



Міністерство освіти і науки України  
Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”  
Факультет інформатики та обчислювальної техніки  
Кафедра інформаційних систем та технологій

## Лабораторна робота №6

### Тестування програмного забезпечення

Виконали:  
Палінчак І.В.  
Хохол М.В.

Перевірив:  
Нечай Д.О.

Київ 2024

Завдання:

1. Повністю покрити якийсь з модулів UNIT-тестами
2. Налаштувати інтеграційні тести таким чином, щоб ви могли запустити всі залежні модулі перед тестами. (наприклад підключитись до бази, запустити хтtp сервер і тестувати його роботу)
3. Налаштувати E2E тестування, що б охоплювало всі елементи системи
4. Провести мутаційне тестування та підготувати невеликий репорт про те як ефективно ви покрили код тестами, яка ефективність цих тестів, що ще варто зробити

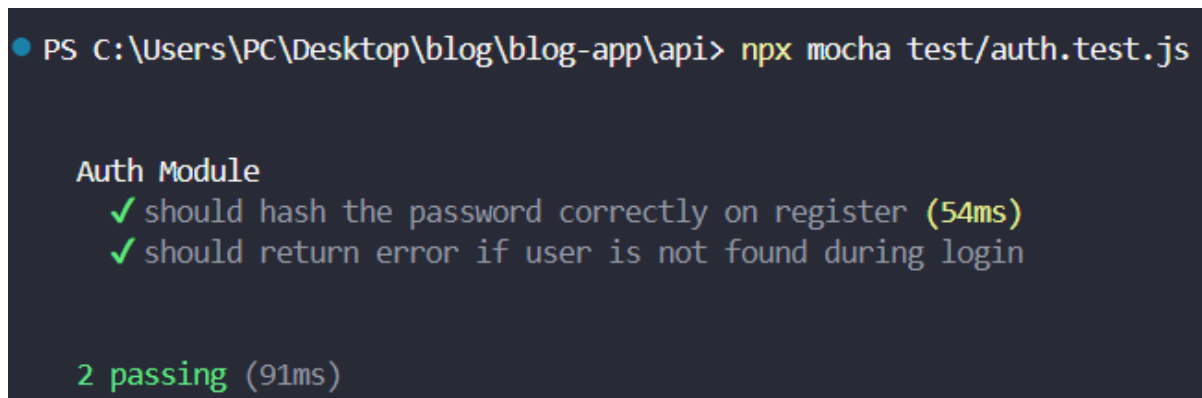
## Хід роботи

Покриваємо логіку реєстрації та логіну, використовуючи такі бібліотеки, як mocha, chai для Node.js.

Встановлено необхідні бібліотеки для тестування:

mocha chai sinon

Створено Unit-тести для перевірки коректного хешування паролю при реєстрації, помилка, якщо користувача не знайдено під час входу в систему,



```
PS C:\Users\PC\Desktop\blog\blog-app\api> npx mocha test/auth.test.js

Auth Module
  ✓ should hash the password correctly on register (54ms)
  ✓ should return error if user is not found during login

2 passing (91ms)
```

Рисунок 1 – Успішне проходження тестів

## Інтеграційні тести

Тестуємо модулі auth, users, posts разом, підключаючи HTTP-сервер і базу даних (MongoDB).

За допомогою попередньо інстальованих бібліотек для тестування в проєкт проведено інтеграційні тести в integration.test.js, перевірки реєстрації користувача в системі та перевірки у разі невдалого входу з неправильними обліковими даними.

```
PS C:\Users\PC\Desktop\blog\blog-app\api> npx mocha test/integration.test.js
Backend is running.

Integration Tests
(node:27136) [DEP0170] DeprecationWarning: The URL mongodb://admin:f6YpNu2vwjW8XmE9@cluster0-shard-00-00.8eqi0.mongodb.net:27017,cluster0-shard-00-02.8eqi0.mongodb.net:27017,cluster0-shard-00-01.8eqi0.mongodb.net:27017/blogapp?authSource=admin&replicaSet=atlas-3s25cg-shard-0&ssl=true is invalid. Future versions of Node.js will throw an error.
(Use `node --trace-deprecation ...` to show where the warning was created)
Connected to MongoDB
Response status: 200
Response body: {
  profilePic: '',
  _id: '677844983b4dd56a005024db',
  username: 'test',
  email: 'test@mail.com',
  password: '$2b$10$zeJkPTJM83tG.1gxtkgBTud0xm4wmn6.3rCfoH1.z.bdr7AU2YDKm',
  createdAt: '2025-01-03T20:12:08.371Z',
  updatedAt: '2025-01-03T20:12:08.371Z',
  __v: 0
}
✓ should register a user successfully (615ms)
✓ should fail login with wrong credentials (144ms)

2 passing (766ms)
```

