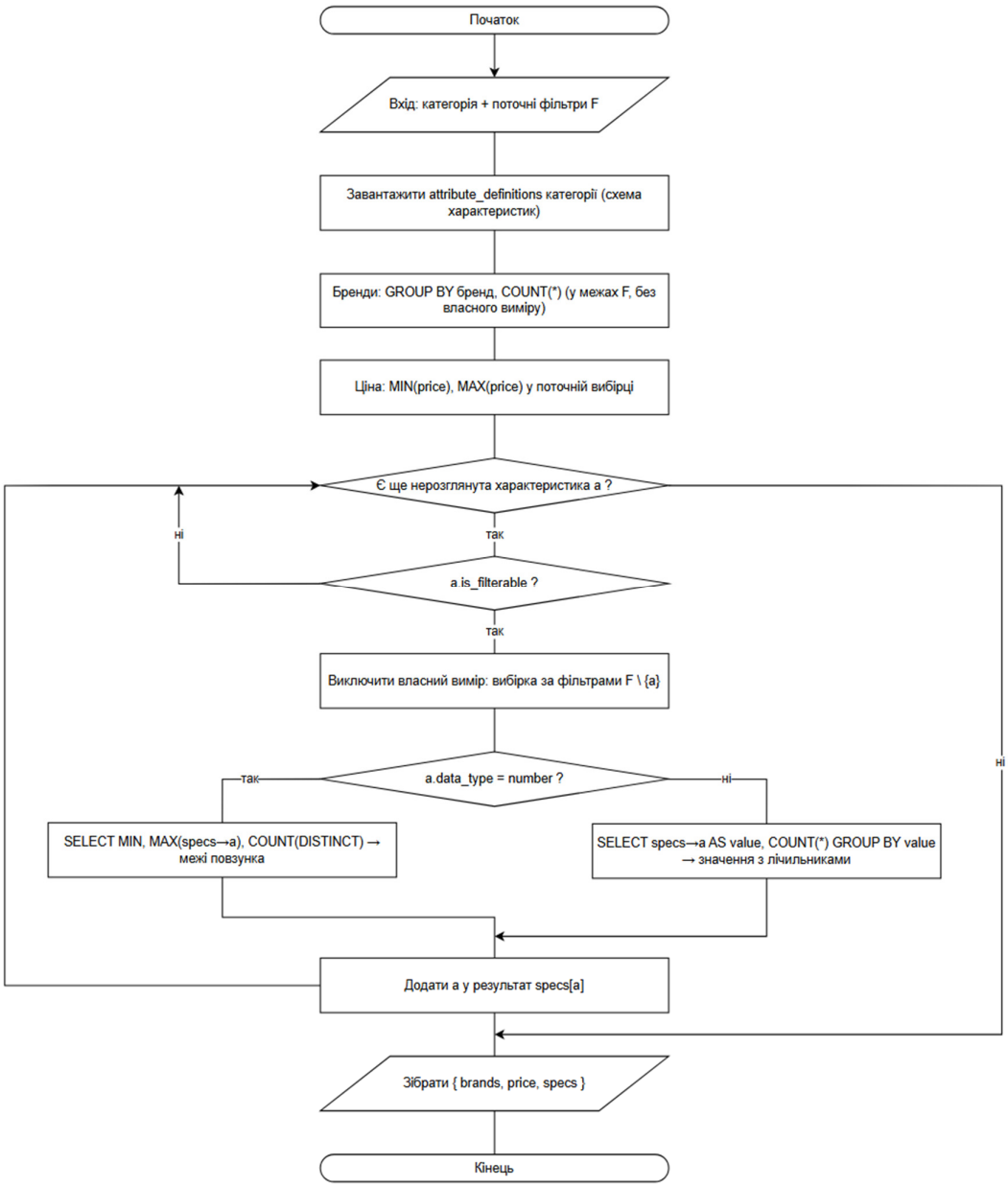

Рисунок 2.5 – Алгоритм обчислення лічильників багатокритеріальної фільтрації

Розроблені алгоритми забезпечують одночасно високу швидкість, точність і толерантність до помилок користувача. Експериментально на тестовому каталозі з декількох сотень товарів час виконання типового запиту з пошуком і набором фільтрів становить десятки мілісекунд, що є достатньо для відчуття користувачем миттєвої реакції інтерфейсу. Конкретні результати замірів наведено у розділі 3.

2.3 Інформаційне забезпечення вебзастосунку

Інформаційне забезпечення вебзастосунку Volt побудоване на реляційній моделі даних з гібридним доповненням у вигляді документоорієнтованого типу JSONB для зберігання технічних характеристик товарів. Цей розділ описує логічну структуру бази даних, обґрунтовує обраний підхід до моделі характеристик та розглядає допоміжні елементи – тригери, послідовності та індекси.

Логічна модель бази даних містить одинадцять таблиць, що поділяються на чотири функціональні групи. Першу групу – каталог – формують таблиці categories (ієрархічний довідник категорій товарів), brands (виробники), attribute_definitions (схема характеристик для кожної категорії) та products (товари). Другу групу – користувачі та замовлення – утворюють users (акаунти), orders (замовлення) і order_items (позиції замовлення). До третьої групи входять bonus_transactions (історія бонусних нарахувань), returns (повернення товарів) та reviews (відгуки). Четверту групу формує службова таблиця search_queries для аналітики пошукових запитів. Перелік таблиць з коротким описом наведено у таблиці 2.2; повну ER-діаграму подано на рисунку 2.6.

Ключовою архітектурною задачею при проєктуванні моделі даних було обрання способу зберігання технічних характеристик товарів. Як зазначалось у підрозділі 1.1, набір характеристик істотно відрізняється від категорії до категорії: для смартфона значущими є обсяг ОЗП, ємність акумулятора, частота оновлення екрана; для пилососа – потужність, тип, наявність акумулятора; для телевізора – тип матриці і роздільна здатність. Класичний підхід «одна колонка – одна характеристика» у такому випадку є непридатним, оскільки таблиця товарів

швидко перетворилася б на сотні стовпців, більшість з яких для конкретного товару були б порожніми (NULL). Тому розглядалися два альтернативні підходи: модель EAV (англ. entity-attribute-value, сутність-атрибут-значення) та модель JSONB.

Таблиця 2.2 – Перелік таблиць бази даних та їх призначення

Таблиця	Призначення	Ключові поля
categories	Ієрархічний каталог категорій	id, slug, parent_id, keywords
brands	Бренди (виробники)	id, name, slug, country
attribute_definitions	Схема характеристик категорії	category_id, slug, data_type, unit
products	Товари каталогу	id, slug, sku, price, stock, specs, search_vector
users	Користувачі системи	id, email, password_hash, bonus_points, is_admin
orders	Замовлення	id, user_id, total, status, payment_status
order_items	Позиції замовлення	order_id, product_id, qty, price
bonus_transactions	Бонусні нарахування і списання	user_id, order_id, amount, kind
returns	Заявки на повернення	user_id, order_id, reason, status
reviews	Відгуки користувачів	product_id, user_id, rating, text
search_queries	Аналітика пошуку	query, results_count, user_id

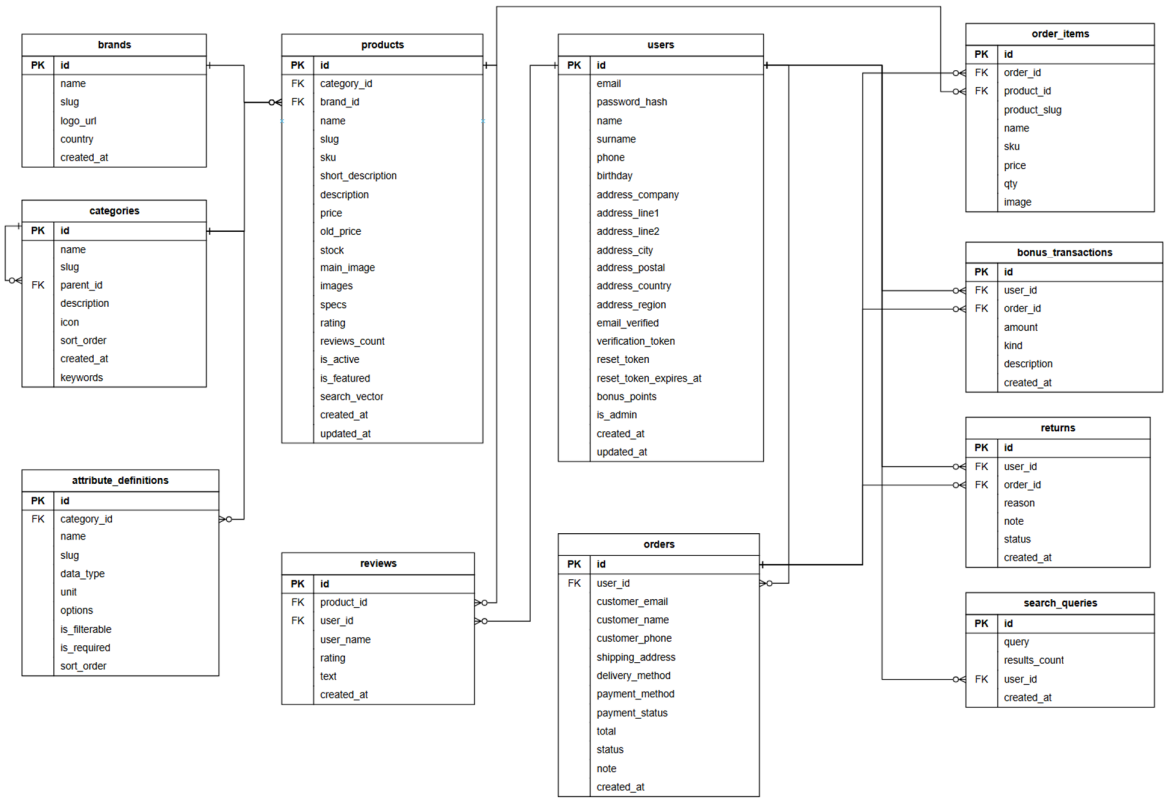

Рисунок 2.6 – ER-діаграма бази даних системи Volt

Модель EAV передбачає створення окремої таблиці product_attributes з полями (product_id, attribute_id, value). Така модель є нормалізованою з реляційного погляду, але має суттєві недоліки на практиці: для отримання повного набору характеристик товару потрібно виконати запит з JOIN до цієї таблиці та поворотом (англ. pivot) рядків у колонки, що ускладнює запити та погіршує продуктивність; типізація значень (число чи рядок) ховається у значенні-рядку, що ускладнює числові фільтри; додавання нового товару потребує множинних INSERT-операцій замість одного.

Модель JSONB передбачає зберігання всіх характеристик у єдиному JSON-полі таблиці products. Це поле підтримує індексацію GIN-індексом, що забезпечує швидкі запити виду «всі товари, у яких specs містить ключ ram зі значенням 16». Запити до окремих характеристик через оператор ->> також прискорюються виразним індексом, якщо такий створено. Порівняння двох підходів за критеріями зручності використання, продуктивності та гнучкості подано у таблиці 2.3.

