

1. restaurant:

The screenshot shows a web browser window with the URL `localhost:4000`. The browser has several tabs open, including "GraphQL Restaurant Data Ex...", "CLI | Strapi Documentation", "Running an Express GraphQL", "Ruru - GraphQL/GraphQL IDE", "Array: length - JavaScript | M...", "My Drive - Google Drive", and "Module 24 - Allison Carlisle".

The main content area displays a GraphQL IDE interface. On the left, there is a query editor with the following code:

```

1 query getRestaurants {
2   restaurants {
3     id
4     name
5     description
6     dishes {
7       name
8       price
9     }
10  }
11 }
12 query getRestaurant($iid: Int!) {
13   restaurant(id: $iid) {
14     name
15     description
16     dishes {
17       name
18       price
19     }
20 }
21 }
22 mutation setRestaurants {
23   setRestaurant(input: {
24     name: "Round Table Pizza"
25     description: "Time-honored recipes and gold-standard ingredients"
26   }) {
27     name
28     description
29   }
30 }
31 mutation deleteRestaurants($iid: Int!) {
32   deleteRestaurant(id: $iid) {
33     ok
34   }
35 }
36 mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
37   editRestaurant(id: $iid, name: $name) {
38     name
39     description
40   }
41 }

```

Below the query editor, there is a "Variables" panel with the following content:

```

1 {
2   "iid": 1
3 }

```

On the right side of the IDE, there is a JSON viewer displaying the result of the query:

```

{
  "data": {
    "restaurants": [
      {
        "id": 0,
        "name": "WoodHill ",
        "description": "American cuisine, farm to table, with fresh produce every day",
        "dishes": [
          {
            "name": "Swordfish grill",
            "price": 27
          },
          {
            "name": "Roasted Broccily ",
            "price": 11
          }
        ]
      },
      {
        "id": 1,
        "name": "Fiorellas",
        "description": "Italian-American home cooked food with fresh pasta and sauces",
        "dishes": [
          {
            "name": "Flatbread",
            "price": 14
          },
          {
            "name": "Carbonara",
            "price": 18
          },
          {
            "name": "Spaghetti",
            "price": 19
          }
        ]
      },
      {
        "id": 2,
        "name": "Karma",
        "description": "Malaysian-Chinese-Japanese fusion, with great bar and bartenders",
        "dishes": [
          {
            "name": "Dragon Roll",
            "price": 12
          },
          {
            "name": "Pancake roll ",
            "price": 11
          },
          {
            "name": "Cod cakes",
            "price": 13
          }
        ]
      }
    ]
  }
}

```

At the bottom of the IDE, there is a footer that reads "Community-funded OSS please sponsor" with a heart icon.

2. restaurants: (I used the id of 1)

The screenshot shows the GraphQL Playground interface in a web browser. The left pane contains a query with several mutations and a query to fetch restaurants. The right pane shows the JSON response for the query, which returns an array of restaurant objects. The bottom pane shows the variables for the query.

```
1 query getRestaurants {
2   restaurants {
3     id
4     name
5     description
6     dishes {
7       name
8       price
9     }
10  }
11 }
12 query getRestaurant($iid: Int!) {
13   restaurant(id: $iid) {
14     name
15     description
16     dishes {
17       name
18       price
19     }
20   }
21 }
22 mutation setRestaurants {
23   setRestaurant(input: {
24     name: "Round Table Pizza"
25     description: "Time-honored recipes and gold-standard ingredients"
26   }) {
27     name
28     description
29   }
30 }
31 mutation deleteRestaurants($iid: Int!) {
32   deleteRestaurant(id: $iid) {
33     ok
34   }
35 }
36 mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
37   editRestaurant(id: $iid, name: $name) {
38     name
39     description
40   }
41 }
```

```
{
  "data": {
    "restaurants": [
      {
        "name": "Fiorellas",
        "description": "Italian-American home cooked food with fresh pasta and sauces",
        "dishes": [
          {
            "name": "Flatbread",
            "price": 14
          },
          {
            "name": "Carbonara",
            "price": 18
          },
          {
            "name": "Spaghetti",
            "price": 19
          }
        ]
      }
    ]
  }
}
```

Variables

```
{
  "iid": 1
}
```

Community-funded OSS [please sponsor](#)

3. setRestaurant: (I added Round Table Pizza Restaurant)

