

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до комп'ютерного практикуму №6
З дисципліни «Основи Back-end технологій»

ВИКОНАЛА:

Студентка групи ІМ-21

Рабійчук Дар'я Олександрівна

№ у списку(варіант) - 4

ПЕРЕВІРИВ:

доц. Голубєв Л.П.

Київ 2025

Лабораторна робота №6

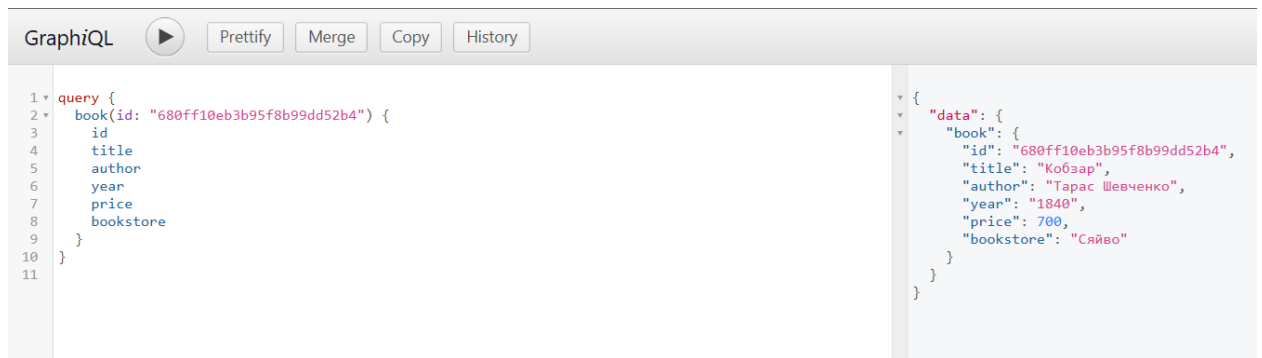
Тема: GraphQL. Створення Schema GraphQL та Resolvers. Створення Query та Mutation.

Завдання.

- На свій БД (розробленої в лаб. роб. #5) за допомогою Schema Definition Language (SDL) створити схему GraphQL.
- Додати Resolvers для виконання операцій GraphQL.
- Створити та виконати Query та Mutation для виконання операцій додавання, редагування та видалення інформації (CRUD) в БД.
- Виконати дослідження роботи створених query та mutation за допомогою Postman.

Результати виконання роботи:

Робота з GraphQL



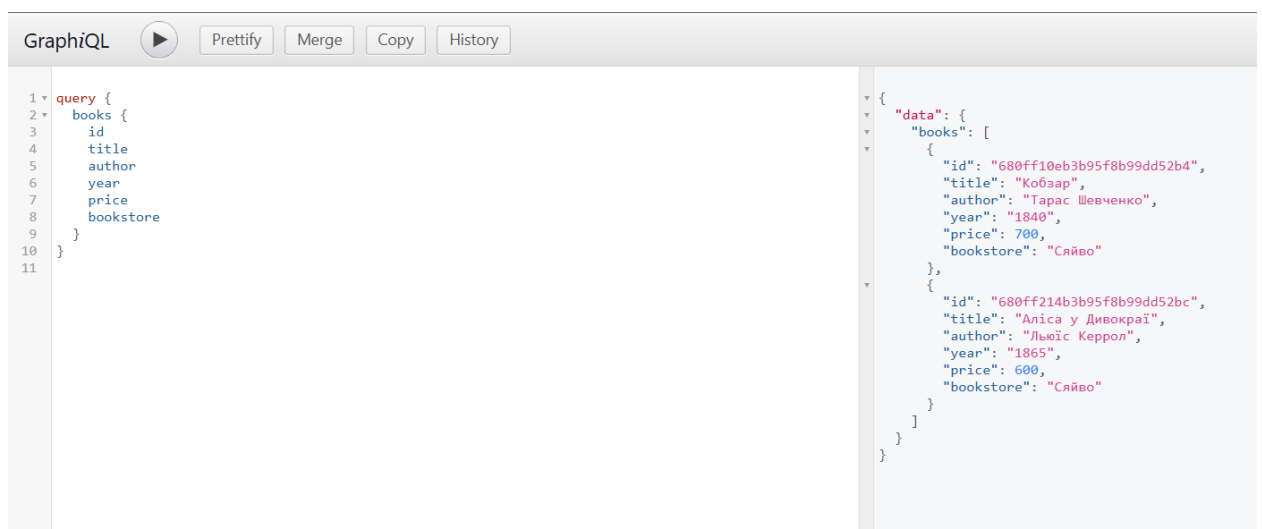
The screenshot shows the GraphQL IDE interface. On the left, a query is entered:

```
1 query {  
2   book(id: "680ff10eb3b95f8b99dd52b4") {  
3     id  
4     title  
5     author  
6     year  
7     price  
8     bookstore  
9   }  
10 }  
11
```