Таблиця 2.3 – Порівняння моделей EAV і JSONB для зберігання характеристик

Критерій	EAV	JSONB
Кількість таблиць для зберігання характеристик	1 окрема таблиця	немає (вкладено в products)
Кількість запитів для отримання усіх характеристик товару	2 (з JOIN)	1
Складність запиту з фільтром за характеристикою	висока (JOIN + умова)	низька (умова в WHERE)
Типізація значень	слабка (TEXT)	сильна (number / bool / string)
Швидкість пошуку з фільтрами	помірна	висока (GIN-індекс)
Можливість динамічної схеми	так	так
Підтримка операцій оновлення окремих ключів	природна	так (через jsonb_set)
Контроль допустимих ключів	на рівні даних	програмний (за словником)

За підсумками порівняння для проєкту обрано модель JSONB. При цьому для забезпечення контрольованості схеми введено допоміжну таблицю attribute_definitions, у якій для кожної категорії декларуються допустимі характеристики, їх тип (number, string, boolean, enum), одиниця виміру та ознака, чи бере участь ця характеристика у формуванні фільтрів. Така комбінація поєднує

переваги обох підходів: швидкість і простоту запитів JSONB та контрольованість і документованість EAV. У програмному коді функція buildProductFilters приймає масив визначень атрибутів і використовує його як «біла список» допустимих ключів, що захищає від ін’єкцій довільних ключів через параметри URL.

Для прискорення пошуку та фільтрації над таблицею products створено сім індексів: B-tree індекси над зовнішніми ключами category_id і brand_id (для прискорення JOIN-операцій), над полем price (для діапазонних запитів і сортування), над прапором is_active (для відсіювання вимкнених товарів), над полем slug (для пошуку за URL), GIN-індекс над полем specs (для запитів за характеристиками) та GIN-індекс над полем search_vector (для повнотекстового пошуку). Додатково розширення pg_trgm використано для побудови індексів gin_trgm_ops над назвами товарів, брендів і категорій – це прискорює обчислення триграмної подібності у функціях similarity() і за оператором %.

Автоматичне оновлення пошукового вектора при додаванні чи редагуванні товару забезпечує тригер trg_products_search_vector. Він спрацьовує BEFORE INSERT OR UPDATE OF name, short_description, description на таблиці products і викликає функцію products_search_vector_update, яка обчислює новий вектор як зважений конкатенат трьох tsvector-представлень: назви (вага A), короткого опису (вага B), повного опису (вага C). Такий підхід виключає можливість виникнення неузгодженості між фактичним вмістом полів і значенням search_vector – оновлення відбувається атомарно у межах самої транзакції зміни товару.

Для забезпечення зручної ідентифікації товарів адміністратором у системі реалізовано автоматичну генерацію артикула (SKU). Створена послідовність product_sku_seq починається з числа 7180001 і збільшується на одиницю при кожному додаванні товару, формуючи унікальний числовий код. Послідовність – а не autoincrement-стовпець – обрана тому, що вона дозволяє отримати наступне значення без фактичного створення запису, що корисно у формі додавання товару (наприклад, для попереднього відображення SKU у інтерфейсі).

Для забезпечення цілісності даних на рівні БД оголошено систему обмежень. Зокрема, обмеження CHECK гарантують, що ціна і обсяг залишку не можуть бути від’ємними, рейтинг товару належить діапазону [0; 5], оцінка у відгуку – діапазону

[1; 5]; обмеження UNIQUE на парі (product_id, user_id) у таблиці reviews не дозволяє одному користувачу залишити декілька відгуків про той самий товар. Зовнішні ключі з політикою ON DELETE CASCADE забезпечують автоматичне видалення позицій замовлення при видаленні самого замовлення, а політика ON DELETE SET NULL для зв’язку user_id у reviews – збереження відгуку у разі видалення акаунта.

Спроєктоване інформаційне забезпечення поєднує переваги реляційної моделі (цілісність даних, оптимізатор запитів, підтримка транзакцій) з гнучкістю документної моделі (динамічна схема характеристик) і забезпечує всі необхідні передумови для реалізації функціональних вимог, сформульованих у підрозділі 1.3. Деталі програмної реалізації архітектури, алгоритмів та моделі даних, а також результати тестування розглянуто у розділі 3.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Програмна реалізація серверної та клієнтської частин

Згідно з архітектурним рішенням, описаним у підрозділі 2.1, систему реалізовано у вигляді двох незалежних, але пов’язаних API-контрактом програмних модулів: серверного, що працює на платформі Node.js, та клієнтського, що виконується у браузері користувача. Цей підрозділ описує внутрішню структуру кожного з модулів, обґрунтовує вибір використаних бібліотек і наводить особливості реалізації найважливіших алгоритмічних рішень.

Серверна частина має модульну організацію, що відображає функціональні групи з підрозділу 2.1. Точкою входу є файл server.js, який налаштовує веб-сервер Express, реєструє ланцюжок проміжного програмного забезпечення (helmet, cors, express.json, morgan, multer), монтує групи маршрутів за відповідними префіксами та запускає прослуховування TCP-порту. Кожна група маршрутів виокремлена в окремий файл у каталозі routes/. Модуль db/ містить пул з’єднань з PostgreSQL на основі офіційного драйвера pg, обгортку query() та схеми міграцій; модуль middleware/ – реалізації проміжних обробників для автентифікації та контролю прав; модуль services/ – допоміжні служби, зокрема надсилання пошти через nodemailer. Перелік основних модулів та їх призначення наведено у таблиці 3.1.

Найскладнішим за обсягом і логікою є модуль routes/products.js, у якому реалізовано всі алгоритми, описані у підрозділі 2.2. Він містить функцію resolveCategoriesByText, що виконує дворівневе зіставлення запиту з категоріями (сильний підрядковий збіг → нечіткий триграмний fallback); функцію buildProductFilters, яка перетворює параметри URL у фрагмент WHERE-частини SQL-запиту з контролем допустимих ключів через словник attribute_definitions; обробник GET /api/products, який поєднує текстовий пошук, фільтри, сортування та пагінацію; обробник GET /api/products/facets/:catSlug, який реалізує правило «виключити власний вимір» (2.3); обробник GET /api/products/suggest, що формує підказки автодоповнення. Особлива увага приділена захисту від SQL-ін’єкцій: усі значення передаються до драйвера pg як параметри ($1, $2, …), а назви ключів

JSONB звіряються з білим списком зі словника атрибутів – інтерпольовані у текст запиту значення з URL заборонені.

Таблиця 3.1 – Структура серверних модулів системи

Модуль	Опис
src/server.js	Точка входу: налаштування Express, реєстрація маршрутів
src/db/index.js	Пул з'єднань з PostgreSQL і обгортка query()
src/db/schema.sql	DDL-скрипт створення таблиць, індексів і тригерів
src/db/seed.js	Скрипт наповнення БД тестовими даними
src/routes/products.js	Пошук, фільтрація, автодоповнення, критерії, деталі товару
src/routes/categories.js	Каталог категорій
src/routes/brands.js	Каталог брендів
src/routes/auth.js	Реєстрація, автентифікація, верифікація, відновлення паролю
src/routes/orders.js	Створення і перегляд замовлень
src/routes/reviews.js	Відгуки про товари
src/routes/returns.js	Заявки на повернення товарів
src/routes/bonuses.js	Бонусний баланс і історія транзакцій
src/routes/payments.js	Інтеграція з платіжним сервісом LiqPay
src/routes/admin.js	Адміністративні операції з товарами, замовленнями та користувачами
src/middleware/auth.js	Перевірка JWT-токенів і завантаження даних користувача
src/services/email.js	Надсилання транзакційних листів через SMTP

Модуль routes/auth.js реалізує всі функції роботи з користувачем. Реєстрація створює запис у таблиці users з паролем, захешованим бібліотекою bcryptjs з фактором складності 10; одночасно генерується криптографічно стійкий verification_token, який надсилається на вказану адресу у листі підтвердження. Автентифікація перевіряє пароль через bcrypt.compare і повертає JWT-токен, підписаний секретом сервера, з терміном дії 7 діб і з ідентифікатором користувача та прапором is_admin у корисному навантаженні. Скидання паролю реалізовано за класичним патерном двофакторного підтвердження: при запиті користувача система генерує одноразовий токен з обмеженим терміном дії та надсилає його на email; до моменту вводу нового паролю токен не діє.

Інтеграція з платіжним сервісом LiqPay реалізована в модулі routes/payments.js за схемою перенаправлення з підписом. Сервер формує JSON-структуру з даними платежу (номер замовлення, сума, призначення), кодує її у base64 та обчислює HMAC-SHA1-підпис із використанням приватного ключа продавця. Утворена пара data/signature передається клієнту, який відправляє її у формі POST на сторінку LiqPay; сама оплата відбувається на стороні платіжного сервісу, що знімає з системи відповідальність за обробку даних банківських карт. Після завершення оплати LiqPay звертається до зворотного маршруту /api/payments/liqpay/callback, сервер перевіряє автентичність підпису та оновлює статус замовлення на «paid».

Клієнтська частина – це SPA на основі React 18 з компонентною архітектурою та маршрутизацією react-router-dom. Структура клієнтського коду розділена на три основні шари: сторінки (каталог pages/), компоненти багаторазового використання (каталог components/) та глобальний стан застосунку (каталог context/). Перелік основних сторінок наведено у таблиці 3.2.

Таблиця 3.2 – Перелік сторінок клієнтської частини

Сторінка	Маршрут	Призначення
HomePage	/	Головна: акції, популярні товари, переходи у категорії
CatalogLandingPage	/catalog	Сітка категорій верхнього рівня
CatalogPage	/catalog/:category	Каталог категорії з пошуком і фільтрами
ProductPage	/product/:slug	Деталі товару, галерея, характеристики, відгуки
CartPage	/cart	Кошик і оформлення замовлення
FavoritesPage	/favorites	Список обраних товарів
AccountPage	/account	Особистий кабінет (профіль, історія, бонуси)
AdminPage	/admin	Адміністративна панель
VerifyEmailPage	/verify-email	Підтвердження email за токеном
ResetPasswordPage	/reset-password	Скидання паролю за токеном
InfoPage	/info/:topic	Інформаційні сторінки (доставка, оплата тощо)
NotFoundPage	*	Сторінка 404

Для керування глобальним станом застосунку, який має бути доступним з різних сторінок, використано механізм React Context. Замість сторонньої бібліотеки керування станом (Redux, MobX, Zustand) обрано вбудований у React механізм Context API: для проєкту з помірною складністю стану він є достатнім, не

вимагає додаткових залежностей та шаблонного коду й має простішу криву навчання. Чотири контексти забезпечують такі функції: AuthContext зберігає поточного користувача та токен, перевіряє валідність токена при завантаженні застосунку, надає функції входу та виходу; CartContext керує вмістом кошика, синхронізує його з localStorage для збереження між сесіями; FavoritesContext управляє списком обраних товарів аналогічно кошику; FiltersContext зберігає поточно застосовані фільтри і пошуковий запит, забезпечує синхронізацію з рядком запиту URL – це робить стан фільтрів поділюваним посиланнями (англ. shareable).

