

Derama, Jeskha Samantha

Users Service

Create User

Operation

```
1 mutation {
2   createUser(name: "John Doe", email: "john@example.com") { ...
3     id
4     name
5     email
6   }
7 }
8
```

Response

```
{
  "data": {
    "createUser": {
      "id": "4",
      "name": "John Doe",
      "email": "john@example.com"
    }
  }
}
```

200 | 23.0ms | 80B

Read Users

Operation

```
1 query {
2   users {
3     id
4     name
5     email
6   }
7 }
8
```

Response

```
{
  "data": {
    "users": [
      {
        "id": "4",
        "name": "John Doe",
        "email": "john@example.com"
      }
    ]
  }
}
```

200 | 17.0ms | 77B

Update User

Operation

```
1 mutation {
2   updateUser(id: 4, name: "Jane Doe") {
3     id
4     name
5   }
6 }
7
```

Response

```
{
  "data": {
    "updateUser": {
      "id": "4",
      "name": "Jane Doe"
    }
  }
}
```

200 | 18.0ms | 53B

Delete User

Operation

```
1 mutation {
2   deleteUser(id: 4) {
3     id
4   }
5 }
6
```

Response

```
{
  "data": {
    "deleteUser": {
      "id": "4"
    }
  }
}
```

200 | 14.0ms | 35B

Posts Service

Create Post

Operation	Response
<pre>1 mutation { 2 createPost(title: "My First Post", content: "This is the 3 content") { 4 id 5 title 6 content 7 } 8 }</pre>	<pre>{ "data": { "createPost": { "id": "3", "title": "My First Post", "content": "This is the content" } } }</pre>

Read Posts

Operation	Response
<pre>1 query { 2 posts { 3 id 4 title 5 content 6 } 7 } 8</pre>	<pre>{ "data": { "posts": [{ "id": "3", "title": "My First Post", "content": "This is the content" }] } }</pre>

Update Post

Operation	Response
<pre>1 mutation { 2 updatePost(id: 3, title: "Updated Title") { 3 id 4 title 5 } 6 } 7</pre>	<pre>{ "data": { "updatePost": { "id": "3", "title": "Updated Title" } } }</pre>

Delete Post

Operation	Response
<pre>1 mutation { 2 deletePost(id: 3) { 3 id 4 } 5 } 6</pre>	<pre>{ "data": { "deletePost": { "id": "3" } } }</pre>

Reflection

a. What do database migrations do and why are they useful?

Database migrations allow you to update the structure of a database while keeping track of changes. They help modify the database schema by adding, removing, or changing tables and fields without losing data. This is especially helpful when using microservices, where each service has its own database and needs to stay updated.

b. How does GraphQL differ from REST for CRUD operations?

GraphQL lets users request exactly the data they need, reducing extra or missing data. Unlike REST, which uses different endpoints for each operation, GraphQL uses one endpoint for all CRUD tasks like reading, updating, and deleting. This makes data retrieval faster and more flexible.