Notes

<https://www.youtube.com/watch?v=ZQL7tL2S0oQ>

<https://www.youtube.com/watch?v=qqzIA1BQ_ys&list=PLillGF-RfqbYZty73_PHBqKRDnv7ikh68&index=2>

INXL+trugao@AVEN-1B7G453 MINGW64 /c/myprojects

$ mkdir react-express-graphql

INXL+trugao@AVEN-1B7G453 MINGW64 /c/myprojects

$ cd react-express-graphql/

INXL+trugao@AVEN-1B7G453 MINGW64 /c/myprojects/react-express-graphql

$ touch server.js

INXL+trugao@AVEN-1B7G453 MINGW64 /c/myprojects/react-express-graphql

$ npm init -y

Wrote to C:\myprojects\react-express-graphql\package.json:

{

"name": "react-express-graphql",

"version": "1.0.0",

"description": "",

"main": "server.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node server.js"

},

"keywords": [],

"author": "",

"license": "ISC"

}

INXL+trugao@AVEN-1B7G453 MINGW64 /c/myprojects/react-express-graphql

$ npm install express nodemon

C:\myprojects\react-express-graphql>npm start

> react-express-graphql@1.0.0 start C:\myprojects\react-express-graphql

> nodemon server.js

[nodemon] 2.0.15

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): \*.\*

[nodemon] watching extensions: js,mjs,json

[nodemon] starting `node server.js`

Server Running on port 5000

Open <http://localhost:5000/>

Graphical user interface, text, application, chat or text message

Description automatically generated

**Repository on github:**

Create new repository on github and then perform following steps:

Graphical user interface, text, application, email

Description automatically generated

C:\myprojects\react-express-graphql>echo "# react-express-graphql" >> README.md

C:\myprojects\react-express-graphql>git init

Initialized empty Git repository in C:/myprojects/react-express-graphql/.git/

C:\myprojects\react-express-graphql>git add README.md

C:\myprojects\react-express-graphql>git commit -m "first commit"

[master (root-commit) 46ee8a2] first commit

1 file changed, 1 insertion(+)

create mode 100644 README.md

C:\myprojects\react-express-graphql>git branch -M main

C:\myprojects\react-express-graphql>git remote add origin https://github.com/truptigaonkar/react-express-graphql.git

C:\myprojects\react-express-graphql>git push -u origin main

1. Basic graphql queries

**server.js**

*const* express *=* *require*("express");

*const* { buildSchema } *=* *require*("graphql");

*const* { graphqlHTTP } *=* *require*("express-graphql");

*const* app *=* express();

*const* schema *=* buildSchema(`

    type User {

        name: String

        age: Int

        college: String

    }

    type Query {

        hello: String

        welcomeMessage(name: String, dayOfWeek: String): String

        getUSer: User

    }

`);

*// Resolver*

*const* root *=* {

  hello: () *=>* {

*return* "hello world";

  },

  welcomeMessage: (args) *=>* {

*return* `Hey, ${args.name} Hows life. Today is ${args.dayOfWeek}`;

  },

  getUSer: () *=>* {

*const* user *=* {

      name: "John Doe",

      age: 26,

      college: "Ruparel",

    };

*return* user;

  },

};

app.use(

  "/graphql",

  graphqlHTTP({

    graphiql: *true*,

    schema: schema,

    rootValue: root,

  })

); *// http://localhost:4000/graphql*

app.listen(1000, () *=>* console.log(`Server on port 1000`));

Run <http://localhost:1000/graphql>

|  |  |
| --- | --- |
| Query | Result |
| {  hello  welcomeMessage(name: "John", dayOfWeek: "Sunday")  getUSer {  name  age  college  }  } | {  "data": {  "hello": "hello world",  "welcomeMessage": "Hey, John Hows life. Today is Sunday",  "getUSer": {  "name": "John Doe",  "age": 26,  "college": "Ruparel"  }  }  } |

Scatter chart

Description automatically generated with low confidence

1. Listing customers and single customer

**server1.tsx**

*const* express *=* *require*("express");

*const* { graphqlHTTP } *=* *require*("express-graphql");

*const* {

  GraphQLSchema,

  GraphQLObjectType,

  GraphQLString,

  GraphQLList,

  GraphQLInt,

  GraphQLNonNull,

} *=* *require*("graphql");

*const* cors *=* *require*( `cors` );

*const* app *=* express();

app.get("/", (req, res) *=>* {

  res.send("Up and running with graphql");

});

*const* customers *=* [

  { id: "1", name: "John Doe", age: 35 },

  { id: "2", name: "Steve Smith", age: 25 },

  { id: "3", name: "Sara Williams", age: 32 },

];

*const* CustomerType *=* *new* GraphQLObjectType({

  name: "Customer",

  description: "This is customer",

  fields: () *=>* ({

    id: { type: GraphQLInt },

    name: { type: GraphQLString },

    age: { type: GraphQLInt },

  }),

});