 On the right, the JSON response is displayed:

```
{  
  "data": {  
    "book": {  
      "id": "680ff10eb3b95f8b99dd52b4",  
      "title": "Кобзар",  
      "author": "Тарас Шевченко",  
      "year": "1840",  
      "price": 700,  
      "bookstore": "Сяйво"  
    }  
  }  
}
```

Query для виведення одної з книг по id



The screenshot shows the GraphQL IDE interface. On the left, a query is entered:

```
1 query {  
2   books {  
3     id  
4     title  
5     author  
6     year  
7     price  
8     bookstore  
9   }  
10 }  
11
```

 On the right, the JSON response is displayed:

```
{  
  "data": {  
    "books": [  
      {  
        "id": "680ff10eb3b95f8b99dd52b4",  
        "title": "Кобзар",  
        "author": "Тарас Шевченко",  
        "year": "1840",  
        "price": 700,  
        "bookstore": "Сяйво"  
      },  
      {  
        "id": "680ff214b3b95f8b99dd52bc",  
        "title": "Аліса у Дивокраї",  
        "author": "Льюїс Керрол",  
        "year": "1865",  
        "price": 600,  
        "bookstore": "Сяйво"  
      }  
    ]  
  }  
}
```

Query для отримання всіх книг



Дослідження результатів роботи GraphQL за допомогою Postman:

The screenshot shows the Postman interface with a GraphQL query named "Books" sent to the endpoint `http://localhost:3000/graphql`. The query is:

```
query Books {  
  books {  
    id  
    title  
    author  
    year  
    author_address  
    publisher_address  
    price  
    bookstore  
  }  
}
```

The query is selected in the left sidebar. The response is displayed in the "Body" tab, showing a JSON object with a list of books:

```
{  
  "data": {  
    "books": [  
      {  
        "id": "680ff10eb3b95f8b99dd52b4",  
        "title": "Кобзар",  
        "author": "Тарас Шевченко",  
        "year": "1840",  
        "author_address": "Україна",  
        "publisher_address": "Київська друкарня",  
        "price": 700,  
        "bookstore": "Сяйво"  
      }  
    ]  
  }  
}
```

The status bar indicates a 200 OK response with a response time of 55.91 ms and a body size of 719 B.

Query для отримання всіх книг

The screenshot shows the Postman interface with a GraphQL query named "Book" sent to the endpoint `http://localhost:3000/graphql`. The query is:

