МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ІНТЕРНЕТ МАГАЗИНУ ЗАСОБАМИ NODE.JS ТА EXPRESS»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.Р. Лісняк
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)
“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Лісняку Максиму Руслановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка серверної частини інтернет магазину засобами Node.js та Express

керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від ««26» січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка серверної частини інтернет магазину засобами Node.js та Express.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гребенюк С. М., завідувач кафедри фундаментальної та прикладної математики		
2	Гребенюк С. М., завідувач кафедри фундаментальної та прикладної математики		
3	Гребенюк С. М., завідувач кафедри фундаментальної та прикладної математики		

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	15.02.2023	
3.	Обробка методичних та теоретичних	27.02.2023	
	джерел.		
4.	Розробка першого та другого розділу.	10.04.2023	
5.	Розробка третього розділу.	08.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	22.06.2023	

Студент _____ М.Р. Лісняк
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка серверної частини інтернет магазину засобами Node.js та Express»: 47 с., 32 рис., 1 табл., 7 джерел.

ІНТЕРНЕТ МАГАЗИН, BACKEND API, EXPRESS, MYSQL, NODE.JS, REST API.

Об’єкт дослідження – процес розробки серверної частини інтернет магазину.

Мета роботи: аналіз предметної області та розробка серверної частини інтернет магазину засобами Node.js та Express.

Метод дослідження – метод об’єктноорієнтованого програмування.

У даній кваліфікаційній роботі розробляється серверна частина інтернет магазину з використанням Node.js та Express. Процес розробки включає аналіз предметної області, проєктування схеми даних, реалізацію API та його тестування.

Робота вносить новизну в область розробки вебзастосунків з використанням Node.js та Express.

Основні конструкторські характеристики включають розробку структури бази даних, проєктування API та взаємодії з клієнтською частиною.

Технологічні характеристики включають використання технологій Node.js та Express для реалізації серверної частини.

Техніко-експлуатаційні характеристики включають можливість масштабування, ефективну обробку запитів та надійність роботи системи.

Ця робота має значущість у контексті розвитку інтернет магазинів та використання технологій Node.js та Express. Вона надає можливість розробникам ефективно створювати та підтримувати серверні частини інтернет магазинів з використанням сучасних технологій.

SUMMARY

Bachelor’s qualifying paper «Development of a Server Part of the Online Store using Node.js and Express»: 47 pages, 32 figures, 1 table, 7 references.

ONLINE STORE, BACKEND API, EXPRESS, MYSQL, NODE.JS, REST API.

Object of the study is the process of development of a Server Part of the Online Store.

Aim of the study: domain analysis and development of a Server Part of the Online Store using Node.js and Express.

Method of research is object-oriented programming method.

In this qualification work, the server-side program of the online store is developed using Node.js and Express. The development process includes domain analysis, data schema design, API implementation and testing.

The work brings novelty to the field of web application development using Node.js and Express.

The main design characteristics include the development of the database structure, the design of the API and interaction with the client-side program.

Technological characteristics include the use of Node.js and Express technologies to implement the server-side program.

Technical and operational characteristics include the possibility of scaling, efficient processing of requests and reliability of system operation.

This work is significant in the context of the development of online stores and the use of Node.js and Express technologies. It enables developers to efficiently create and maintain server-side programs of online stores using modern technologies.

ЗМІСТ

ВСТУП

Сучасний розвиток засобів розробки вебзастосунків дає можливість обирати серед великої кількості технологій для використання. Одними із них є Node.js та Express.

Node.js – це популярне середовище виконання JavaScript, яке дозволяє запускати його на стороні сервера, що робить його добре придатним для створення масштабованих і ефективних вебзастосунків. Express – це швидкий та потужний вебфреймворк для Node.js, який надає надійний набір функцій для створення вебзастосунків і API.

Метою кваліфікаційної роботи бакалавра є розробка серверної частини інтернет магазину засобами Node.js та Express.

Задачі, які необхідно розв’язати для досягнення поставленої мети:

– проаналізувати предметну область;
– сформувати вимоги до проєкту;
– спроєктувати схему даних;
– реалізувати API;
– протестувати API.

Об’єкт дослідження – процес розробки серверної частини інтернет магазину.

Предмет дослідження – розробка серверної частини інтернет магазину засобами Node.js та Express.

Методи дослідження – методи об’єктноорієнтованого програмування.

Структурно робота складається з 3 розділів.

У першому розділі роботи надається огляд засобів, які були використані під час розробки серверної частини інтернет магазину з використанням Node.js та Express. В цьому розділі обґрунтовується вибір цих конкретних технологій та розглядаються аналогічні рішення. Огляд засобів дозволяє пояснити, чому саме Node.js та Express були обрані для реалізації серверної частини. Також розглянуті

інші важливі компоненти розробки вебзастосунків, які були використані в процесі роботи. Серед них – СУБД MySQL, та бібліотека knex.js.

Другий розділ роботи детально описує процес проєктування серверної частини інтернет магазину засобами Node.js та Express. В ньому розглядаються маршрутизація та структура API (Інтерфейс прикладного програмування). Маршрутизація визначає, які URL-шляхи будуть доступні для клієнтів та які дії будуть виконуватись на сервері при обробці запитів. Кожен маршрут має свій унікальний шлях та метод HTTP, які вказують на конкретну функціональність, що повинна бути виконана. В цьому розділі також представлено схему даних та описані міграції, які були реалізовані за допомогою knex.js. Схема даних визначає структуру та взаємозв’язки між різними сутностями у базі даних. Вона вказує, які поля є в кожній таблиці та які типи даних вони мають.

Третій розділ надає приклади роботи серверної частини інтернет магазину, яка була розроблена з використанням Node.js та Express. Цей розділ містить приклади роботи маршрутів, демонструє запити до системи та відповідні відповіді, що отримує клієнт від API.

1 ОГЛЯД ЗАСОБІВ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ ІНТЕРНЕТ МАГАЗИНУ

1.1 Огляд Node.js

Згідно з [1] Node.js є кросплатформовим виконавчим середовищем з відкритим вихідним кодом для виконання JavaScript, що базується на двигуні V8 JavaScript, яке є ядром Google Chrome, але працює поза межами браузера. Ця особливість надає Node.js велику потужність.

Програма Node.js працює в одному процесі, не створюючи новий потік для кожного запиту. Вона надає набір асинхронних примітивів введення/виведення у своїй стандартній бібліотеці, які запобігають блокуванню виконання JavaScript-коду. Бібліотеки в Node.js, в цілому, написані з використанням неблокуючих парадигм, що призводить до того, що блокування стає винятком, а не нормою.

Коли Node.js виконує операцію введення-виведення, наприклад, читання з мережі, доступ до бази даних або файлової системи, вона не блокує потік виконання і не затримує центральний процесор на очікуванні. Замість цього, Node.js продовжує виконувати інші операції й повертається до обробки відповіді, коли вона стає доступною. Це дозволяє Node.js обробляти тисячі одночасних з’єднань з одним сервером, не створюючи складностей управління паралелізмом потоків, що є частою причиною помилок.

Node.js має унікальну перевагу, оскільки мільйони фронт-енд розробників, які вже програмують на JavaScript для браузера, можуть використовувати його для написання серверного коду, не потребуючи вивчення іншої мови програмування.

У Node.js можна використовувати нові стандарти ECMAScript, а також включати певні експериментальні функції, запускаючи Node.js з необхідними прапорцями.

1.2 Огляд Express

Згідно з [2] Express є найпопулярнішим вебфреймворком для Node.js і є основною бібліотекою для ряду інших відомих вебфреймворків для Node.js. Він надає механізми для:

– маршрутизації;

– інтеграції з “двигунами відображення” (view rendering engines), щоб генерувати відповіді, вставляючи дані в шаблони;

– додавання додаткових “middleware” для обробки запитів в будь-якій точці потоку обробки запиту;

– підключення бібліотек для роботи з “cookies”, сесіями, авторизацією користувачів, параметрами URL, POST-даними, заголовками безпеки та багатьма іншими.

Далі наведено як працює Express сервер. Вебсервер очікує HTTP-запити від вебпереглядача (або іншого клієнта). При отриманні запиту застосунок визначає необхідні дії на основі шаблону URL і, можливо, пов’язаної інформації, що міститься в POST-даних або GET-даних. Залежно від вимог, він може читати або записувати інформацію з бази даних або виконувати інші необхідні завдання для задоволення запиту. Після цього застосунок повертає відповідь вебпереглядачу, часто динамічно створюючи сторінку HTML для відображення вебпереглядачем, вставляючи отримані дані в заповнювачі шаблону HTML.

1.3 Порівняння Node.js фреймворків з аналогами

Нижче у таблиці 1.1 наведено переваги та недоліки інших популярних Node.js фреймворків згідно з [3].

Для виконання цієї кваліфікаційної роботи було обрано саме Express тому, що він є зручним для використання, розширюваним та дозволяє впроваджувати багато пакетів сторонніх розробників.

Таблиця 1.1 – Порівняння Node.js фреймворків

Назва	Переваги	Недоліки
Koa.js	– має кращий підхід до обробки помилок; – усуває “callback hell” за допомогою “promises”.	– має невелику спільноту розробників з відкритим кодом; – його “middleware” несумісне з іншими “middleware” фреймворків Node.js, включаючи Express.js.
Meteor	– забезпечує безперервну комунікацію між сервером і клієнтською частиною; – реалізовує “hot reload”; – забезпечує можливість повторного використання коду – веб та мобільні застосунки можуть бути створені на основі одного коду.	– не надає підтримку серверного рендерингу для рівня “views”; – для синхронізації між серверною і клієнтською частиною застосунку потрібне стабільне інтернет-з’єднання.
Nest.js	– є розширюваним та гнучким, що дозволяє використовувати інші бібліотеки JavaScript; – використовує сучасні можливості JavaScript для реалізації кращих патернів проєктування.	– може бути складним для розробників-початківців.

1.4 Обґрунтування вибору інструментарію

Для виконання кваліфікаційної роботи було обрано виконавче середовище Node.js, фреймворк Express. Розглянемо нижче «плюси», що вплинули на вибір.

Висока продуктивність: Node.js використовує неблокуючу асинхронну модель, що дозволяє обробляти багато запитів одночасно без блокування потоку виконання, що робить його ефективним для високонавантажених вебзастосунків, які потребують швидкої обробки запитів.

Багатофункціональний фреймворк: Express надає простий та гнучкий спосіб створення вебзастосунків та API. Він має маршрутизацію, підтримку “middleware”, шаблонізатори та інші корисні функції, що полегшують розробку серверної частини додатка.

Велика спільнота та екосистема: Node.js та Express мають велику активну спільноту розробників, яка надає безліч додаткових модулів, пакетів та рішень для різних потреб, що дозволяє швидко розширювати функціональність додатка та використовувати перевірені рішення.

Також для виконання роботи було обрано СУБД MySQL з кількох причин.

MySQL є відомою своєю стабільністю, що робить її надійним рішенням для зберігання та управління даними. Вона має довгу історію розробки та випробувань, що сприяє її надійності і стійкості до відмов.

Ще однією перевагою MySQL є її реляційність. Це означає, що база даних MySQL використовує структуру даних, що базується на реляціях між таблицями. Цей підхід дозволяє ефективно організувати та зв’язувати дані.

MySQL має широку підтримку середніх до великих навантажень, що дозволяє оптимізувати продуктивність інтернет магазину. Вона може обробляти велику кількість одночасних запитів та з’єднань, що дозволяє забезпечити швидку відповідь на запити користувачів та зменшити час завантаження сторінок.

Для проєктування схеми даних і написання міграції було використано Knex.js. Knex.js є потужним інструментом для сворення SQL-запитів у

середовищі Node.js. Ця бібліотека забезпечує зручний спосіб взаємодії з реляційними базами даних, використовуючи JavaScript. Knex.js дозволяє будувати складні запити, виконувати транзакції, використовувати міграції та проводити інші операції з базою даних.

Всі ці фактори роблять Node.js, Express та MySQL потужними інструментами для розробки швидких та масштабованих вебзастосунків.

Висновки до розділу 1

В цьому розділі було наведено огляд інструментарію для реалізації серверної частини інтернет магазину. Обрано Node.js, Express, MySQL та Knex.js з таких причин: Node.js за його потужність, високу продуктивність, Express як легкий та багатофункціональний фреймворк, MySQL як надійне та реляційне рішення для зберігання даних, а Knex.js для зручного та експресивного інтерфейсу роботи з базою даних у Node.js. Також наведено порівняння Express з аналогічними рішеннями.

