



Міністерство освіти і науки України  
Національний технічний університет України „КПІ  
імені Ігоря Сікорського ”  
Факультет інформатики та обчислювальної техніки  
Кафедра інформатики і програмної інженерії

## **ЗВІТ**

лабораторної роботи №5  
з курсу «Основи WEB - технологій»

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

Перевірив:

Викл. Альбрехт Й.О.

Виконала:

ст. Мельникова  
Катерина гр. ІС-12

Київ 2024

## 1. Завдання 1.

### Завдання.

- На своїй БД (розробленої в лаб. роб. #4) 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

Повний код проєкту можна переглянути за посиланням:  
<https://github.com/katerynamelnikova/web-technologies-labs>

Файл worker.resolver.js:

```
import { buildSchema } from 'graphql';
import { WorkerModel } from '../models/worker.model.js';

const schema = buildSchema(`
  type Worker {
    id: ID!
    name: String!
    room: String!
    department: String!
    computer: String!
    date: String
  }

  type Query {
    getWorkerById(id: ID!): Worker
    getAllWorkers: [Worker]
  }

  type Mutation {
    createWorker(name: String!, room: String!, department: String!, computer: String!): Worker
    updateWorker(id: ID!, name: String, room: String, department: String, computer: String):
Worker
    deleteWorker(id: ID!): Worker
  }
`);
```

```

const resolver = {
  getWorkerById: async ({ id }) => {
    try {
      const worker = await WorkerModel.findOne({ _id: id });
      return worker;
    } catch (error) {
      throw new Error('Error fetching worker by ID');
    }
  },
  getAllWorkers: async () => {
    try {
      const workers = await WorkerModel.find();
      return workers;
    } catch (error) {
      throw new Error('Error fetching all workers');
    }
  },
  createWorker: async ({ name, room, department, computer }) => {
    try {
      const newWorker = new WorkerModel({ name, room, department, computer });
      return await newWorker.save();
    } catch (error) {
      throw new Error('Error creating worker');
    }
  },
  updateWorker: async ({ id, name, room, department, computer }) => {
    try {
      return await WorkerModel.findOneAndUpdate(
        { _id: id },
        { name, room, department, computer },
        { new: true }
      );
    } catch (error) {
      throw new Error('Error updating worker');
    }
  },
  deleteWorker: async ({ id }) => {
    try {
      return await WorkerModel.findOneAndDelete({ _id: id });
    } catch (error) {
      throw new Error('Error deleting worker');
    }
  }
};

```

```
export { schema, resolver };
```

## Отримані результати:

### GetAllWorkers:

The screenshot shows the GraphQL Playground interface. At the top, the URL is `http://localhost:4000/graphql`. The query editor contains the following query:

```
query GetAllWorkers {
  getAllWorkers {
    id
    name
    room
    department
    computer
    date
  }
}
```

The left sidebar shows the schema for `Worker` with fields `id` (ID!), `name` (String!), `room` (String!), and `department` (String!). The response tab at the bottom shows the following JSON:

```
{
  "data": {
    "getAllWorkers": [
      {
        "id": "66e3370050a82d9f91646a5a",
        "name": "Мельник",
        "room": "350",
        "department": "IT",
        "computer": "Macbook Pro",
        "date": "1726180128590"
      },
      {
        "id": "66e33bd650a82d9f91646a63",
        "name": "Івашенко",
        "room": "100",
        "department": "IT",
        "computer": "Macbook Air",
        "date": "1726180128590"
      }
    ]
  }
}
```

The status bar at the bottom indicates a 200 OK response with a time of 14.92 ms and a size of 702 B.

### GetWorkerById:

web-lab-5 / <http://localhost:4000/graphql> Save Share

<http://localhost:4000/graphql> Query

Query Authorization Headers Schema Scripts

Search fields

Query

☒ getWorkerById Worker

☒ id ID! 66e3370050a82d9f916 ARG

☒ id ID!

