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

Звіт до комп'ютерного практикуму №6

З дисципліни «Основи Back-end технологій»

Виконав(ла) *ІП-24 Малий Олександр Сергійович*
(шифр, прізвище, ім'я, по батькові)

Прийняв *викладач Зубко Р. А.*
(посада, прізвище, ім'я, по батькові)

Київ 2025

Лабораторна робота №6

GraphQL. Створення Schema GraphQL та Resolvers. Створення Query та Mutation.

Завдання.

1. на своїй БД (розробленої в лаб. роб. #5) за допомогою Schema Definition Language (SDL) створити схему GraphQL;
2. додати Resolvers для виконання операцій GraphQL;
3. створити та виконати Query та Mutation для виконання операцій додавання, редагування та видалення інформації (CRUD) в БД;
4. виконати дослідження роботи створених query та mutation за допомогою Postman.

Хід роботи

Для початку встановлюю необхідні пакети (рис. 1). Створюю schema.js для схеми GraphQL та resolvers.js для резолверів в папці graphql (рис. 2). Імпортую вміст файлів та створюю підсервер через ApolloServer (рис. 3). Перевіряємо підключення перейшовши за адресою серверу (рис. 4).

```
"dependencies": {
  "apollo-server-express": "^3.13.0",
  "bcrypt": "^5.1.1",
  "cookie-parser": "^1.4.7",
  "cors": "^2.8.5",
  "dotenv": "^16.4.7",
  "express": "^4.21.2",
  "express-graphql": "^0.12.0",
  "express-validator": "^7.2.1",
  "graphql": "^16.10.0",
  "http-errors": "^2.0.0",
  "http-status-codes": "^2.3.0",
  "jsonwebtoken": "^9.0.2",
  "mongoose": "^8.10.0",
  "swagger-jsdoc": "^6.2.8",
  "swagger-ui-express": "^5.0.1"
},
```

Рисунок 1 – Нові залежності проєкту

Рисунок 2 – Нові файли для graphql

```
39  const server : ApolloServer<any>  = new ApolloServer( config: {
40    typeDefs,
41    resolvers,
42  });
43
44  await server.start();
45  server.applyMiddleware( {app, ...rest}: { app });
```