The screenshot shows the Ruru GraphQL IDE interface. The left pane displays a GraphQL schema with the following queries and mutations:

```
1 query getRestaurants {
2   restaurants {
3     id
4     name
5     description
6     dishes {
7       name
8       price
9     }
10  }
11 }
12 query getRestaurant($iid: Int!) {
13   restaurant(id: $iid) {
14     name
15     description
16     dishes {
17       name
18       price
19     }
20 }
21 }
22 mutation setRestaurants {
23   setRestaurant(input: {
24     name: "Round Table Pizza"
25     description: "Time-honored recipes and gold-standard ingredients"
26   }) {
27     name
28     description
29   }
30 }
31 mutation deleteRestaurants($iid: Int!) {
32   deleteRestaurant(id: $iid) {
33     ok
34   }
35 }
36 mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
37   editRestaurant(id: $iid, name: $name) {
38     name
39     description
40   }
41 }
```

The right pane shows the JSON response for the `setRestaurant` mutation:

```
{
  "data": {
    "setRestaurant": {
      "name": "Round Table Pizza",
      "description": "Time-honored recipes and gold-standard ingredients"
    }
  }
}
```

Below the schema, the 'Variables' section shows the input for the `setRestaurant` query:

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

The bottom of the interface includes a footer that reads: "Community-funded OSS [please sponsor](#) ❤️".

4. deleteRestaurant: (I used the id of 1 again)

The screenshot shows the Ruru GraphQL IDE interface. The left pane contains a GraphQL query with the following structure:

```
1 query getRestaurants {
2   restaurants {
3     id
4     name
5     description
6     dishes {
7       name
8       price
9     }
10  }
11 }
12 query getRestaurant($iid: Int!) {
13   restaurant(id: $iid) {
14     name
15     description
16     dishes {
17       name
18       price
19     }
20 }
21 }
22 mutation setRestaurants {
23   setRestaurant(input: {
24     name: "Round Table Pizza"
25     description: "Time-honored recipes and gold-standard ingredients"
26   }) {
27     name
28     description
29   }
30 }
31 mutation deleteRestaurants($iid: Int!) {
32   deleteRestaurant(id: $iid) {
33     ok
34   }
35 }
36 mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
37   editRestaurant(id: $iid, name: $name) {
38     name
39     description
40   }
41 }
```

The right pane shows the JSON response for the query:

```
{
  "data": {
    "deleteRestaurant": {
      "ok": true
    }
  }
}
```

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

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

The bottom of the interface features a footer that reads: "Community-funded OSS 🙌 please sponsor ❤️".

5. editRestaurant: (Because I just deleted my restaurant with an id of 1, I used an id of 2 and changed its name to Patxi's Pizza)

The screenshot shows the Ruru GraphQL IDE interface. On the left, a GraphQL query is defined with the following structure:

```
1 query getRestaurants {
2   restaurants {
3     id
4     name
5     description
6     dishes {
7       name
8       price
9     }
10  }
11 }
12 query getRestaurant($iid: Int!) {
13   restaurant(id: $iid) {
14     name
15     description
16     dishes {
17       name
18       price
19     }
20 }
21 }
22 mutation setRestaurants {
23   setRestaurant(input: {
24     name: "Round Table Pizza"
25     description: "Time-honored recipes and gold-standard ingredients"
26   }) {
27     name
28     description
29   }
30 }
31 mutation deleteRestaurants($iid: Int!) {
32   deleteRestaurant(id: $iid) {
33     ok
34   }
35 }
36 mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
37   editRestaurant(id: $iid, name: $name) {
38     name
39     description
40   }
41 }
```

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

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

On the right, the JSON response for the query is displayed:

```
{
  "data": {
    "getRestaurants": {
      "restaurants": [
        {
          "id": 1,
          "name": "Patxi's Pizza",
          "description": "Malaysian-Chinese-Japanese fusion, with great bar and bartenders"
        }
      ]
    }
  }
}
```

The interface includes a top navigation bar with various tabs and a bottom status bar that reads "Community-funded OSS, please sponsor ❤️".

6. One more restaurants so you can see what the array looks like now:

The screenshot shows the Ruru GraphQL IDE interface. On the left, a query is defined with the following schema:

```

1 query getRestaurants {
2   restaurants {
3     id
4     name
5     description
6     dishes {
7       name
8       price
9     }
10  }
11 }
12 query getRestaurant($iid: Int!) {
13   restaurant(id: $iid) {
14     name
15     description
16     dishes {
17       name
18       price
19     }
20   }
21 }
22 mutation setRestaurants {
23   setRestaurant(input: {
24     name: "Round Table Pizza"
25     description: "Time-honored recipes and gold-standard ingredients"
26   }) {
27     name
28     description
29   }
30 }
31 mutation deleteRestaurants($iid: Int!) {
32   deleteRestaurant(id: $iid) {
33     ok
34   }
35 }
36 mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
37   editRestaurant(id: $iid, name: $name) {
38     name
39     description
40   }
41 }