Рисунок 2 – Успішне проходження інтеграційних тестів

За допомогою інструменту тестування Cypress виконано E2E тестування

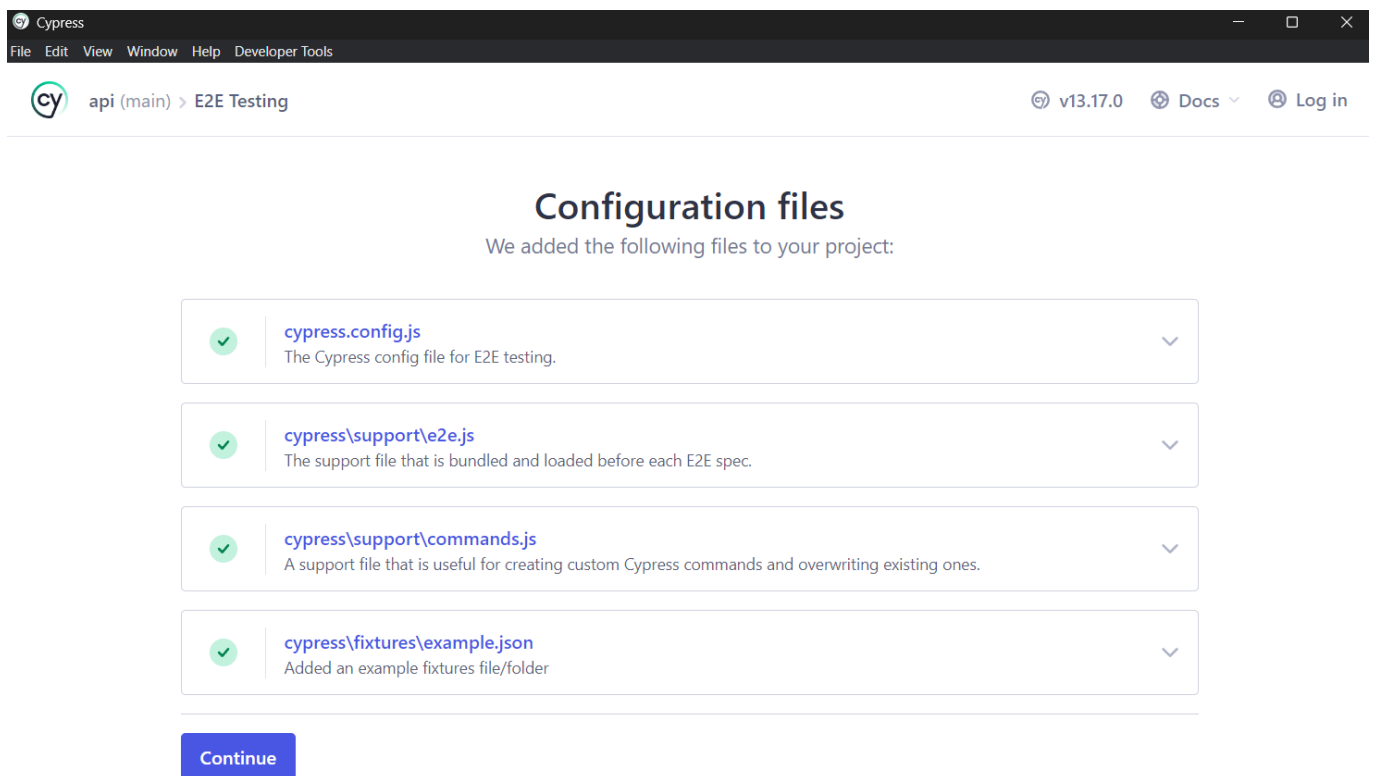


Рисунок 3 – Вигляд налаштування конфігураційних файлів для Cypress

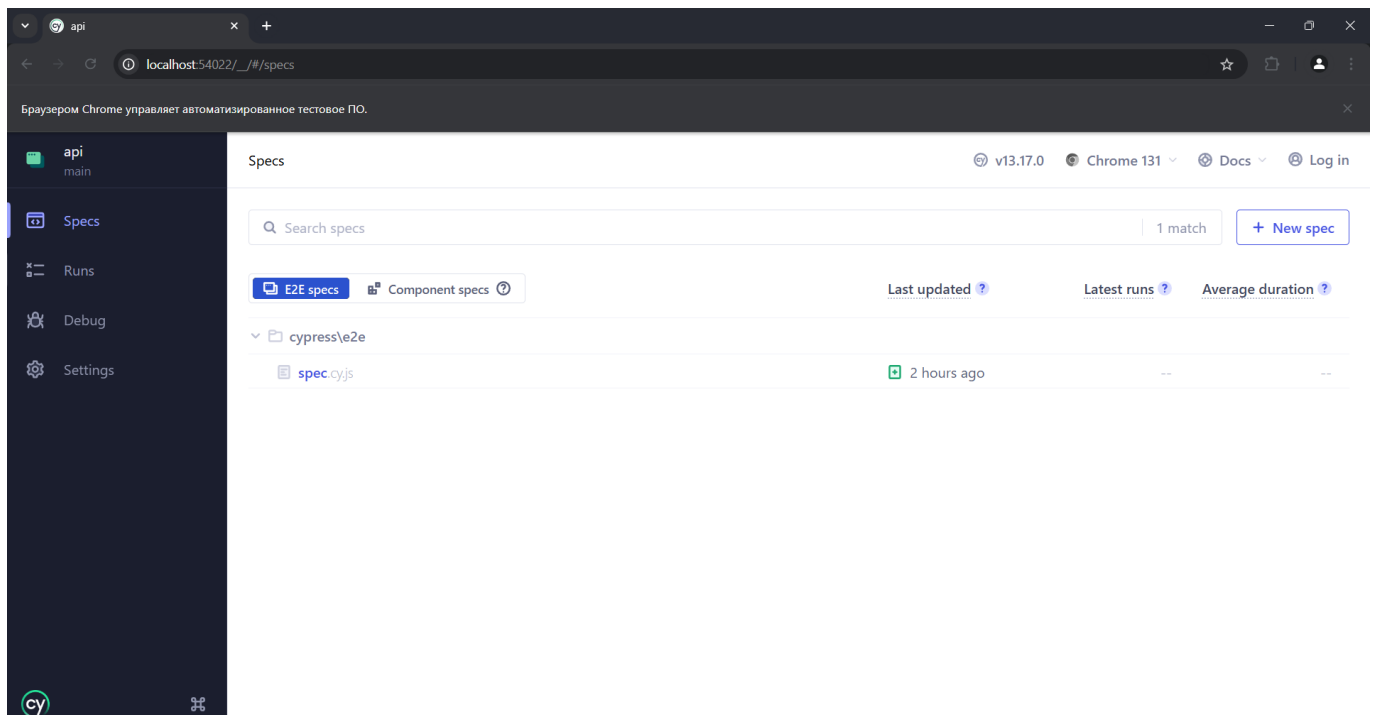


Рисунок 4 –Створений E2E specs тест

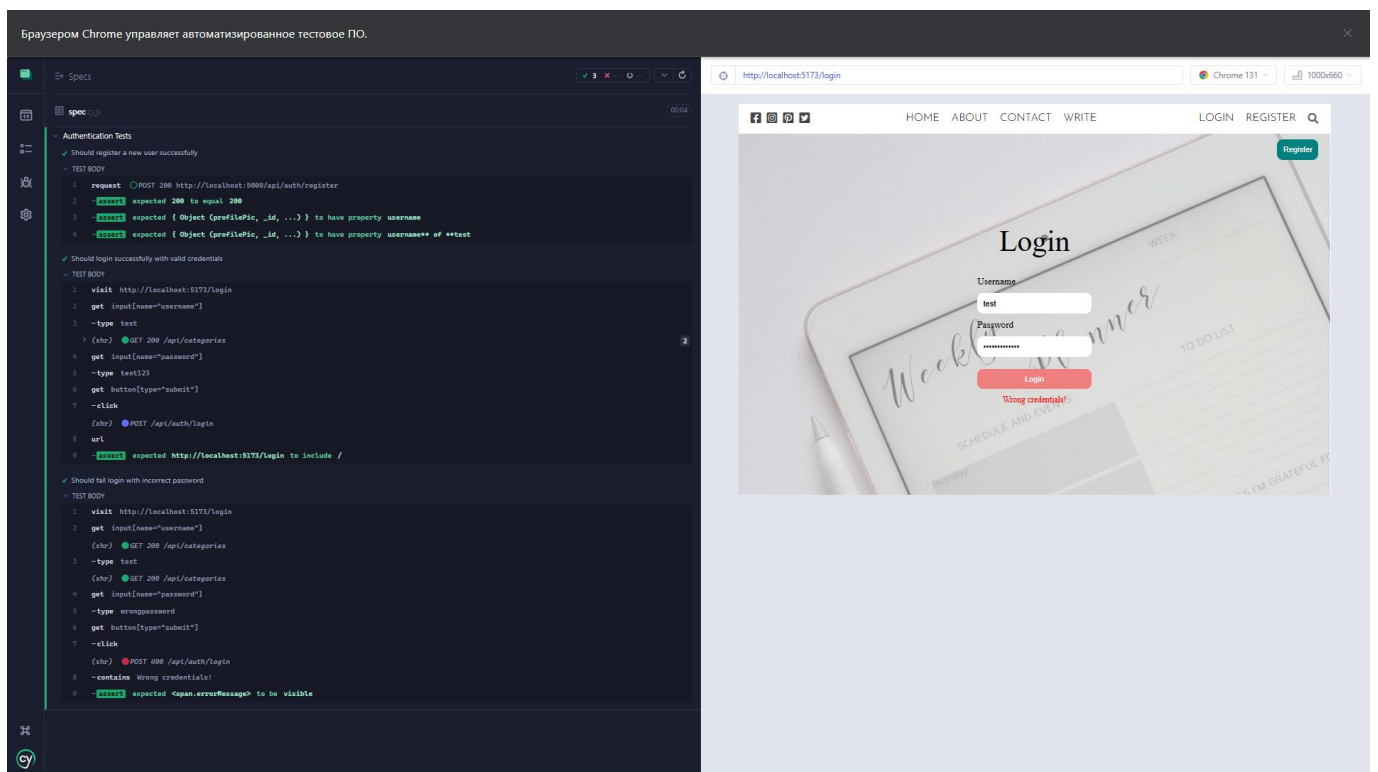


Рисунок 5 – Успішне проходження всіх тестів