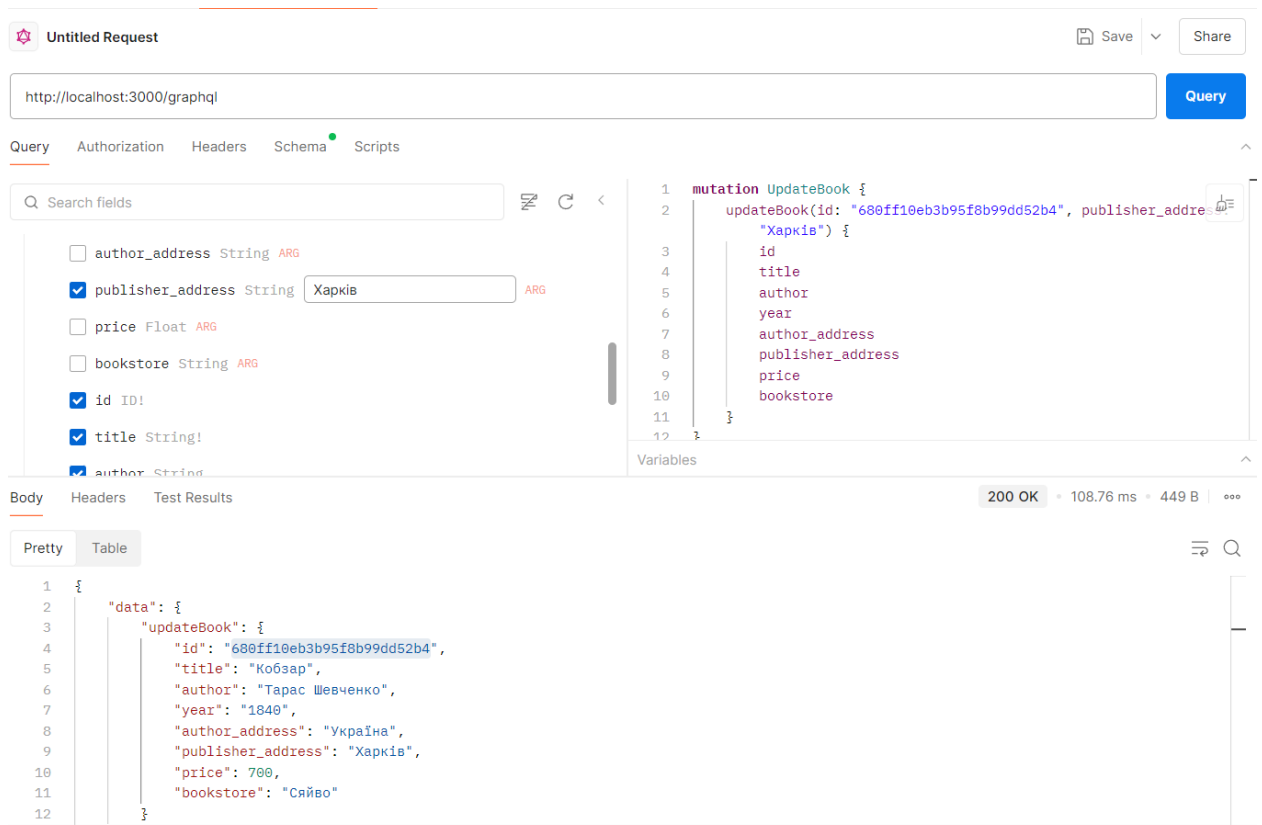
```
query Book {  
  book(id: "680ff10eb3b95f8b99dd52b4") {  
    id  
    title  
    author  
    year  
    author_address  
    publisher_address  
    price  
    bookstore  
  }  
}
```

The query is selected in the left sidebar. The response is displayed in the "Body" tab, showing a JSON object with a single book:

```
{  
  "data": {  
    "book": {  
      "id": "680ff10eb3b95f8b99dd52b4",  
      "title": "Кобзар",  
      "author": "Тарас Шевченко",  
      "year": "1840",  
      "author_address": "Україна",  
      "publisher_address": "Київська друкарня",  
      "price": 700,  
      "bookstore": "Сяйво"  
    }  
  }  
}
```

The status bar indicates a 200 OK response with a response time of 52.72 ms and a body size of 464 B.

Query для отримання однієї книги



Мутація для оновлення книги

Кобзар

Автор: Тарас Шевченко

Рік видання: 1840

Адреса автора: Україна

Адреса видавництва: Харків

Ціна: 700 грн

Книготорговельна фірма: Сяйво

РедагуватиВидалити

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('680ff10eb3b95f8b99dd52b4')
title: "Кобзар"
author: "Тарас Шевченко"
year: "1840"
author_address: "Україна"
publisher_address: "Харків"
price: 700
bookstore: "Сяйво"
created_at: 2025-04-28T21:20:14.271+00:00
__v: 0
```

Перевірка оновлення на сайті та базі даних

Untitled Request

Save Share

http://localhost:3000/graphql

Query Authorization Headers Schema Scripts

Search fields

Query

- books [Book]
- book Book

Mutation

- ☒ addBook Book
 - ☒ title String! Тіні забутих предків ARG
 - ☒ author String! Михайло Коцюбинський ARG