Рисунок 3 – Створення підсервера через ApolloServer

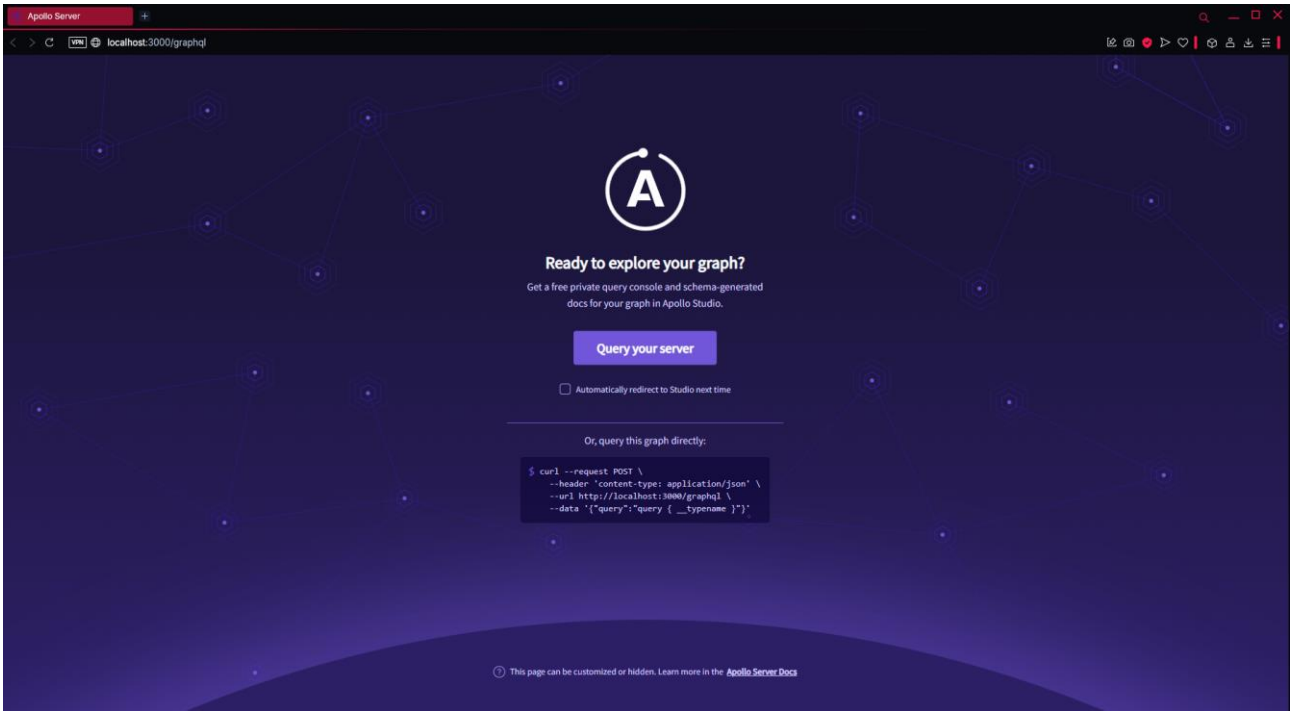

Рисунок 4 – Перевірка роботи graphql за посиланням

Після успішного встановлення серверу з graphql переходимо до тестування через Postman (рис. 5 – 11).

