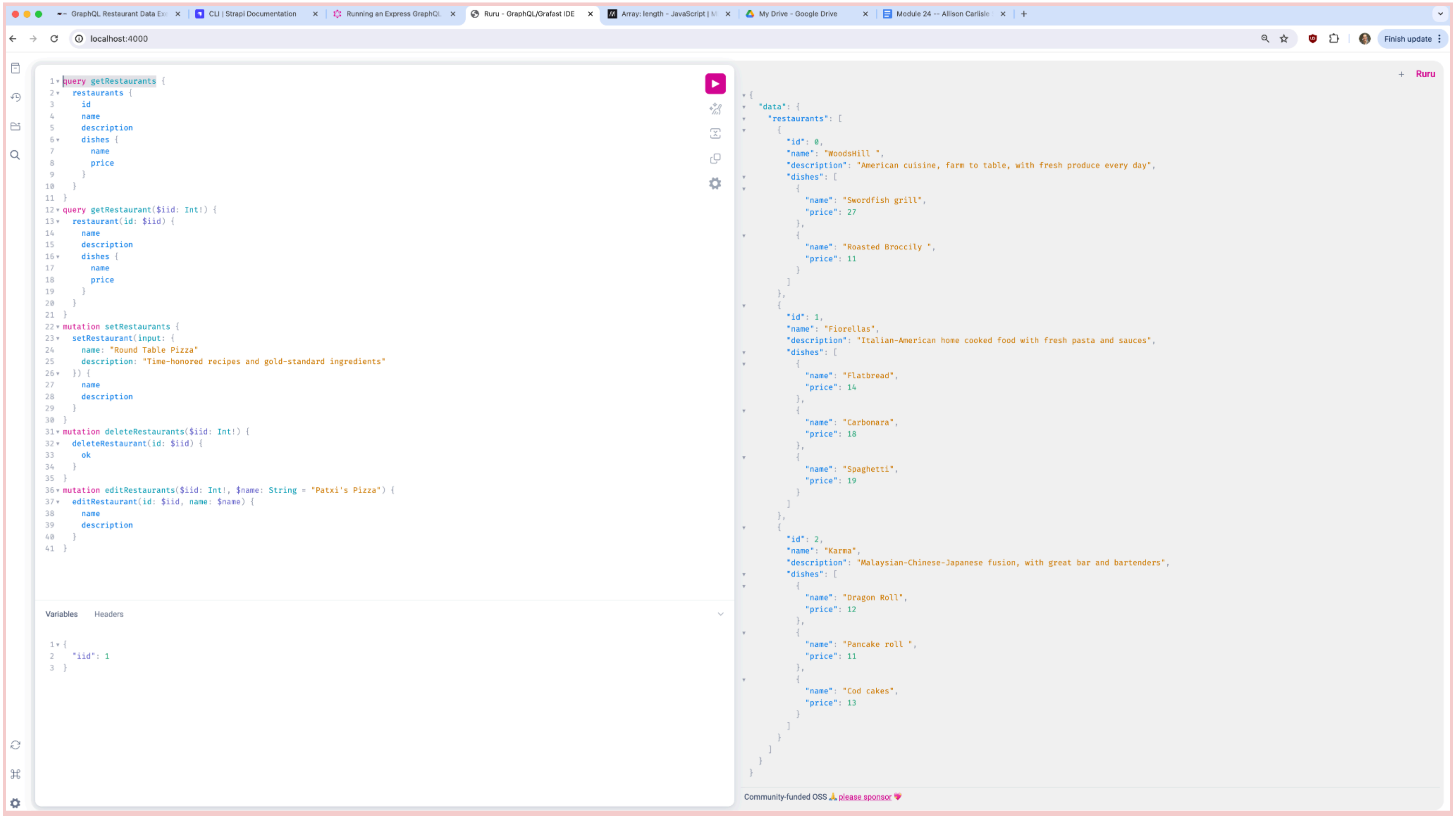
Allison Carlisle

1. restaurant:



```
query getRestaurants {
  restaurants {
    id
    name
    description
    dishes {
      name
      price
    }
  }
}
query getRestaurant($iid: Int!) {
  restaurant(id: $iid) {
    name
    description
    dishes {
      name
      price
    }
  }
}
mutation setRestaurants {
  setRestaurant(input: {
    name: "Round Table Pizza"
    description: "Time-honored recipes and gold-standard ingredients"
  }) {
    name
    description
  }
}
mutation deleteRestaurants($iid: Int!) {
  deleteRestaurant(id: $iid) {
    ok
  }
}
mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
  editRestaurant(id: $iid, name: $name) {
    name
    description
  }
}
```