2 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ІНТЕРНЕТ МАГАЗИНУ

2.1 Огляд функціонала

Серверна частина виконана у формі REST API, що дає доступ до функцій системи для сторонніх клієнтських застосунків. Функції системи є наступними:

– реєстрація, авторизація, редагування, видалення користувачів;

– перегляд інформації про користувача;

– перегляд, створення, редагування, видалення продуктів;

– фільтрування продуктів за категоріями, брендами, ціною, характеристиками;

– отримання списку доступних фільтрів для категорії;

– додавання та видалення продуктів до/зі списку улюблених користувача;

– додавання та видалення характеристик до/зі списку характеристик продуктів;

– створення, редагування, видалення ключових слів;

– додавання та видалення ключових слів до/зі списку ключових слів продуктів;

– пошук продуктів за ключовими словами;

– перегляд, створення, редагування, видалення коментарів;

– створення, редагування, видалення ключових слів;

– створення, редагування, видалення категорій продуктів;

– перегляд кореневих категорій та підкатегорій;

– додавання та видалення продуктів до категорій;

– перегляд, створення, редагування, видалення брендів.

2.2 Огляд структури проєкту

Розроблений проєкт має структуру, зображену нижче на рисунку 2.1.

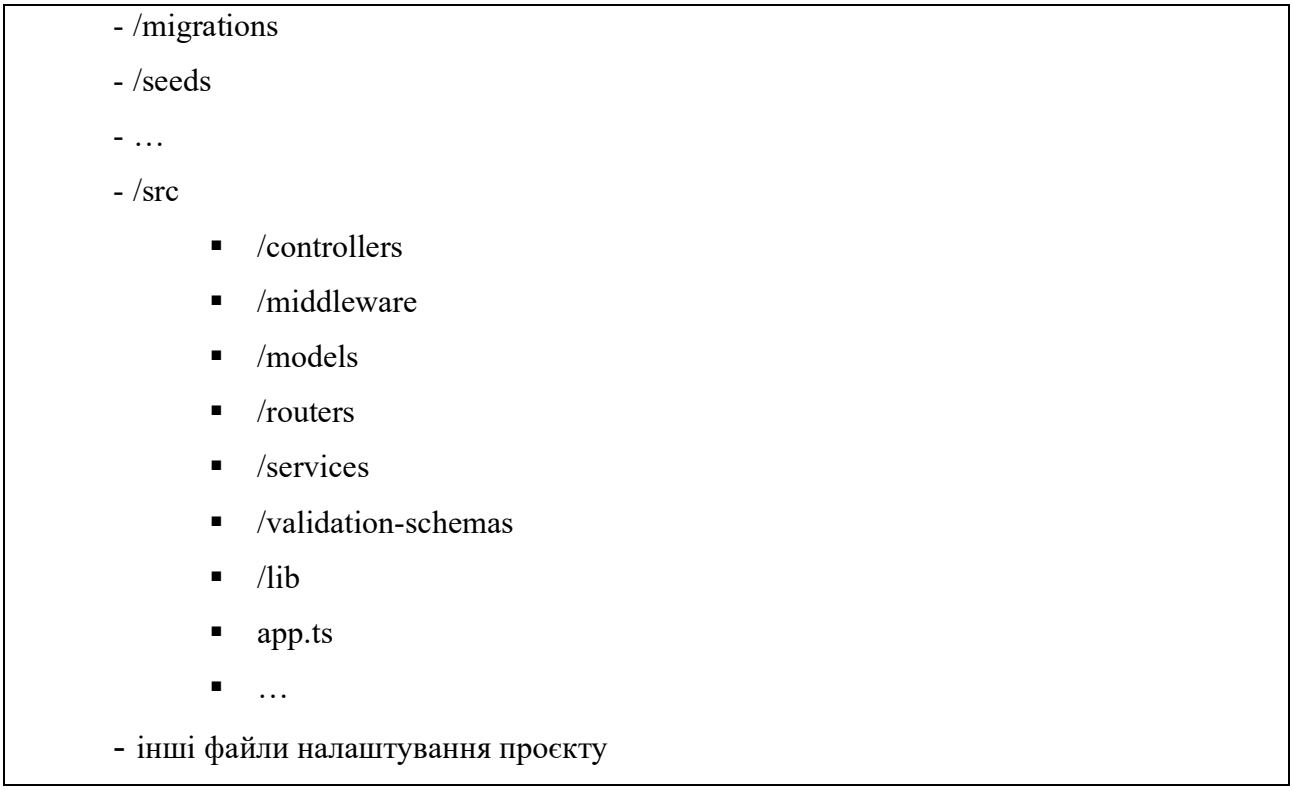

Рисунок 2.1 – Діаграма структури проєкту

Директорія /migrations містить файли-міграції для роботи з базою даних. Ці файли містять скрипти, які дозволяють змінювати схему бази даних, створювати та оновлювати таблиці, поля тощо.

Директорія /seeds містить файли для заповнення таблиць бази даних наперед заданими значеннями. Ці файли можуть містити початкові дані, що допомагають ініціалізувати базу даних з певними значеннями або тестові дані для виконання тестів.

Файли директорії /controllers містять функції-контролери для обробки запитів до API. Контролери приймають вхідні дані, виконують необхідну логіку та повертають відповідь клієнту або викликають відповідні сервіси.

Файли директорії /middleware містять проміжні функції. Вони слугують для перевірки, авторизації, обробки помилок тощо.

Файли директорії /models містять моделі для роботи з сутностями бази даних. Моделі описують структуру та взаємозв’язки між таблицями бази даних і надають методи для роботи з цими даними, такі як створення, читання, оновлення, видалення записів.

Файли директорії /routers містять маршрути API, та задають функції-обробники. Маршрути визначають шляхи до різних ендпоінтів API та пов’язані з ними функції-обробники, які будуть виконуватись при отриманні відповідного запиту.

Файли директорії /services містять функції для роботи з даними, що викликаються з функцій-контролерів. Сервіси зазвичай містять бізнес-логіку, обробляють дані та здійснюють взаємодію з базою даних.

Файли директорії /validation-schemas містять схеми валідації даних при запиті до API. Ці схеми описують правила та обмеження для вхідних даних, такі як обов’язкові поля, типи даних, максимальна чи мінімальна довжина тощо.

Файли директорії /lib містять додаткові файли проєкту, такі як константи, інтерфейси чи типи.

Файл app.ts містить базове налаштування проєкту та код, що запускає програму.

2.3 Проєктування схеми даних

Схема даних складається з деяких сутностей. Нижче наведено їх назви та короткий опис.

«Users» – сутність, що зберігає користувачів системи, які мають акаунти і можуть здійснювати різні дії, такі як реєстрація, авторизація, додавання коментарів, додавання товарів до улюблених тощо.

«Comments» – сутність, що зберігає коментарі до продуктів, які можуть бути залишені користувачами. Коментарі зберігають інформацію про текст коментаря, автора, дату та інші відомості.

«Keywords» – сутність, що зберігає ключові слова, які використовуються для пошуку продуктів. Ключові слова можуть бути прив’язані до конкретних продуктів і допомагають знаходити їх за відповідними ознаками чи описом.

«Keyword_to_product» – сутність, що використовується для встановлення зв’язку між конкретним ключовим словом і продуктом, що має це ключове слово.

«Favorites» – сутність, що зберігає улюблені товари користувачів, а саме – список товарів, що їм сподобалися або на які вони звертають увагу. Ця сутність дозволяє користувачам зручно організовувати та отримувати доступ до своїх улюблених товарів.

«Products» – сутність, що зберігає продукти, які доступні у системі. Кожен продукт має свою унікальну ідентифікаційну інформацію, а також включає в себе дані про бренд та інші відомості.

«Brands» – сутність, що зберігає дані про різні бренди, що представлені у системі.

«Categories» – сутність, що зберігає категорії, до яких можуть належати продукти.

«Category_to_product» – сутність, що використовується для встановлення зв’язку між конкретною категорією і продуктом, який відноситься до цієї категорії.

«Attribute_names» – сутність, що зберігає список назв доступних характеристик, які можуть бути призначені для продуктів.

«Attribute_values» – сутність, що містить дані про значення характеристик, які призначені для продуктів.

«Attribute_to_product» – сутність, що використовується для встановлення зв’язку між конкретною характеристикою і продуктом, який має цю характеристику.

Нижче на рисунку 2.2 наведена ER-діаграма схеми даних.

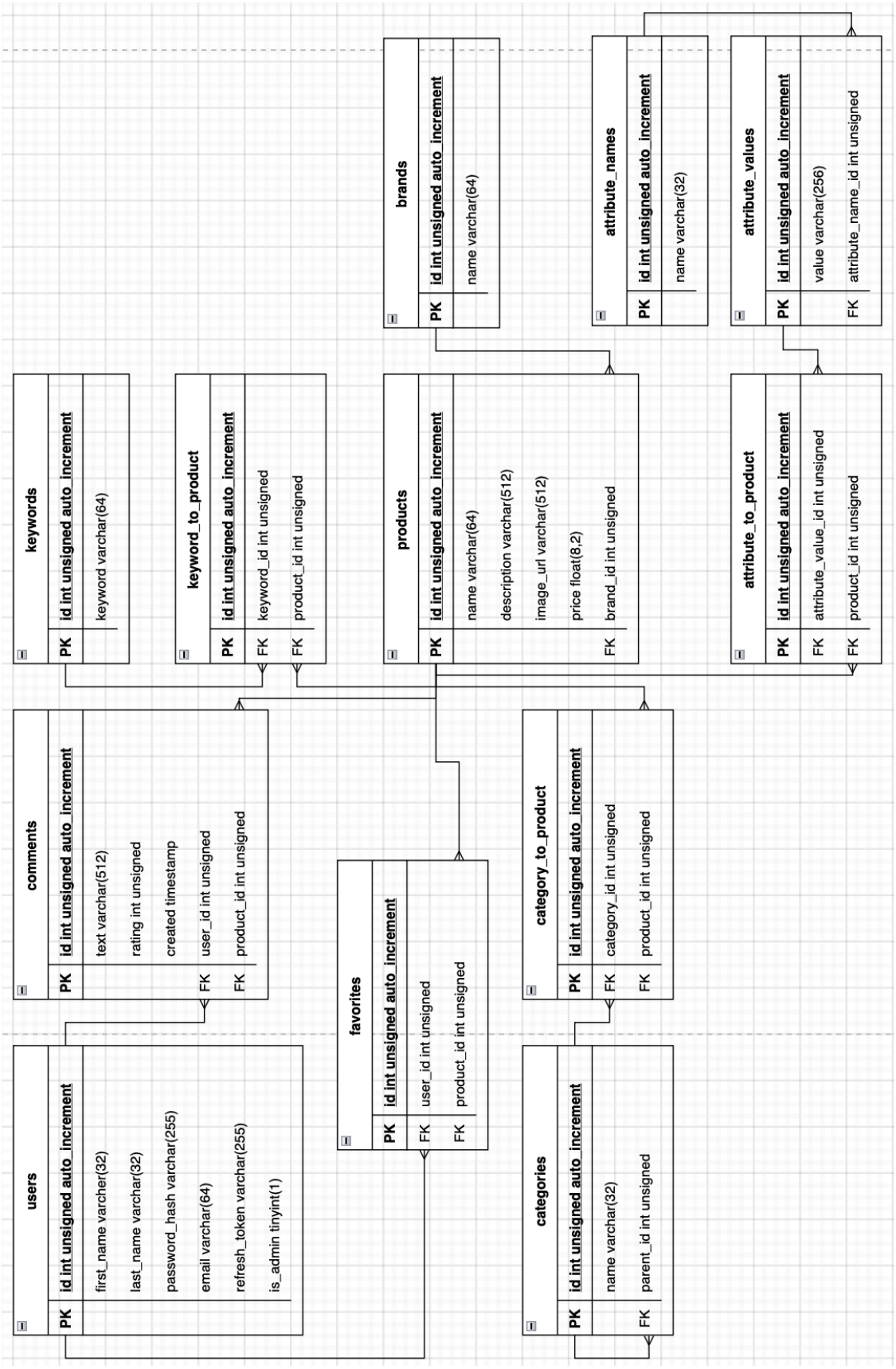

Рисунок 2.2 – ER-діаграма схеми даних

Далі на рисунках 2.3 – 2.14 наведено код міграції вищезазначених сутностей створених з використанням knex.js.

