



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

Звіт до комп'ютерного практикуму №6

З дисципліни «Основи Back-end технологій»

Прийняв:

Зубко Роман Анатолійович

Виконав:

Студент 3 курсу,
Котенко Ярослав
гр. ІМ-13

2024 р.

Лабораторна робота №6. Тема: 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

На своїй БД (розробленої в лаб. роб. #5) за допомогою Schema Definition Language (SDL) створив схему GraphQL.

```
schema > ✖ schema.graphql
1  type Detail {
2    id : ID!
3    name : String!
4    description : String!
5  }
6
7  type ProcType {
8    id : ID!
9    name : String!
10   description : String!
11 }
12
13 type TechCard {
14   id : ID!
15   detail_ID : Detail!
16   procType_ID : ProcType!
17   duration : Int!
18   creationDate : String
19   description : String!
20 }
21
22 type Query {
23   getTechCardById(id: ID): TechCard
24   getProcTypeById(id: ID): ProcType
25   getDetailById(id: ID): Detail
26   getAllTechCards : [TechCard!]
27   getAllProcTypes : [ProcType!]
28   getAllDetails : [Detail!]
29 }
30
```

```

schema > schema.graphql
30
31 input newTechCard {
32   detail_ID : ID
33   procType_ID : ID
34   duration : Int
35   creationDate : String
36   description : String
37 }
38
39 input newDetail {
40   name : String
41   description : String
42 }
43
44 input newProcType {
45   name : String
46   description : String
47 }
48
49 type Mutation {
50   addTechCard(techCard: newTechCard): [TechCard!]!
51   addDetail(detail: newDetail): [Detail!]!
52   addProcType(procType: newProcType): [ProcType!]!
53   updateTechCard(id: ID!, body: newTechCard): TechCard
54   updateProcType(id: ID!, body: newProcType): ProcType
55   updateDetail(id: ID!, body: newDetail): Detail
56   deleteTechCard(id: ID!): [TechCard!]!
57   deleteProcType(id: ID!): [ProcType!]!
58   deleteDetail(id: ID!): [Detail!]!
59 }

```

Додав **Resolvers** для виконання операцій GraphQL. В Resolvers створив та виконав **Query** та **Mutation** щоб описати **CRUD** операції для головної таблиці(**TechCard**) та допоміжних таблиць(**ProcessingTypes**, **Details**).

```

schema > JS resolver.js > resolvers > Query > getTechCardById
1 import techCards from '../models/techCards.js'
2 import processingTypes from '../models/processingTypes.js'
3 import details from '../models/details.js'
4
5 const resolvers = {
6   Query: {
7     getTechCardById: async (_, { id }) => {
8       return await techCards.findById(id)
9     },
10    getProcTypeById: async (_, { id }) => {
11      return await processingTypes.findById(id)
12    },
13    getDetailById: async (_, { id }) => {
14      return await details.findById(id)
15    },
16    getAllTechCards: async () => {
17      return await techCards.find()
18    },
19    getAllProcTypes: async () => {
20      return await processingTypes.find()
21    },
22    getAllDetails: async () => {
23      return await details.find()
24    }
25  },
26   Mutation: {
27     addTechCard: async (_, { techCard: { detail_ID, procType_ID, duration, creationDate, description } }) => {
28       const date = creationDate ? creationDate : new Date(Date.now())
29       try {
30         const newTechCard = new techCards({
31           detail_ID,
32           procType_ID,
33           duration,
34           creationDate: date,
35           description
36         })
37         await newTechCard.save()
38         return await techCards.find()
39       } catch (err) {
40         throw new Error(err)
41       }
42     },
43     addDetail: async (_, { detail: { name, description } }) => {
44       try {
45         const newDetail = new details({
46           name,
47           description
48         })
49         await newDetail.save()
50         return await details.find()
51       } catch (err) {
52         throw new Error(err)
53       }
54     },