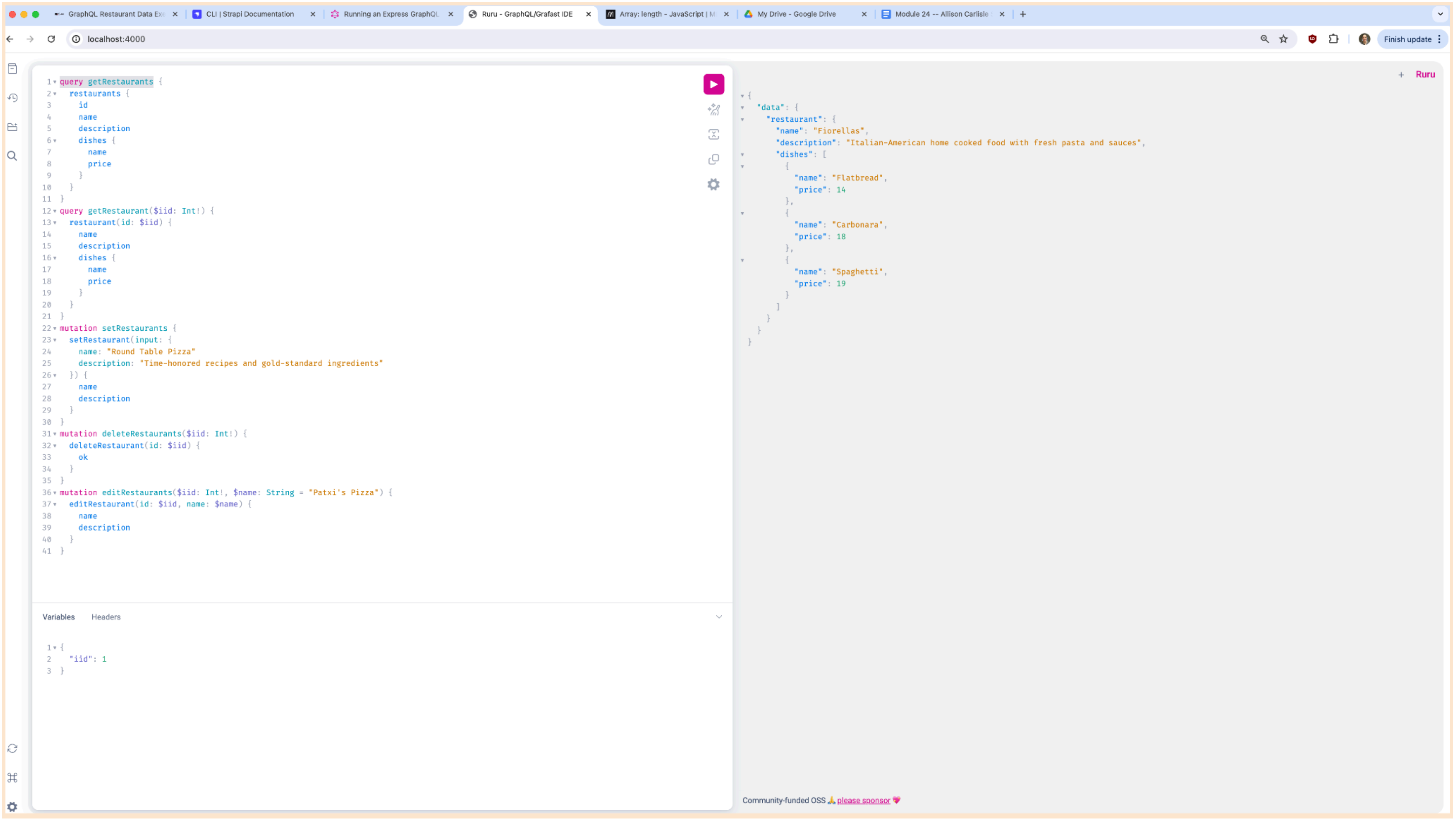
Variables Headers

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

```
{
  "data": {
    "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
          }
        ]
      }
    ]
  }
}
```

