



Міністерство освіти і науки України  
Національний технічний університет України „КПІ 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»**

Прийняв:

Викладач

пос. Зубко Р. А.

«29» квітня 2024 р.

Виконала:

Студентка 3 курсу, гр. ІМ-11

Бащак Ярина

**2024 р.**

## Лабораторна робота №6.

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

#### Завдання.

- На своїй 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.

#### Хід роботи

1. Завантажуємо необхідні пакети: graphql, express-graphql
2. Створюємо схему GraphQL

```
const { buildSchema } = require('graphql');

const schema = buildSchema(`
  type Post {
    id: ID!
    title: String
    author: String
    text: String
    createdAt: String
    updatedAt: String
  }

  type Query {
    getPost(id: ID!): Post
    listPosts: [Post]
  }

  type Mutation {
    createPost(title: String!, author: String!, text: String!): Post
    updatePost(id: ID!, title: String, author: String, text: String): Post
    deletePost(id: ID!): Post
  }
`);

module.exports = schema;
```

3. Створюємо Resolvers

```

const Post = require('../models/Post');

const resolvers = {
  getPost: ({ id }) => {
    return Post.findById(id);
  },
  listPosts: () => {
    return Post.find();
  },
  createPost: ({ title, author, text }) => {
    const newPost = new Post({ title, author, text });
    return newPost.save();
  },
  updatePost: ({ id, title, author, text }) => {
    return Post.findByIdAndUpdate(id, { title, author, text }, { new: true });
  },
  deletePost: ({ id }) => {
    return Post.findByIdAndDelete(id);
  }
};

module.exports = resolvers;

```

#### 4. Додаємо новий шлях /graphql у server.js

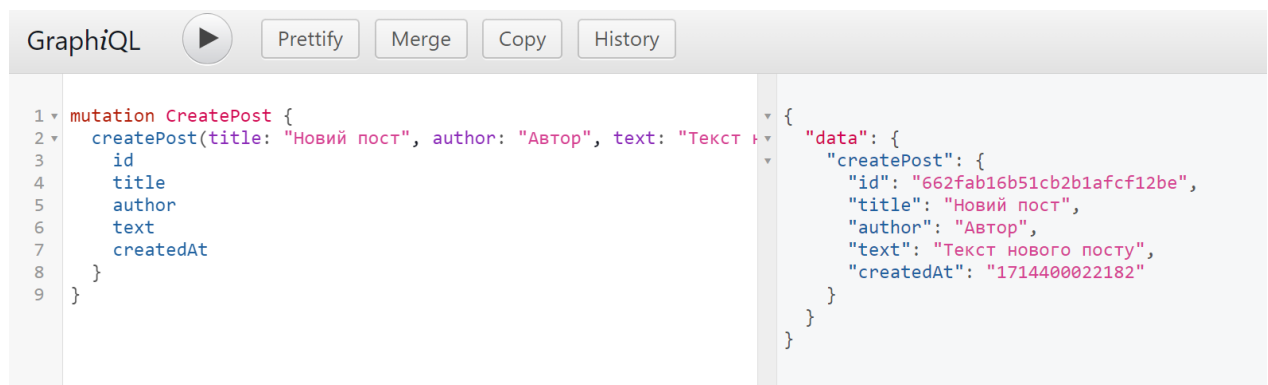
```

const { graphqlHTTP } = require('express-graphql');
const schema = require('./graphql/schema');
const resolvers = require('./graphql/resolvers');

app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: resolvers,
  graphiql: true,
}));

```

#### 5. Результати



The screenshot shows the GraphQL Playground interface. On the left, the query is defined as a mutation to create a new post. On the right, the JSON response is displayed, showing the created post with its ID, title, author, text, and creation timestamp.

```

1 mutation CreatePost {
2   createPost(title: "Новий пост", author: "Автор", text: "Текст") {
3     id
4     title
5     author
6     text
7     createdAt
8   }
9 }

```

```

{
  "data": {
    "createPost": {
      "id": "662fab16b51cb2b1afcf12be",
      "title": "Новий пост",
      "author": "Автор",
      "text": "Текст нового посту",
      "createdAt": "1714400022182"
    }
  }
}