*const* RootQueryType *=* *new* GraphQLObjectType({

  name: "Query",

  description: "Root Query",

  fields: () *=>* ({

    customers: {

      type: *new* GraphQLList(CustomerType),

      description: "List of all customers",

      resolve: () *=>* customers,

    },

    customer:{

        type:CustomerType,

        description: "Single customer",

        args:{ id:{type:GraphQLInt} },

        resolve(parentValue, args){

*for*(*let* i *=* 0;i *<* customers.length;i*++*){

*if*(customers[i].id *==* args.id){

*return* customers[i];

                }

            }

        }

    },

  }),

});

*const* schema *=* *new* GraphQLSchema({

  query: RootQueryType,

});

app.use( cors() );

app.use(

    "/graphql",

    graphqlHTTP({

      graphiql: *true*,

      schema: schema,

    })

  );

app.listen(2000, () *=>* console.log("Server Running on port 2000"));

Result: <http://localhost:2000/graphql>

|  |  |
| --- | --- |
| Query | Result |
| {  customers {  id  name  age  }  } | {  "data": {  "customers": [  {  "id": 1,  "name": "John Doe",  "age": 35  },  {  "id": 2,  "name": "Steve Smith",  "age": 25  },  {  "id": 3,  "name": "Sara Williams",  "age": 32  }  ]  }  } |

**React** to show list of customers

C:\myprojects\react-express-graphql\client>npm start’

<http://localhost:3000/>

**Result:**

Graphical user interface, text

Description automatically generated

1. **Apollo server:** <https://www.apollographql.com/docs/apollo-server/getting-started/>

**server2.jsx**

*const* { ApolloServer, gql } *=* *require*('apollo-server');

*const* typeDefs *=* gql`

  type Book {

    title: String

    author: String

  }

  type Query {

    books: [Book]

  }

`;

*const* books *=* [

  {

    title: 'The Awakening',

    author: 'Kate Chopin',

  },

  {

    title: 'City of Glass',

    author: 'Paul Auster',

  },

];

*const* resolvers *=* {

  Query: {

    books: () *=>* books,

  },

};

*const* server *=* *new* ApolloServer({ typeDefs, resolvers });

server.listen().then(({ url }) *=>* {

  console.log(`🚀  Server ready at ${url}`);

});

Result:

<http://localhost:4000/>

|  |  |
| --- | --- |
| Query | Result |
| query ExampleQuery {    books {      title      author    }  } | {    "data": {      "books": [        {          "title": "The Awakening",          "author": "Kate Chopin"        },        {          "title": "City of Glass",          "author": "Paul Auster"        }      ]    }  } |

1. Todos with apollo server (Node) and client (React)

**server3-todos.js**

*const* express *=* *require*('express');

*const* { ApolloServer, gql } *=* *require*('apollo-server');

*const* cors *=* *require*('cors');

*let* todos *=* [

  {

    id: Date.now().toString(),

    text: 'Hello from GraphQL',

    completed: *true*,

  },

];

*const* typeDefs *=* gql`

  type Todo {

    id: String

    text: String

    completed: Boolean

  }

  type Query {

    todos: [Todo]!

  }

  type Mutation {

    createTodo(text: String!):String

    removeTodo(id: String!):String

    updateTodo(id: String!):String

  }

`;

*const* resolvers *=* {

  Query: {

    todos: () *=>* todos,

  },

  Mutation: {

    createTodo: (parent, args, context, info) *=>* {

*return* todos.push({

        id: Date.now().toString(),

        text: args.text,

        completed: *false*,

      });

    },

    removeTodo: (parent, args, context, info) *=>* {

*for* (*let* i *in* todos) {

*if* (todos[i].id *===* args.id) {

          todos.splice(i, 1);

        }

      }

*return* args.id;

    },

    updateTodo: (parent, args, context, info) *=>* {

*for* (*let* i *in* todos) {

*if* (todos[i].id *===* args.id) {

          todos[i].completed *=* *!*todos[i].completed;

        }

      }

*return* args.id;

    }

  }

};

*const* server *=* *new* ApolloServer({ typeDefs, resolvers });

server.listen(2000, () *=>* console.log("Server Running on port 2000"));

Result:

<http://localhost:2000/> which will connect to <https://studio.apollographql.com/sandbox/explorer>

|  |  |
| --- | --- |
| Query | Result |
| query Query {  todos {  id  text  completed  }  } | {    "data": {      "todos": [        {          "id": "1640603293343",          "text": "Hello from GraphQL",          "completed": true        }      ]    }  } |
| mutation {    createTodo (text: "Do washing")  } | {    "data": {      "createTodo": "2"    }  } |
| mutation {    removeTodo (id: "1640603527561")  } | {    "data": {      "removeTodo": "1640603527561"    }  } |