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“users”, function (table) {
      table.increments();
      table.string(“first_name”, 32).notNullable();
      table.string(“last_name”, 32);
      table.string(“password_hash”).notNullable();
      table.string(“email”, 64).notNullable().unique();
      table.string(“refresh_token”);
      table.boolean(“is_admin”).defaultTo(false);
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable(“users”)
}
```

Рисунок 2.3 – Код міграції для сутності “users”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“categories”, function (table) {
      table.increments();
      table.string("name", 32).unique().notNullable();
      table.integer("parent_id").unsigned();
      table.foreign(“parent_id”)
        .references("id")
        .inTable("categories").onDelete("SET NULL");
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable(“categories”)
}
```

Рисунок 2.4 – Код міграції для сутності “categories”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“category_to_product”, function (table) {
      table.increments();
      table.integer("category_id").unsigned();
      table.integer("product_id").unsigned();
      table.foreign("category_id")
        .references("id").inTable("categories").onDelete("CASCADE");
      table.foreign("product_id")
        .references("id").inTable("products").onDelete("CASCADE");
      table.unique([“category_id”, “product_id”])
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("category_to_product")
};
```

Рисунок 2.5 – Код міграції для сутності “category_to_product”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“products”, function (table) {
      table.increments();
      table.string(“name”, 64).notNullable();
      table.string(“description”, 512);
      table.string(“image_url”);
      table.float(“price”);
      table.integer("brand_id").unsigned();
      table.foreign("brand_id").references("id")
        .inTable("brands").onDelete("SET NULL");
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("products")
}
```

Рисунок 2.6 – Код міграції для сутності “products”

```
exports.up = function (knex) {
  return knex.schema.createTable(“comments”, function (table) {
     table.increments();
     table.string(“text”, 512).notNullable();
     table.integer(“rating”).unsigned().nullable();
     table.timestamp(“created”).notNullable().defaultTo(knex.fn.now())
     table.integer("user_id").unsigned();
     table.integer("product_id").unsigned();
     table.foreign("user_id")
       .references("id").inTable("users").onDelete("SET NULL");
     table.foreign("product_id")
       .references("id").inTable("products").onDelete("CASCADE");
   })
}
exports.down = function (knex) {
  return knex.schema.dropTable("comments")
}
```

Рисунок 2.7 – Код міграції для сутності “comments”

```
exports.up = function (knex) {
  return knex.schema.createTable(“favorites”, function (table) {
     table.increments();
     table.integer("user_id").unsigned();
     table.integer("product_id").unsigned();
     table.foreign("user_id")
       .references("id").inTable("users").onDelete("CASCADE");
     table.foreign("product_id")
       .references("id").inTable("products").onDelete("CASCADE");
   })
}
exports.down = function (knex) {
  return knex.schema.dropTable("favorites")
}
```

Рисунок 2.8 – Код міграції для сутності “favorites”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“keywords”, function (table) {
      table.increments();
      table.string("keyword", 64)
.notNullable();
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("keywords")
}
```

Рисунок 2.9 – Код міграції для сутності “keywords”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“keyword_to_product”, function (table) {
      table.increments();
      table.integer("keyword_id")
.unsigned();
      table.integer("product_id")
.unsigned();
      table.foreign("keyword_id")
        .references("id")
.inTable("keywords")
.onDelete("CASCADE");
      table.foreign("product_id")
        .references("id")
.inTable("products")
.onDelete("CASCADE");
      table.unique([“keyword_id”, “product_id”])
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("keyword_to_product")
};
```

Рисунок 2.10 – Код міграції для сутності “keyword_to_product”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“attribute_names”, function (table) {
      table.increments();
      table.string("name", 32)
.notNullable()
.unique();
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("attribute_names")
}
```

Рисунок 2.11 – Код міграції для сутності “attribute_names”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“attribute_values”, function (table) {
      table.increments();
      table.string("value", 256);
      table.integer("attribute_name_id").unsigned();
      table.foreign("attribute_name_id")
        .references("id").inTable("attribute_names")
        .onDelete("SET NULL");
      table.unique([“value”, “attribute_name_id”])
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("attribute_values")
}
```

Рисунок 2.12 – Код міграції для сутності “attribute_values”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“attribute_to_product”, function (table) {
      table.increments();
      table.integer("attribute_value_id").unsigned();
      table.integer("product_id").unsigned();
      table.foreign("attribute_value_id")
        .references("id").inTable("attribute_values")
        .onDelete("CASCADE");
      table.foreign("product_id").references("id")
        .inTable("products").onDelete("CASCADE");
      table.unique([“attribute_value_id”, “product_id”])
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("attribute_to_product")
};
```

Рисунок 2.13 – Код міграції для сутності “product_to_attribute”