Community-funded OSS 🙏 please sponsor 💖

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



localhost:4000

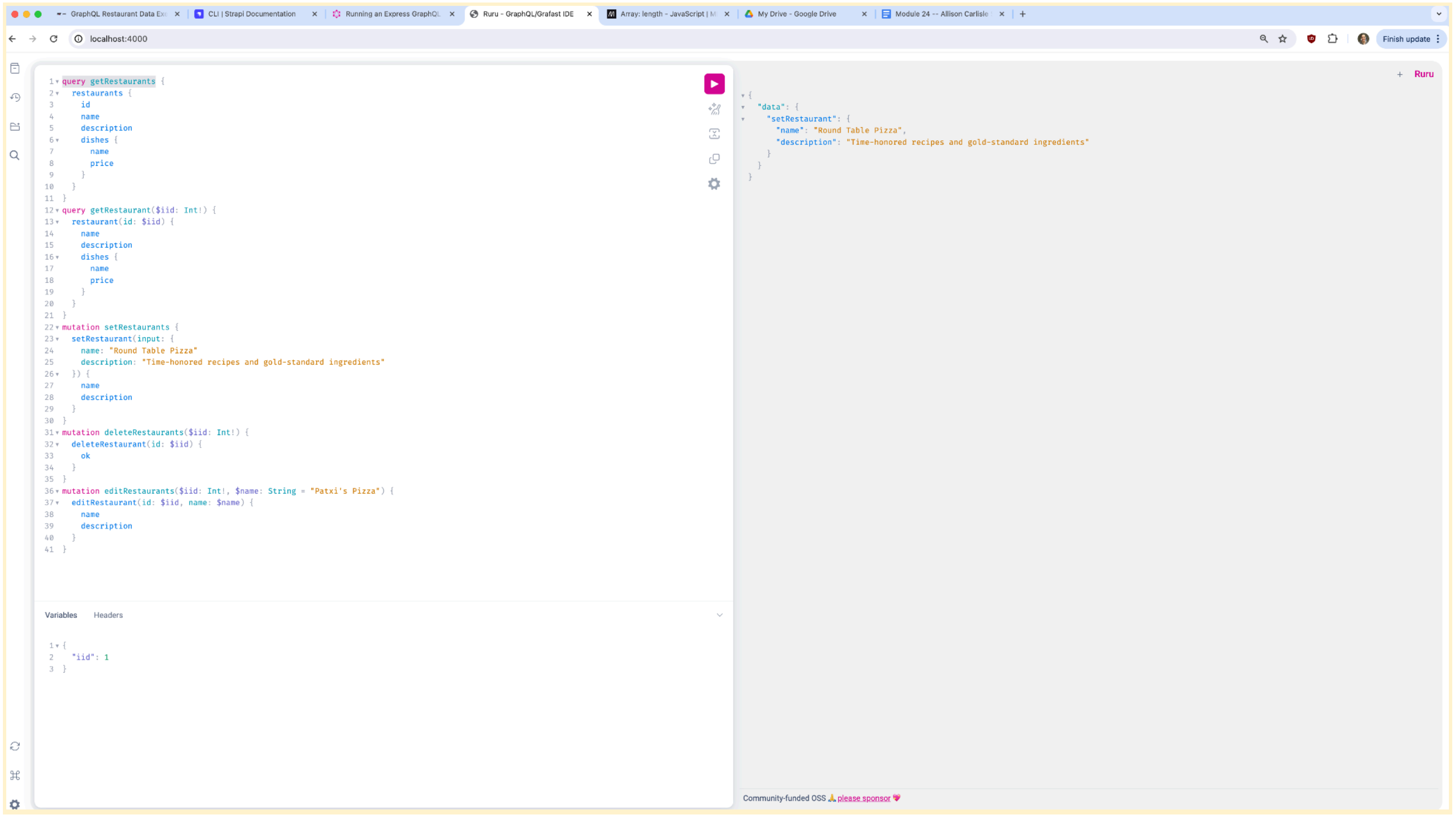
```graphql
query getRestaurants {
  restaurants {
    id
    name
    description
    dishes {
      name
      price
    }
  }
}
query getRestaurant($iid: Int!) {
  restaurant(id: $iid) {
    name
    description
    dishes {
      name
      price
    }
  }
}
mutation setRestaurants {
  setRestaurant(input: {
    name: "Round Table Pizza"
    description: "Time-honored recipes and gold-standard ingredients"
  }) {
    name
    description
  }
}
mutation deleteRestaurants($iid: Int!) {
  deleteRestaurant(id: $iid) {
    ok
  }
}
mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
  editRestaurant(id: $iid, name: $name) {
    name
    description
  }
}
```

