

# User Service

Create:

The screenshot shows the GraphQL Playground interface. The top bar indicates the URL is `http://localhost:4001/graphic`. The main editor contains a mutation query:

```
1 mutation($name: String!, $email: String!){
2   createUser(name: $name, email: $email) {
3     id
4     name
5     email
6   }
7 }
```

The 'Run' button is highlighted. The 'Response' tab on the right shows the JSON output:

```
{
  "data": {
    "createUser": {
      "id": 1,
      "name": "John Lorenz",
      "email": "jla@gmail.com"
    }
  }
}
```

Below the query editor, the 'Variables' tab is active, showing the input variables:

```
1 {
2   "name": "John Lorenz",
3   "email": "jla@gmail.com"
4 }
```

The status bar at the bottom right of the response tab shows a 200 status code, 55.0ms execution time, and 77B response size.

Read:

The screenshot shows the GraphQL Playground interface. The top bar indicates the URL is `http://localhost:4001/graphic`. The main editor contains a query:

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

The 'Run' button is highlighted. The 'Response' tab on the right shows the JSON output:

```
{
  "data": {
    "users": [
      {
        "id": 1,
        "name": "John Lorenz",
        "email": "jla@gmail.com"
      }
    ]
  }
}
```

Below the query editor, the 'Variables' tab is active, showing the input variables:

```
1 {
2   "name": "John Lorenz",
3   "email": "jla@gmail.com"
4 }
```

The status bar at the bottom right of the response tab shows a 200 status code, 20.0ms execution time, and 74B response size.

Update:

The screenshot shows the GraphQL Playground interface with a mutation query. The query is:

```
1 mutation($updateUserId: Int!, $name: String!, $email: String!){
2   updateUser(id: $updateUserId, name: $name, email: $email){
3     id
4     name
5     email
6   }
7 }
```

The response is:

```
{
  "data": {
    "updateUser": {
      "id": 1,
      "name": "John Lorenz CHANGED",
      "email": "jlchanged@gmail.com"
    }
  }
}
```

The status bar indicates a 200 status code, 23.0ms execution time, and 92B response size. The variables section is empty.

Read:

The screenshot shows the GraphQL Playground interface with a query. The query is:

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

The response is:

```
{
  "data": {
    "users": [
      {
        "id": 1,
        "name": "John Lorenz CHANGED",
        "email": "jlchanged@gmail.com"
      },
      {
        "id": 2,
        "name": "Stephanie Recla",
        "email": "stephanie@gmail.com"
      }
    ]
  }
}
```

The status bar indicates a 200 status code, 9.00ms execution time, and 153B response size. The variables section is empty.

Delete:

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4001/graphql`. The operation is a mutation to delete a user by ID.

```
1 mutation($deleteUserId: Int!){
2   deleteUser(id: $deleteUserId) {
3     id
4   }
5 }
```

The response is a JSON object:

```
{
  "data": {
    "deleteUser": {
      "id": 1
    }
  }
}
```

The variables section shows:

```
1 {
2   "deleteUserId": 1
3 }
```

Read:

The screenshot shows the GraphQL Playground interface. The URL is `http://localhost:4001/graphql`. The operation is a query to fetch all users.

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

The response is a JSON object:

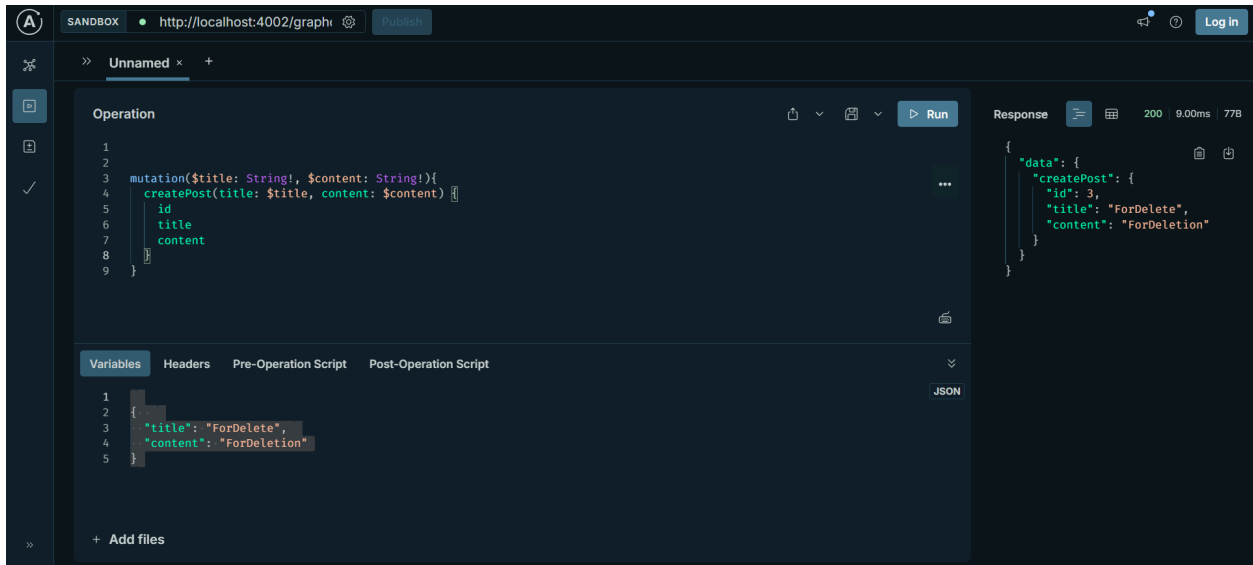
```
{
  "data": {
    "users": [
      {
        "id": 2,
        "name": "Stephanie Recla",
        "email": "stephanie@gmail.com"
      }
    ]
  }
}
```

The variables section shows:

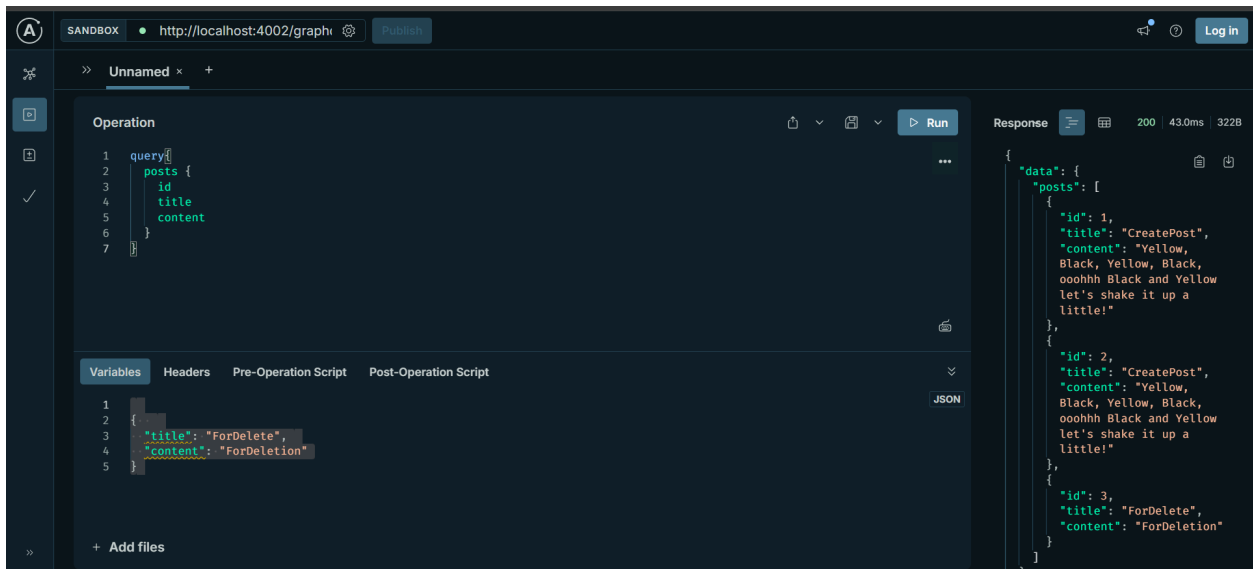
```
1 {
2   "deleteUserId": 1
3 }
```

## Post Service

Create:



Read:



Update:

The screenshot shows the GraphQL Sandbox interface with a successful update mutation. The URL is `http://localhost:4002/graphql`. The operation is a mutation to update a post.

```
1 mutation($updatePostId: Int!, $title: String!, $content: String!){
2   updatePost(id: $updatePostId, title: $title, content: $content) {
3     id
4     title
5     content
6   }
7 }
```

The response is a JSON object:

```
{
  "data": {
    "updatePost": {
      "id": 2,
      "title": "THE FIRST FILIPINO",
      "content": "BY LEON MARIA GUERRERO"
    }
  }
}
```

The variables section shows the input data:

```
1 {
2   "title": "THE FIRST FILIPINO",
3   "content": "BY LEON MARIA GUERRERO",
4   "updatePostId": 2
5 }
6 }
```

The status bar indicates a 200 status code, 17.0ms execution time, and 97B response size.

Read:

The screenshot shows the GraphQL Sandbox interface with a successful query to read posts. The URL is `http://localhost:4002/graphql`. The operation is a query to fetch posts.

```
1 query{
2   posts {
3     id
4     title
5     content
6   }
7 }
```

The response is a JSON object:

```
{
  "data": {
    "posts": [
      {
        "id": 1,
        "title": "CreatePost",
        "content": "Yellow, Black, Yellow, Black,
        ooohhh Black and Yellow
        let's shake it up a
        little!"
      },
      {
        "id": 2,
        "title": "THE FIRST FILIPINO",
        "content": "BY LEON MARIA GUERRERO"
      },
      {
        "id": 3,
        "title": "ForDelete",
        "content": "ForDeletion"
      }
    ]
  }
}
```

The variables section shows the input data:

```
1 {
2   "title": "THE FIRST FILIPINO",
3   "content": "BY LEON MARIA GUERRERO",
4   "updatePostId": 2
5 }
6 }
```

The status bar indicates a 200 status code, 10.0ms execution time, and 271B response size.

Delete:

The screenshot shows the GraphQL Playground interface. The top bar indicates the URL is `http://localhost:4002/graphql`. The main editor is titled "Unnamed" and contains a GraphQL mutation:

```
mutation($deletePostId: Int!){
  deletePost(id: $deletePostId) {
    id
  }
}
```

The "Variables" tab is selected, showing the following JSON:

```
{
  "deletePostId": 3
}
```

The "Response" tab shows the JSON response:

```
{
  "data": {
    "deletePost": {
      "id": 3
    }
  }
}
```

The status bar at the top right shows a 200 status code, 12.0ms execution time, and 338 bytes of data.

Read:

The screenshot shows the GraphQL Playground interface. The top bar indicates the URL is `http://localhost:4002/graphql`. The main editor is titled "Unnamed" and contains a GraphQL query:

```
query{
  posts {
    id
    title
    content
  }
}
```

The "Variables" tab is selected, showing the following JSON:

```
{
  "deletePostId": 3
}
```

The "Response" tab shows the JSON response:

```
{
  "data": {
    "posts": [
      {
        "id": 1,
        "title": "CreatePost",
        "content": "Yellow, Black, Yellow, Black, ooohhh Black and Yellow let's shake it up a little!"
      },
      {
        "id": 2,
        "title": "THE FIRST FILIPINO",
        "content": "BY LEON MARIA GUERRERO"
      }
    ]
  }
}
```

The status bar at the top right shows a 200 status code, 8.00ms execution time, and 2188 bytes of data.

## **Reflection:**

### **What do database migrations do and why are they useful?**

- Database migrations are like a set of instructions that help you update your database structure as your application grows. They let you add, change, or remove tables and columns in a controlled way. They're super helpful because they keep your database organized, make it easy to track changes, and ensure everyone on your team is working with the same setup. Plus, if something goes wrong, you can roll back to a previous version.

### **How does GraphQL differ from REST for CRUD operations?**

- GraphQL lets you ask for exactly the data you want in one request, so you don't get extra stuff you don't need. REST, on the other hand, is like ordering from a fixed menu. You have to go to different places like endpoints to get different things, and sometimes you get extra stuff you didn't need. REST can mean more work. With GraphQL, you just use one URL and tell it what you want to do like read, create, or update in the request.