```
exports.up = function (knex) {
  return knex.schema
    .createTable(“brands”, function (table) {
      table.increments();
      table.string("name", 64).notNullable().unique();
    })
}
exports.down = function (knex) {
  return knex.schema.dropTable("brands")
}
```

Рисунок 2.14 – Код міграції для сутності “brands”

2.4 Проєктування схеми маршрутів API

Інтерфейс прикладного програмування (API) – це спосіб взаємодії двох чи більше комп'ютерних програм одна з одною. Це різновид програмного інтерфейсу, який пропонує послуги іншим частинам програмного забезпечення. Документ або стандарт, який описує, як створити чи використовувати таке з'єднання чи інтерфейс, називається специфікацією API, згідно з [4].

Для виконання кваліфікаційної роботи було створено API що дозволяє стороннім клієнтам працювати з системою – запитувати, створювати, редагувати та видаляти дані.

Для доступу до API використовуються HTTP запити.

HTTP (Hypertext Transfer Protocol) є протоколом передачі даних в мережі Інтернет, і використовується для комунікації між клієнтами і серверами. У HTTP існує кілька типів запитів (HTTP verbs), які визначають тип дії, що потрібно виконати з вказаним ресурсом на сервері, згідно з [5, 6, 7]:

– GET – це HTTP метод, який використовується для отримання даних з сервера;

– POST – цей метод використовується для створення нових ресурсів на сервері;

– DELETE – цей метод використовується для видалення існуючих ресурсів на сервері;

– PATCH – цей метод використовується для часткового оновлення існуючих ресурсів на сервері.

В ході виконання роботи було створено маршрути та функції-контролери, що виконуються при запиті до відповідного маршруту та повертають відповідь клієнту.

Нижче наведено опис маршрутів які використовується в API, що було створено.

2.4.1 Маршрути для роботи з користувачами

GET /users/me – маршрут, що використовується для отримання даних про поточного користувача. Він захищений “middleware” функцією “verifyAuthorization”, яка перевіряє дійсність токена доступу (access token). Маршрут викликає функцію “findByToken” з контролера UsersController.

GET /users/:id – маршрут, що використовується для отримання даних певного користувача. Він викликає функцію “findById” з контролера UsersController.

GET /users/ – маршрут, що використовується для отримання даних всіх користувачів за параметрами пошуку. Він викликає функцію “find” з контролера UsersController.

POST /users/sign-up – маршрут, що використовується для реєстрації користувача. Він викликає функцію “signUp” з контролера UsersController.

POST /users/sign-in – маршрут, що використовується для входу користувача в систему. Він викликає функцію “signIn” з контролера UsersController.

POST /users/sign-out – маршрут, що використовується для виходу користувача з системи. Він викликає функцію “signOut” з контролера UsersController.

POST /users/refresh – маршрут, що використовується для оновлення токена доступу. Він викликає функцію “handleRefreshToken” з контролера UsersController.

POST /users/favorite-products/:product_id – маршрут, що використовується для додавання продукту до списку улюблених користувача. Він захищений “middleware” функцією “verifyAuthorization”. Він викликає функцію “addFavoriteProduct” з контролера UsersController.

PATCH /users/me – маршрут, що використовується для оновлення даних поточного користувача. Він захищений “middleware” функцією

“verifyAuthorization”. Він викликає функцію “updateByToken” з контролера UsersController.

DELETE /users/favorite-products/:product_id – маршрут, що використовується для видалення продукту зі списку улюблених користувача. Він захищений “middleware” функцією “verifyAuthorization”. Він викликає функцію “removeFavoriteProduct” з контролера UsersController.