Тестування створеного API за допомогою Postman. **Postman** — це популярний інструмент для тестування API, який дозволяє здійснювати запити до веб-сервісів і аналізувати відповіді. Він підтримує всі основні методи HTTP (GET, POST, PUT, DELETE тощо) і надає зручний інтерфейс для роботи з параметрами запитів, заголовками та іншими важливими аспектами API.

Тестування реєстрації користувача:

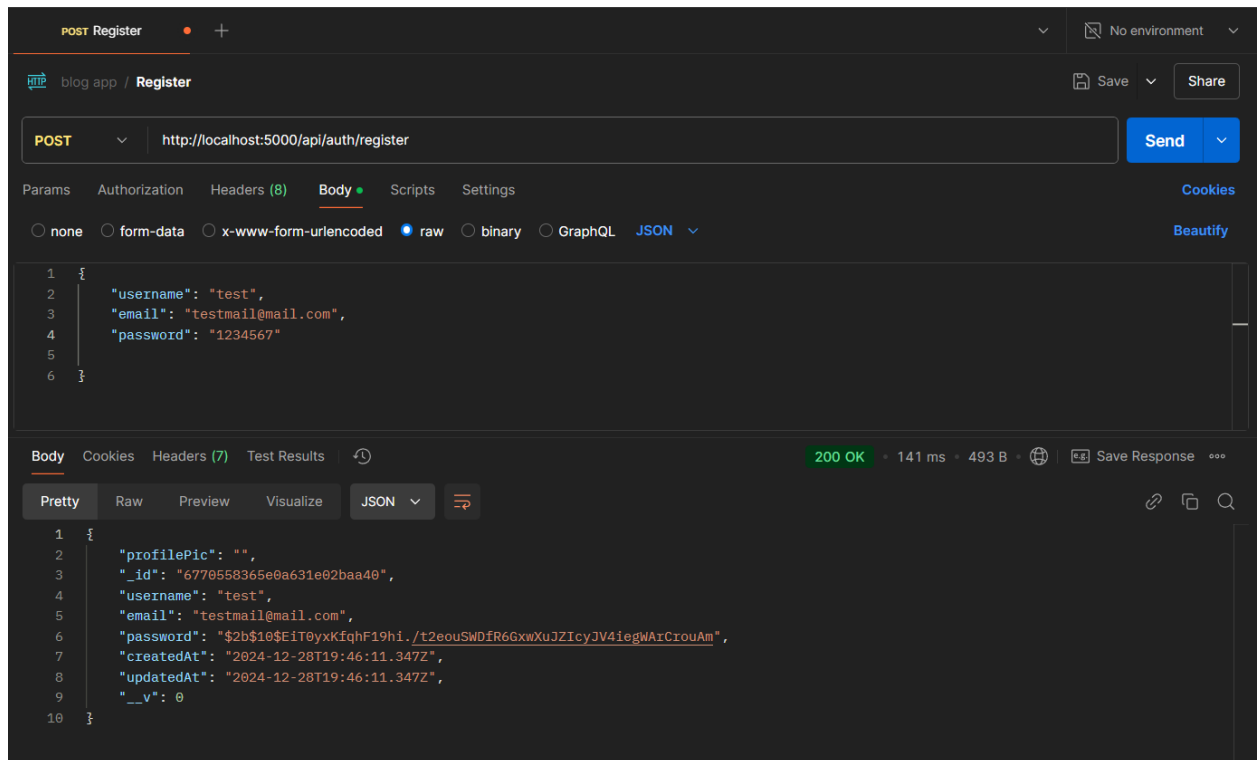


Рисунок 6 – Успішна реєстрація користувача

Аутентифікація в акаунт користувача:

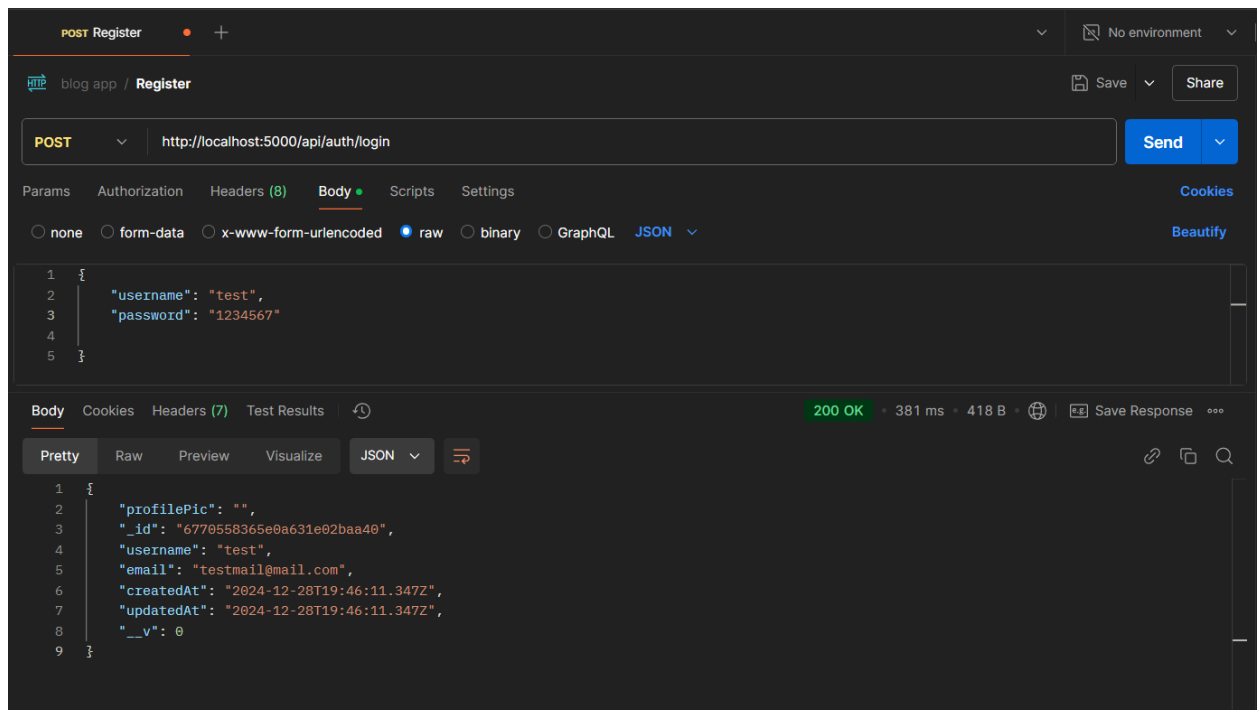


Рисунок 7 – Успішна авторизація створеного користувача

Перевірка на негативний сценарій:

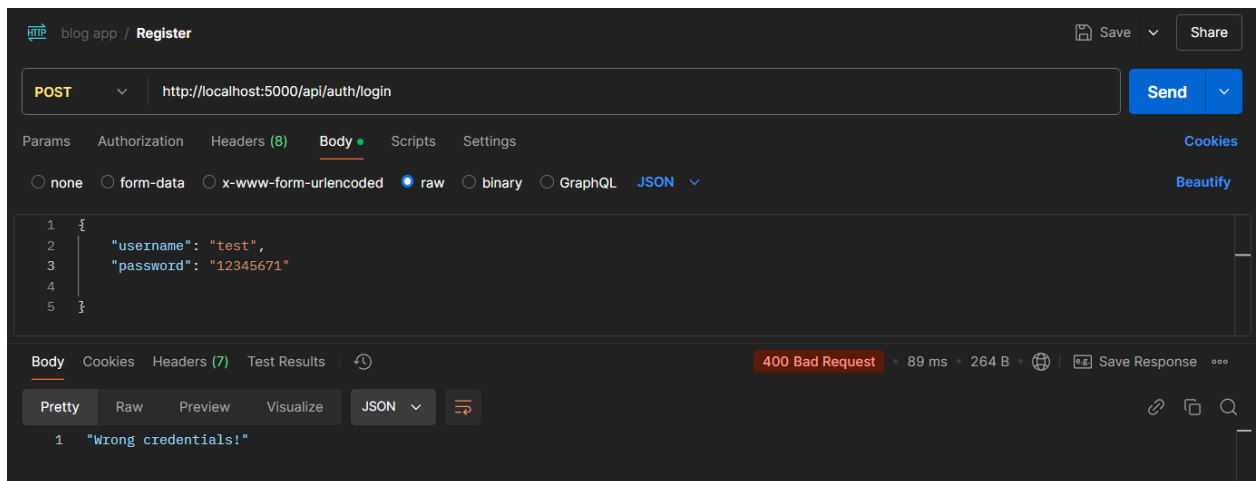


Рисунок 8 – Помилка при вводиті даних для авторизації

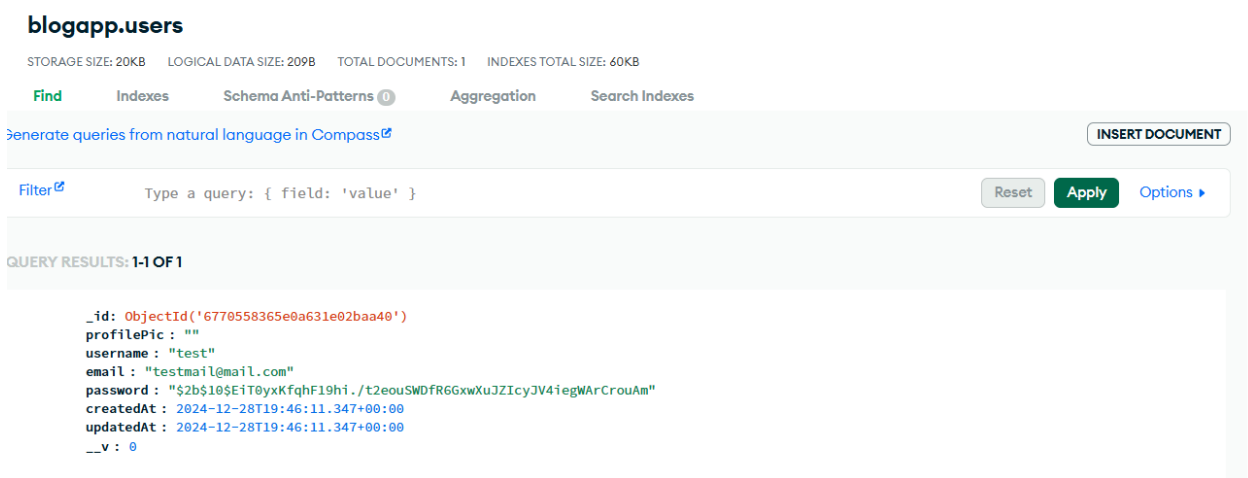


Рисунок 9 – Вигляд створеного користувача в БД

Перегляд користувача:

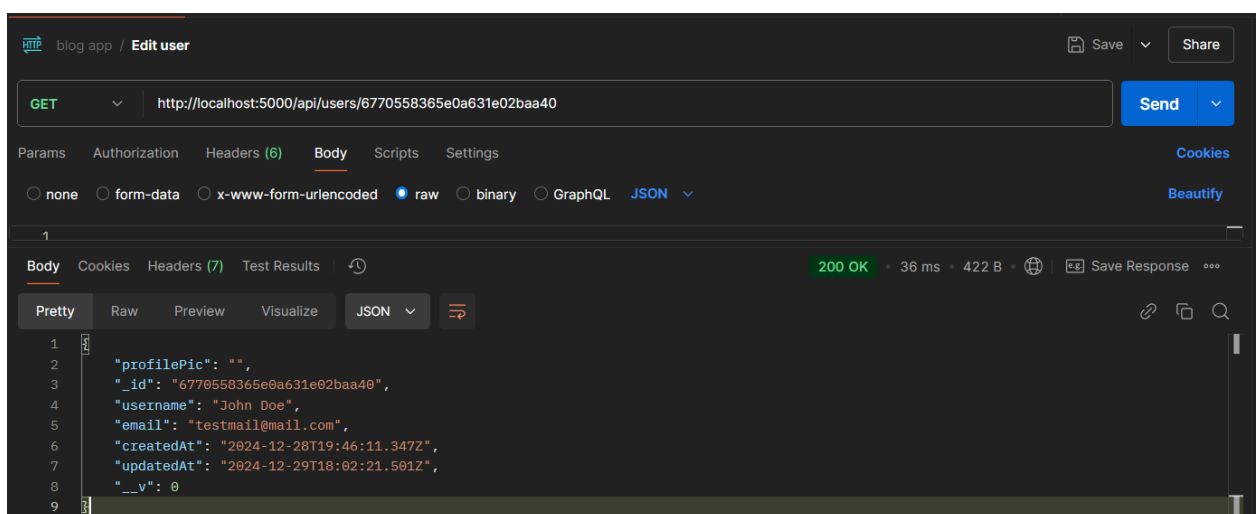


Рисунок 10 – Вигляд створеного користувача в БД за ідентифікатором

Зміна інформації про користувача:

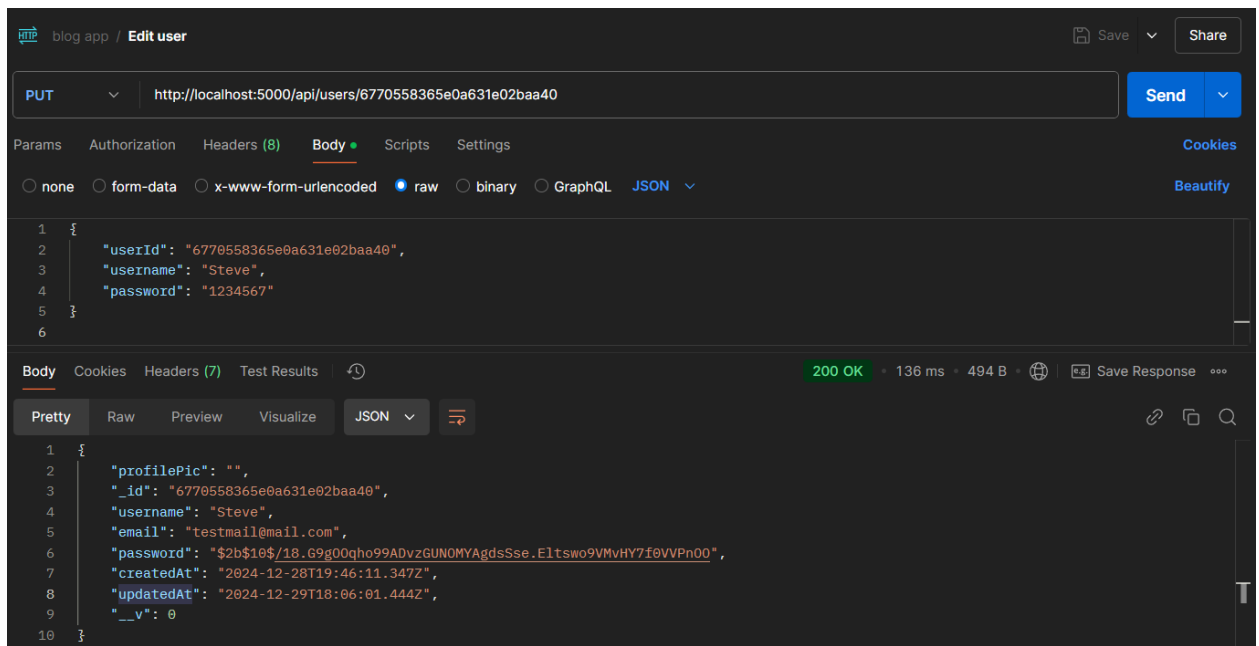


Рисунок 11 – Успішна зміна інформації про користувача

Перевірка негативного сценарію:

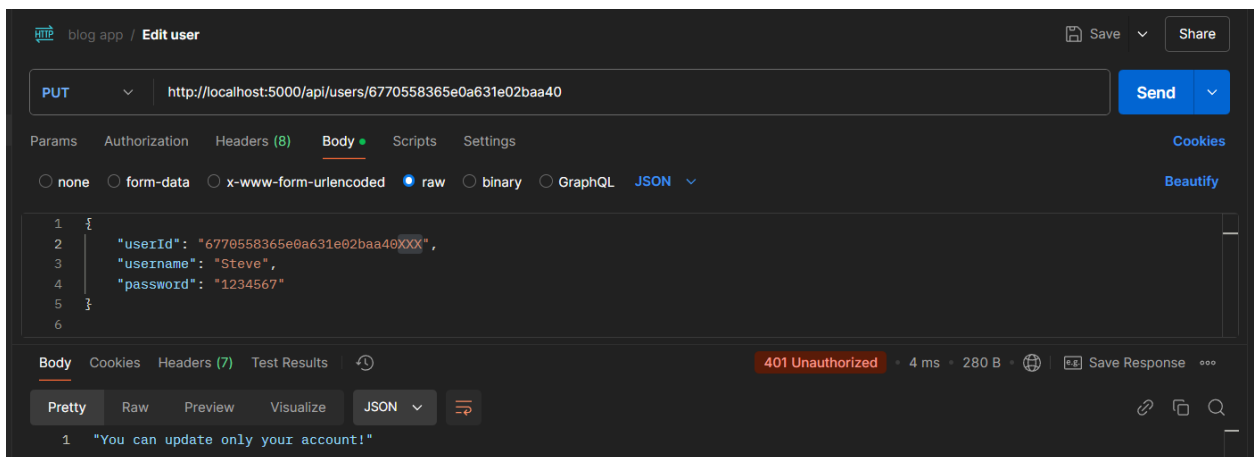


Рисунок 11 – Зміна інформації про користувача не вдалася через некоректні дані

Кінцевий вигляд після внесення змін в БД:



Рисунок 12 – Вигляд інформації в БД

Видалення профілю користувача:

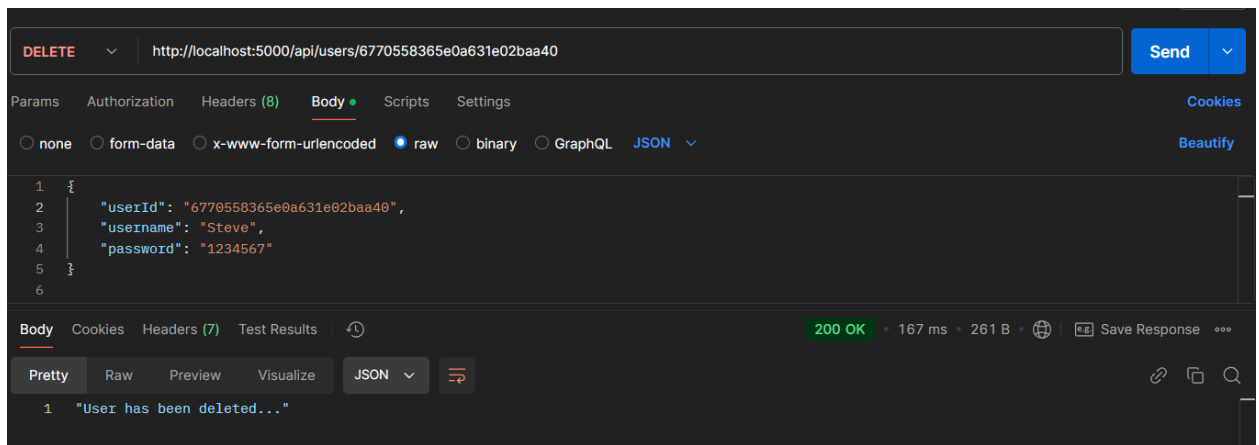


Рисунок 13 – Успішне видалення профілю користувача

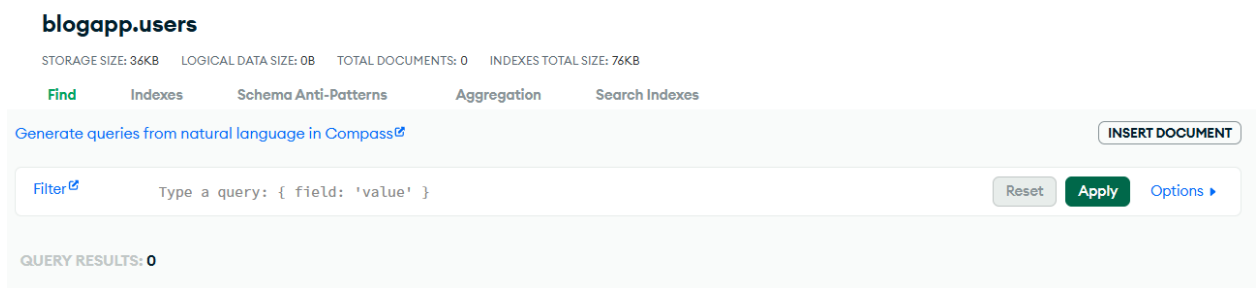


Рисунок 13 – Кінцевий вигляд в БД

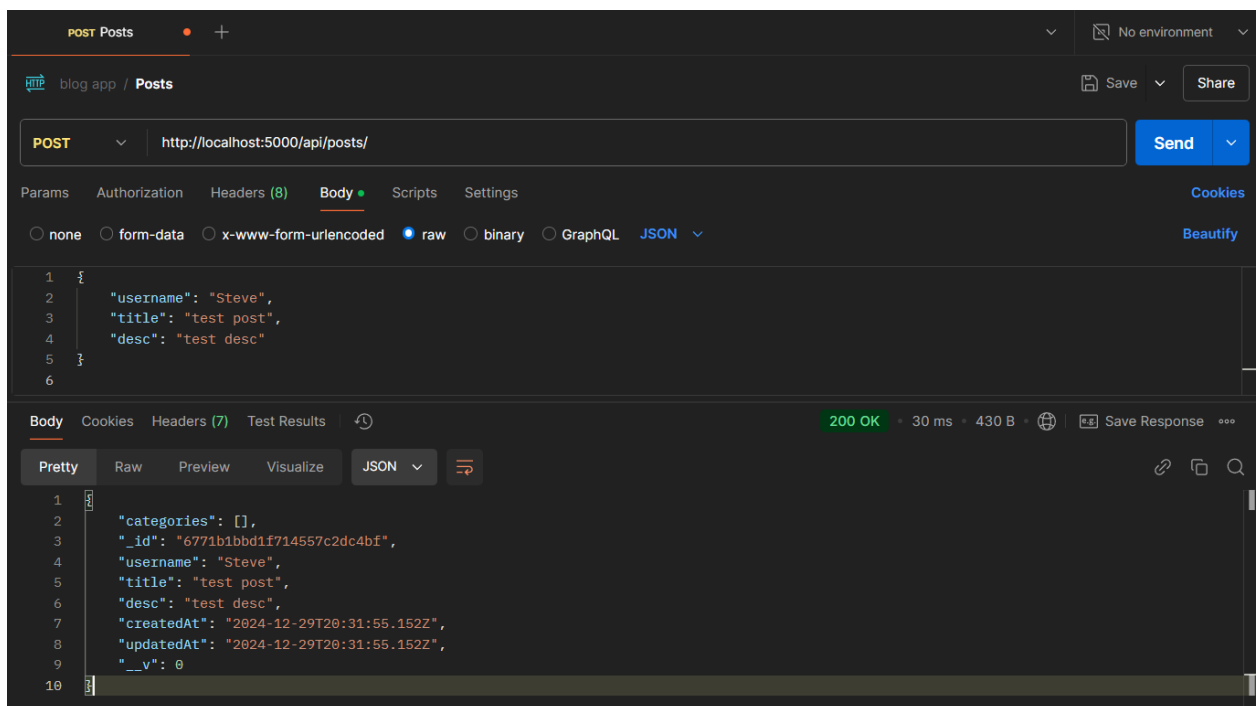


Рисунок 14 – Створення поста



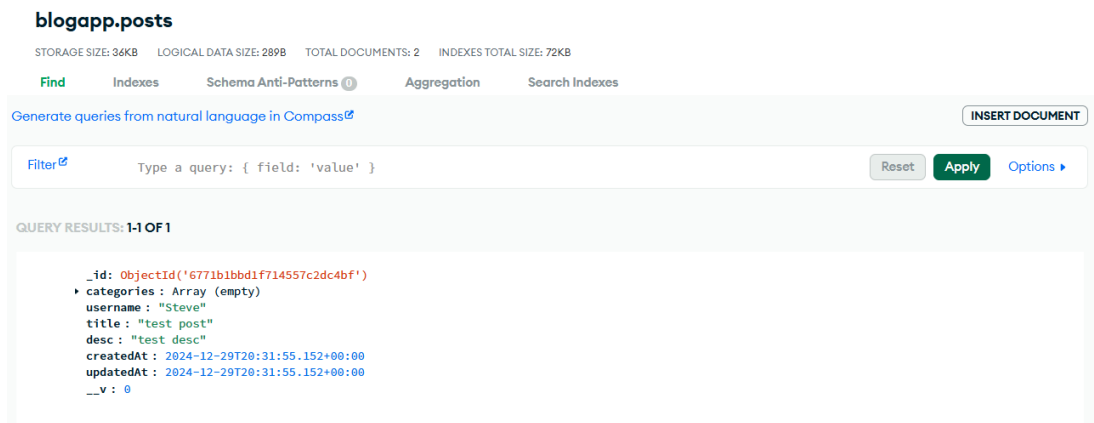


Рисунок 15 – Вигляд поста в БД

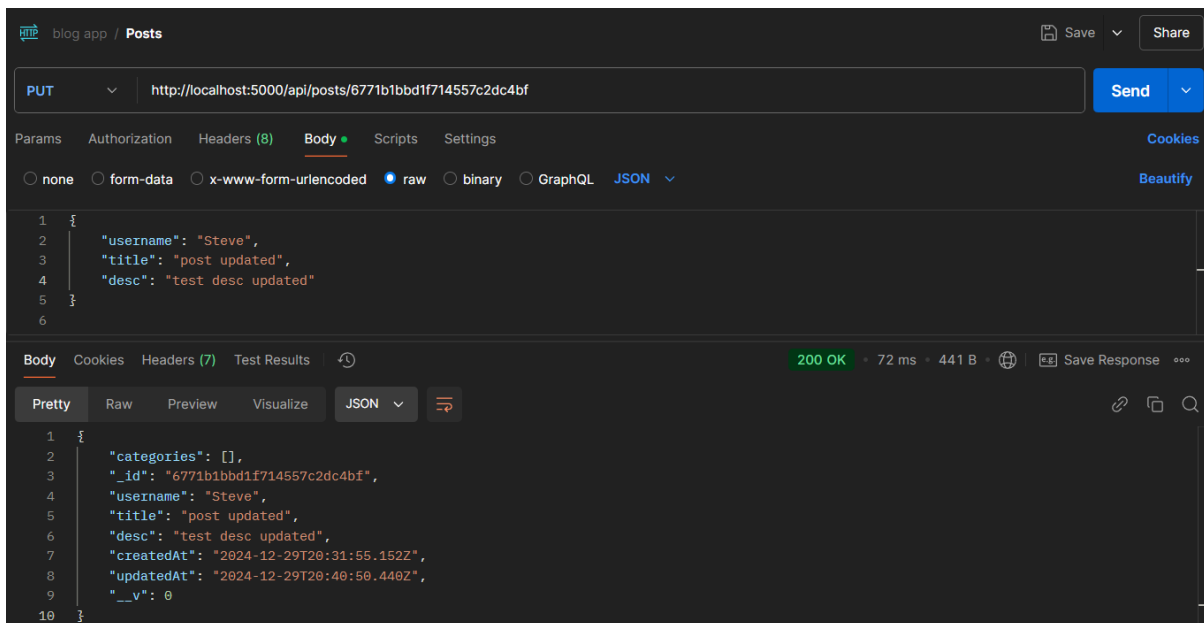


Рисунок 16 – Редагування поста

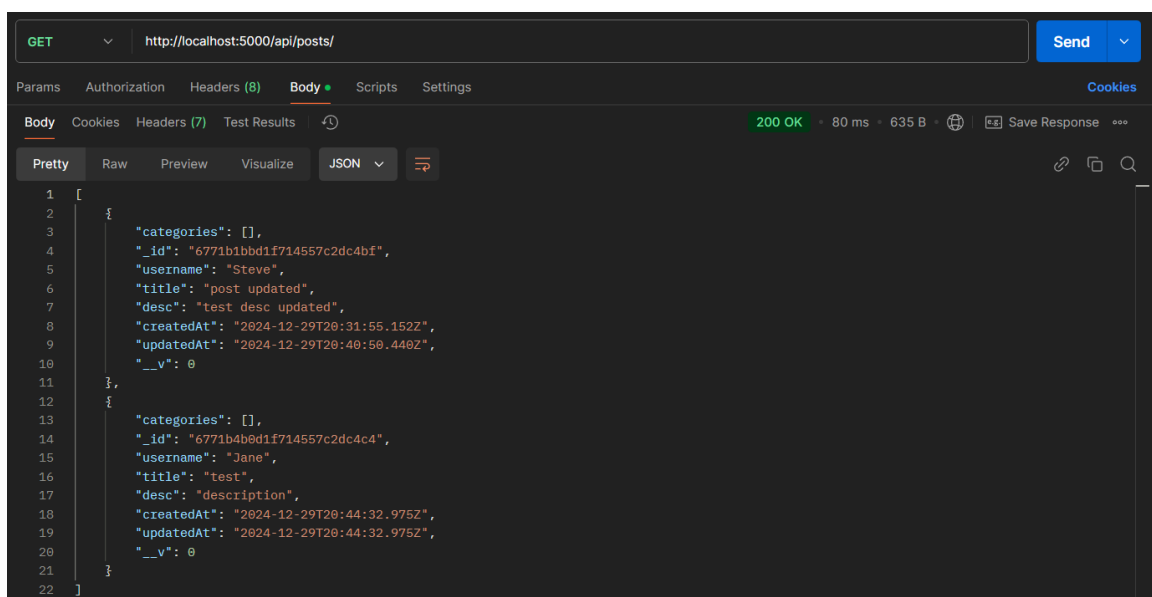


Рисунок 17 – Перегляд всіх постів

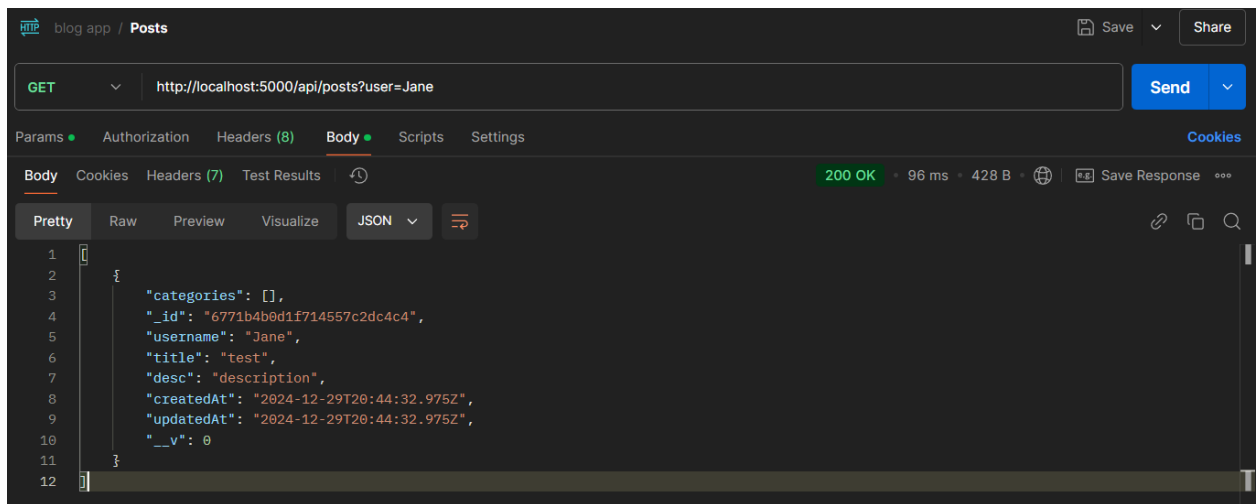


Рисунок 18 – Перегляд постів від користувача “Jane”