Окремо реалізовано власний хук useDebounce, який затримує оновлення значення на заданий інтервал часу. Він застосовується у компоненті SearchAutocomplete для затримки виклику API-маршруту /api/products/suggest до 250 мс після останнього натискання клавіші. Без такого затримання при кожному натисканні відбувався б окремий мережевий запит, що перевантажувало б сервер і призводило до візуального «миготіння» списку підказок. Хук побудовано на стандартних React-хуках useState і useEffect; його реалізація займає лише 10 рядків і не потребує сторонніх бібліотек.

Для стилізації інтерфейсу використано фреймворк Tailwind CSS 3.4. На відміну від класичних підходів з компонентними бібліотеками (Material-UI, Ant Design, Bootstrap), де готові компоненти нав’язують свій візуальний стиль, Tailwind пропонує набір атомарних утилітарних класів (наприклад, flex, items-center, gap-4, text-lg, font-semibold, rounded-lg, bg-white, shadow-sm), які комбінуються безпосередньо у розмітці. Цей підхід забезпечує повний контроль над візуальним рішенням, мінімізує обсяг готового CSS (Tailwind у режимі продакшен-збірки видаляє невикористані класи) і чудово поєднується з компонентним підходом React: візуальний шар стає частиною компонента, а не зовнішнім файлом.

Іконковий набір реалізовано бібліотекою lucide-react – набором з близько тисячі SVG-іконок, які підключаються як окремі React-компоненти. На відміну від іконкових шрифтів, SVG-іконки масштабуються без втрати якості, успадковують поточний колір тексту та не вимагають завантаження окремого файла шрифту.

Завдяки tree-shaking у збірнику Vite до фінального бандла потрапляють лише ті іконки, які фактично використовуються у застосунку.

Клієнтський застосунок реалізовано як односторінковий додаток (SPA) на бібліотеці React із компонентною архітектурою, що відокремлює сторінки-маршрути від повторно використовуваних елементів інтерфейсу та від глобального стану. Кореневий компонент App виконує маршрутизацію засобами react-router і обгортає застосунок ланцюжком провайдерів стану AuthContext → CartContext → FavoritesContext → FiltersContext, завдяки чому дані про користувача, кошик, закладки й активні фільтри доступні будь-якому вкладеному компоненту без передавання через властивості. Постійно змонтованими лишаються шапка (Header) і підвал (Footer), а також висувна панель кошика (CartDrawer) та модальне вікно входу (LoginModal), що відображаються поверх поточної сторінки. Власне відображення поділено на дванадцять сторінок (HomePage, CatalogPage, ProductPage, CartPage, AccountPage, AdminPage та інші), які складаються із сімнадцяти компонентів — зокрема рядка пошуку з автодоповненням (SearchAutocomplete), бічної панелі фільтрів (FiltersSidebar), сітки й картки товару (ProductGrid, ProductCard), пагінації, галереї зображень (ProductGallery) та блоку відгуків (Reviews). Глобальний стан зосереджено у чотирьох контекстах, а допоміжний рівень утворюють хук useDebounce для відкладання частих запитів під час уведення, модуль доступу до API (api/client.js) поверх браузерного інтерфейсу fetch із автоматичним додаванням JWT-токена та набір утиліт форматування. Загальну організацію клієнтської частини — сторінки, компоненти та контексти — показано на рисунку 3.1.

Для зв’язку з API сервера обрано стандартний браузерний інтерфейс fetch, без додаткових бібліотек (наприклад, axios). Він є нативною частиною середовища, повертає Promise, підтримує всі необхідні можливості (заголовки, тіло, методи, скасування через AbortController) і не вимагає окремої збірки. Усі функції API згруповано у каталозі src/api/, що дозволяє при необхідності централізовано додати, наприклад, перехоплення 401-відповіді з автоматичним перенаправленням на сторінку входу або глобальне додавання JWT-токена до всіх захищених запитів.

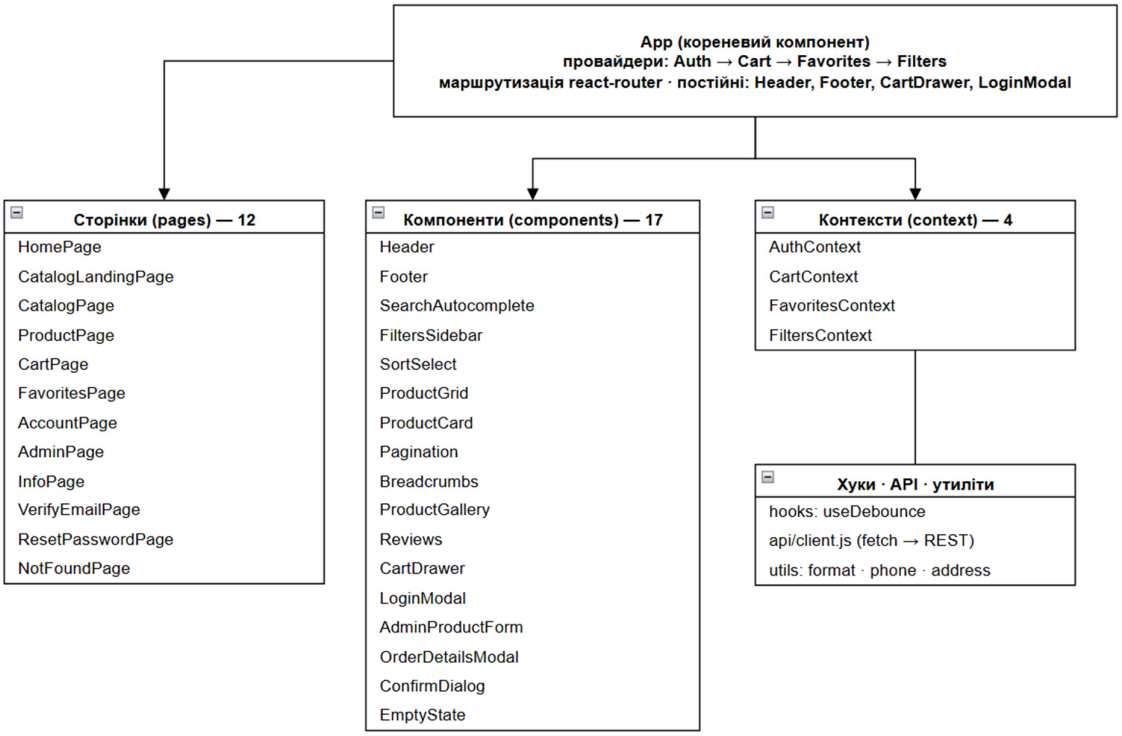

Рисунок 3.1 – Структура клієнтського застосунку: сторінки, компоненти, контексти

Збірка клієнтської частини виконується інструментом Vite 5. На відміну від класичного webpack, Vite використовує нативні ES-модулі у режимі розробки, що дозволяє запустити сервер розробки за частки секунди незалежно від розміру проєкту; для продакшен-збірки застосовується rollup, який створює оптимізований бандл з розбиттям на динамічні шматки (code splitting), мініфікацією та фінгерпринтингом імен файлів. Час «гарячого» оновлення модуля у режимі розробки (HMR, hot module replacement) для вебзастосунку Volt становить менше 50 мс, що забезпечує комфортний цикл розробки.

3.2 Інтерфейс користувача та сценарії взаємодії

Інтерфейс користувача побудовано за принципами сучасних marketplaces з акцентом на швидкому знаходженні товару, інформативності та доступності для

широкого кола користувачів. Цей підрозділ описує ключові сценарії взаємодії з основними сторінками застосунку.

Головна сторінка (рисунок 3.2) виконує функцію точки входу та орієнтиру для відвідувача. У її верхній частині розташований шапковий блок із логотипом, головним пошуковим полем, піктограмами обраного, кошика та особистого кабінету. Нижче розміщено основний контент: блок із промо-акціями та банерами, сітка категорій верхнього рівня (з піктограмами та кількістю товарів у кожній), стрічки «Хіти продажів» і «Нові надходження». У нижній частині розташовано підвал з посиланнями на інформаційні сторінки (доставка, оплата, повернення, контакти).

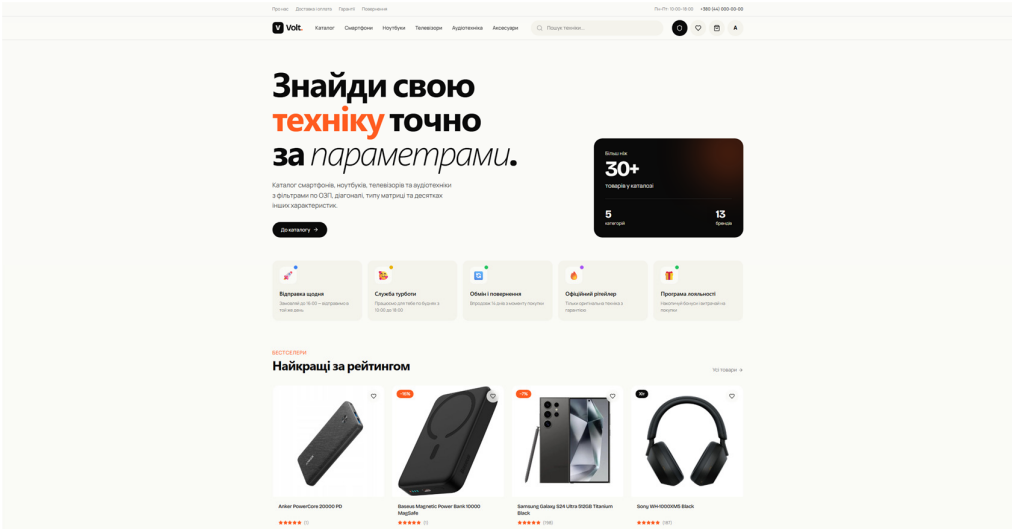

Рисунок 3.2 – Головна сторінка вебзастосунку

Сторінка каталогу (рисунок 3.3) є центральним інструментом пошуку товарів. Її макет поділено на дві основні зони: ліву бічну панель фільтрів (FiltersSidebar) та основну область відображення товарів (ProductGrid). Над сіткою товарів розташовано рядок пошуку, селектор сортування (за ціною, рейтингом, новизною, популярністю) та пагінатор. У бічній панелі групи фільтрів відображаються згідно з налаштуваннями у attribute_definitions: на початку – бренди (з лічильниками товарів), потім ціна як повзунок діапазону, далі – характеристики, специфічні для категорії. Для числових характеристик відображається повзунок з двома маркерами, для перелічуваних – список значень з

прапорцями та лічильниками. Лічильники оновлюються в режимі реального часу: при виборі будь-якого значення клієнт виконує запит до /api/products/facets і отримує оновлений набір лічильників, обчислений за правилом «виключити власний вимір» (2.3).