DELETE /users/me – маршрут, що використовується для видалення облікового запису поточного користувача. Він захищений “middleware” функцією “verifyAuthorization”. Він викликає функцію “deleteByToken” з контролера UsersController.

2.4.2 Маршрути для роботи з категоріями

GET /categories/:id – маршрут, що використовується для отримання інформації про категорію за її ідентифікатором. Він викликає функцію “findById” з контролера CategoriesController.

GET /categories – маршрут, що використовується для отримання списку всіх кореневих категорій. Він викликає функцію “findRootCategories” з контролера CategoriesController.

POST /categories/:category_id/products/:product_id – маршрут, що використовується для додавання продукту до категорії. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “addProduct” з контролера CategoriesController.

POST /categories – маршрут, що використовується для створення нової категорії. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “create” з контролера CategoriesController.

PATCH /categories/:id – маршрут, що використовується для оновлення інформації про категорію за її ідентифікатором. Він захищений “middleware”

функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “updateById” з контролера CategoriesController.

DELETE /categories/:category_id/products/:product_id – маршрут, що використовується для видалення продукту з категорії. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “removeProduct” з контролера CategoriesController.

DELETE /categories/:id – маршрут, що використовується для видалення категорії за її ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “deleteById” з контролера CategoriesController.

2.4.3 Маршрути для роботи з продуктами

GET /products/filter-list – маршрут, що використовується для отримання списку можливих фільтрів для продуктів. Він викликає функцію “getFilterList” з контролера ProductsController.

GET /products/:id – маршрут, що використовується для отримання інформації про продукт за його ідентифікатором. Він викликає функцію “findById” з контролера ProductsController.

POST /products/filter – маршрут, що використовується для пошуку продуктів за заданими фільтрами. Він викликає функцію “findByFilters” з контролера ProductsController.

POST /products – маршрут, що використовується для створення нового продукту. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “create” з контролера ProductsController.

PATCH /products/:id – маршрут, що використовується для оновлення інформації про продукт за його ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію

“updateById” з контролера ProductsController.

DELETE /products/:id – маршрут, що використовується для видалення продукту за його ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “deleteById” з контролера ProductsController.

POST /products/:product_id/attributes – маршрут, що використовується для додавання атрибута до продукту за його ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “addAttribute” з контролера ProductsController.

DELETE /products/:product_id/attributes/:attribute_id – маршрут, що використовується для видалення атрибута продукту за ідентифікаторами продукту та атрибуту. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “removeAttribute” з контролера ProductsController.

2.4.4 Маршрути для роботи з коментарями

GET /comments – маршрут, що використовується для отримання списку коментарів. Він викликає функцію “find” з контролера CommentsController.

POST /comments – маршрут, що використовується для створення нового коментаря. Він захищений “middleware” функцією “verifyAuthorization”. Він викликає функцію “create” з контролера CommentsController.