```

```

55   addProcType: async (_, { procType: { name, description } }) => {
56     try {
57       const newProcType = new processingTypes({
58         name,
59         description
60       })
61       await newProcType.save()
62       return await processingTypes.find()
63     } catch (err) {
64       throw new Error(err)
65     }
66   },
67   updateTechCard: async (_, { id, body }) => {
68     try {
69       await techCards.findByIdAndUpdate(id, body)
70       return techCards.findById(id)
71     } catch (err) {
72       throw new Error(err)
73     }
74   },
75   updateProcType: async (_, { id, body }) => {
76     try {
77       await processingTypes.findByIdAndUpdate(id, body)
78       return processingTypes.findById(id)
79     } catch (err) {
80       throw new Error(err)
81     }
82   }

```

```

schema > JS resolver.js > resolvers > Mutation > updateDetail
5   const resolvers = {
26   Mutation: {
42     //
43     addDetail: async (_, { detail: { name, description } }) => {
44       try {
45         const newDetail = new details({
46           name,
47           description
48         })
49         await newDetail.save()
50         return await details.find()
51       } catch (err) {
52         throw new Error(err)
53       }
54     },
55     addProcType: async (_, { procType: { name, description } }) => {
56       try {
57         const newProcType = new processingTypes({
58           name,
59           description
60         })
61         await newProcType.save()
62         return await processingTypes.find()
63       } catch (err) {
64         throw new Error(err)
65       }
66     },
67     updateTechCard: async (_, { id, body }) => {
68       try {
69         await techCards.findByIdAndUpdate(id, body)
70         return techCards.findById(id)
71       } catch (err) {
72         throw new Error(err)
73       }
74     },
75     updateProcType: async (_, { id, body }) => {
76       try {
77         await processingTypes.findByIdAndUpdate(id, body)
78         return processingTypes.findById(id)
79       } catch (err) {
80         throw new Error(err)
81       }
82     }

```

```

81   }
82 },
83 updateDetail: async (_, { id, body }) => {
84   try {
85     await details.findByIdAndUpdate(id, body)
86     return details.findById(id)
87   } catch (err) {
88     throw new Error(err)
89   }
90 },
91 deleteTechCard: async (_, { id }) => {
92   await techCards.findByIdAndDelete(id)
93   return await techCards.find()
94 },
95 deleteProcType: async (_, { id }) => {
96   await processingTypes.findByIdAndDelete(id)
97   return await processingTypes.find()
98 },
99 deleteDetail: async (_, { id }) => {
100   await details.findByIdAndDelete(id)
101   return await details.find()
102 },
103 },
104 TechCard: {
105   detail_ID: async (parent) => await details.findById(parent.detail_ID),
106   procType_ID: async (parent) => await processingTypes.findById(parent.procType_ID)
107 }
108 }
109
110 export default resolvers

```

Головний файл де я запускаю сервер

```
JS app.js > [0] start
1  import express from 'express'
2  import mongoose from 'mongoose'
3  import { graphqlHTTP } from 'express-graphql'
4  import schema from './schema/schema.js'
5
6  const app = express()
7  const PORT = process.env.PORT || 4000
8  const url = `mongodb+srv://yarikkotenkoim13:LUG03nhtc1ZpvEST@cluster0.qbd8tfw.mongodb.net/lab_6`
9
10 app.use('/graphql', graphqlHTTP({
11   schema,
12   graphiql: true
13 }))
14
15 const start = async () => {
16   await mongoose.connect(url)
17   app.listen(PORT, (err) => {
18     err ? console.log(err) : console.log(`Server is running on port: ${PORT}`)
19   })
20 }
21
22 try {
23   start()
24 } catch (e) {
25   console.log(e)
26 }
```

Допоміжний файл для створення схеми

```
schema > JS schema.js > ...
1  import { makeExecutableSchema } from '@graphql-tools/schema'
2  import resolvers from './resolver.js'
3  import { readFileSync } from 'fs'
4  import { dirname, resolve } from 'path'
5  import { fileURLToPath } from 'url'
6
7  const __filename = fileURLToPath(import.meta.url)
8  const __dirname = dirname(__filename)
9
10 const schemaFilePath = resolve(__dirname, 'schema.graphql')
11
12 const typeDefs = readFileSync(schemaFilePath, 'utf-8')
13
14 const schema = makeExecutableSchema({ typeDefs, resolvers })
15
16 export default schema
```

Тут я об'єдную resolvers з описом GraphQL типів та передаю його до головного файлу серверу.