```
1 mutation AddBook {
2   addBook(
3     title: "Тіні забутих предків\"
4     author: "Михайло Коцюбинський"
5     year: "1911"
6     author_address: "Україна"
7     publisher_address: "Львів: Світ"
8     bookstore: "Книгарня Є"
9   ) {
10     id
11     title
12   }
13 }
```

Variables

Body Headers Test Results

200 OK • 67.81 ms • 315 B

Pretty Table

```
1 {
2   "data": {
3     "addBook": {
4       "id": "6810bd899ce78f0ba32dde01",
5       "title": "Тіні забутих предків\"
6     }
7   }
8 }
```

Мутація для додавання нової книги

POST New Request

Untitled Request

Save Share

http://localhost:3000/graphql

Query Authorization Headers Schema Scripts

Search fields

- ☐ publisher_address String
- ☐ price Float
- ☐ bookstore String
- ☐ updateBook Book
- ☒ deleteBook String
 - ☒ id ID! 6810bd899ce78f0ba32dde0 ARG

```
1 mutation DeleteBook {
2   deleteBook(id: "6810bd899ce78f0ba32dde01")
3 }
4 }
```

Variables

Body Headers Test Results

200 OK • 54.63 ms • 262 B

Pretty Table

```
1 {
2   "data": {
3     "deleteBook": "Книга видалена"
4   }
5 }
```

Мутація для видалення

app.js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const bookRoutes = require('./routes/bookRoutes');
const { graphqlHTTP } = require('express-graphql');
const { buildSchema } = require('graphql');
const Book = require('./models/Book'); // підключаємо модель книги

const app = express();
const PORT = 3000;

// Підключення до MongoDB Atlas
mongoose.connect('mongodb+srv://rabiychukdaria:P2dhnrluW2R8Y@cluster0.lffllwe.mongodb.net/library?retryWrites=true&w=majority&appName=Cluster0')
  .then(() => console.log("MongoDB Atlas підключено"))
  .catch(err => console.log("Помилка MongoDB:", err));

// Налаштування Express
app.set('view engine', 'ejs');
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: true }));

// Роутинг для класичного сайту
app.use('/', bookRoutes);

// Оголошення GraphQL-схеми
const schema = buildSchema(`
  type Book {
    id: ID!
    title: String!
    author: String
    year: String
    author_address: String
    publisher_address: String
    price: Float
    bookstore: String
  }

  type Query {
    books: [Book]
    book(id: ID!): Book
  }

  type Mutation {
    addBook(
      title: String!
      author: String
      year: String
      author_address: String
    )
  }
`);
```

```

        publisher_address: String
        price: Float
        bookstore: String
    ): Book

    updateBook(
        id: ID!
        title: String
        author: String
        year: String
        author_address: String
        publisher_address: String
        price: Float
        bookstore: String
    ): Book

    deleteBook(id: ID!): String
}
`);

// Resolvers для операцій
const root = {
  books: async () => await Book.find(),

  book: async ({ id }) => await Book.findById(id),

  addBook: async (args) => {
    const book = new Book(args);
    await book.save();
    return book;
  },

  updateBook: async ({ id, ...args }) => {
    await Book.findByIdAndUpdate(id, args);
    return await Book.findById(id);
  },

  deleteBook: async ({ id }) => {
    await Book.findByIdAndDelete(id);
    return "Книга видалена";
  }
};

// Підключення GraphQL-сервера
app.use('/graphql', graphqlHTTP({
  schema,
  rootValue: root,
  graphiql: true, // Увімкнути вбудований GraphQL-браузер
})));

// Запуск сервера

```



```
app.listen(PORT, () => {  
  console.log(`Сервер запущено на http://localhost:${PORT}`);  
});
```

Висновок

У ході лабораторної роботи було реалізовано вебдодаток на платформі Node.js з використанням бази даних MongoDB Atlas. За допомогою бібліотек GraphQL та express-graphql було створено схему GraphQL, що описує структуру даних про книги, а також resolvers для виконання CRUD-операцій. Створені запити типу Query та Mutation дозволяють отримувати, додавати, редагувати та видаляти записи з бази даних. Тестування запитів було виконано за допомогою Postman, що підтвердило коректність роботи додатку.