```

Мал.1 Створення поста

<pre> 1 mutation UpdatePost { 2   updatePost(id: "662fab16b51cb2b1afcf12be", 3             title: "Оновлений заголовок", 4             author: "Оновлений автор", 5             text: "Оновлений текст") { 6     id 7     title 8     author 9     text 10  } 11 }</pre>	<pre> {   "data": {     "updatePost": {       "id": "662fab16b51cb2b1afcf12be",       "title": "Оновлений заголовок",       "author": "Оновлений автор",       "text": "Оновлений текст"     }   } }</pre>
--	--

Мал.2 Редагування поста

<pre> 1 query ListPosts { 2   listPosts { 3     id 4     title 5     author 6     text 7   } 8 }</pre>	<pre> {   "data": {     "listPosts": [       {         "id": "66142e05b480116c7788da13",         "title": "Title1",         "author": "Author1",         "text": "Article 1, article 1"       },       {         "id": "66166ef05b26ec96079f3a98",         "title": "hjfdk 34543",         "author": "dfds4",         "text": "kgjg"       },       {         "id": "662fab16b51cb2b1afcf12be",         "title": "Оновлений заголовок",         "author": "Оновлений автор",         "text": "Оновлений текст"       }     ]   } }</pre>
--	--

Мал.3 Вивід списку постів

<pre> 1 query GetPost { 2   getPost(id: "662fab16b51cb2b1afcf12be") { 3     id 4     title 5     author 6     text 7   } 8 }</pre>	<pre> {   "data": {     "getPost": {       "id": "662fab16b51cb2b1afcf12be",       "title": "Оновлений заголовок",       "author": "Оновлений автор",       "text": "Оновлений текст"     }   } }</pre>
--	---

Мал.4 Вивід поста за ідентифікатором

<pre> 1 mutation DeletePost { 2   deletePost(id: "662fab16b51cb2b1afcf12be") { 3     id 4   } 5 }</pre>	<pre> {   "data": {     "deletePost": {       "id": "662fab16b51cb2b1afcf12be"     }   } }</pre>
---	--

Мал.5 Видалення поста

## 6. Тестування всіх query і mutation в Postman

Untitled Request

Save

http://localhost:3000/graphql

Query

Query

Authorization

Headers

Schema

Scripts

Search fields

Query

getPost Post

listPosts [Post]

id ID!

title String

author String

text String

createdAt String

updatedAt String

Mutation

createPost Post

updatePost Post

deletePost Post

```
1 query ListPosts {
2   listPosts {
3     id
4     title
5     author
6     text
7     createdAt
8     updatedAt
9   }
10 }
11
```

Variables

Body

Headers

Test Results

Status: 200 OK Time: 345.17 ms Size: 720 B Save as Example

Pretty

Table

```
1 {
2   "data": {
3     "listPosts": [
4       {
5         "id": "66142e05b480116c7788da13",
6         "title": "Title1",
7         "author": "Author1",
8         "text": "Article 1, article 1",
9         "createdAt": "1712598533377",
10        "updatedAt": "1712598533378"
11      },
12      {
13        "id": "66166ef05b26ec96079f3a98",
14        "title": "hjfdk 34543",
15        "author": "dfds4",
16        "text": "kgjg",
17        "createdAt": "1712746224451",
18        "updatedAt": "1712746238104"
19      },
20      {
21        "id": "662fb77cb51cb2b1afcf12cc",
22        "title": "Новий пост",
23        "author": "Автор",
24        "text": "Текст нового посту",
25        "createdAt": "1714403196121",
26        "updatedAt": "1714403196121"
27      }
28    ]
29  }
30 }
```

**Висновок:** при виконанні даної роботи ми інтегрували GraphQL API у наявний Node.js проект з MongoDB. Це дало нам змогу створити більш гнучкий та ефективний інтерфейс для взаємодії з даними, забезпечуючи оптимізацію запитів і краще управління даними через єдиний ендпойнт. Також ці запити були протестовані через Postman. Завдання комп'ютерного практикуму було виконано, а всі результати, у вигляді скріншотів, представлені у звіті.