Дослідження

Отримання технологічної карти по id:

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4000/graphql`. The query is:

```
1 query GetTechCardById {
2   getTechCardById(id: "66141311b778274508665f8a") {
3     duration
4     createdAt
5     description
6     detail_ID {
7       id name description
8     }
9     procType_ID {
10      id name description
11    }
12  }
13 }
14
```

The response body is shown in the 'Pretty' tab:

```
1 {
2   "data": {
3     "getTechCardById": {
4       "duration": 45,
5       "createdAt": "171259163460",
6       "description": "test description",
7       "detail_ID": {
8         "id": "66140ab73lead8db197c0f96",
9         "name": "M8 Bolt",
10        "description": "Standard bolt with a diameter of 8 mm"
11      },
12      "procType_ID": {
13        "id": "66140b65d3c003e3fe2f62e5",
14        "name": "Turning",
15        "description": "Processing using a lathe machine"
16      }
17    }
18  }
19 }
```

Status: 200 OK Time: 395.57 ms Size: 523 B

Отримання типу обробки по id:

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4000/graphql`. The query is:

```
1 query GetTechCardById {
2   getProcTypeById(id: "66140b65d3c003e3fe2f62e5") {
3     id
4     name
5     description
6   }
7 }
8
```

The response body is shown in the 'Pretty' tab:

```
1 {
2   "data": {
3     "getProcTypeById": {
4       "id": "66140b65d3c003e3fe2f62e5",
5       "name": "Turning",
6       "description": "Processing using a lathe machine"
7     }
8   }
9 }
```

Status: 200 OK Time: 44.75 ms Size: 311 B

Отримання деталі по id:

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4000/graphql`. The query is:

```
query GetDetailById {
  getDetailById(id: "66140ab731ead0db197c0f96") {
    id
    name
    description
  }
}
```

The variables section is empty. The response is shown in the 'Body' tab, formatted as JSON:

```
{
  "data": {
    "getDetailById": {
      "id": "66140ab731ead0db197c0f96",
      "name": "M8 Bolt",
      "description": "Standard bolt with a diameter of 8 mm"
    }
  }
}
```

Status: 200 OK, Time: 46.53 ms, Size: 314 B.

Створення нової технологічної карти:

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4000/graphql`. The mutation is:

```
mutation AddTechCard {
  addTechCard(
    techCard: {
      detail_ID: "66140ab731ead0db197c0f96"
      procType_ID: "66140b65d3c003e3fe2f62e5"
      duration: 60
      description: "lab6_test_description"
    }
  ) {
    id
    duration
    creationDate
    description
    detail_ID {
      name
      id
    }
    procType_ID {
      id
      name
    }
  }
}
```

The variables section is empty. The response is shown in the 'Body' tab, formatted as JSON:

```
{
  "data": {
    "addTechCard": [
      {
        "id": "66141311b778274508665f8a",
        "duration": 45,
        "creationDate": "1712591633400",
        "description": "test description",
        "detail_ID": {
          "name": "M8 Bolt",
          "id": "66140ab731ead0db197c0f96"
        }
      }
    ]
  }
}
```

Status: 200 OK, Time: 139.24 ms, Size: 695 B.