☒ name String!

```
1 query GetWorkerById {
2   getWorkerById(id: "66e3370050a82d9f91646a5a") {
3     id
4     name
5     room
6     department
7     computer
8     date
9   }
10 }
```

Variables

Body Headers Test Results 200 OK • 8.06 ms • 437 B Save Response

Pretty Table

```
1 {
2   "data": {
3     "getWorkerById": {
4       "id": "66e3370050a82d9f91646a5a",
5       "name": "Мельник",
6       "room": "350",
7       "department": "IT",
8       "computer": "Macbook Pro",
9       "date": "1726180128590"
10     }
11   }
12 }
```

## CreateWorker:

web-lab-5 / <http://localhost:4000/graphql> Save Share

<http://localhost:4000/graphql> Query

Query Authorization Headers Schema Scripts

Search fields

☒ name String! Бойко ARG

☒ room String! 70 ARG

☒ department String! IT ARG

☒ computer String! HP ARG

☒ id ID!

```
1 mutation CreateWorker {
2   createWorker(name: "Бойко", room: "70", department: "IT",
3     computer: "HP") {
4     id
5     name
6     room
7     department
8     computer
9     date
10  }
11 }
12 }
```

Variables

Body Headers Test Results 200 OK • 13.91 ms • 422 B Save Response

Pretty Table

```
1 {
2   "data": {
3     "createWorker": {
4       "id": "66e75886a7e5c2e072a685bf",
5       "name": "Бойко",
6       "room": "70",
7       "department": "IT",
8       "computer": "HP",
9       "date": "1726437510603"
10     }
11   }
12 }
```

Filter ⓘ ⓘ

Type a query: { field: 'value' }

Reset Find ↩ More Options ▶

ADD DATA ▾

EXPORT COLLECTION

1 - 4 of 4 ↺ < > ⋮ {} | ⌂

department: "IT"

computer: "Macbook Air"

date: 2024-09-12T23:05:36.761+00:00

\_\_v: 0

▶

\_id: ObjectId('66e741765cc9685707ae8231')

name: "Мазып"

room: "23"

department: "HR"

computer: "HP"

date: 2024-09-15T20:20:06.602+00:00

\_\_v: 0

✎

🔍

📄

🗑

HP

\_id: ObjectId('66e75886a7e5c2e072a685bf')

name: "Бойко"

room: "70"

department: "IT"

computer: "HP"

date: 2024-09-15T21:58:30.603+00:00

\_\_v: 0

UpdateWorker:

web-lab-5 / http://localhost:4000/graphql

Save ▾ Share

http://localhost:4000/graphql

Query

Query Authorization Headers Schema ⬢ Scripts

Search fields

☐ name String ARG

☒ room String 60 ARG

☐ department String ARG

☐ computer String ARG

☒ id ID!

1 mutation UpdateWorker {

2   updateWorker(id: "66e75886a7e5c2e072a685bf", room: "60") {

3     id

4     name

5     room

6     department

7     computer

8     date

9   }

10 }

Variables

Body Headers Test Results

200 OK • 9.55 ms • 422 B 📄 Save Response ▾

Pretty Table

1 {

2   "data": {

3     "updateWorker": {

4       "id": "66e75886a7e5c2e072a685bf",

5       "name": "Бойко",

6       "room": "60",

7       "department": "IT",

8       "computer": "HP",

9       "date": "1726437510603"

10     }

11   }

12 }

## DeleteWorker:

The screenshot shows the Postman interface for a GraphQL API. The URL is `http://localhost:4000/graphql`. The query editor shows a mutation named `DeleteWorker` with the following query:

```
1 mutation DeleteWorker {
2   deleteWorker(id: "66e75886a7e5c2e072a685bf") {
3     id
4     name
5     room
6     department
7     computer
8     date
9   }
10 }
```

The variables section is empty. The response is a 200 OK status with a response time of 7.16 ms and a body size of 422 B. The response body is shown in the 'Body' tab, formatted as JSON:

```
1 {
2   "data": {
3     "deleteWorker": {
4       "id": "66e75886a7e5c2e072a685bf",
5       "name": "Бойко",
6       "room": "60",
7       "department": "IT",
8       "computer": "HP",
9       "date": "1726437510603"
10     }
11   }
12 }
```

## Висновок

Під час виконання даної лабораторної роботи я створила схему GraphQL за допомогою SDL, додала resolvers для виконання CRUD операцій у БД та виконала дослідження створених query та mutations за допомогою Postman.