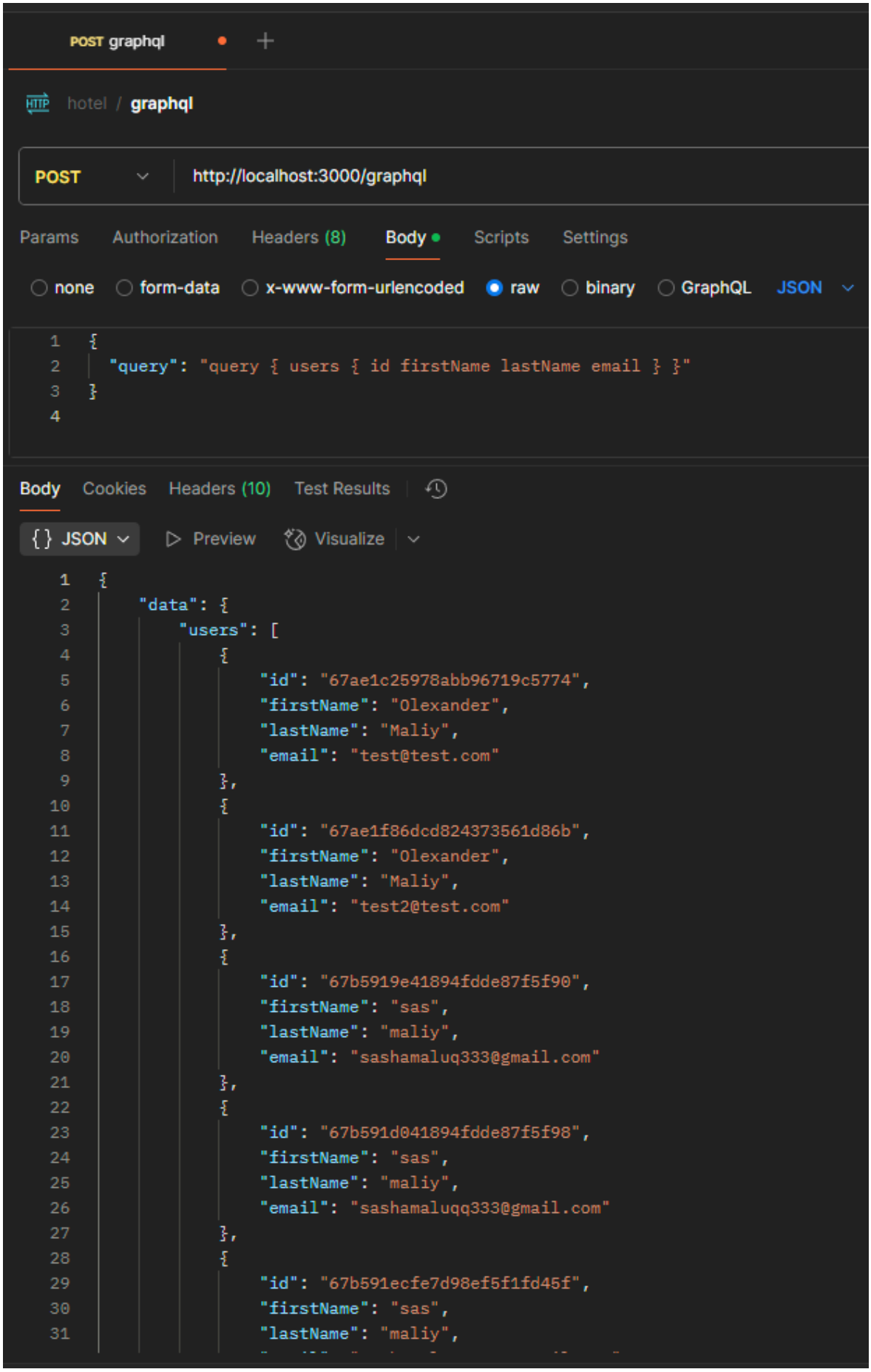

Рисунок 5 – Запит отримання користувачів

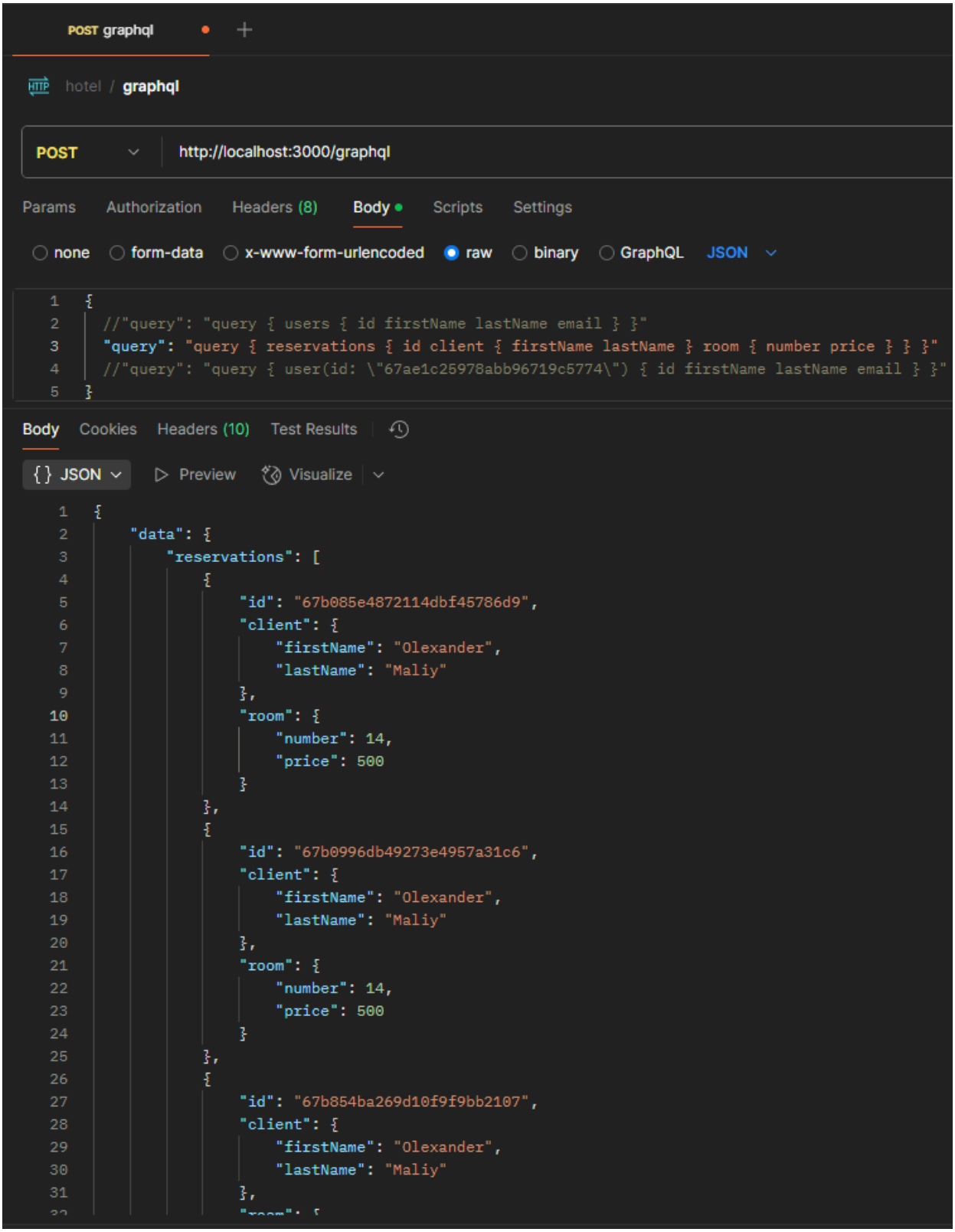

Рисунок 6 – Запит отримання резервацій з клієнтом та кімнатою

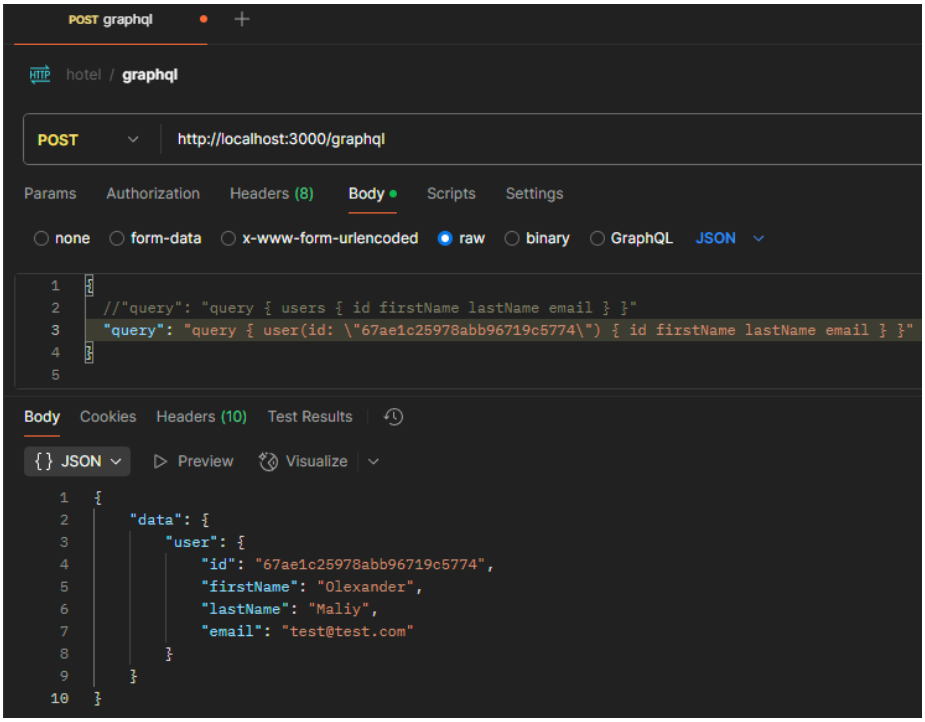

Рисунок 7 – Запит отримання користувача за id