PATCH /comments/:id – маршрут, що використовується для оновлення коментаря за його ідентифікатором. Він захищений “middleware” функцією “verifyAuthorization”. Він викликає функцію “updateById” з контролера CommentsController.

DELETE /comments/:id – маршрут, що використовується для видалення коментаря за його ідентифікатором. Він захищений “middleware” функцією

“verifyAuthorization”. Він викликає функцію “deleteById” з контролера CommentsController.

2.4.5 Маршрути для роботи з ключовими словами

GET /keywords/:id – маршрут, що використовується для отримання інформації про ключове слово за його ідентифікатором. Він викликає функцію “findById” з контролера KeywordsController.

GET /keywords – маршрут, що використовується для отримання списку всіх ключових слів. Він викликає функцію “find” з контролера KeywordsController.

POST /keywords/:keyword_id/products/:product_id – маршрут, що використовується для додавання ключового слова до продукту. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “addToProduct” з контролера KeywordsController.

POST /keywords – маршрут, що використовується для створення нового ключового слова. Він викликає функцію “create” з контролера KeywordsController.

PATCH /keywords/:id – маршрут, що використовується для оновлення інформації про ключове слово за його ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “updateById” з контролера KeywordsController.

DELETE /keywords/:keyword_id/products/:product_id – маршрут, що використовується для видалення ключового слова з продукту. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “removeFromProduct” з контролера KeywordsController.

DELETE /keywords/:id – маршрут, що використовується для видалення ключового слова за його ідентифікатором. Він захищений “middleware”

функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “deleteById” з контролера KeywordsController.

2.4.6 Маршрути для роботи з брендами

GET /brands/:id – маршрут, що використовується для отримання інформації про бренд за його ідентифікатором. Він викликає функцію “findById” з контролера BrandsController.

GET /brands – маршрут, що використовується для отримання списку всіх брендів. Він викликає функцію “find” з контролера BrandsController.

POST /brands – маршрут, що використовується для створення нового бренду. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “create” з контролера BrandsController.

PATCH /brands/:id – маршрут, що використовується для оновлення інформації про бренд за його ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “updateById” з контролера BrandsController.

DELETE /brands/:id – маршрут, що використовується для видалення бренду за його ідентифікатором. Він захищений “middleware” функціями “verifyAuthorization” та “verifyAdmin”. Він викликає функцію “deleteById” з контролера BrandsController.

Висновки до розділу 2

У цьому розділі було оглянуто функціонал серверної частини інтернет магазину та реалізовано API для роботи з цією системою. Було представлено огляд функціоналу та описано структуру даних, наведено опис сутностей бази даних. Також розглянуті основні маршрути для роботи з сутностями та

функціями, системи. Кожен маршрут був пояснений та описаний. Вказано, які HTTP методи використовуються для відповідних маршрутів та які функції контролера викликаються для обробки цих маршрутів.

3 ТЕСТУВАННЯ API

3.1 Тестування маршрутів для роботи з користувачами

Запит на авторизацію виконується за маршрутом “/users/sign-in”.

У тілі запиту користувач відправляє адресу електронної пошти та пароль. Сервер надсилає у відповідь дані користувача, “accessToken”, та “refreshToken” як “cookie”. “accessToken” використовується для доступа до захищених URL, він підтверджує факт авторизації. Нижче, на рисунку 3.1, наведено приклад успішного запиту на авторизацію.