```

Below the query, the 'Variables' section shows:

```

1 {
2   "iid": 2
3 }

```

On the right, the JSON response is displayed:

```

{
  "data": {
    "restaurants": [
      {
        "id": 0,
        "name": "WoodHill ",
        "description": "American cuisine, farm to table, with fresh produce every day",
        "dishes": [
          {
            "name": "Swordfish grill",
            "price": 27
          },
          {
            "name": "Roasted Broccily ",
            "price": 11
          }
        ]
      },
      {
        "id": 2,
        "name": "Patxi's Pizza",
        "description": "Malaysian-Chinese-Japanese fusion, with great bar and bartenders",
        "dishes": [
          {
            "name": "Dragon Roll",
            "price": 12
          },
          {
            "name": "Pancake roll ",
            "price": 11
          },
          {
            "name": "Cod cakes",
            "price": 13
          }
        ]
      },
      {
        "id": 3,
        "name": "Round Table Pizza",
        "description": "Time-honored recipes and gold-standard ingredients",
        "dishes": null
      }
    ]
  }
}

```

At the bottom right, there is a link: [Community-funded OSS, please sponsor](#).

Here's my code:

```
// This project works as of june 27, 2024 (I edited the starter file)
// Step 1: in a terminal, navigate to the folder this file is in (currently called GraphQLExpress)
// Step 2: in that terminal window, type npm install express graphql-http graphql --save
// Step 3: in that terminal window, type npm install --save ruru
// Step 4: in that terminal window, type node index.js
// Step 5: in the browser, navigate to http://localhost:4000
// Step 6: Press and hold on the play button to select different actions

var express = require("express")
var { createHandler } = require("graphql-http/lib/use/express")
var { buildSchema } = require("graphql")

var restaurants = [
  {
    "id": 0,
    "name": "WoodsHill ",
    "description": "American cuisine, farm to table, with fresh produce every day",
    "dishes": [
      {
        "name": "Swordfish grill",
        "price": 27
      },
      {
        "name": "Roasted Broccily ",
        "price": 11
      }
    ]
  }
],
```

```
{
  "id": 1,
  "name": "Fiorellas",
  "description": "Italian-American home cooked food with fresh pasta and sauces",
  "dishes": [
    {
      "name": "Flatbread",
      "price": 14
    },
    {
      "name": "Carbonara",
      "price": 18
    },
    {
      "name": "Spaghetti",
      "price": 19
    }
  ]
},
{
  "id": 2,
  "name": "Karma",
  "description": "Malaysian-Chinese-Japanese fusion, with great bar and bartenders",
  "dishes": [
    {
      "name": "Dragon Roll",
      "price": 12
    },
    {
      "name": "Pancake roll ",

```



```
        "price": 11
    },
    {
        "name": "Cod cakes",
        "price": 13
    }
]
}
```

```
// Construct a schema, using GraphQL schema language
var schema = buildSchema(`
    type Query {
        restaurant(id: Int): restaurant
        restaurants: [restaurant]
    },
    type restaurant {
        id: Int
        name: String
        description: String
        dishes: [Dish]
    },
    type Dish {
        name: String
        price: Int
    }
    input restaurantInput {
        name: String
        description: String
    }
`)
```

```

}
type DeleteResponse {
  ok: Boolean!
}
type Mutation {
  setRestaurant(input: restaurantInput): restaurant
  deleteRestaurant(id: Int!): DeleteResponse
  editRestaurant(id: Int!, name: String!): restaurant
}
`)
// The root provides a resolver function for each API endpoint
var root = {
  restaurant(arg) {
    return restaurants[arg.id]
  },
  restaurants() {
    return restaurants
  },
  setRestaurant({input}) {
    restaurants.push({id: restaurants.length, name: input.name, description: input.description})
    return input
  },
  deleteRestaurant({id}) {
    const ok = restaurants.some(element => element.id === id)
    let delr = restaurants.find(element => element.id === id);
    let index = restaurants.indexOf(delr);
    restaurants.splice(index, 1)
    console.log(JSON.stringify(delr))
    return {ok}
  },

```

```

editRestaurant({id, ...restaurant}) {
  if(!restaurants.some(element => element.id === id)) {
    throw new Error("That restaurant doesn't exist.")
  }
  restToEdit = restaurants.find(element => element.id === id)
  index = restaurants.indexOf(restToEdit);
  restaurants[index] = {
    ...restToEdit, ...restaurant
  }

  console.log(JSON.stringify(restaurants[index]))
  return restaurants[index]
}
}

var app = express()

// Create and use the GraphQL handler.
app.all(
  "/graphql",
  createHandler({
    schema: schema,
    rootValue: root,
  })
)

var { ruruHTML } = require("ruru/server")
// Serve the GraphiQL IDE.
app.get("/", (_req, res) => {
  res.type("html")
  res.end(ruruHTML({ endpoint: "/graphql" }))
})

```

```
})  
// Start the server at port  
app.listen(4000)  
console.log("Running a GraphQL API server at http://localhost:4000")
```