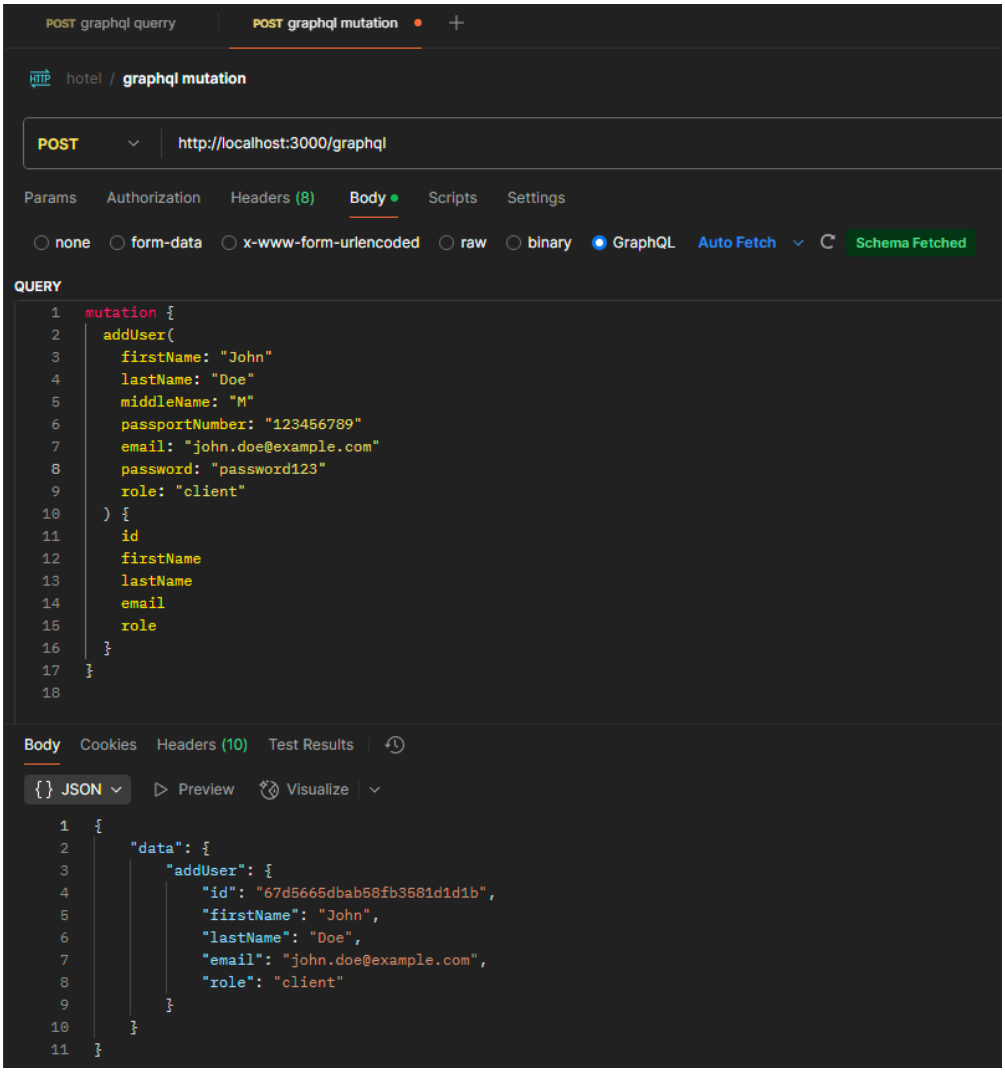

Рисунок 8 – Запит додавання нового користувача

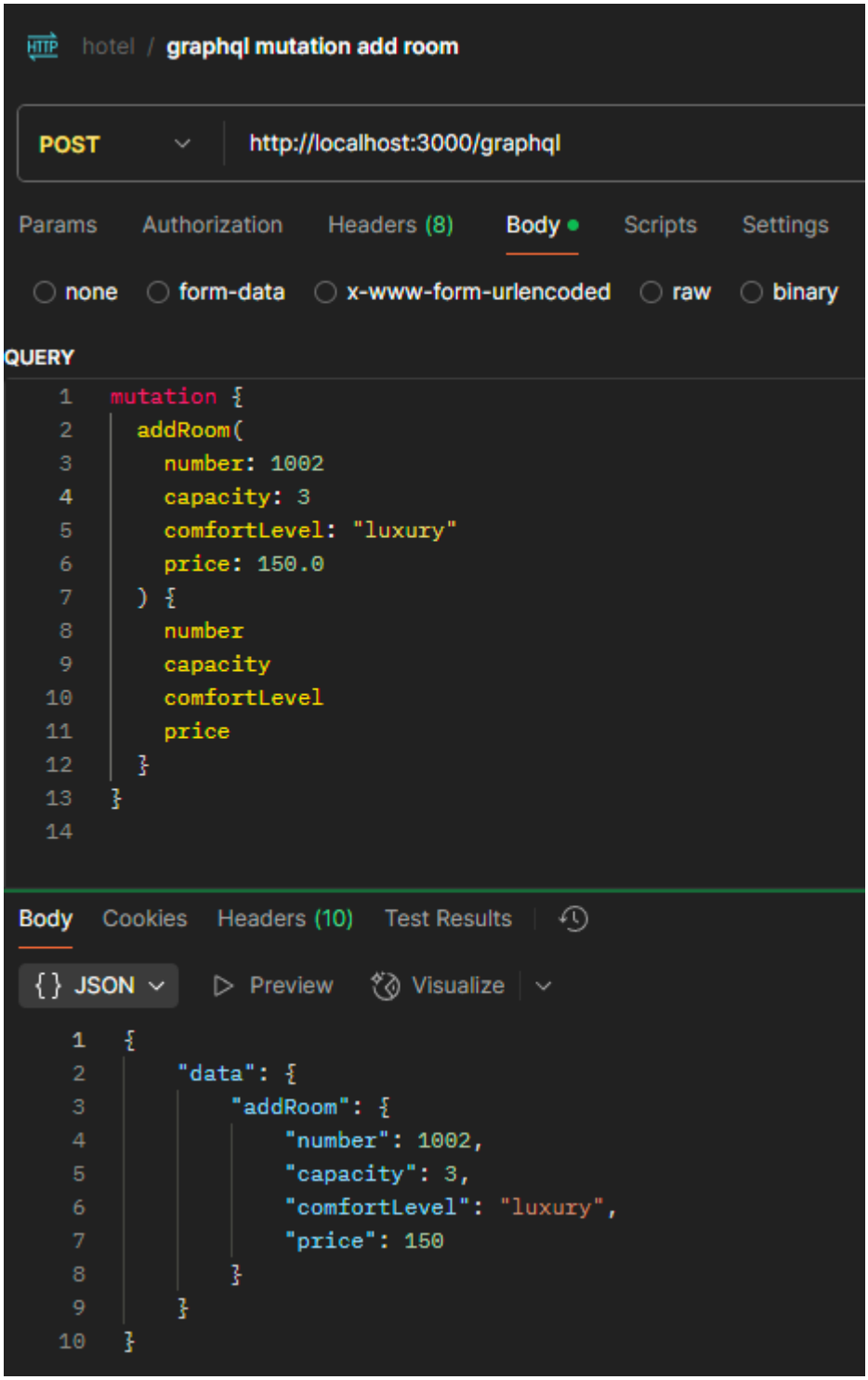

Рисунок 9 – Запит додавання нової кімнати

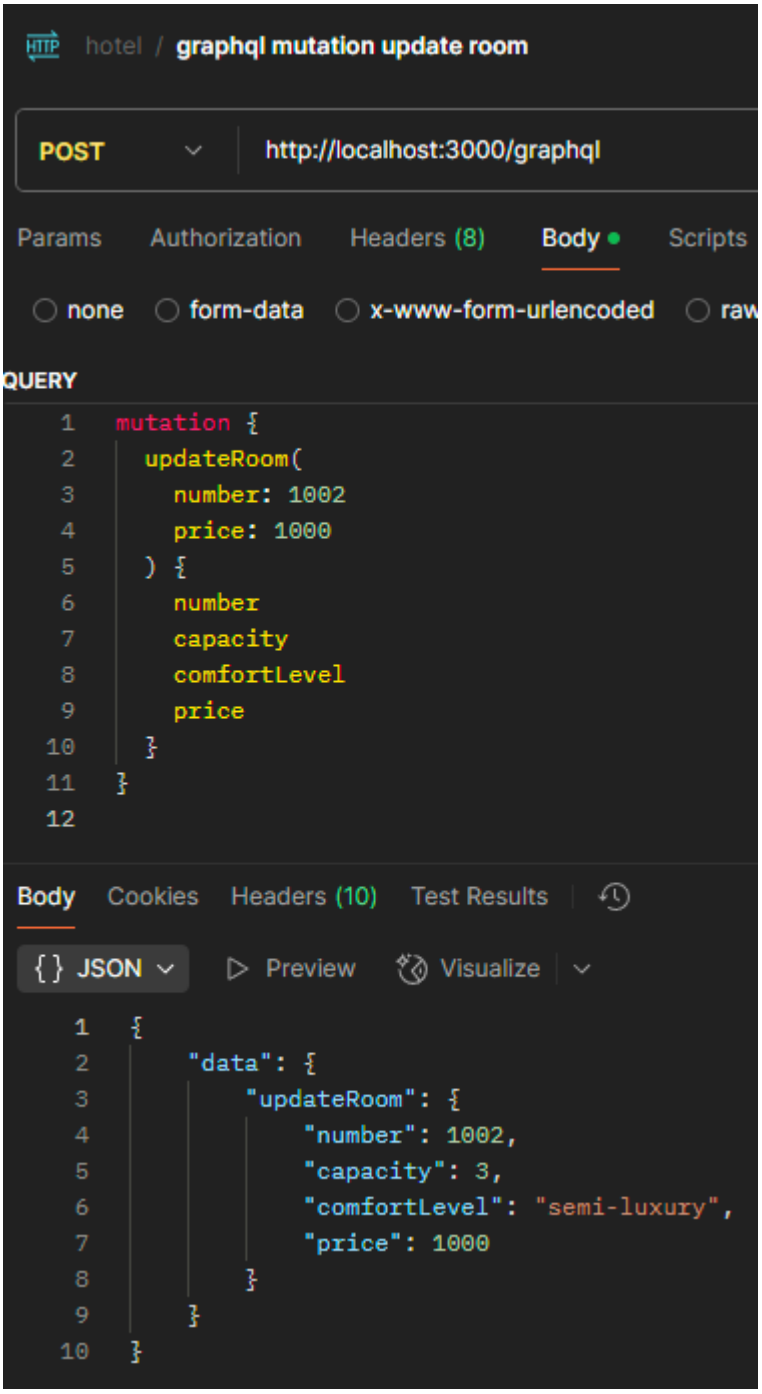