Untitled Request

Save

http://localhost:4000/graphql

Query

Body Headers Test Results

Status: 200 OK Time: 139.24 ms Size: 695 B Save as Example

Pretty Table

```
1 {
2   "data": {
3     "addTechCard": [
4       {
5         "id": "66141311b776274508665f8a",
6         "duration": 45,
7         "creationDate": "1712591633400",
8         "description": "test description",
9         "detail_ID": {
10          "name": "MG Bolt",
11          "id": "66148ab731ead8db197c0f96"
12        },
13         "procType_ID": {
14          "id": "66140b65d3c003e3fe2f62e5",
15          "name": "Turning"
16        }
17       },
18       {
19         "id": "6614259dae3308f8aa8a9a70",
20         "duration": 60,
21         "creationDate": "1712596372985",
22         "description": "lab6_test_description",
23         "detail_ID": {
24          "name": "MG Bolt",
25          "id": "66148ab731ead8db197c0f96"
26        },
27         "procType_ID": {
28          "id": "66140b65d3c003e3fe2f62e5",
29          "name": "Turning"
30        }
31       }
32     ]
33   }
34 }
```

Postbot Runner Capture requests Cookies Trash

Створення нового типу обробки:

Untitled Request

Save

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

addDetail [Detail]!

addProcType [ProcType]!

procType newProcType ARS

name String Milling ARS

description String Processing using a milling machine ARS

id ID!

name String!

description String!

mutation AddProcType {

addProcType {

procType: { name: "Milling", description: "Processing using a milling machine" }

} {

id

name

description

}

Variables

Status: 200 OK Time: 91.43 ms Size: 411 B Save as Example

Pretty Table

```
1 {
2   "data": {
3     "addProcType": [
4       {
5         "id": "66140b65d3c003e3fe2f62e5",
6         "name": "Turning",
7         "description": "Processing using a lathe machine"
8       },
9       {
10        "id": "66142652ae3308f8aa8a9a7a",
11        "name": "Milling",
12        "description": "Processing using a milling machine"
13      }
14     ]
15   }
16 }
```

Postbot Runner Capture requests Cookies Trash

Створення нової деталі:

Untitled Request

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

addTechCard [TechCard!]

addDetail [Detail!]

detail newDetail ARG

name String M10 Hex Nut ARG

description String Nut with a hexagonal hole with a diameter of 10 mm ARG

id ID!

name String!

```
1 mutation AddDetail {
2   addDetail {
3     detail {
4       name: "M10 Hex Nut"
5       description: "Nut with a hexagonal hole with a diameter of 10 mm"
6     }
7   }
8   {
9     id
10    name
11    description
12  }
13 }
```

Status: 200 OK Time: 94.37 ms Size: 434 B Save as Example

Body Headers Test Results

Pretty Table

```
1 {
2   "data": {
3     "addDetail": [
4       {
5         "id": "66140ab731ead0db197c0f96",
6         "name": "M8 Bolt",
7         "description": "Standard bolt with a diameter of 8 mm"
8       },
9       {
10        "id": "66142617ae3308f8aa5a9a77",
11        "name": "M10 Hex Nut",
12        "description": "Nut with a hexagonal hole with a diameter of 10 mm"
13      }
14    ]
15  }
16 }
```

Postbot Runner Capture requests Cookies Trash

Отримання всіх технологічних карток:

Untitled Request

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

id ID!

detail_ID Detail!

id ID!

name String!

description String!

procType_ID ProcType!

id ID!

```
1 query GetAllTechCards {
2   getAllTechCards {
3     id
4     duration
5     createDate
6     description
7     detail_ID {
8       id name description
9     }
10    procType_ID {
11      id name description
12    }
13  }
14 }
```

Status: 200 OK Time: 416.92 ms Size: 2.00 KB Save as Example

Body Headers Test Results

Pretty Table

```
1 {
2   "data": {
3     "getAllTechCards": [
4       {
5         "id": "66141311b778274508665f8a",
6         "duration": 45,
7         "createDate": "1712591633400",
8         "description": "test description",
9         "detail_ID": {
10          "id": "66140ab731ead0db197c0f96",
11          "name": "M8 Bolt",
12          "description": "Standard bolt with a diameter of 8 mm"
13        },
14         "procType_ID": {
15          "id": "66140b65d3c003e3fe2f62e5",
16          "name": "Turning",
17          "description": "Processing using a lathe machine"
18        }
19      }
20    ]
21  }
22 }
```