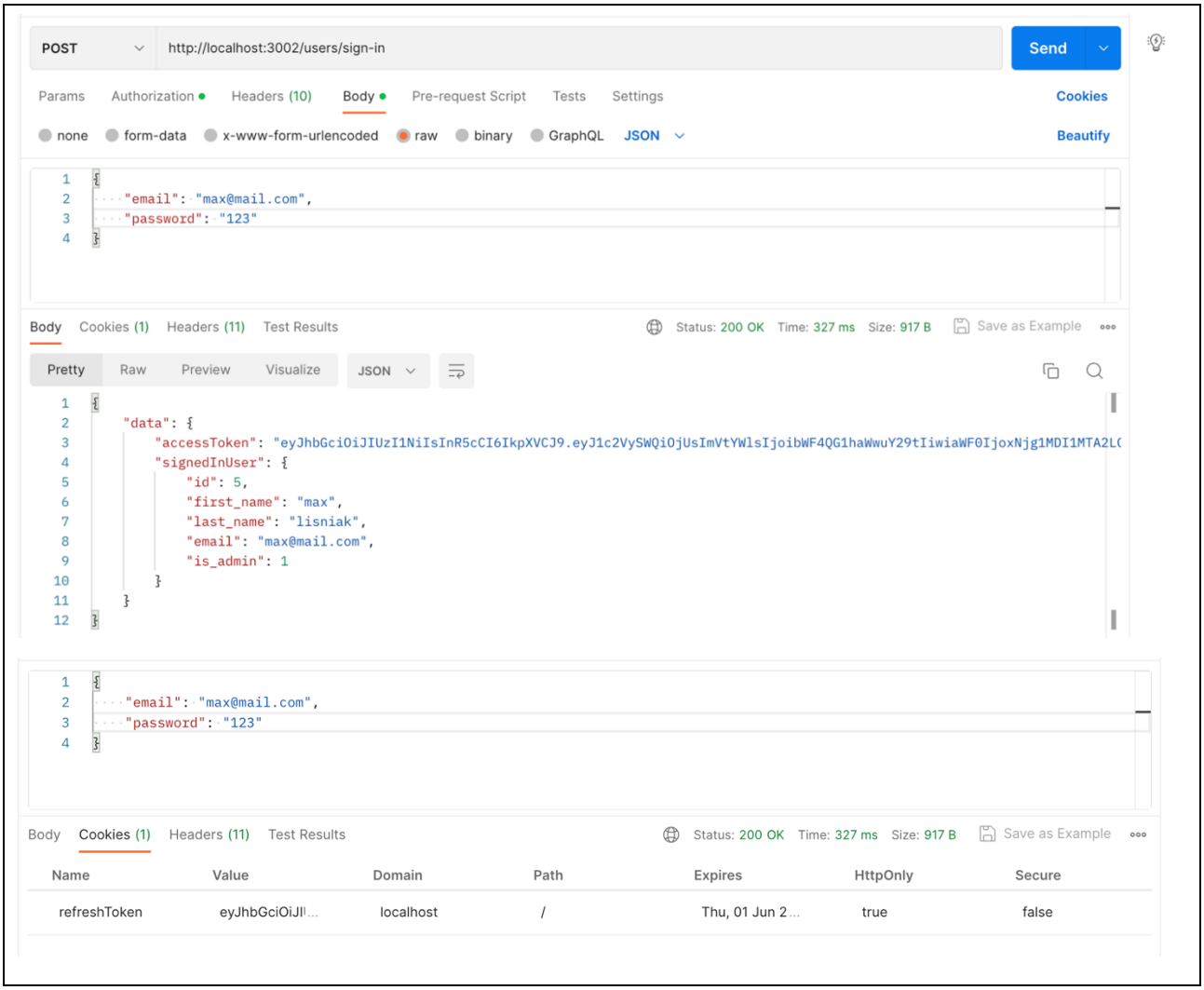

Рисунок 3.1 – Успішний запит на авторизацію

Запит на авторизацію може також бути невдалим якщо передати неправильні дані користувача. На рисунку 3.2 показано приклад невдалої авторизації, де сервер надсилає повідомлення про неправильність даних.

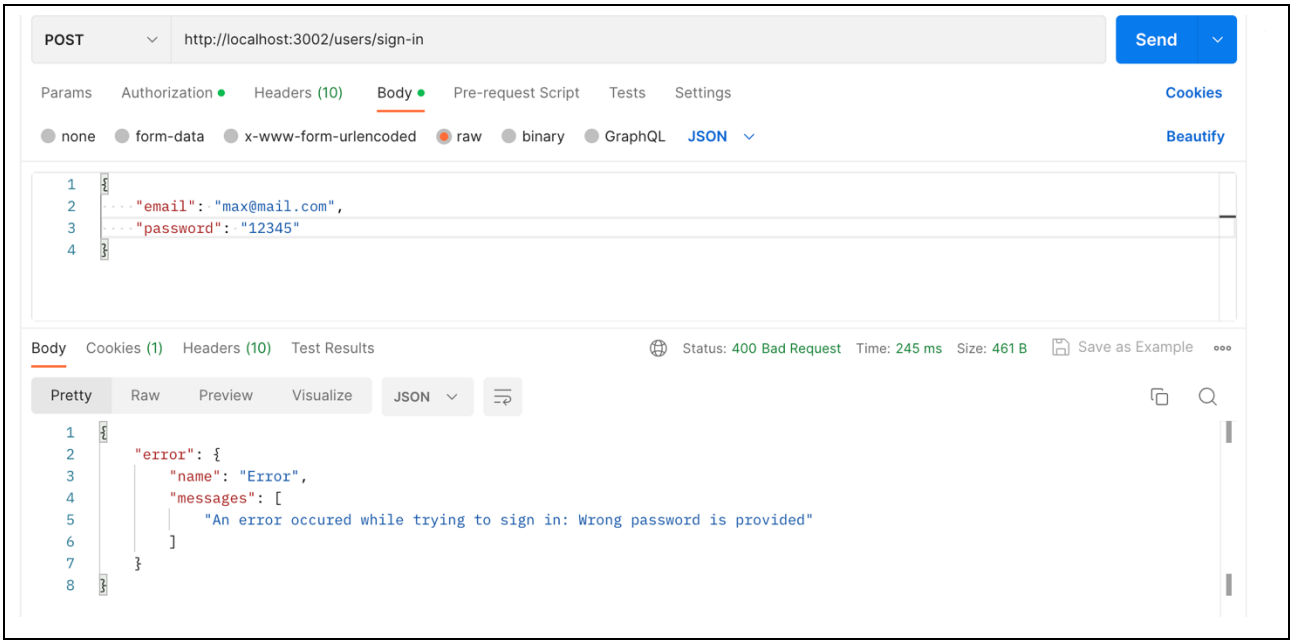

Рисунок 3.2 – Невдалий запит на авторизацію

“AccessToken” має строк придатності, тож, коли він вичерпався, необхідно його оновити. Запит за маршрутом “/users/refresh” дозволяє це зробити. На рисунку 3.3 показано приклад оновлення “accessToken” за допомогою “refreshToken”.

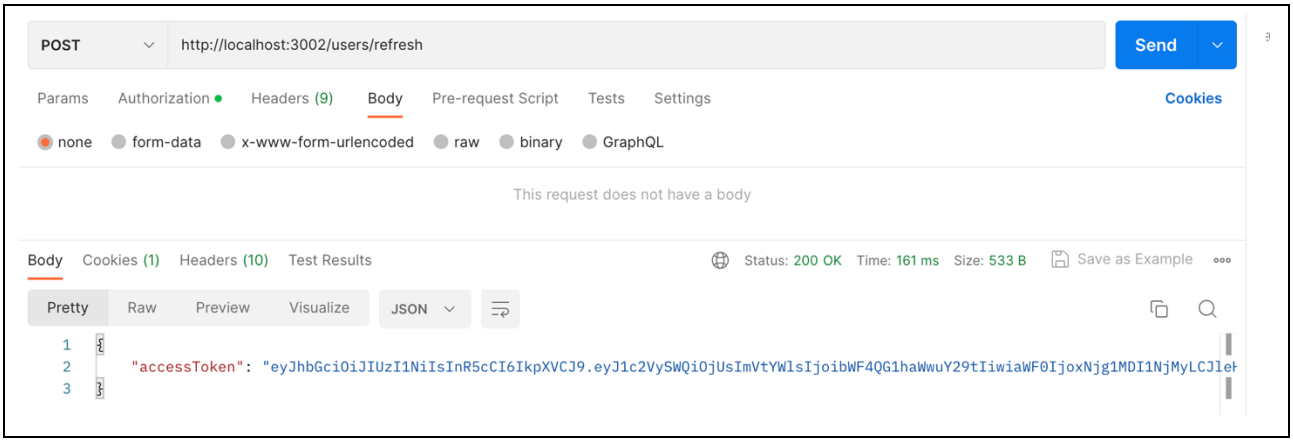

Рисунок 3.3 – Запит на оновлення “accessToken”

3.2 Тестування маршрутів для роботи з категоріями

Однією з функцій системи є отримання категорій продуктів. Категорії можуть бути кореневими або підкатегоріями інших категорії. На рисунку 3.4 наведено приклад запиту на отримання кореневих категорій.

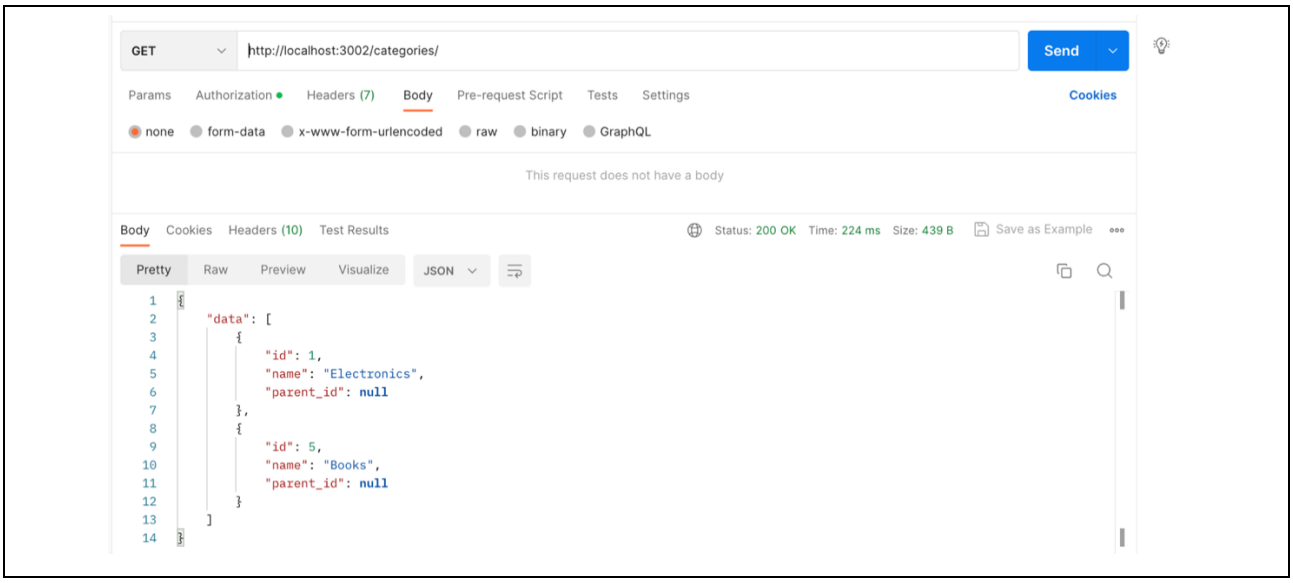

Рисунок 3.4 – Запит на отримання кореневих категорій

На рисунку 3.5 наведено приклад запиту на отримання підкатегорій за ідентифікатором категорії.

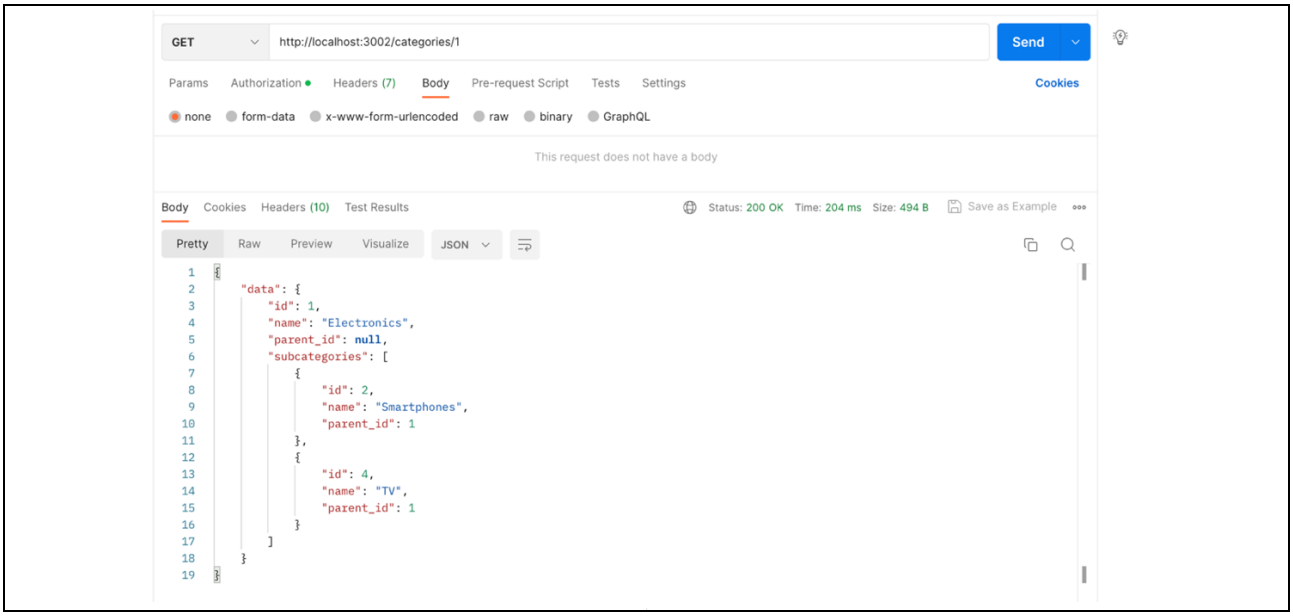

Рисунок 3.5 – Запит на отримання підкатегорій за ідентифікатором категорії

На рисунку 3.6 наведено приклад запитів на створення підкатегорії, та перегляду списку категорії цього ж рівня вкладеності.

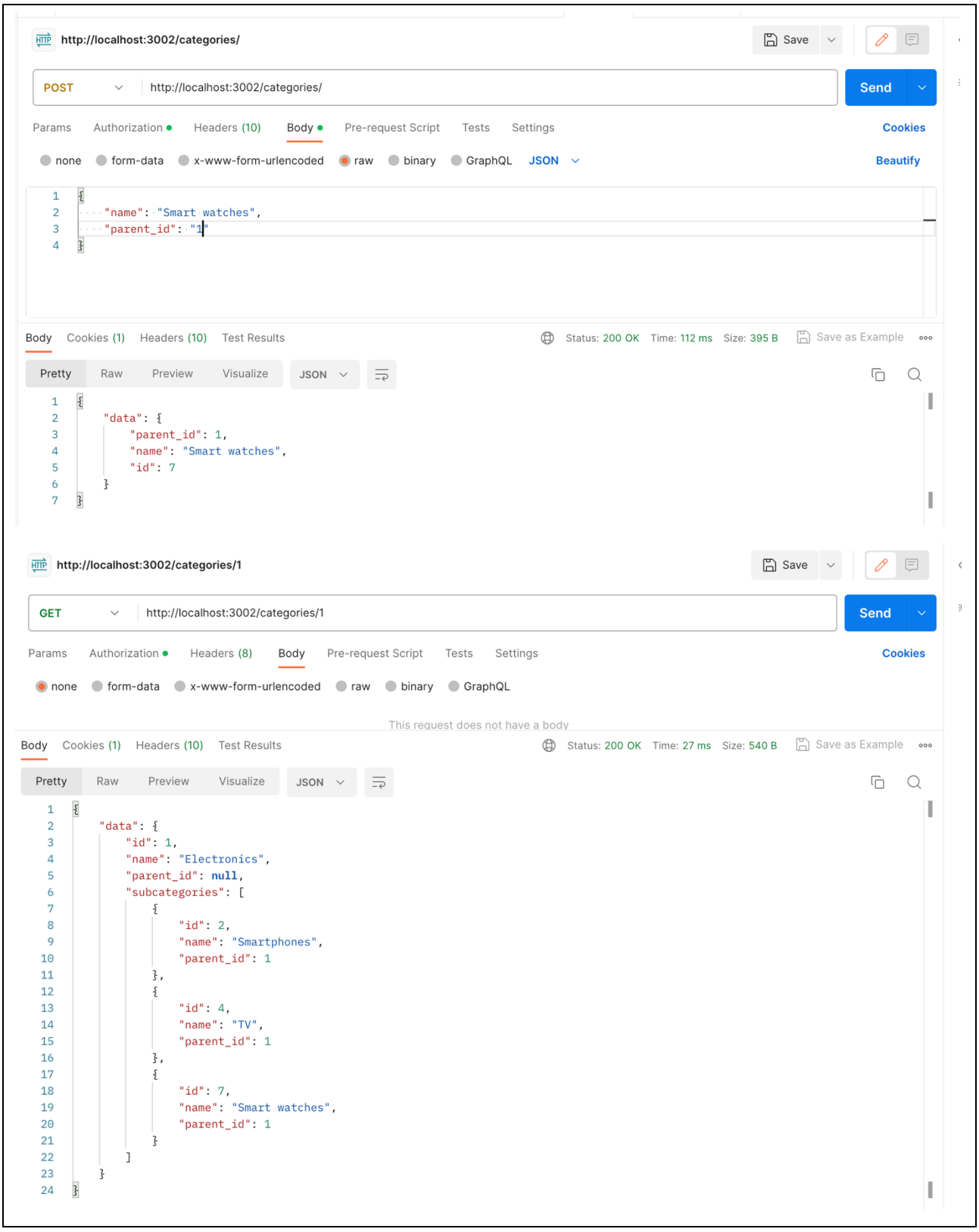

Рисунок 3.6 – Запит на створення та перегляд підкатегорій

3.3 Тестування маршрутів для роботи з продуктами

Система дозволяє фільтрувати продукти за широким набором фільтрів. На рисунку 3.7 наведено приклад запиту на отримання списку фільтрів для продуктів.

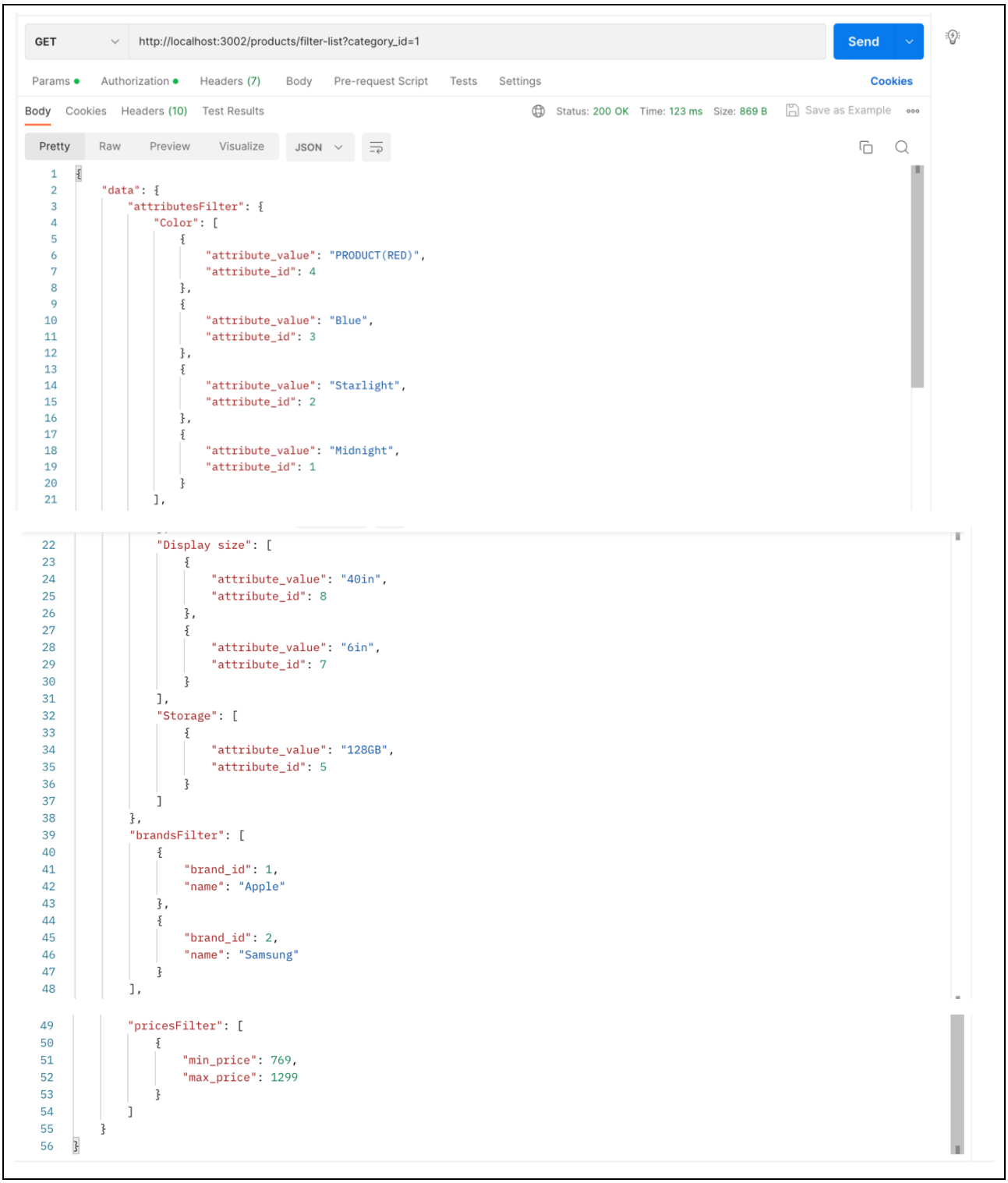

Рисунок 3.7 – Запит на отримання списку фільтрів для продуктів

Система дозволяє фільтрувати продукти за характеристиками, брендом, ціною. На рисунку 3.8 показано приклад запиту на фільтрацію продуктів.

Тіло запиту складається з параметрів фільтрації:

– attribute_filters – масив ідентифікаторів характеристик;

– brands – масив ідентифікаторів брендів;

– price – об’єкт ціни, що складається з мінімальної та максимальної ціни.

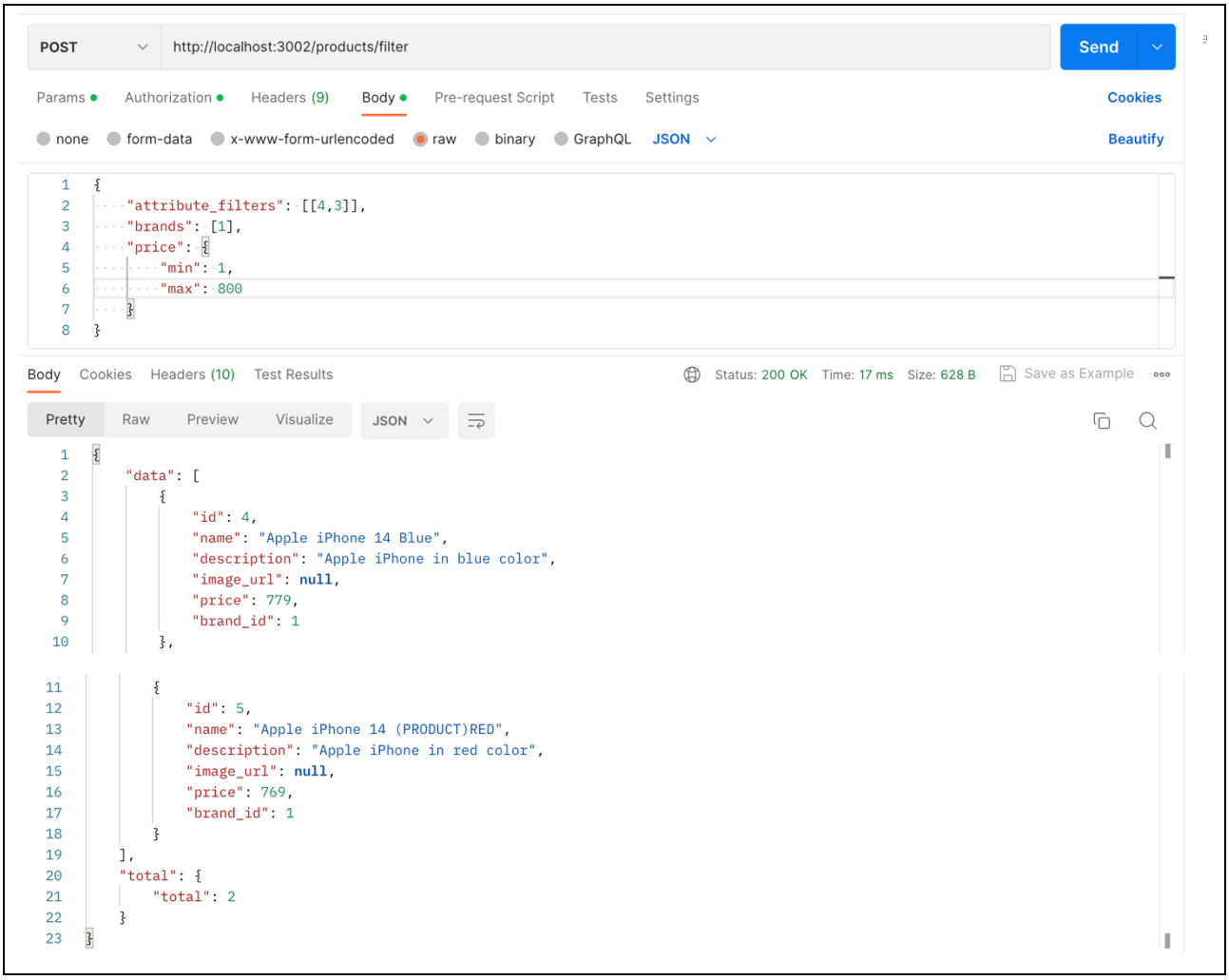

Рисунок 3.8 – Запит на фільтрацію продуктів

Також система дозволяє шукати товари за назвою або ключовими словами. На рисунку 3.9 показано приклад запиту на пошук продуктів.

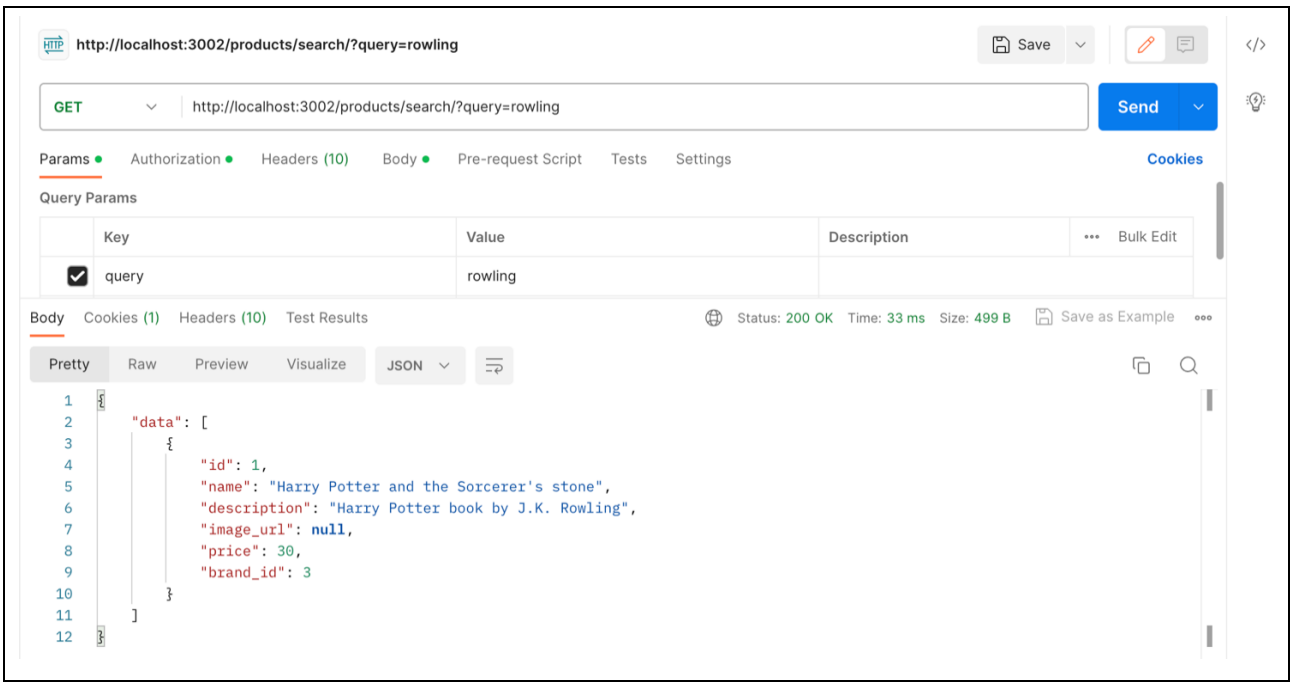

Рисунок 3.9 – Запит на пошук продуктів

На рисунку 3.10 наведено приклади запитів на створення продукту.

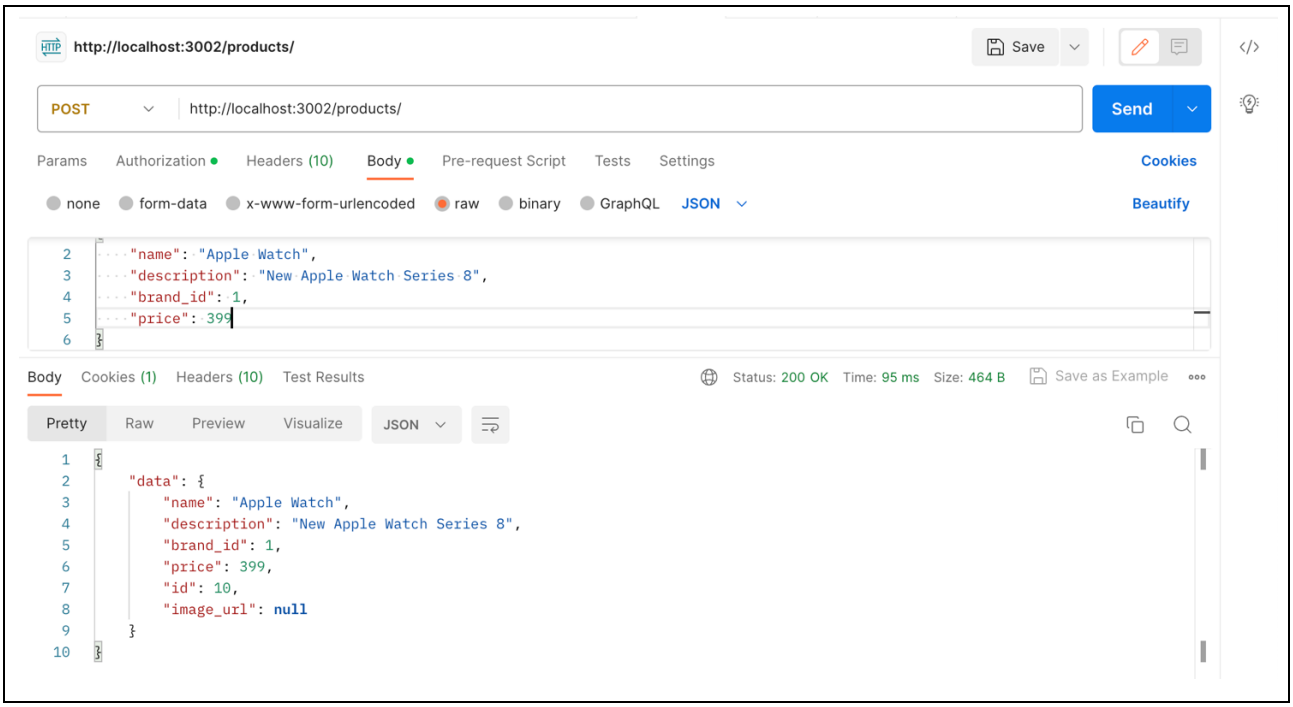

Рисунок 3.10 – Запит на створення продукту

На рисунку 3.11 наведено приклад запитів на додавання характеристики до продукту та перегляд інформації про продукт.

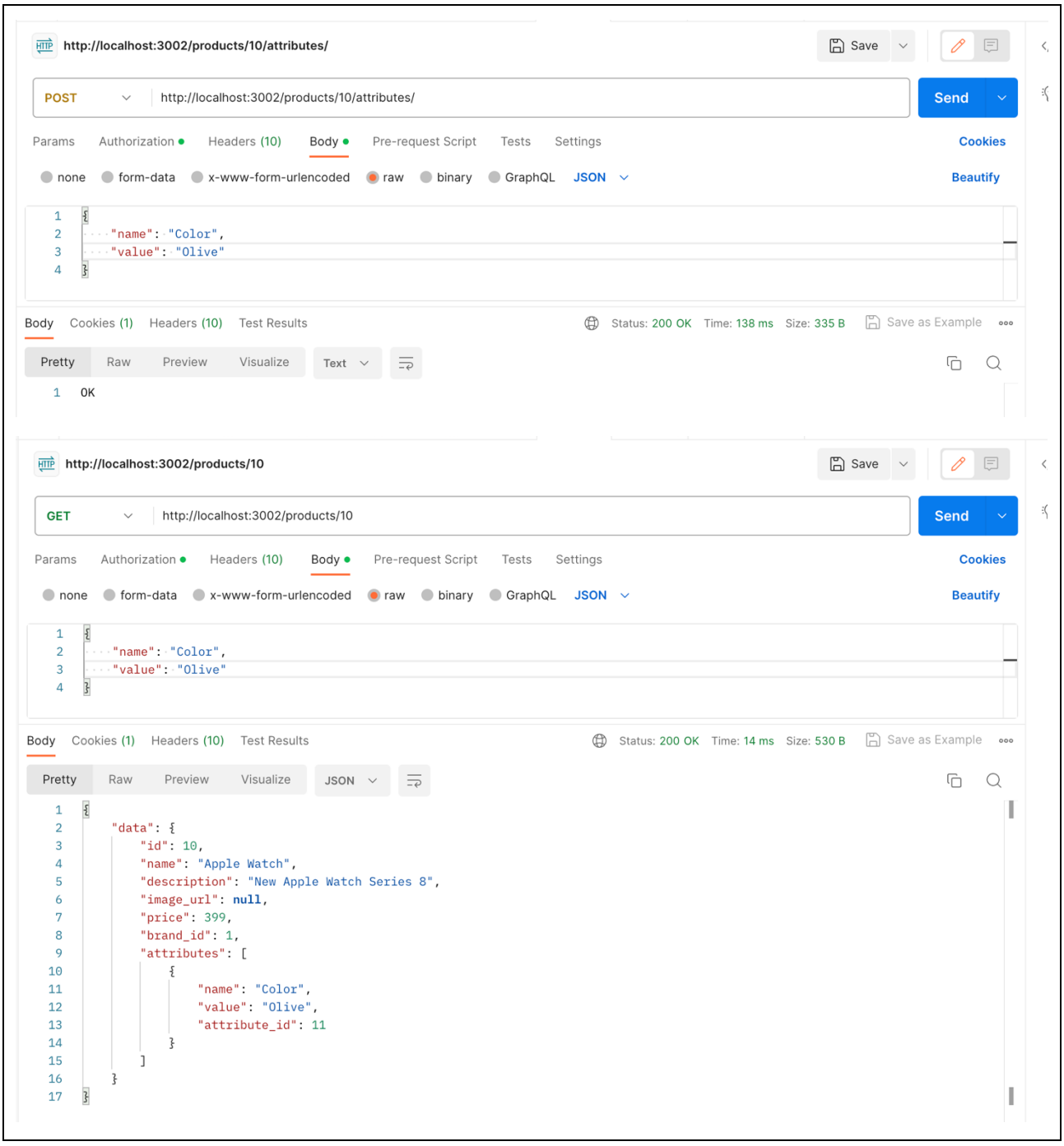

Рисунок 3.11 – Запит на додавання характеристики до продукту та перегляд інформації про продукт

3.4 Тестування маршрутів для роботи з коментарями

Система дозволяє переглядати та додавати коментарі до товарів. На рисунку 3.12 показано приклад запиту на отримання коментарів до певного продукту.

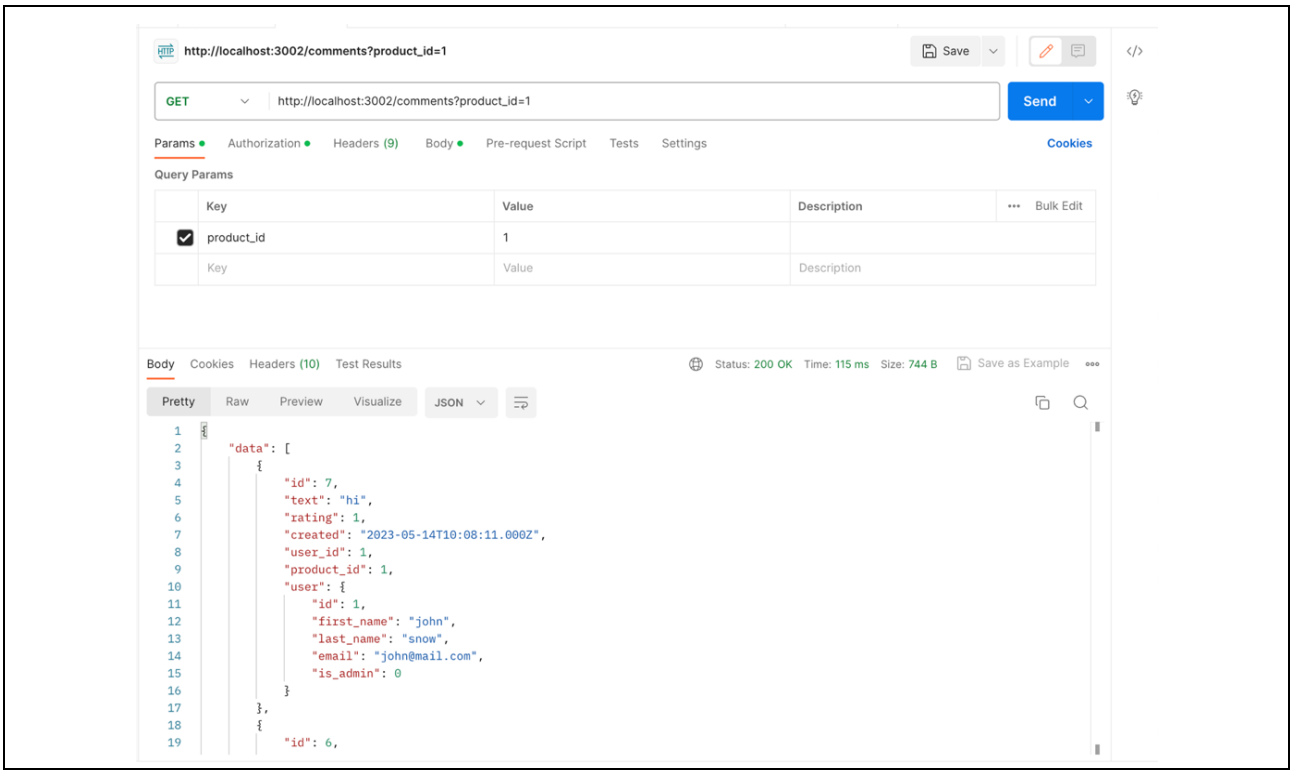

Рисунок 3.12 – Запит на отримання коментарів до продукту

На рисунку 3.13 показано приклад запиту на додавання коментаря до продукту.

У сутності коментар у поле “user_id” значення підставляється автоматично в залежності від авторизованого користувача.

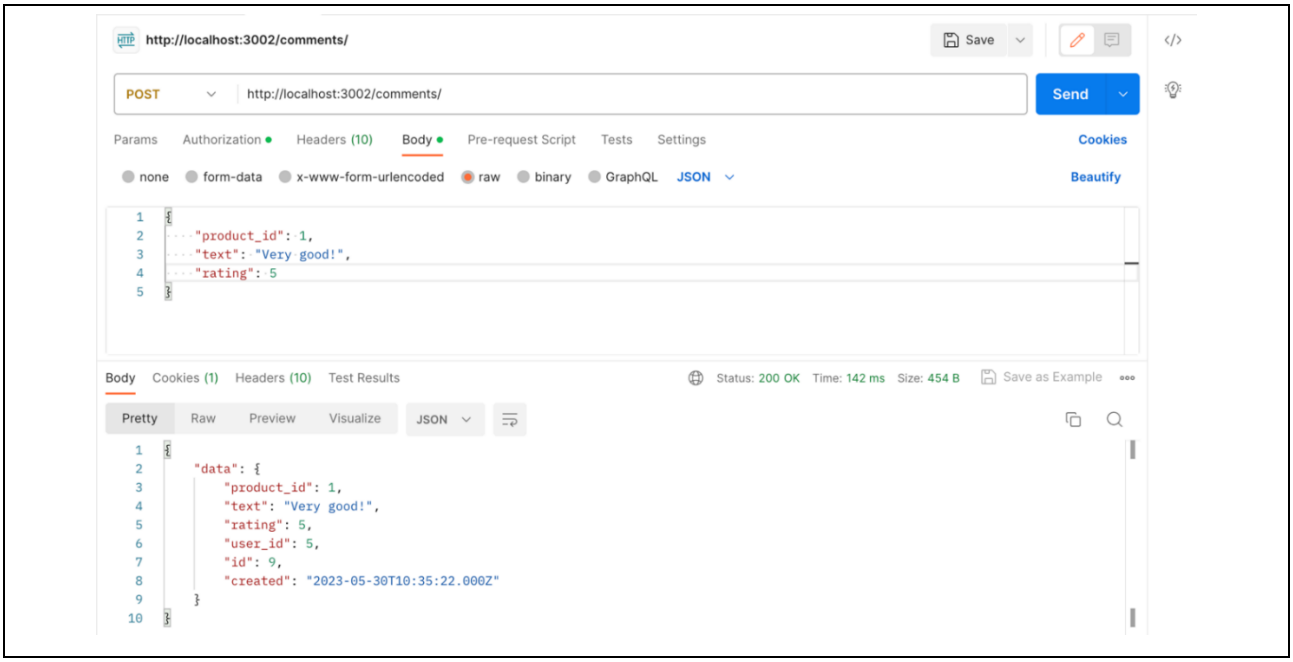

Рисунок 3.13 – Запит на додавання коментаря до продукту

3.5 Тестування маршрутів для роботи з ключовими словами

Система дозволяє створювати, додавати, та від’єднувати ключові слова від продуктів. На рисунку 3.14 показано приклад запиту на додавання ключового слова до продукту.

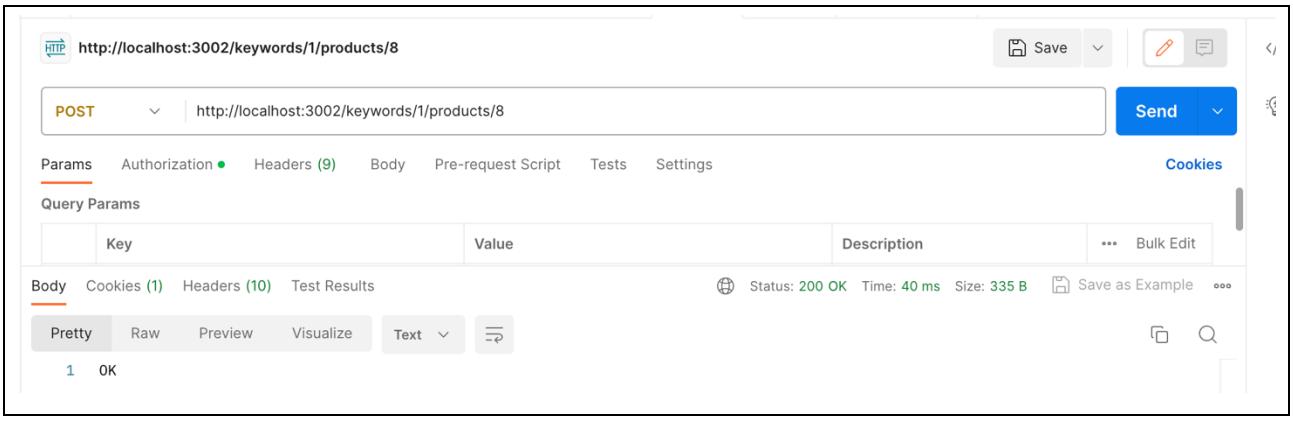