Variables   Headers

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

```json
{
  "data": {
    "restaurant": {
      "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
        }
      ]
    }
  }
}
```

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



localhost:4000

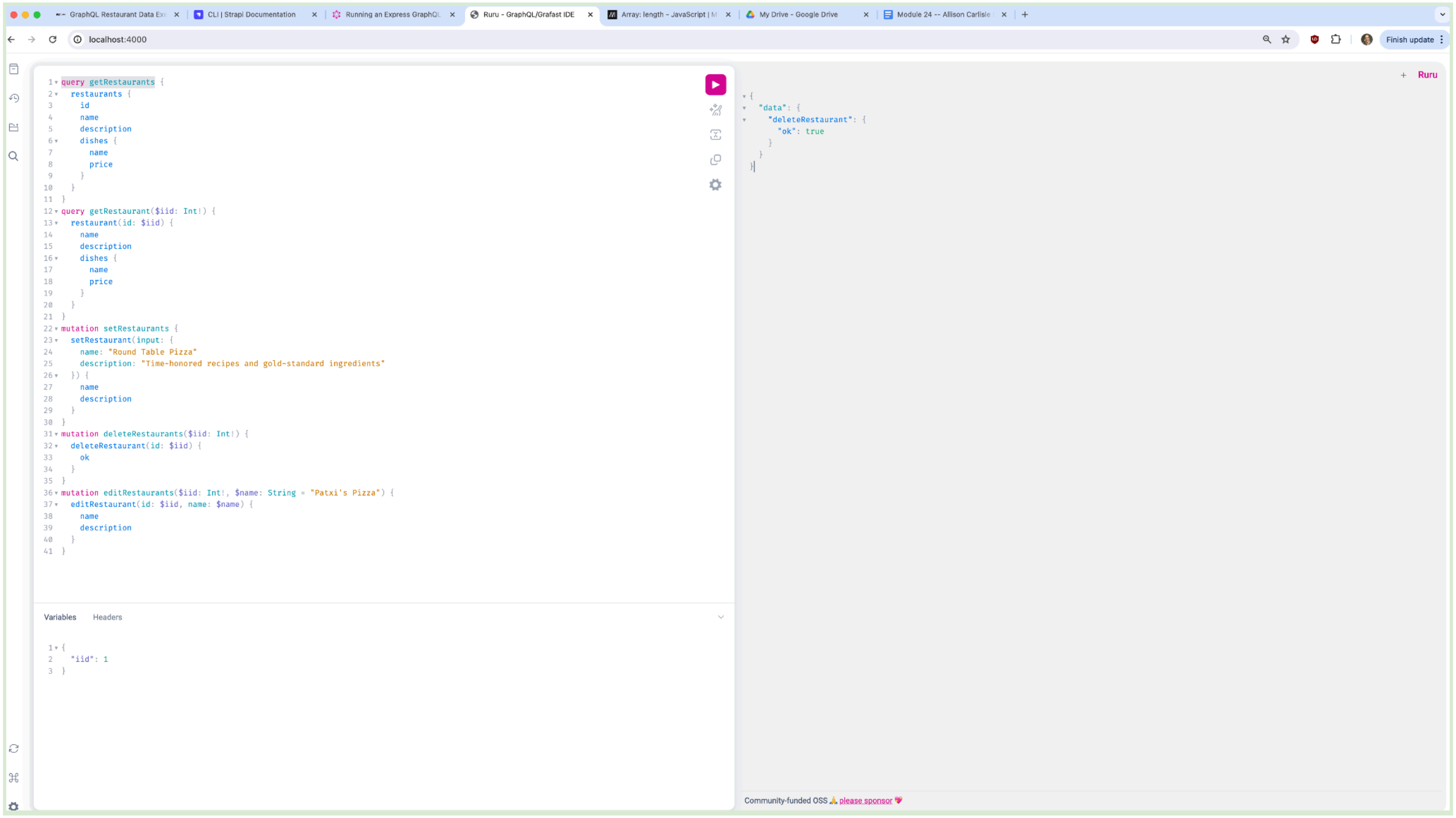
```
query getRestaurants {
  restaurants {
    id
    name
    description
    dishes {
      name
      price
    }
  }
}
query getRestaurant($iid: Int!) {
  restaurant(id: $iid) {
    name
    description
    dishes {
      name
      price
    }
  }
}
mutation setRestaurants {
  setRestaurant(input: {
    name: "Round Table Pizza"
    description: "Time-honored recipes and gold-standard ingredients"
  }) {
    name
    description
  }
}
mutation deleteRestaurants($iid: Int!) {
  deleteRestaurant(id: $iid) {
    ok
  }
}
mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
  editRestaurant(id: $iid, name: $name) {
    name
    description
  }
}
```