Postbot Runner Capture requests Cookies Trash

Untitled Request

Save

http://localhost:4000/graphql

Query

Body Headers Test Results

Status: 200 OK Time: 416.92 ms Size: 2.00 KB Save as Example

Pretty Table

```
29  },
30  "procType_ID": {
31    "id": "66140b65d3c003e3fe2f62e5",
32    "name": "Turning",
33    "description": "Processing using a lathe machine"
34  },
35  },
36  },
37  {
38    "id": "66142758ae3308f8aa8a9a95",
39    "duration": 55,
40    "creationDate": "1712596824559",
41    "description": "test_3",
42    "detail_ID": {
43      "id": "661426f3ae3308f8aa8a9a89",
44      "name": "Drive Shaft",
45      "description": "Critical component for transmitting motion"
46    },
47    "procType_ID": {
48      "id": "6614268aae3308f8aa8a9a7d",
49      "name": "Welding",
50      "description": "Joining two metal parts together by melting and fusing them"
51    },
52  },
53  {
54    "id": "66142773ae3308f8aa8a9a98",
55    "duration": 55,
56    "creationDate": "1712596851045",
57    "description": "test_4",
58    "detail_ID": {
59      "id": "661426f3ae3308f8aa8a9a89",
60      "name": "Drive Shaft",
61      "description": "Critical component for transmitting motion"
62    },
63  },
64  "procType_ID": {
65    "id": "661426adac3308f8aa8a9a83",
66    "name": "Heat Treatment",
```

Postbot Runner Capture requests Cookies Trash

Отримання всіх типів обробки:

Untitled Request

Save

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

☐ getDetailById Detail

☐ getAllTechCards [TechCard[]]

☒ getAllProcTypes [ProcType[]]

☒ id ID:

```
1 query GetAllProcTypes {
2   getAllProcTypes {
3     id
4     name
5     description
6   }
7 }
8
```

Variables

Body Headers Test Results

Status: 200 OK Time: 45.62 ms Size: 951 B Save as Example

Pretty Table

```
1 {
2   "data": {
3     "getAllProcTypes": [
4       {
5         "id": "66140b65d3c003e3fe2f62e5",
6         "name": "Turning",
7         "description": "Processing using a lathe machine"
8       },
9       {
10        "id": "66142652ae3308f8aa8a9a7a",
11        "name": "Milling",
12        "description": "Processing using a milling machine"
13      },
14      {
15        "id": "6614268aae3308f8aa8a9a7d",
16        "name": "Welding",
17        "description": "Joining two metal parts together by melting and fusing them"
18      },
19      {
20        "id": "6614269cae3308f8aa8a9a80",
21        "name": "Plating",
22        "description": "Applying a thin layer of metal onto a surface for protection or decoration"
23      },
24      {
25        "id": "661426adac3308f8aa8a9a83",
```

Postbot Runner Capture requests Cookies Trash

Отримання всіх деталей:

Untitled Request

Save

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

getProcTypes [ProcType]

getDetails [Detail]

id ID!

name String!

```
1 query GetAllProcTypes {
2   getAllDetails {
3     id
4     name
5     description
6   }
7 }
```

Variables

Status: 200 OK Time: 44.24 ms Size: 893 B Save as Example

Body Headers Test Results

Pretty Table

```
1 {
2   "data": {
3     "getAllDetails": [
4       {
5         "id": "66140ab731ead0db197c0f96",
6         "name": "MS Bolt",
7         "description": "Standard bolt with a diameter of 8 mm"
8       },
9       {
10        "id": "66142617ae3308f8aa8a9a77",
11        "name": "M10 Hex Nut",
12        "description": "Nut with a hexagonal hole with a diameter of 10 mm"
13      },
14      {
15        "id": "661426f3ae3308f8aa8a9a89",
16        "name": "Drive Shaft",
17        "description": "Critical component for transmitting motion"
18      },
19      {
20        "id": "66142704ae3308f8aa8a9a8c",
21        "name": "Gear Wheel",
22        "description": "Circular toothed part used in machinery"
23      },
24      {
25        "id": "66142712ae3308f8aa8a9a8f",
26        "name": "Piston Rod",
27      }
28    ]
29  }
30 }
```