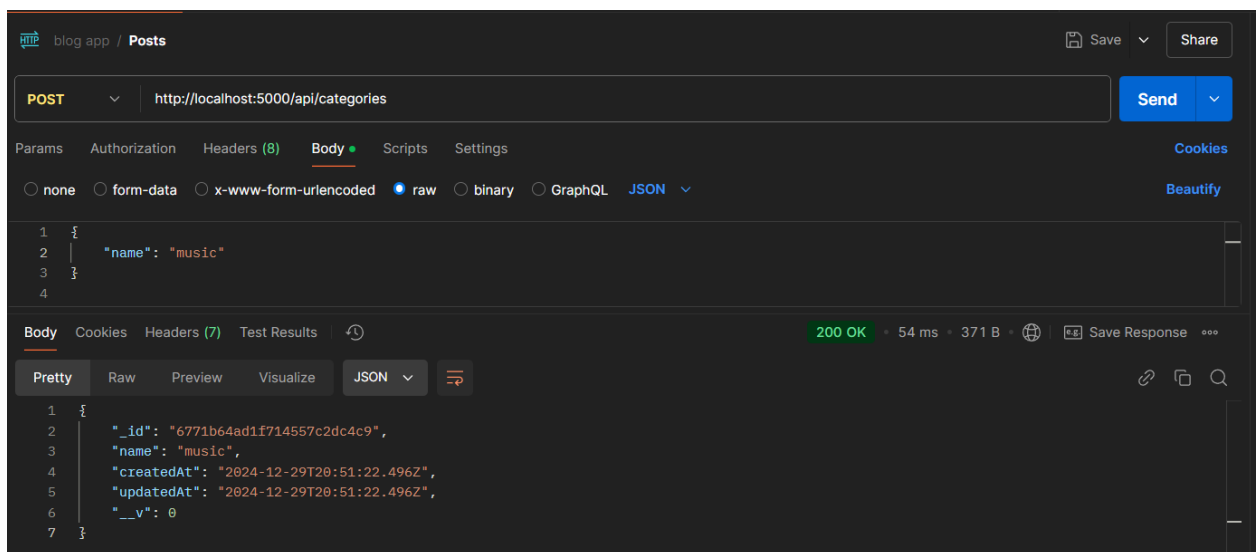


Рисунок 19 – Створення категорії

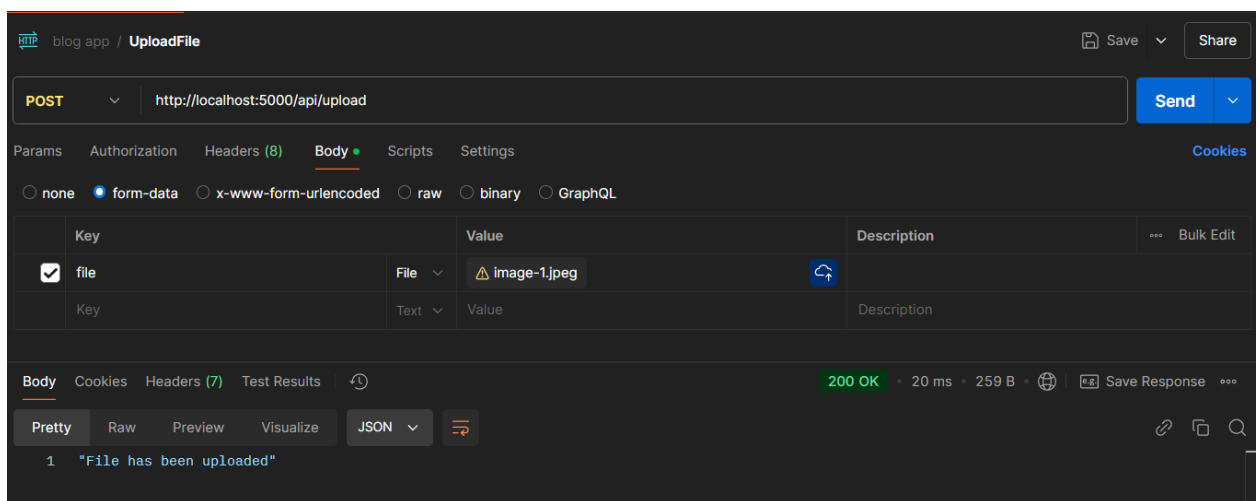


Рисунок 20 – Створення категорії

Висновок:

У результаті виконаної роботи були досягнуті значні успіхи в покритті тестами та забезпеченні якості коду. Один із ключових модулів проекту було повністю покрито UNIT-тестами, що дозволяє забезпечити правильність його роботи на рівні окремих функцій. Інтеграційні тести були налаштовані таким чином, щоб перевірити взаємодію з іншими модулями, включаючи підключення до бази даних та запуск HTTP-сервера, що забезпечує правильну роботу всіх залежних компонентів. Також було налаштовано E2E тестування, яке охоплює всі елементи системи та дозволяє перевірити коректність роботи програми в цілому.

Для підвищення ефективності тестів було проведено мутаційне тестування, що дозволило оцінити, наскільки добре покриття тестами здатне виявляти дефекти в коді. Було підготовлено звіт, в якому зазначено ефективність проведених тестів, а також рекомендації щодо подальшого поліпшення покриття та додавання нових тестів для підвищення надійності системи.