Variables Headers

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

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

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



localhost:4000

```graphql
query getRestaurants {
  restaurants {
    id
    name
    description
    dishes {
      name
      price
    }
  }
}
query getRestaurant($iid: Int!) {
  restaurant(id: $iid) {
    name
    description
    dishes {
      name
      price
    }
  }
}
mutation setRestaurants {
  setRestaurant(input: {
    name: "Round Table Pizza"
    description: "Time-honored recipes and gold-standard ingredients"
  }) {
    name
    description
  }
}
mutation deleteRestaurants($iid: Int!) {
  deleteRestaurant(id: $iid) {
    ok
  }
}
mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
  editRestaurant(id: $iid, name: $name) {
    name
    description
  }
}
```

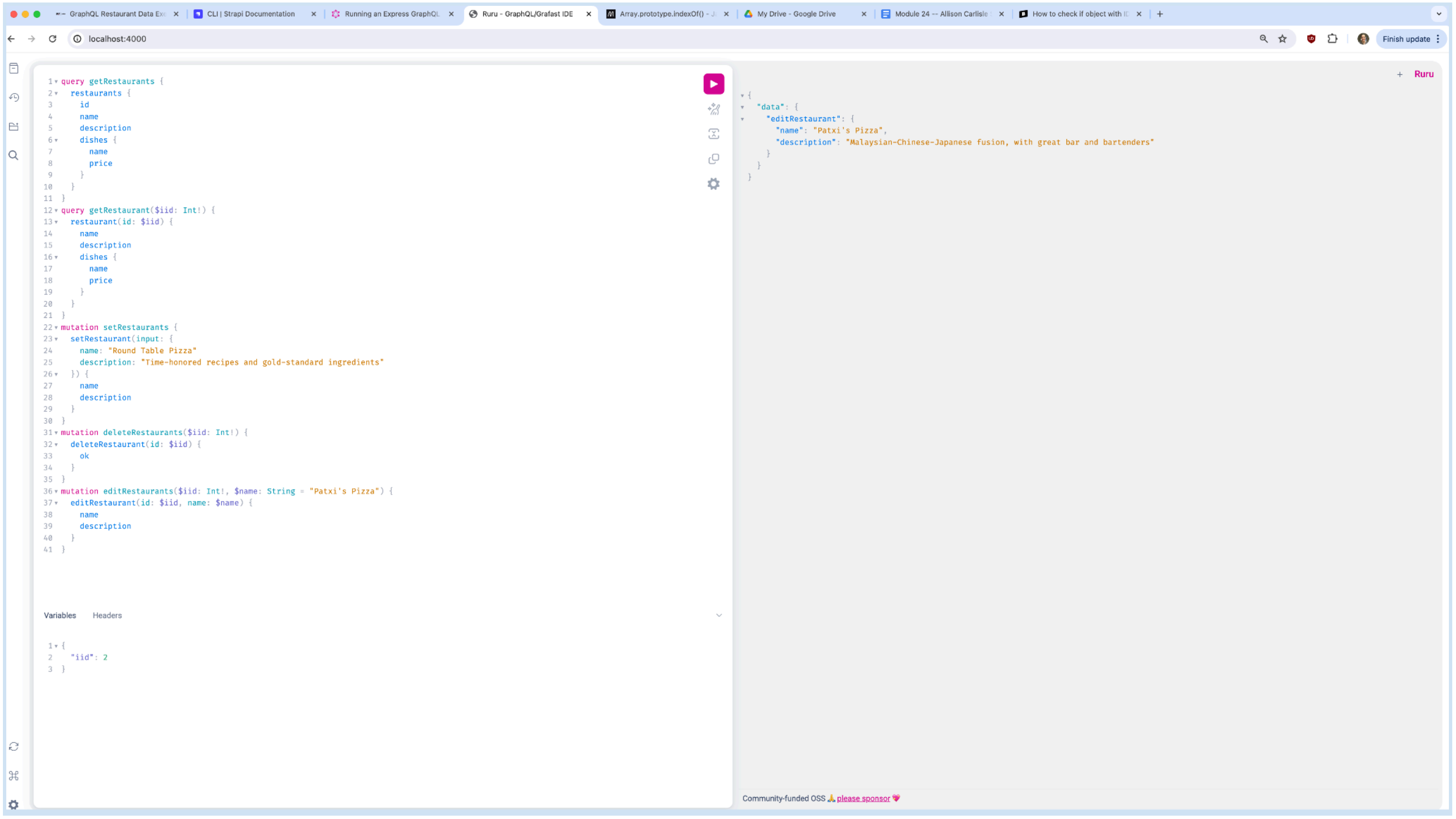
Variables Headers

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

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

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)