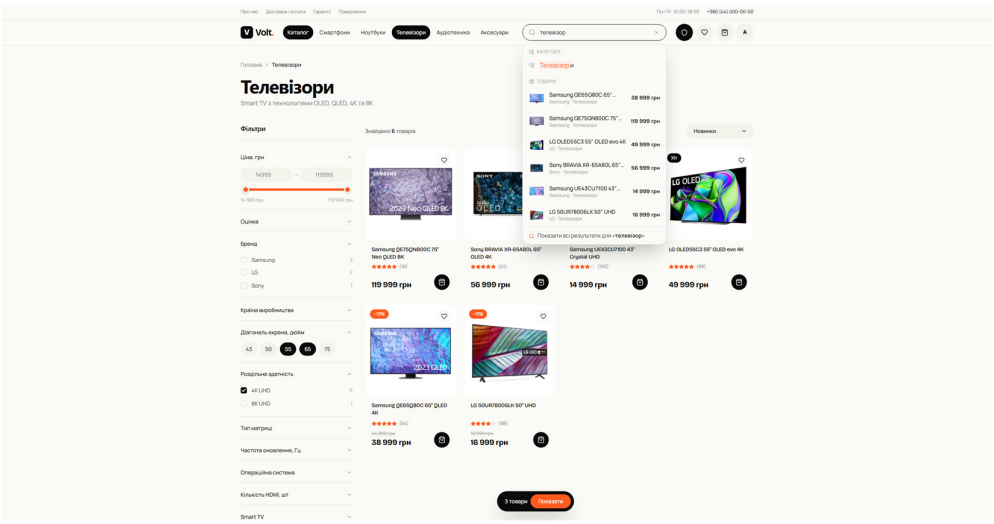

Рисунок 3.3 – Сторінка каталогу з пошуком, фільтрами та сіткою товарів

Рядок пошуку оснащено компонентом автодоповнення (SearchAutocomplete). Після введення перших символів і затримки 250 мс (реалізованої через хук useDebounce) клієнт звертається до маршруту /api/products/suggest. Підказки розгортаються у випадаючий список з групуванням за типом: окремо товари (з мініатюрою, ціною), категорії, бренди. Користувач може обрати елемент мишею або клавіатурою (стрілки ↑/↓, Enter), після чого здійснюється перехід відповідно до типу – на сторінку товару, у каталог категорії або у каталог товарів обраного бренда.

Сторінка товару (рисунки 3.4 - 3.5) деталізовано представляє конкретну позицію каталогу. Зверху ліворуч розташована галерея зображень (ProductGallery) із головним кадром і стрічкою мініатюр; зверху праворуч – інформаційний блок із назвою, артикулом, рейтингом, ціною, кнопками «У кошик» та «До обраного». Нижче, у вкладках, відображаються характеристики (повний перелік ключ-значення з поля specs з підстановкою назв і одиниць виміру з attribute_definitions), детальний опис і відгуки. Розділ відгуків (Reviews) показує середній рейтинг,

гістограму розподілу оцінок та список відгуків з можливістю додавання власного для автентифікованих користувачів, які ще не залишали відгук про цей товар (відповідне обмеження UNIQUE забезпечує його контроль на рівні бази даних).

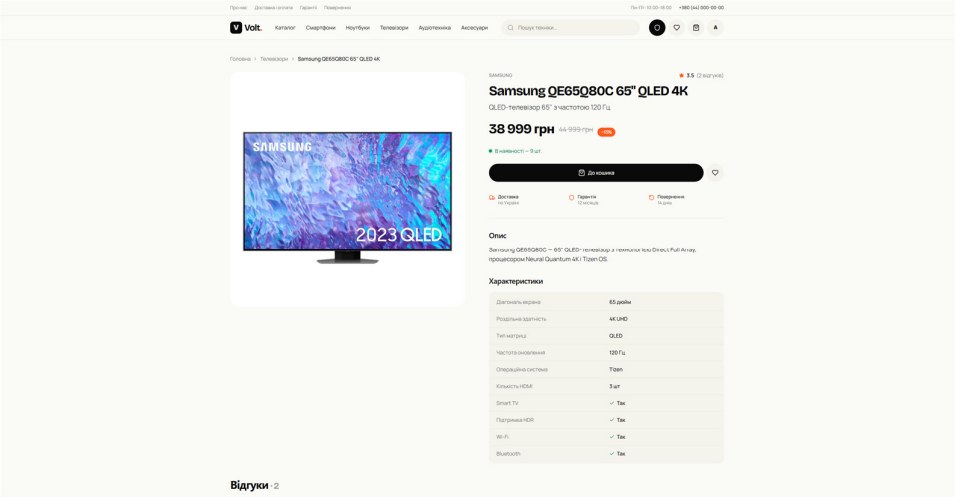

Рисунок 3.4 – Сторінка товару з галереєю, характеристиками і відгуками

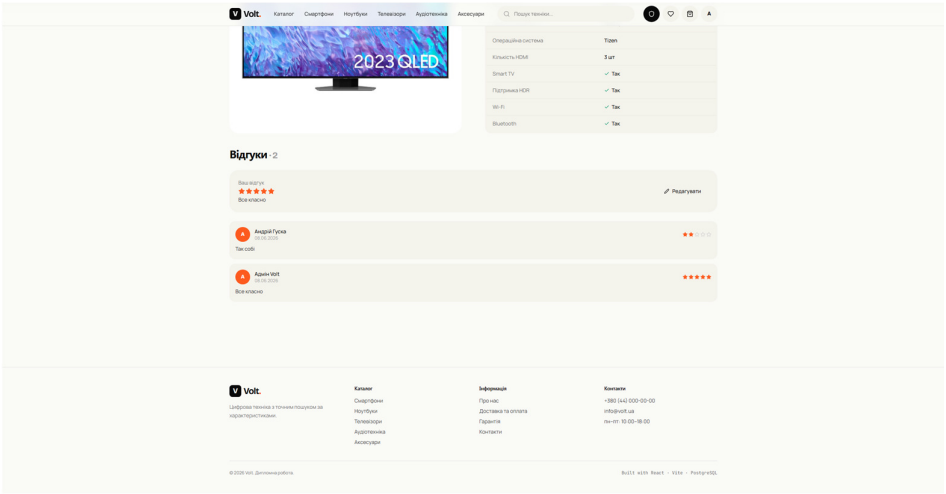

Рисунок 3.5 – Відгуки на сторінці товару

Кошик (рисунки 3.6 - 3.7) існує у двох формах: компактній (CartDrawer – висувна панель з переліком товарів, доступна з шапки на будь-якій сторінці) та повноекранній (CartPage – сторінка оформлення замовлення). На сторінці оформлення користувач переглядає вміст кошика з можливістю зміни кількості або видалення позицій, обирає спосіб доставки і оплати, для автентифікованого користувача дані доставки автоматично підставляються з профілю. Для гостьових

замовлень обов’язково заповнюються email, ім’я та телефон. Після підтвердження клієнт викликає POST /api/orders; сервер створює запис у таблицях orders та order_items в одній транзакції, для автентифікованого користувача нараховує бонусні бали (3% від суми), за необхідності перенаправляє на сторінку оплати LiqPay.

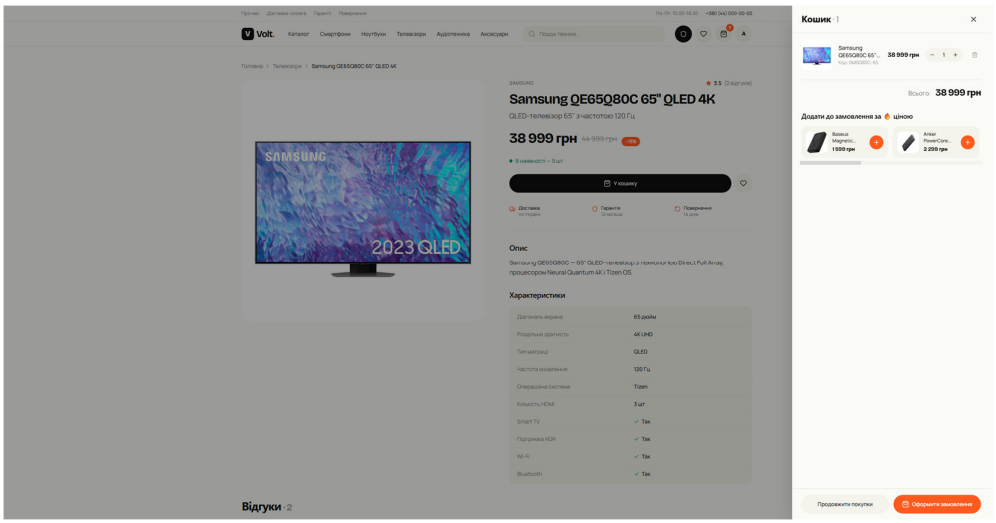

Рисунок 3.6 – Висувна панель кошику

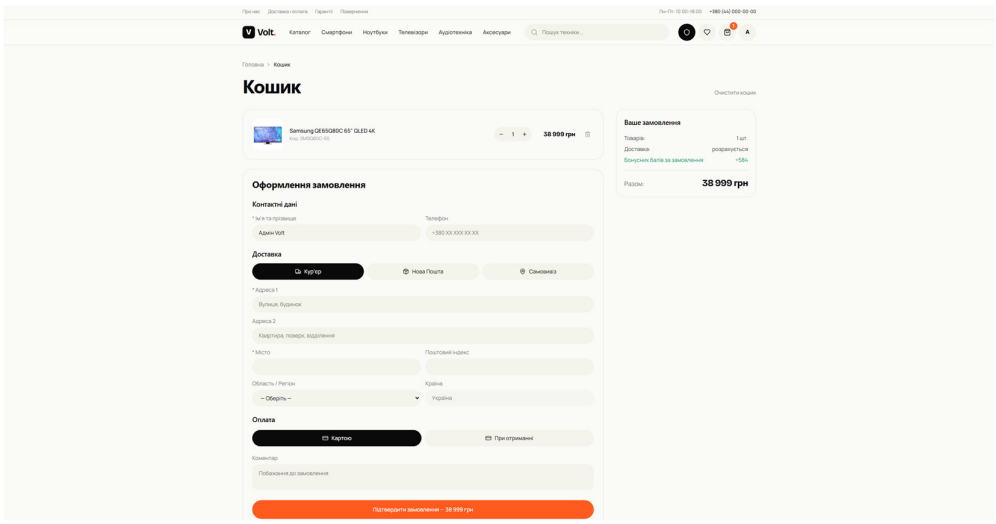

Рисунок 3.7 – Сторінка оформлення замовлення

Особистий кабінет (рисунок 3.8) розділений на вкладки: загальна інформація профілю з можливістю редагування, історія замовлень з деталями кожного, бонусний баланс і історія транзакцій, форми зміни паролю та подання заявки на

повернення. Сторінка доступна лише для автентифікованих користувачів – при спробі звернутися без токена користувач перенаправляється на головну сторінку з відкритим модальним вікном входу.