Рисунок 3.14 – Запит на додавання ключового слова до продукту

На рисунку 3.15 показано приклад запиту на від’єднання ключового слова від продукту.

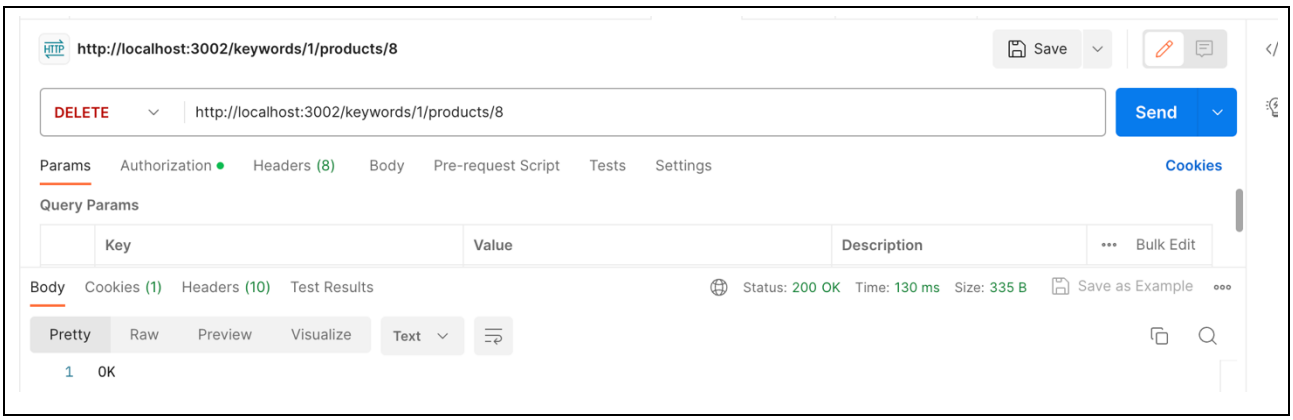

Рисунок 3.15 – Запит на від’єднання ключового слова від продукту

На рисунку 3.16 наведено приклад запиту на отримання продуктів, зв’язаних з ключовим словом.

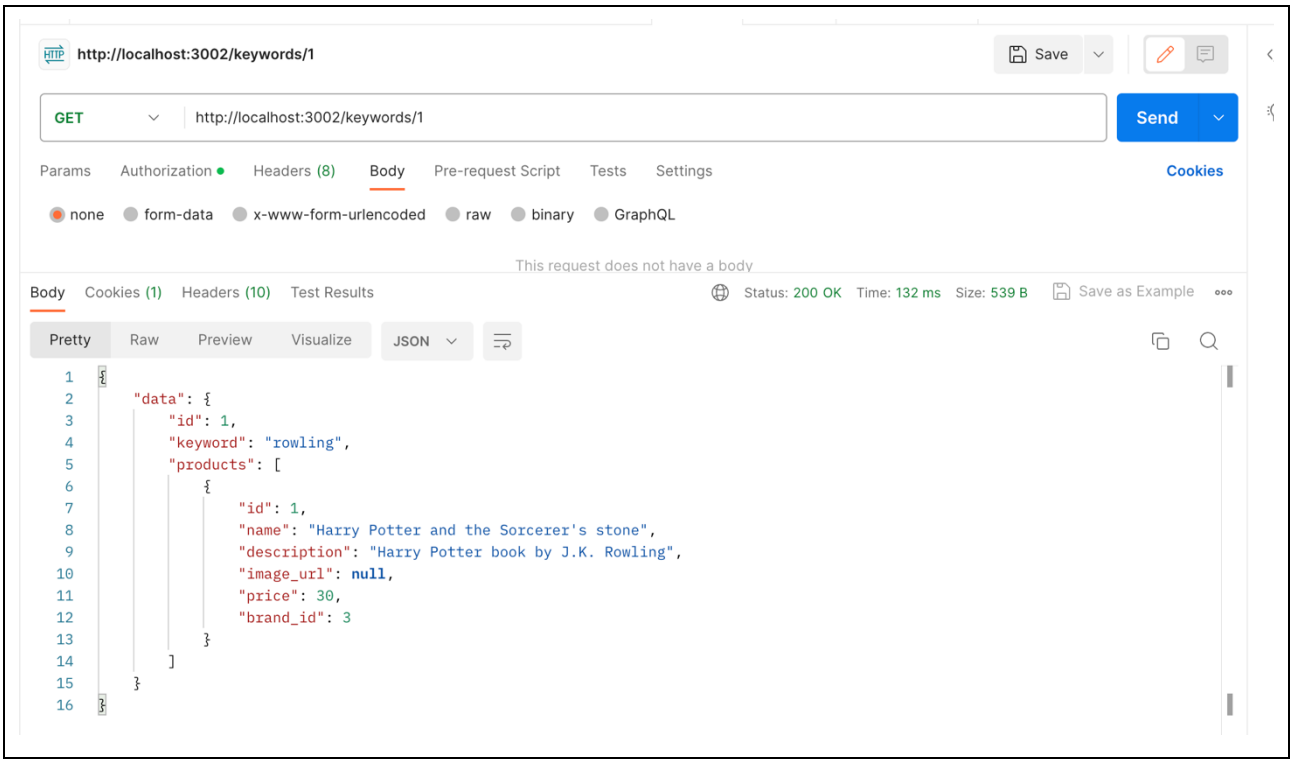

Рисунок 3.16 – Запит на отримання продуктів ключового слова

3.6 Тестування маршрутів для роботи з брендами

На рисунку 3.17 показано приклад запиту на отримання списку брендів.

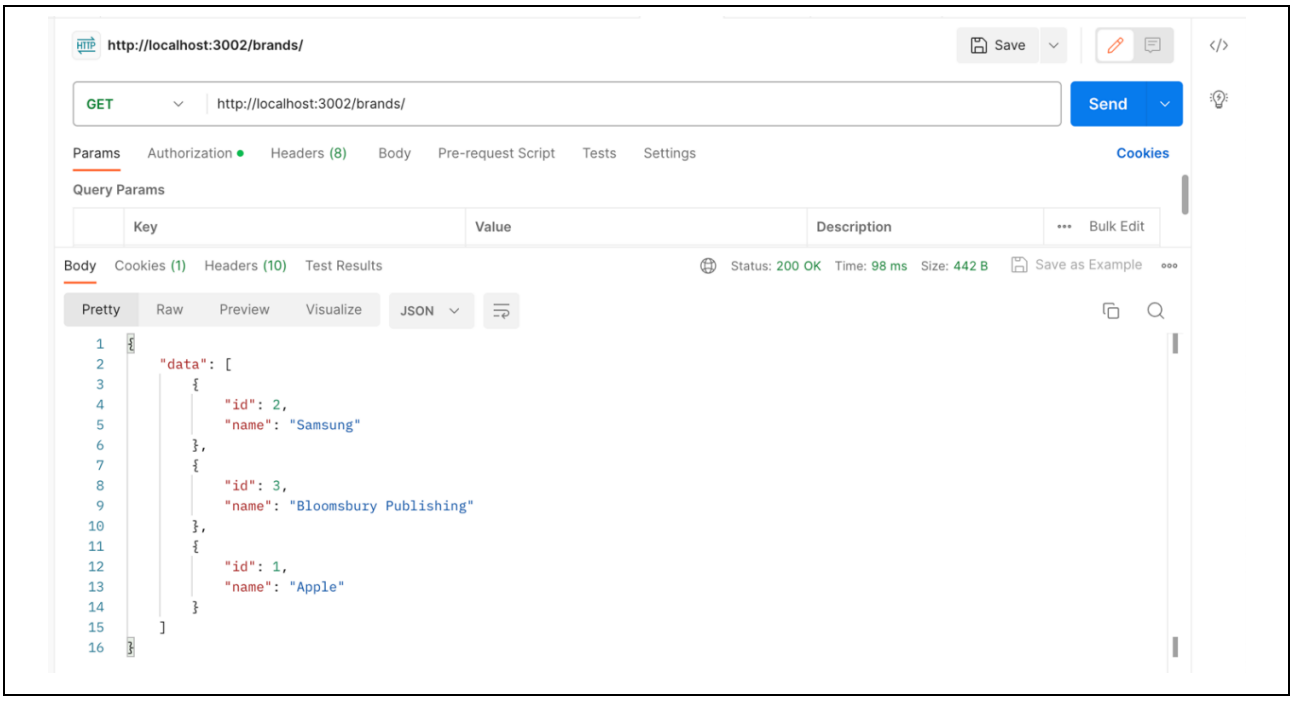

Рисунок 3.17 – Запит на отримання списку брендів

На рисунку 3.18 показано приклад запиту на отримання деталей про бренд.

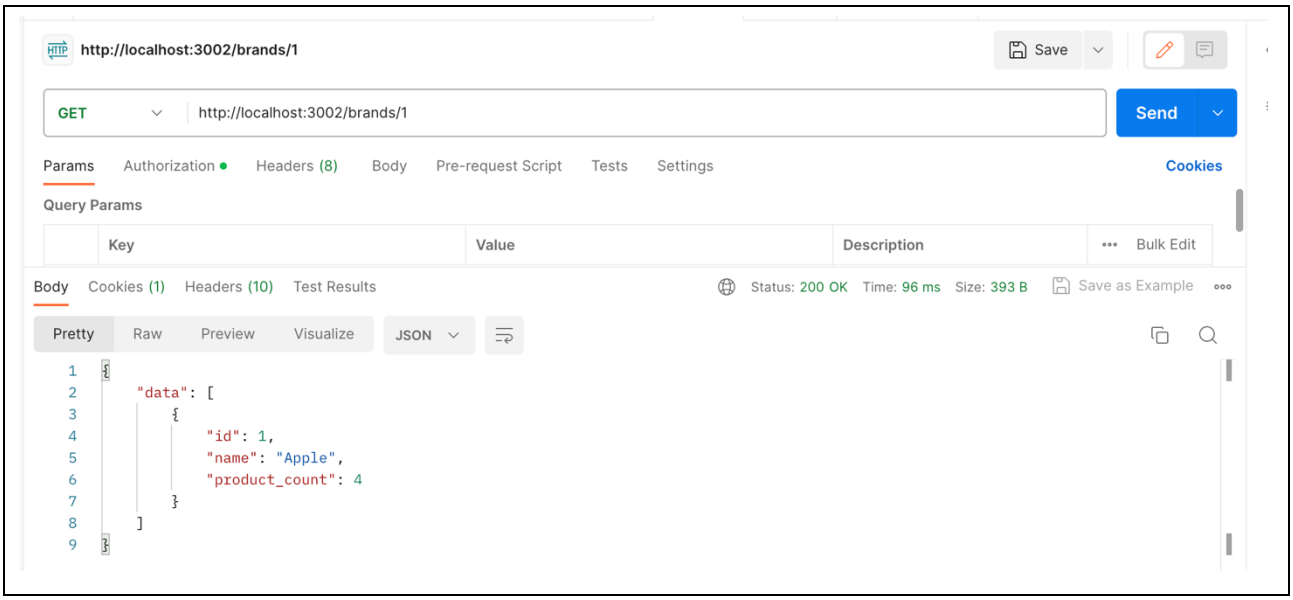

Рисунок 3.18 – Запит на отримання деталей про бренд

Висновки до розділу 3

У цьому розділі було проведено тестування різних маршрутів, які відповідають за роботу з користувачами, категоріями, продуктами, коментарями, ключовими словами та брендами. Маршрути було перевірено на правильну реалізацію функціоналу та відповідність очікуваним результатам. Тестування допомогло виявити й виправити можливі помилки та проблеми, а також підтвердити правильну роботу API. Завершення розділу тестування API є важливим кроком у розробці інтернет магазину, оскільки воно гарантує, що API працює належним чином та забезпечує необхідний функціонал для взаємодії з клієнтами.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи була проведена розробка серверної частини інтернет магазину. Було оглянуто функціонал, розроблено API для роботи з системою, оглянуто схему даних та схему маршрутів API. Було використано потужні технології, такі як Node.js, Express, MySQL та Knex.js, для забезпечення ефективності та надійності роботи системи. Також було наведено огляд цих та аналогічних технологій, що дозволило вибрати оптимальний набір для реалізації поставлених завдань.

У роботі були сформульовані та виконані вимоги до програми, що детально описують необхідний функціонал. Було розроблено серверну частину інтернет магазину, яка забезпечує взаємодію з клієнтами. Розроблені маршрути API дозволяють користувачам працювати з різними сутностями, такими як користувачі, категорії, продукти, коментарі, ключові слова та бренди. Компоненти API були перевірені та протестовані для забезпечення їх надійності та коректності роботи.

Результатом виконання цієї роботи є створений потужний серверний застосунок для роботи інтернет магазину, який здатний задовольнити потреби користувачів та забезпечити зручну та надійну платформу для покупок. Розроблені компоненти та функціонал серверної частини можуть бути подальше розширені та вдосконалені для відповідності потребам та вимогам користувачів. Програма відповідає поставленим вимогам. Результати роботи API-маршрутів підтверджують їх правильну роботу та функціональність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Introduction to Node.js. URL: https://nodejs.dev/en/learn/ (дата звернення: 23.02.2023).

2. Express/Node introduction. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction (дата звернення: 23.02.2023)

3. Comparison of TOP 14 Node.js Frameworks (2023) – Inventorsoft. URL: https://inventorsoft.co/blog/top-14-node-js-frameworks-comparisson (дата звернення: 23.02.2023).

4. API. URL: https://en.wikipedia.org/wiki/API (дата звернення: 24.02.2023).

5. 9 HTTP methods and how to use them. URL: https://testfully.io/blog/http-methods/ (дата звернення: 24.02.2023).

6. 9 Method Definitions. URL: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html (дата звернення: 24.02.2023).

7. HTTP request methods. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (дата звернення: 24.02.2023).