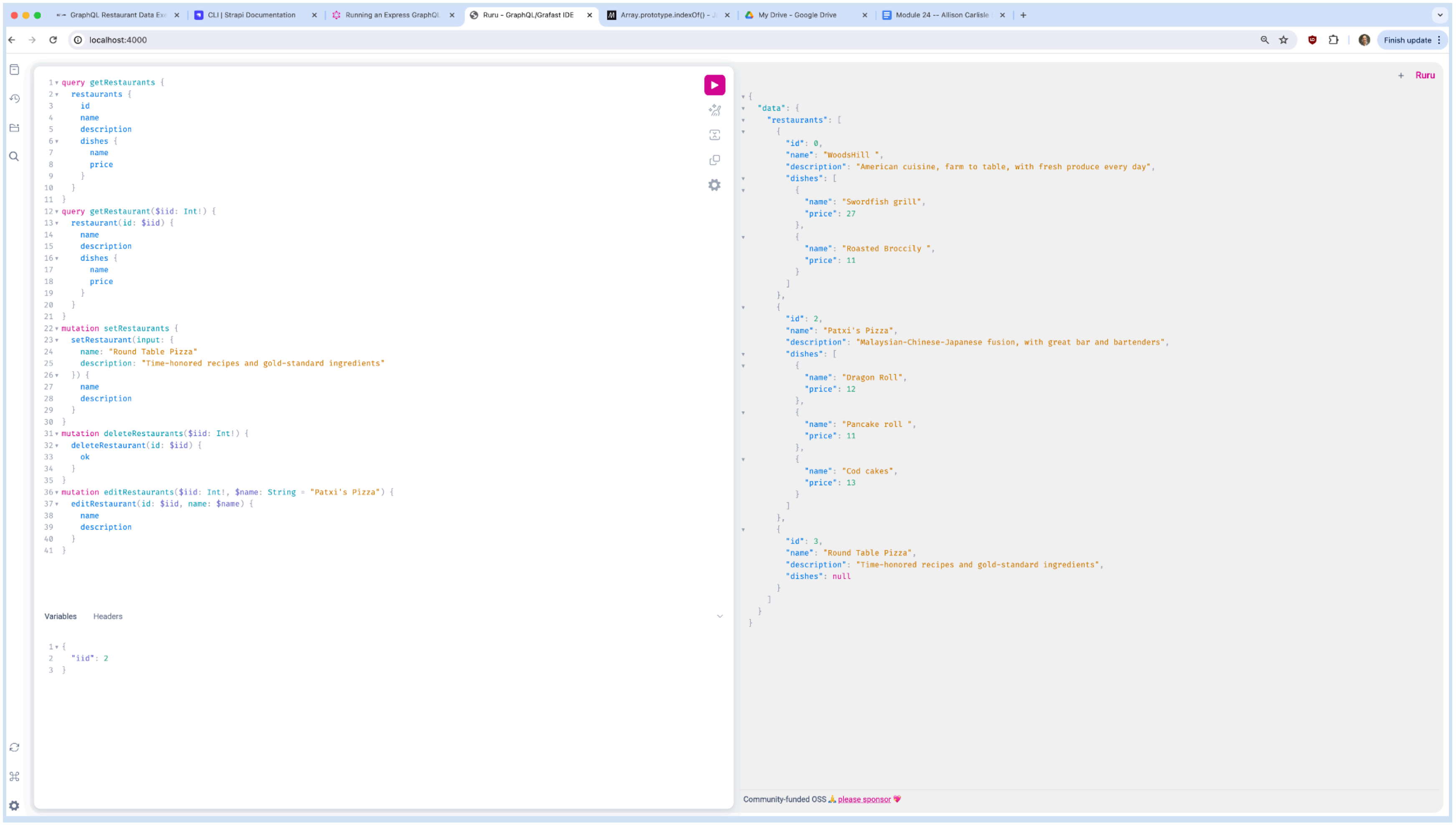

GraphQL Restaurant Data Ex
CLI | Strapi Documentation
Running an Express GraphQL
Ruru - GraphQL/Grafast IDE
Array.prototype.indexOf() - J
My Drive - Google Drive
Module 24 -- Allison Carlisle
How to check if object with I
localhost:4000
Finish update
Ruru
Variables
Headers
Community-funded OSS please sponsor

```graphql
query getRestaurants {
  restaurants {
    id
    name
    description
    dishes {
      name
      price
    }
  }
}
query getRestaurant($iid: Int!) {
  restaurant(id: $iid) {
    name
    description
    dishes {
      name
      price
    }
  }
}
mutation setRestaurants {
  setRestaurant(input: {
    name: "Round Table Pizza"
    description: "Time-honored recipes and gold-standard ingredients"
  }) {
    name
    description
  }
}
mutation deleteRestaurants($iid: Int!) {
  deleteRestaurant(id: $iid) {
    ok
  }
}
mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
  editRestaurant(id: $iid, name: $name) {
    name
    description
  }
}
```

```json
{
  "iid": 2
}
```

```json
{
  "data": {
    "editRestaurant": {
      "name": "Patxi's Pizza",
      "description": "Malaysian-Chinese-Japanese fusion, with great bar and bartenders"
    }
  }
}
```

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



```
query getRestaurants {
  restaurants {
    id
    name
    description
    dishes {
      name
      price
    }
  }
}
query getRestaurant($iid: Int!) {
  restaurant(id: $iid) {
    name
    description
    dishes {
      name
      price
    }
  }
}
mutation setRestaurants {
  setRestaurant(input: {
    name: "Round Table Pizza"
    description: "Time-honored recipes and gold-standard ingredients"
  }) {
    name
    description
  }
}
mutation deleteRestaurants($iid: Int!) {
  deleteRestaurant(id: $iid) {
    ok
  }
}
mutation editRestaurants($iid: Int!, $name: String = "Patxi's Pizza") {
  editRestaurant(id: $iid, name: $name) {
    name
    description
  }
}
```

Variables Headers

```
{
  "iid": 2
}
```

```
{
  "data": {
    "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": 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
      }
    ]
  }
}
```

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")
```