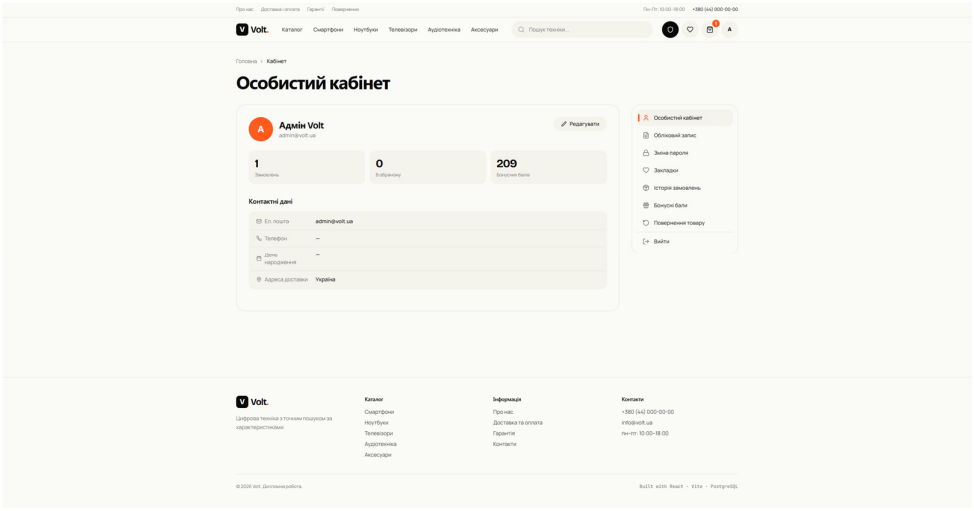

Рисунок 3.8 – Особистий кабінет користувача

Адміністративна панель (рисунки 3.9 - 3.11) є окремим розділом для користувачів з прапором is_admin. Вона забезпечує управління каталогом (додавання, редагування та видалення товарів, категорій і брендів), управління замовленнями (зміна статусу, перегляд деталей), управління користувачами (надання та зняття адміністративних прав), модерацію відгуків і оброблення заявок на повернення. Окремий розділ присвячено аналітиці пошуку: він показує найпопулярніші пошукові запити за останні 30 днів та запити з нульовою видачею – це дозволяє адміністратору вчасно виявити прогалини у каталозі (товари, які користувачі шукають, але не знаходять) та реагувати на них.

Уся клієнтська частина побудована як адаптивна: за допомогою утилітарних класів Tailwind задано три основні точки переходу (sm: 640 px, md: 768 px, lg: 1024 px), які перебудовують макет залежно від ширини екрана. На пристроях з шириною < 768 px бічна панель фільтрів згортається у модальне вікно, що відкривається кнопкою, а сітка товарів переходить з трьох колонок у одну. Усі інтерактивні елементи мають достатній розмір (не менш як 44 × 44 px) для зручного тапу на сенсорних екранах.

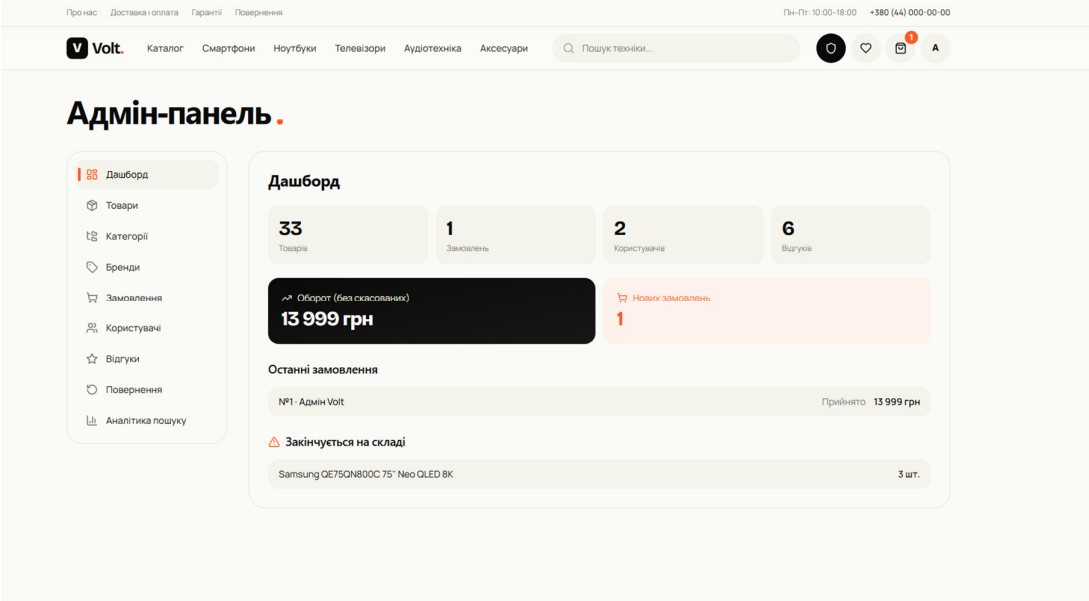

Рисунок 3.9 – Адміністративна панель: дашборд

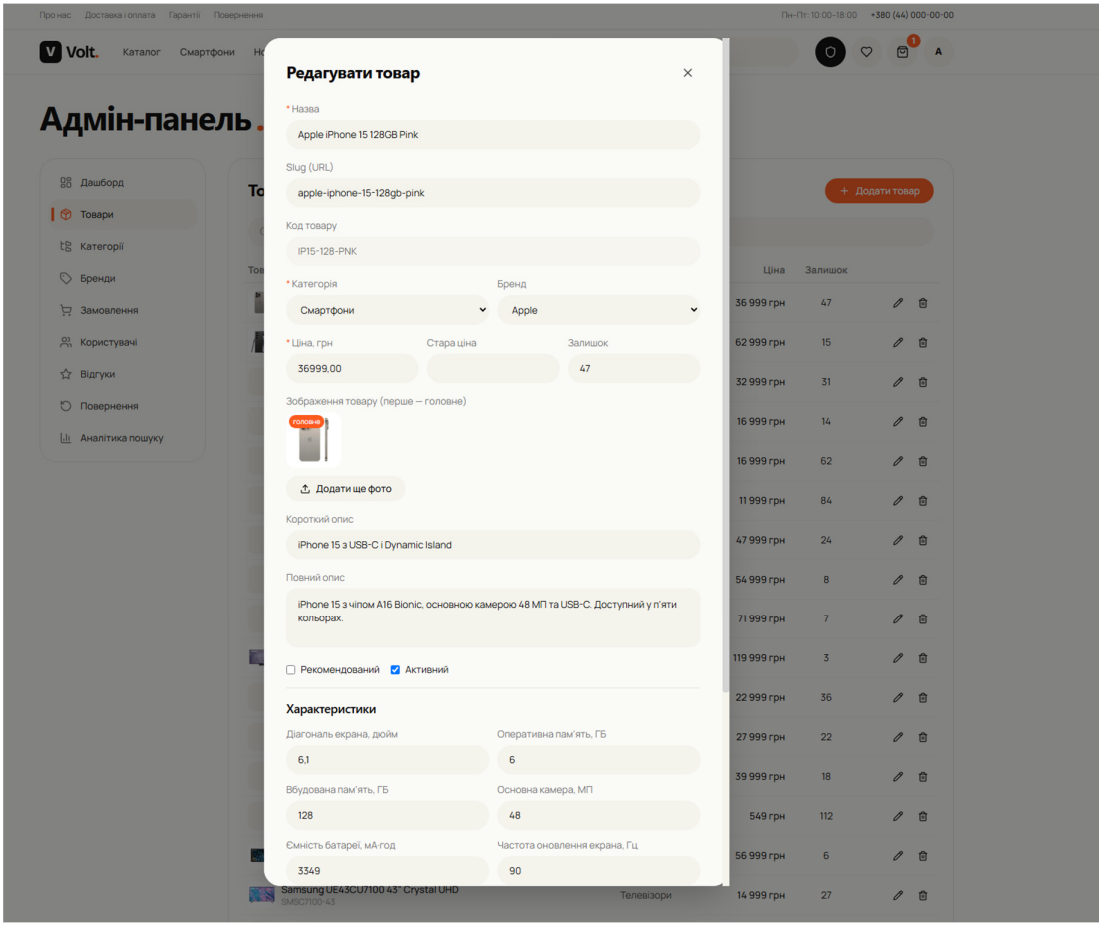

Рисунок 3.10 – Адміністративна панель: керування товаром

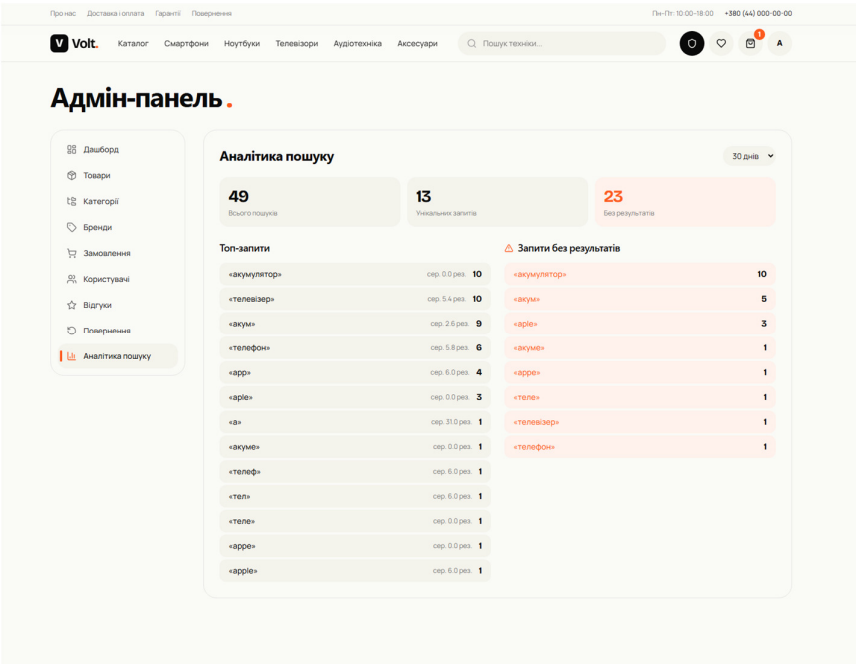

Рисунок 3.11 – Адміністративна панель: аналітика пошуку

3.3 Тестування системи та оцінка результатів

Розроблену систему піддано комплексному тестуванню, що охоплює функціональну перевірку ключових сценаріїв роботи, оцінку коректності алгоритму пошуку з різними типами запитів, вимірювання швидкодії та чутливості інтерфейсу, а також перевірку крос-браузерної сумісності клієнтської частини. Тестування виконувалося у браузері Google Chrome з використанням вбудованих інструментів розробника (вкладка Performance) та інструмента аудиту Google Lighthouse; швидкодію інтерфейсу оцінювали за галузевими метриками Core Web Vitals.

Функціональне тестування полягало у послідовному проходженні всіх типових сценаріїв взаємодії з системою. Усі сценарії пройшли успішно. Підсумкові результати наведено у таблиці 3.3.

Тестування алгоритму пошуку проводилося окремо – на наборі з типових пошукових запитів з різним рівнем коректності написання. Запити включали повні правильні назви, скорочені форми, синоніми, запити з одруками 1–3 літер. Результати наведено у таблиці 3.4.