Рисунок 10 – Запит редагування кімнати за номером

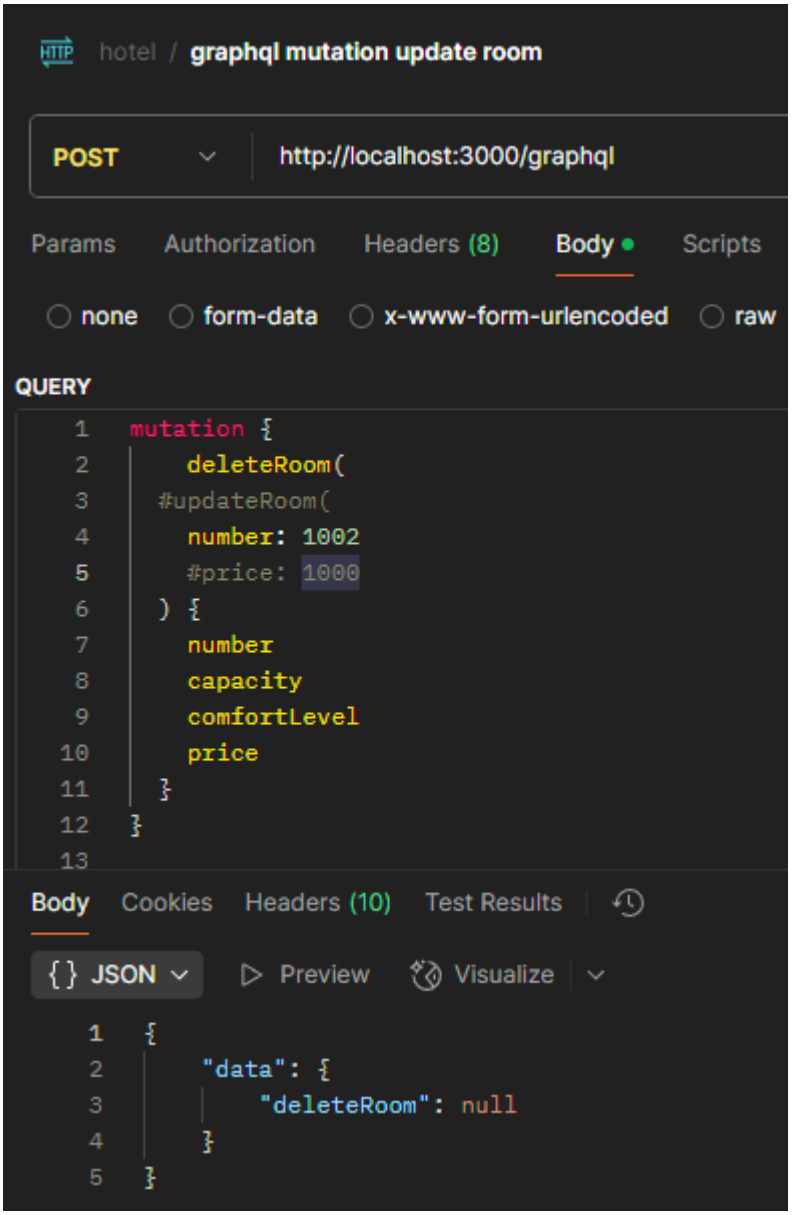

Рисунок 11 – Запит видалення кімнати за номером

Висновок.

У процесі виконання лабораторної роботи я розробив та налаштував GraphQL API для управління готелем. Створив відповідні моделі для користувачів, кімнат та бронювань, а також реалізував мутації для CRUD-операцій. Після налаштування запитів і мутацій я перевірив їхню працездатність через Postman, де протестував різні сценарії додавання, оновлення та видалення даних.

Робота дозволила набути навички роботи з GraphQL, а також показала, як організувати ефективну обробку запитів та мутацій у реальному API.

Скрипти.

Код застосунку знаходиться на віддаленому репозиторію за посиланням:

https://github.com/JessFreak/hotel-management-backend.