Postbot Runner Capture requests Cookies Trash

Оновлення технологічної картки:

Untitled Request

Save

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

detail_ID ID ARG

procType_ID ID ARG

duration Int ARG

creationDate String ARG

description String

id ID!

detail_ID Detail!

procType_ID ProcType!

duration Int!

```
1 mutation UpdateTechCard {
2   updateTechCard(
3     id: "6614278fae3308f8aa8a9a9b"
4     body: { description: "UPDATED tech card" }
5   ) {
6     id
7     duration
8     creationDate
9     description
10    detail_ID {
11      id name description
12    }
13    procType_ID {
14      id name description
15    }
16  }
17 }
```

Variables

Status: 200 OK Time: 139.37 ms Size: 605 B Save as Example

Body Headers Test Results

Pretty Table

```
1 {
2   "data": {
3     "updateTechCard": {
4       "id": "6614278fae3308f8aa8a9a9b",
5       "duration": 55,
6       "creationDate": "1712596879062",
7       "description": "UPDATED tech card",
8       "detail_ID": {
9         "id": "66142724ae3308f8aa8a9a92",
10        "name": "Sprocket",
11        "description": "Toothed wheel used in bicycles or machinery"
12      },
13      "procType_ID": {
14        "id": "661426adde3308f8aa8a9a83",
15        "name": "Heat Treatment",
16        "description": "Applying controlled heating and cooling to alter material properties"
17      }
18    }
19  }
20 }
```

Postbot Runner Capture requests Cookies Trash

Оновлення типу обробки:

Untitled Request

Save

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

updateTechCard TechCard

updateProcType ProcType

id ID! 66142652ae3308f8aa8a9a7a ARG

body newProcType ARG

name String Milling UPDATED ARG

description String ARG

id ID!

name String!

```
1 mutation UpdateProcType {
2   updateProcType(
3     id: "66142652ae3308f8aa8a9a7a"
4     body: { name: "Milling UPDATED" }
5   ) {
6     id
7     name
8     description
9   }
10 }
11
```

Variables

Status: 200 OK Time: 89.01 ms Size: 320 B Save as Example

Body Headers Test Results

Pretty Table

```
1 {
2   "data": {
3     "updateProcType": {
4       "id": "66142652ae3308f8aa8a9a7a",
5       "name": "Milling UPDATED",
6       "description": "Processing using a milling machine"
7     }
8   }
9 }
```

Postbot Runner Capture requests Cookies Trash

Оновлення деталі:

Untitled Request

Save

http://localhost:4000/graphql

Query Authorization Headers Schema Scripts

Search fields

updateTechCard TechCard

updateProcType ProcType

updateDetail Detail

id ID! 66140ab731ead0db197c0f96 ARG

body newDetail ARG

name String M8 Bolt UPDATED ARG

description String ARG

id ID!

```
1 mutation UpdateDetail {
2   updateDetail(id: "66140ab731ead0db197c0f96", body: { name: "M8 Bolt UPDATED" }) {
3     id
4     name
5     description
6   }
7 }
8
```