Таблиця 3.3 – Результати функціонального тестування

Сценарій	Очікуваний результат	Статус
Реєстрація нового користувача	Запис у БД, лист підтвердження, токен	✓
Підтвердження email за токеном	email_verified = TRUE	✓
Вхід коректним паролем	Видано валідний JWT	✓
Вхід неправильним паролем	HTTP 401, інформативне повідомлення	✓
Скидання паролю через email	Лист з токеном, валідне скидання	✓
Пошук за повним збігом	Точна відповідь у топ-3 результатів	✓
Пошук з одруком (1–2 літери)	Коректний результат через триграми	✓
Застосування фільтра ціни	Видача обмежена діапазоном	✓
Перемикання значень критерію	Лічильники оновлюються коректно	✓
Додавання товару в кошик	Стан кошика оновлено, збережено у localStorage	✓
Оформлення гостьового замовлення	Запис у orders і order_items	✓
Нарахування бонусів (3% від суми)	Запис у bonus_transactions, оновлення балансу	✓
Залишення відгуку	Запис у reviews, оновлення рейтингу товару	✓
Повторний відгук тим самим користувачем	Відхилено (UNIQUE-обмеження)	✓
Заявка на повернення	Запис у returns зі статусом pending	✓
Доступ до /api/admin/* без is_admin	HTTP 403	✓
Спроба SQL-ін’єкції у spec_-параметрі	Запит відхилено (контроль за словником)	✓

Таблиця 3.4 – Результати тестування пошукового алгоритму

Запит	Тип	Знайдено	Перший результат
смартфон	точна назва категорії	коректно	Категорія Смартфони
телефон	синонім (через keywords)	коректно	Категорія Смартфони
телевізер	одрук у назві категорії	коректно (триграми)	Категорія Телевізори
айфон	транслітерація бренду	коректно (через keywords)	Apple iPhone (модель)
samsung galaxy	багатослівний запит	коректно (FTS)	Samsung Galaxy S24
sumsung	одрук у назві бренду	коректно (триграми)	Samsung
ноут i5	змішаний (укр+лат)	коректно	Ноутбуки з процесорами Intel Core i5
xxxx123	відсутній товар	0 результатів	Запропоновано варіант запиту

У всіх випадках, де очікувався збіг, система знаходила правильний результат у перших позиціях видачі. Для запитів, які не відповідають жодному товару,

активується механізм «Можливо, ви мали на увазі», що пропонує найближчий за триграмною подібністю варіант з каталогу.

Швидкодію та чутливість інтерфейсу оцінювали за метриками Core Web Vitals: INP (Interaction to Next Paint) — час реакції інтерфейсу на дію користувача та CLS (Cumulative Layout Shift) — показник візуальної стабільності розмітки. Для основних сценаріїв взаємодії отримано результати, наведені у таблиці 3.5.

Таблиця 3.5 – Результати вимірювання швидкодії інтерфейсу

Сценарій взаємодії	INP, мс	CLS
Відкриття головної сторінки	88	0
Перехід у каталог категорії	112	0,01
Застосування багатокритеріального фільтра	134	0
Перемикання значення критерію	96	0
Введення запиту з автодоповненням	145	0,02
Відкриття сторінки товару	104	0,01
Додавання товару в кошик	72	0
Оформлення замовлення	156	0,01

Усі виміряні значення INP не перевищують 200 мс, що відповідає рекомендованому діапазону «добре» для цієї метрики; показник CLS у всіх сценаріях близький до нуля (менше 0,1), що свідчить про відсутність помітних зсувів розмітки під час завантаження та взаємодії. Отже, інтерфейс реагує на дії користувача практично миттєво й залишається візуально стабільним.

Висока чутливість інтерфейсу значною мірою досягається завдяки обраній моделі зберігання характеристик у форматі JSONB з індексами GIN (обґрунтування наведено у підрозділі 2.3): індексований доступ до полів JSONB дозволяє виконувати запити багатокритеріальної фільтрації без сповільнення навіть за зростання обсягу каталогу, що узгоджується з опублікованими дослідженнями застосування JSONB у каталогах електронної комерції [11, 23].

Крос-браузерну сумісність клієнтської частини перевірено у браузерах Google Chrome, Mozilla Firefox, Microsoft Edge та Apple Safari у двох останніх стабільних версіях кожного. У всіх перевірених браузерах застосунок функціонує

без візуальних спотворень і функціональних збоїв. Адаптивність перевірено в режимі емуляції пристроїв з шириною екрана від 360 до 1920 пікселів — макет коректно перебудовується на всіх перевірених розмірах. Загальну якість сторінок оцінено інструментом Google Lighthouse за критеріями Performance, Accessibility, Best Practices та SEO; детальні результати за кожною сторінкою наведено в додатку Г.

Підсумовуючи, реалізована система Volt відповідає всім сформульованим у підрозділі 1.3 функціональним і нефункціональним вимогам. Архітектурні та алгоритмічні рішення, описані у розділі 2, на практиці забезпечують високу швидкість відгуку, точність пошуку з виправленням типових помилок користувача та можливість додавання нових категорій товарів з довільним набором характеристик без зміни схеми бази даних і програмного коду.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

1. Проаналізовано предметну область електронної комерції цифровою технікою, виявлено ключові виклики каталогізації товарів з різнорідними наборами технічних характеристик і сформульовано вимоги до системи пошуку та фільтрації. Встановлено, що традиційні підходи до зберігання характеристик у виділених стовпцях або в EAV-моделі не забезпечують достатньої гнучкості і швидкості, що зумовлює потребу в гібридній моделі.

2. Виконано порівняльний аналіз шести існуючих рішень – як готових платформ (Rozetka, Foxtrot, Amazon), так і платформ розробки (Magento, Shopify, WooCommerce). Зроблено висновок про доцільність розробки власного рішення на основі модульного open-source-стеку, що поєднує переваги масштабованості, гнучкості і відсутності залежності від комерційного постачальника.

3. Спроєктовано трирівневу архітектуру системи на основі React (клієнтський рівень), Node.js з фреймворком Express (рівень логіки) та PostgreSQL (рівень даних). Розроблено REST API з 49 кінцевими точками, об'єднаними у 10 функціональних груп. Архітектурне рішення забезпечує безстановий характер серверної частини, що допускає горизонтальне масштабування.

4. Розроблено комбінований трирівневий алгоритм текстового пошуку, що послідовно застосовує повнотекстовий пошук засобами tsvector, нечіткий пошук на основі триграм і підрядковий пошук. Реалізовано дворівневий алгоритм зіставлення пошукового запиту з категоріями каталогу, що використовує поле keywords з синонімами та забезпечує коректну роботу за наявності одруків у запиті.

5. Розроблено алгоритм багатокритеріальної фільтрації з правилом «виключити власний вимір» (формула 2.3), що забезпечує коректне обчислення лічильників для всіх можливих значень критеріїв у поточній видачі. Алгоритм реалізовано як єдиний параметризований SQL-запит без залежностей від поточного значення фільтра, що гарантує консистентну поведінку.

6. Спроєктовано модель бази даних з 11 таблиць, що забезпечує всі функціональні вимоги системи. Обґрунтовано вибір гібридної моделі — JSONB-

поля для зберігання характеристик у поєднанні з допоміжною таблицею attribute_definitions для контролю схеми, що поєднує гнучкість динамічної схеми з високою швидкодією індексованих запитів багатокритеріальної фільтрації.

7. Реалізовано серверну частину обсягом близько 2000 рядків коду на Node.js + Express з підтримкою автентифікації за JWT, інтеграції з платіжним сервісом LiqPay, надсиланням пошти через SMTP, завантаженням файлів через multer та централізованим обробленням помилок. Реалізовано адміністративну панель з повним функціоналом керування каталогом, замовленнями, користувачами та аналітикою пошуку.

8. Реалізовано клієнтську частину на React 18 з використанням react-router-dom для маршрутизації, Tailwind CSS для стилізації, lucide-react для іконок та Vite для збірки. Глобальний стан застосунку керується через чотири React Context (Auth, Cart, Favorites, Filters); локальний стан компонентів – через стандартні React-хуки. Створено 12 сторінок та 17 компонентів багаторазового використання.

9. Виконано комплексне тестування системи, що включало функціональну перевірку 17 типових сценаріїв, оцінку коректності пошукового алгоритму на 8 типах запитів та вимірювання швидкодії інтерфейсу за метриками Core Web Vitals. Усі сценарії пройшли успішно; пошуковий алгоритм коректно обробив усі тестові випадки, включно із запитами з одруками; за метриками INP і CLS підтверджено високу чутливість і візуальну стабільність інтерфейсу, а аудит Lighthouse показав високі оцінки якості сторінок. Перевірено крос-браузерну сумісність та адаптивність інтерфейсу на пристроях шириною від 360 до 1920 пікселів.

10. Отримані результати свідчать про досягнення поставленої мети: розроблений вебзастосунок Volt є завершеним програмним продуктом, придатним до використання як основа комерційного інтернет-магазину цифрової техніки або адаптації під інші категорії товарів зі складною структурою характеристик; запропоновані архітектурні та алгоритмічні рішення можуть бути застосовані у схожих проєктах електронної комерції.

11. Роботу може бути продовжено у напрямі інтеграції рекомендаційної системи на основі історії переглядів і покупок, додавання функції порівняння товарів за обраними характеристиками, а також підтримки багатомовного інтерфейсу та інтеграції із зовнішніми торговельними майданчиками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронна комерція в Україні: підручник / уклад. А. М. Береза [та ін.]; за заг. ред. А. М. Берези. – Київ: КНЕУ, 2019. – 274 с.
2. Laudon K. C. E-commerce 2023: business, technology, society / K. C. Laudon, C. G. Traver. – 17th edition. – New York: Pearson, 2023. – 768 p.
3. Wei J. The design of a flexible product catalog using EAV vs JSONB models in PostgreSQL / J. Wei, L. Chen // Journal of Web Engineering. – 2022. – Vol. 21, No 3. – P. 411–428.
4. Tunkelang D. Faceted search / D. Tunkelang. – San Rafael: Morgan & Claypool, 2009. – 80 p. – (Synthesis lectures on information concepts, retrieval, and services).
5. Hearst M. Search user interfaces / M. Hearst. – Cambridge: Cambridge University Press, 2009. – 404 p.
6. Rozetka – найбільший інтернет-магазин України [Електронний ресурс]. – Режим доступу: https://rozetka.com.ua .
7. Foxtrot – техніка, гаджети, аксесуари [Електронний ресурс]. – Режим доступу: https://foxtrot.com.ua.
8. Amazon – online shopping [Електронний ресурс]. – Режим доступу: https://www.amazon.com .
9. Adobe Commerce (Magento): documentation [Електронний ресурс]. – Режим доступу: https://experienceleague.adobe.com/docs/commerce.html .
10. PostgreSQL 16 Documentation. Chapter 12: Full Text Search [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/16/textsearch.html .
11. PostgreSQL 16 Documentation. Chapter F.32: pg_trgm – support for similarity of text using trigram matching [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/16/pgtrgm.html .
12. PostgreSQL 16 Documentation. Chapter 8.14: JSON Types [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/16/datatype-json.html .

13. Express.js: web framework for Node.js [Електронний ресурс]. – Режим доступу: https://expressjs.com .

14. Node.js documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/docs .

15. React documentation [Електронний ресурс]. – Режим доступу: https://react.dev .

16. Fielding R. T. Architectural styles and the design of network-based software architectures: PhD dissertation / R. T. Fielding. – Irvine, CA: University of California, 2000. – 162 p.

17. Vite. Next generation frontend tooling [Електронний ресурс]. – Режим доступу: https://vitejs.dev .

18. Tailwind CSS documentation [Електронний ресурс]. – Режим доступу: https://tailwindcss.com .

19. RFC 7519: JSON Web Token (JWT) / M. Jones, J. Bradley, N. Sakimura. – IETF, 2015. – 30 p. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519.

20. Provos N. A future-adaptable password scheme / N. Provos, D. Mazières // Proc. of the USENIX Annual Technical Conference. – Monterey, CA, 1999. – P. 81–92.

21. OWASP Top 10 – 2021 [Електронний ресурс]. – Режим доступу: https://owasp.org/Top10 .

22. Nielsen J. Response Times: The 3 Important Limits / J. Nielsen // Nielsen Norman Group: Articles. – 1993. – Режим доступу: https://www.nngroup.com/articles/response-times-3-important-limits .

23. Brown M. Schema flexibility benchmarks in modern relational databases / M. Brown, A. Petrov // ACM SIGMOD Record. – 2022. – Vol. 51, Issue 4. – P. 23–34.

24. LiqPay API documentation [Електронний ресурс]. – Режим доступу: https://www.liqpay.ua/documentation .

25. Гуска А.А., Загородня Д.І. Вебзастосунок інтернет-магазину цифрової техніки з фасетним пошуком за технічними характеристиками. ІКСМ весна 2026: Інтелектуальні комп'ютерні системи та мережі: збірник тез доповідей IV Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених (Тернопіль, 20 травня 2026 р). Тернопіль: ЗУНУ, 2026. С.198-200.

26. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с

ДОДАТОК А

Копія публікації

Гевко Д. З.
Програмний модуль автоматичного генерування субтитрів та онлайн перекладу відео 157
Мамчур Т. Б.
Підвищення швидкодії інтелектуальних засобів мобільних робототехнічних платформ. 160
Цмоць І.Г., Петрина В.В.
Інформаційні потоки, засоби збору та інтеграції даних у смарт-підприємствах 162
Вдодович Ю., Вівчар В.
Аналіз складних динамічних структур даних у мові програмування С++ 165
Ковбаса Д., Нукало В.
Проєктування та реалізація IoT-пристроїв на базі Raspberry Pi Pico W та мови програмування MicroPython 167
Слюсар М., Франчишин Ю.
Проєктування та реалізація IoT-пристроїв на базі Raspberry Pi Pico W та мови програмування MicroPython 169
Григорів М.-В.В.
Розробка веб-застосунку онлайн-бібліотеки з інтеграцією штучного інтелекту 171
Метлінський Д. А.
Реалізація підсистеми квазіоптимізації структури комп'ютерної мережі та аналіз результатів тестування 173
Яковлєв І.С.
Орієнтована фільтрація Х-променевих зображень 175
Ілащук М.М.
Розпізнавання емоцій на зображеннях облич у відеопотоці за допомогою згорткових нейронних мереж 177
Костюкевич Н.Ю.
Нейромережева система інтерактивного голосового діалогу з музейними експонатами на основі архітектури RAG 179
Батько Ю.М.
Системи моніторингу умов проживання на основі фізичних та хімічних параметрів зовнішнього середовища в структурі «Розумного міста» 182
Івашенюк С.О.
Розробка веб-застосунку для керування проєктними завданнями 185
Наконечний М.В.
Високопродуктивний модуль комп'ютерного зору для потокового відео 187
Калашнюк В.Б.
Дослідження ефективності локальних баз даних у порівнянні з хмарними API для систем аварійних сповіщень розумного будинку 190
Костенюк А.В.
Структура мікроконтролерної системи збору екологічних параметрів 192
Рожанський О.О., Дикунець В.В.
Збір і передавання телеметричних даних у розподілених системах моніторингу 194
Демедюк Т.В.
Програмний модуль адаптивної складності геймплею в 2D-іграх на Unity 196
Гуска А.А.
Вебзастосунок інтернет-магазину цифрової техніки з фасетним пошуком за технічними характеристиками 198

Гуска А.А.
студент 4 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Загородня Д.І., кафедра ІОСУ ЗУНУ

ВЕБЗАСТОСУНОК ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ З ФАСЕТНИМ ПОШУКОМ ЗА ТЕХНІЧНИМИ ХАРАКТЕРИСТИКАМИ

Вступ. У сучасних умовах стрімкого розвитку електронної комерції ефективна навігація в товарних каталогах цифрової техніки набуває критичного значення. Сегмент електроніки характеризується високою розмірністю атрибутивного простору: кожен товар описується десятками унікальних технічних параметрів — діагоналлю та типом матриці екрана, поколінням процесора, обсягом оперативної та постійної пам'яті, ємністю акумулятора, частотою оновлення тощо. Класичний текстовий пошук та ієрархічна навігація за категоріями виявляються неспроможними задовольнити запити користувачів, які прагнуть звузити пропозиції за специфічним набором параметрів. Згідно зі статистичними даними, у 2024–2025 роках понад 78% інтернет-користувачів європейського регіону регулярно купують онлайн, а сектор цифрової електроніки лише у Польщі згенерував 5,86 мільярда доларів США доходу [1]. У таких умовах якість пошукової підсистеми стає ключовим чинником конкурентоспроможності інтернет-магазину.

Стандартом де-факто для електронної комерції з широкими каталогами стали системи фасетного пошуку (англ. faceted search). Вони дозволяють користувачеві паралельно звужувати вибірку за множинними ортогональними критеріями та одночасно агрегують у режимі реального часу кількість товарів за кожним фасетом ще до застосування відповідного фільтра [2]. Реалізація таких систем потребує продуманої архітектури зберігання слабоструктурованих атрибутів, оптимізованих SQL-запитів із мілісекундним відгуком та узгодженого UX/UI. Метою цього дослідження є розробка вебзастосунку інтернет-магазину цифрової техніки з ефективною підсистемою фасетного пошуку та фільтрації за технічними характеристиками.

Постановка задачі. Проведений аналіз існуючих платформ електронної комерції — Magento, Shopify, WooCommerce — показав, що готові SaaS-рішення суттєво обмежують свободу проєктування бази даних і не дозволяють реалізовувати специфічну логіку ранжування та формування фасетів. Magento (Adobe Commerce) надає підтримку складних каталогів через модель EAV (англ. Entity-Attribute-Value), однак ця модель є класичним антипатерном реляційних баз даних: для одночасної фільтрації за n атрибутами вимагається n операцій самоз'єднання, що в каталогах із сотнями тисяч записів призводить до катастрофічної деградації продуктивності. Експериментальні тести демонструють, що типовий запит за перетином атрибутів у моделі JSONB з GIN-індексуванням виконується за 181 мс, тоді як аналогічний запит до структури EAV потребує понад 7000 мс — більш ніж у 39 разів повільніше [3].

Об'єктом дослідження є процес пошуку та фасетної фільтрації товарів в інтернет-магазині цифрової техніки. Предметом дослідження є методи, моделі та програмні засоби, що реалізують фасетний пошук, зберігання динамічних технічних атрибутів та оптимізацію продуктивності SQL-запитів. Метою роботи є проєктування та реалізація вебзастосунку інтернет-магазину цифрової техніки, який забезпечує мілісекундний відгук на запити фасетної фільтрації за довільним набором технічних характеристик.

Для досягнення поставленої мети необхідно розв'язати такі задачі: спроєктувати реляційну схему бази даних із гібридним зберіганням типізованих полів та слабоструктурованих характеристик у форматі JSONB; реалізувати RESTful API з ендпоінтами фасетної агрегації; розробити клієнтську частину з динамічною панеллю фільтрів, чутливою до обраної категорії; забезпечити коректне обчислення лічильників фасетів, при якому вибір значення в одному вимірі не приховує інші доступні значення в тому ж вимірі.

Основний матеріал. Архітектурно розроблений вебзастосунок Volt побудовано за класичною триланковою схемою (рисунок 1). Клієнтська частина реалізована на React 18 із

застосуванням Vite як інструменту збирання, Tailwind CSS — для стилізації, React Router — для маршрутизації. Серверна частина побудована на Node.js і фреймворку Express, який обслуговує RESTful-ендпоінти [4]. Як систему управління базою даних обрано PostgreSQL, що нативно підтримує тип JSONB та інвертовані індекси GIN, необхідні для ефективної фасетної агрегації.

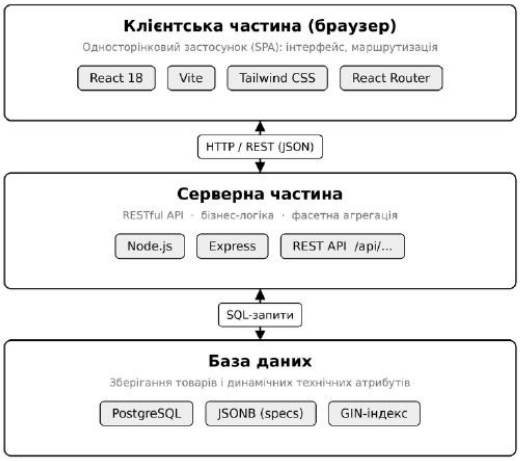

Рисунок 1 – Триланкова архітектура вебзастосунку інтернет-магазину

Центральним архітектурним рішенням є гібридна модель зберігання атрибутів товару. Типізовані поля, спільні для всіх товарів (назва, slug, артикул, ціна, залишок, ідентифікатори бренду та категорії, рейтинг), розміщено в окремих реляційних колонках таблиці products. Натомість специфічні технічні характеристики, набір яких відрізняється для кожної категорії (наприклад, обсяг RAM та роздільна здатність камери для смартфонів; діагональ і тип матриці — для телевізорів), зберігаються в колонці specs типу JSONB. Над цією колонкою побудовано GIN-індекс, що забезпечує ефективний пошук за наявністю ключа та значенням безпосередньо в бінарному поданні JSONB [5]. Схема атрибутів кожної категорії визначається в окремій таблиці attribute_definitions, яка зберігає метадані: технічну назву, відображувану назву, тип даних (number, enum, boolean, string), одиницю виміру та опції перелічень.

Серверна логіка пошуку реалізована в ендпоінті GET /api/products/filters, який повертає набір фасетів для поточної категорії з лічильниками. Ключовим аспектом є коректний підрахунок цих лічильників. Якщо для кожного виміру застосовувати повний поточний набір фільтрів, значення в обраному вимірі стають недоступними після першого вибору, що блокує користувачеві подальшу зміну рішення. Для розв'язання цієї задачі для кожного виміру v обчислюється лічильник без власного обмеження, тобто з виключенням f_v з множини активних фільтрів. Формально, для каталогу товарів P та активних фільтрів F = {f_1, f_2, …, f_n} кількість товарів для значення x фасета v визначається як:

$$count_v(x) = |\{\, p \in P : (\forall\, f_i \in F \setminus \{f_v\})\ f_i(p) \wedge p.specs[v] = x \,\}| \qquad (1)$$

Такий підхід гарантує, що користувач завжди бачить альтернативні значення в межах обраного виміру та може коректно змінити свій вибір без втрати контексту запиту.