Variables

Status: 200 OK Time: 93.79 ms Size: 321 B Save as Example

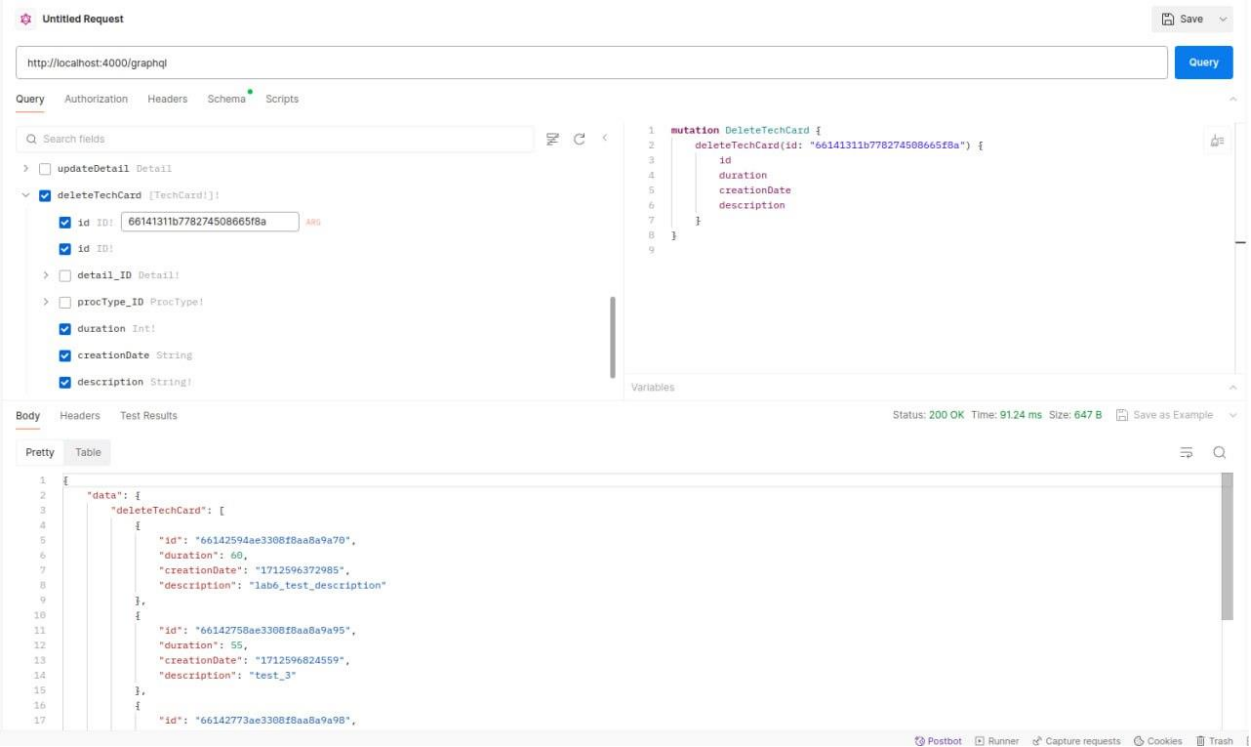
Body Headers Test Results

Pretty Table

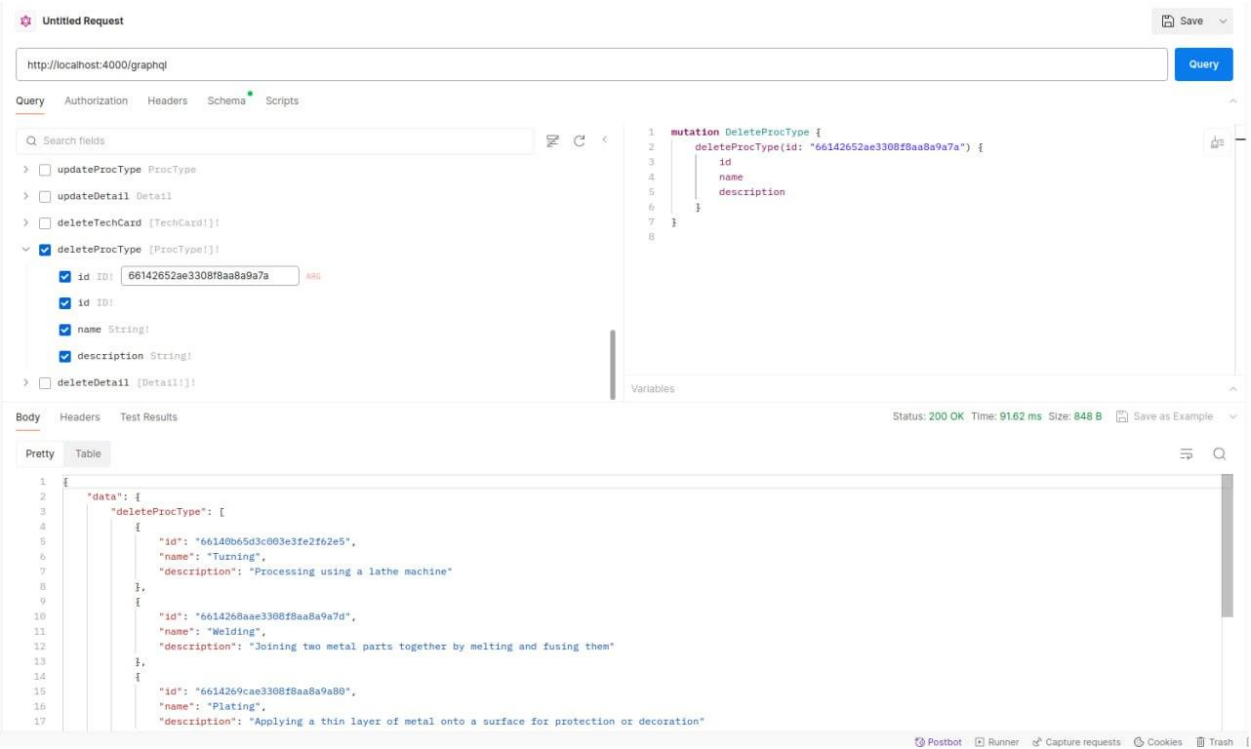
```
1 {
2   "data": {
3     "updateDetail": {
4       "id": "66140ab731ead0db197c0f96",
5       "name": "M8 Bolt UPDATED",
6       "description": "Standard bolt with a diameter of 8 mm"
7     }
8   }
9 }
```

Postbot Runner Capture requests Cookies Trash

Видалення першої технологічної картки (буде повертатись список тих карток, що залишилось, на скріншотах вище можна буде переконатись, що видалена картка дійсно існувала):



Видалення типу обробки:



Видалення деталі (Gear Wheel):

```
17      "description": "Critical component for transmitting motion"
18    },
19    {
20      "id": "66142704ae3308f8aa8a9a8c",
21      "name": "Gear Wheel",
22      "description": "Circular toothed part used in machinery"
23    },
24  ],
25  {
26    "id": "66142712ae3308f8aa8a9a8f",
27    "name": "Piston Rod",
28    "description": "Cylindrical component used in hydraulic systems"
29  },
30  {
31    "id": "66142724ae3308f8aa8a9a92",
32    "name": "Sprocket",
33    "description": "Toothed wheel used in bicycles or machinery"
34  }
35 }
```

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4000/graphql`. The query editor contains a `mutation DeleteDetail { deleteDetail(id: "66142704ae3308f8aa8a9a8c") { id name description } }`. The variables section is empty. The response, shown in the 'Body' tab, is a JSON object indicating a successful deletion: `{ "data": { "deleteDetail": { "id": "66142704ae3308f8aa8a9a8c", "name": "Gear Wheel", "description": "Circular toothed part used in machinery" } } }`. The status bar at the bottom shows 'Status: 200 OK, Time: 94.16 ms, Size: 790 B'.

Висновок:

Під час виконання лабораторної роботи з GraphQL я отримав досвід у створенні схеми GraphQL та реалізації resolvers для операцій CRUD та використання Query та Mutation для взаємодії з базою даних. Це дозволило мені отримати базові знання та навички у роботі з GraphQL.