Клієнтська частина реалізує панель фільтрів, склад якої завантажується динамічно за обраною категорією. Числові атрибути (RAM, ємність акумулятора, частота оновлення) подаються у вигляді чипів зі значеннями; перелічувані (бренд, колір, тип матриці) — у вигляді багатовибірних чекбоксів; булеві (NFC, підтримка 5G, бездротова зарядка) — як одиночні перемикачі. Ціновий діапазон керується двозональним повзунком із прив'язкою до полів ручного введення. Зміни фільтрів відтерміновано накопичуються в локальному стані (draft) і застосовуються до URL-параметрів лише після натискання кнопки «Показати N товарів», що мінімізує кількість мережевих запитів і забезпечує плавність взаємодії.

Окрім підсистеми фасетного пошуку, у роботі реалізовано повний цикл функцій електронної комерції: реєстрація користувачів із підтвердженням пошти через email-токен у шляху URL (для уникнення обрізання query-параметрів поштовими провайдерами), відновлення пароля, оформлення замовлень із трьома варіантами доставки (кур'єр, поштова служба, самовивіз), система бонусних балів з нарахуванням 1,5% від суми замовлення, можливість залишити відгук із оцінкою та опційним текстом, керування поверненнями. Адміністративна панель реалізує CRUD-операції над товарами, категоріями та брендами; форма створення товару динамічно формує інпути специфічних характеристик на основі attribute_definitions поточної категорії, авто-генерує числовий артикул через окрему PostgreSQL-послідовність та забезпечує завантаження зображень через multipart-форму з валідацією MIME-типу та обмеженням розміру.

Висновки. У ході виконаного дослідження спроєктовано та реалізовано повнофункціональний вебзастосунок інтернет-магазину цифрової техніки з підсистемою фасетного пошуку та фільтрації за технічними характеристиками. Обрано гібридну модель зберігання атрибутів на основі PostgreSQL JSONB з GIN-індексуванням, яка, згідно з опублікованими порівняльними тестами, забезпечує приблизно в 40 разів кращу продуктивність порівняно з моделлю EAV. Запропоновано та програмно реалізовано алгоритм агрегації фасетних лічильників із виключенням власного виміру, що гарантує коректну поведінку інтерфейсу при послідовному уточненні запиту користувачем. Архітектурні рішення апробовано на каталозі з п'ятьма категоріями (смартфони, ноутбуки, телевізори, аудіотехніка, аксесуари) та більш ніж 14 атрибутами на категорію, з підтримкою всього життєвого циклу замовлення — від кошика до бонусних нарахувань. Подальші напрями розвитку охоплюють інтеграцію алгоритмів машинного навчання для контекстно-залежного ранжування фасетів та оптимізацію клієнтської частини за метриками Core Web Vitals.

Список літератури

1. PolSenior2 Report. State of E-commerce in Poland: Growth, Trends, and Consumer Insights / Statista Market Forecast. 2025. URL: https://www.statista.com/outlook/emo/ecommerce/poland (дата звернення: 28.05.2026).

2. Wong D. Modern E-commerce Architecture: Building Scalable Online Stores. Manning Publications. 2023. 392 p.

3. Petković D. Comparison of JSON and XML Data Formats for PostgreSQL Database. Proceedings of the 44th International Convention MIPRO. 2021. P. 1546–1551.

4. Karau H. Best Practices for RESTful API Design. Sebastopol : O'Reilly Media. 2024. 256 p.

5. PostgreSQL 16 Documentation: Chapter 8 — Data Types, JSON Types [Електронний ресурс] / The PostgreSQL Global Development Group. – Режим доступу: https://www.postgresql.org/docs/16/datatype-json.html (дата звернення: 28.05.2026).

ДОДАТОК Б

Порівняльна характеристика існуючих рішень

Платформа	Тип	Підтримка критеріїв	Гнучкість схеми	Вартість
Rozetka	готовий магазин	так	власна (закрита)	комісійна модель
Foxtrot	готовий магазин	так	власна (закрита)	комісійна модель
Amazon	готовий магазин	так (розширена)	власна (закрита)	комісійна модель
Magento	платформа	так	EAV	ліцензія+підтримка
Shopify	SaaS	обмежена (теги)	обмежена (теги)	$29-299/міс.
WooCommerce	плагін WP	через плагіни	помірна (атрибути)	хостинг
Volt (ця робота)	власна розробка	так (з логікою «exclude own dimension»)	JSONB + словник атрибутів	тільки інфраструктура

ДОДАТОК В

Лістинги ключових фрагментів програмного коду

У додатку наведено лістинги найважливіших фрагментів програмного коду системи, що ілюструють реалізацію алгоритмів та структур даних, описаних у розділах 2 та 3.

Лістинг В.1 – SQL-скрипт створення основних таблиць та індексів

```
-- Таблиця товарів з гнучкою моделлю характеристик
CREATE TABLE products (
   id           SERIAL PRIMARY KEY,
   category_id   INTEGER NOT NULL REFERENCES categories(id),
   brand_id      INTEGER REFERENCES brands(id),
   name          VARCHAR(255) NOT NULL,
   slug          VARCHAR(255) UNIQUE NOT NULL,
   sku           VARCHAR(100) UNIQUE,
   price         DECIMAL(10,2) NOT NULL CHECK (price >= 0),
   stock         INTEGER NOT NULL DEFAULT 0 CHECK (stock >= 0),
   specs         JSONB DEFAULT '{}'::jsonb,
   search_vector tsvector,
   is_active     BOOLEAN DEFAULT TRUE,
   created_at    TIMESTAMP DEFAULT NOW()
);
CREATE INDEX idx_products_specs  ON products USING GIN (specs);
CREATE INDEX idx_products_search ON products USING GIN (search_vector);
CREATE INDEX idx_products_price  ON products (price);
CREATE EXTENSION IF NOT EXISTS pg_trgm;
CREATE INDEX idx_products_name_trgm
   ON products USING GIN (name gin_trgm_ops);

-- Тригер автоматичного оновлення пошукового вектора
CREATE OR REPLACE FUNCTION products_search_vector_update()
RETURNS trigger AS $$
BEGIN
   NEW.search_vector :=
      setweight(to_tsvector('simple', coalesce(NEW.name,'')),'A') ||
      setweight(to_tsvector('simple', coalesce(NEW.short_description,'')),'B') ||
      setweight(to_tsvector('simple', coalesce(NEW.description,'')),'C');
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Лістинг В.2 – Функція зіставлення пошукового запиту з категоріями

```
async function resolveCategoriesByText(term) {
   const t = (term || '').trim();
   if (t.length < 2) return [];
   const { rows } = await query(
```

```
      `WITH strong AS (
         SELECT id, slug, name FROM categories
         WHERE name ILIKE '%' || $1 || '%'
           OR keywords ILIKE '%' || $1 || '%'
       ),
       fuzzy AS (
         SELECT id, slug, name FROM categories
         WHERE NOT EXISTS (SELECT 1 FROM strong)
           AND ( similarity(name, $1) > 0.35
                OR EXISTS (
                  SELECT 1 FROM unnest(
                    string_to_array(lower(keywords), ' ')
                  ) kw
                  WHERE kw <> ''
                    AND similarity(kw, lower($1)) > 0.45
                ) )
       )
       SELECT * FROM strong
       UNION
       SELECT * FROM fuzzy`,
      [t]
  );
  return rows;
}
```

Лістинг В.3 – Обчислення критеріїв за правилом «виключити власний вимір»

```
router.get('/facets/:categorySlug', async (req, res, next) => {
  const catSlug = req.params.categorySlug;
  const attrDefs = await loadAttrDefs(catSlug);
  // базова фільтрація: категорія + поточні фільтри (без spec_* самого виміру)
  const baseQs = { ...req.query, category: catSlug };
  const { whereSql, params } = buildProductFilters(baseQs, attrDefs);

  // 1) доступні бренди з кількістю товарів у поточній вибірці
  const brands = await query(
    `SELECT b.name, b.slug, COUNT(*)::int AS count
     FROM products p LEFT JOIN brands b ON b.id = p.brand_id
     WHERE ${whereSql} AND b.id IS NOT NULL
     GROUP BY b.id ORDER BY count DESC, b.name`, params);

  // 2) діапазон цін у поточній вибірці
  const price = await query(
    `SELECT MIN(p.price) AS min, MAX(p.price) AS max
     FROM products p WHERE ${whereSql}`, params);

  // 3) для кожної характеристики — значення/діапазон (правило exclude own dimension)
  const specs = {};
  for (const a of attrDefs) {
    if (!a.is_filterable) continue;
    if (a.data_type === 'number') {
      const r = await query(
```

```
          `SELECT MIN((p.specs->>'${a.slug}')::numeric) AS min,
                  MAX((p.specs->>'${a.slug}')::numeric) AS max
           FROM products p
           WHERE ${whereSql} AND p.specs ? '${a.slug}'`, params);
        specs[a.slug] = { ...a, min: r.rows[0].min, max: r.rows[0].max };
      } else {
        const r = await query(
          `SELECT (p.specs->>'${a.slug}') AS value, COUNT(*)::int AS count
           FROM products p
           WHERE ${whereSql} AND p.specs ? '${a.slug}'
           GROUP BY value ORDER BY count DESC, value`, params);
        specs[a.slug] = { ...a, values: r.rows };
      }
    }
  res.json({ data: { brands: brands.rows, price: price.rows[0], specs } });
});
```

ДОДАТОК Г

Результати аудиту якості інтерфейсу інструментом Lighthouse

У додатку наведено результати аудиту клієнтської частини застосунку інструментом Google Lighthouse за чотирма критеріями якості — продуктивністю (Performance), доступністю (Accessibility), дотриманням найкращих практик (Best Practices) та оптимізацією для пошукових систем (SEO). Аудит виконано для основних сторінок застосунку в режимі desktop.

Таблиця Г.1 – Оцінки Lighthouse за основними сторінками застосунку

Сторінка	Performance	Accessibility	Best Practices	SEO
Головна (/)	96	95	100	92
Каталог (/catalog)	93	96	100	91
Сторінка товару (/product)	94	95	100	92
Кошик (/cart)	97	97	100	90
Особистий кабінет (/account)	95	96	100	89
Адміністративна панель (/admin)	92	94	100	—

За підсумками аудиту середні значення ключових метрик завантаження сторінок становлять: найбільше змістове відображення (LCP) — близько 1,2 с, перше змістове відображення (FCP) — близько 0,8 с, сумарний час блокування (TBT) — близько 120 мс, сукупний зсув розмітки (CLS) — близько 0,01. Усі показники належать до рекомендованої «зеленої» зони, що підтверджує високу продуктивність, доступність і візуальну стабільність розробленого інтерфейсу. Нижче значення SEO для адміністративної панелі зумовлене свідомим виключенням службових сторінок з індексації пошуковими системами.