graphql

# Graphql react

## 1. Start:

Link course: <https://www.udemy.com/graphql-with-react-course/>

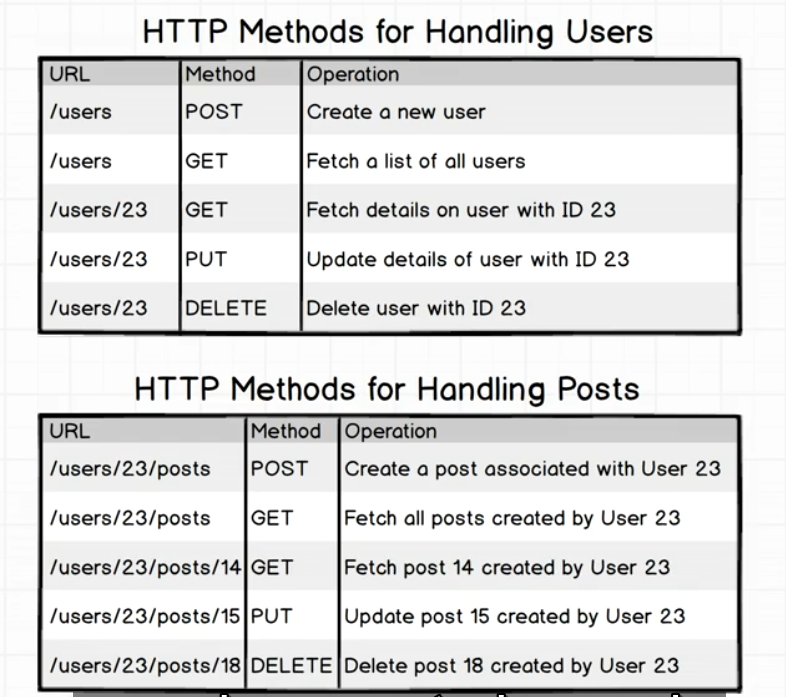
Tạo biểu đồ: basamiq.com

If you get stuck at any point you can always reference the completed code for all examples here:

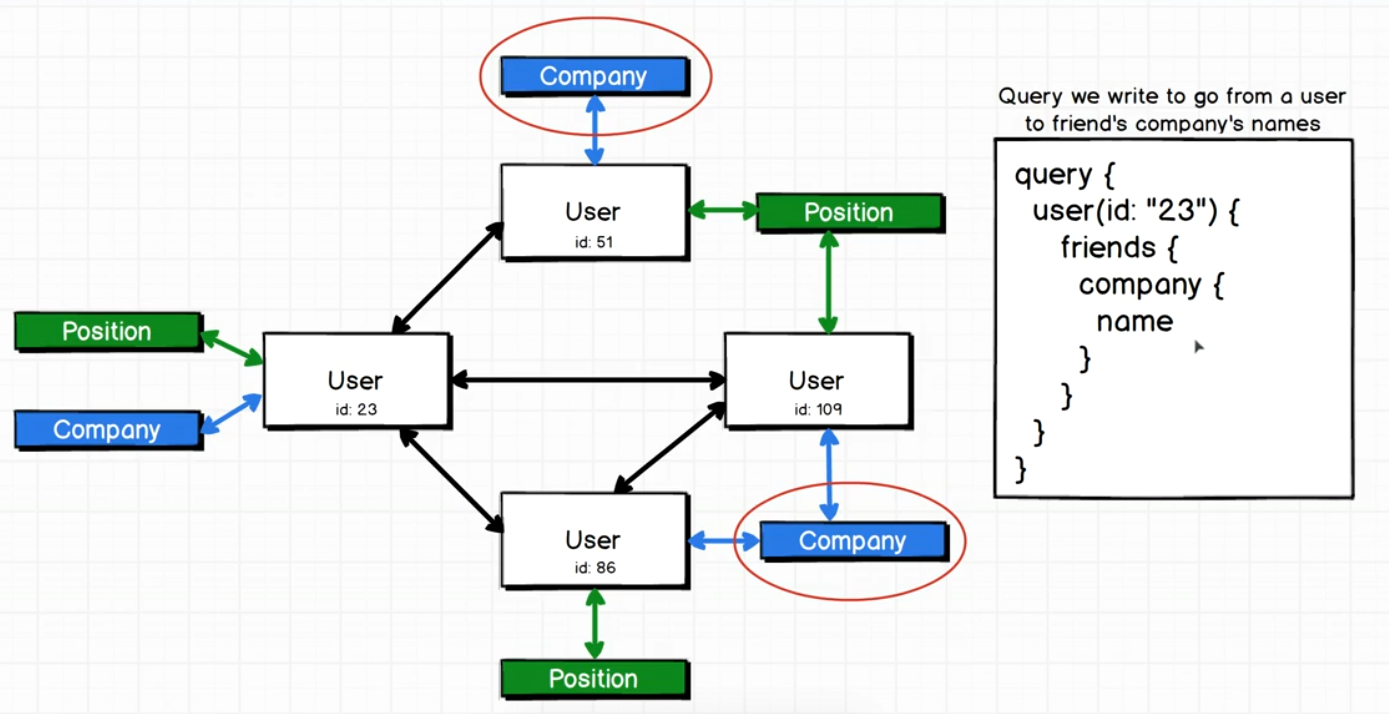
<https://github.com/StephenGrider/GraphQLCasts>

## 2. A REST-ful Routing Primer

Restful routing?



## 3. On To GraphQL



Mkir users

Cd users

Npm init

Npm install –-save express express-grapqhl graphql lodash

Code .

Node server.js

5. Writing a GraphQL Schema

Tạo file server.js

const express = require('express');

const expressGraphQL = require('express-graphql');

const schema = require('./schema/schema');

const app = express();

// nói với express app rằng mọi req đến sẽ tìm trong /graphql để xử lí nó; graphiql là 1 development tool

app.use('/graphql', expressGraphQL({

  schema, // nếu k có gọi url sẽ báo lỗi thiếu khi trả về data

  graphiql: true

}));

app.listen(4000, () => {

  console.log('Listening');

});

Chi tiết lỗi nếu không có schema: <http://localhost:4000/graphql>

{"errors":[{"message":"GraphQL middleware options must contain a schema."}]}

Lưu ý: các tên biến phải viết đúng theo quy tắc

Tạo file schema mô tả obj look like

const graphql = require('graphql');

const axios = require('axios');

const {

  GraphQLObjectType,

  GraphQLString,

  GraphQLInt,

  GraphQLSchema,

  GraphQLList,

  GraphQLNonNull

} = graphql;

const CompanyType = new GraphQLObjectType({

  name: 'Company',

  fields: () => ({

    id: { type: GraphQLString },

    name: { type: GraphQLString },

    description: { type: GraphQLString },

    users: {

      type: new GraphQLList(UserType),

      resolve(parentValue, args) {

        return axios.get(`http://localhost:3000/companies/${parentValue.id}/users`)

          .then(res => res.data)

      }

    }

  })

});

const UserType = new GraphQLObjectType({

  name: 'User',

  fields: () => ({

    id: { type: GraphQLString },

    firstName: { type: GraphQLString },

    age: { type: GraphQLInt },

    company: {

      type: CompanyType,

      resolve(parentValue, args) {

        return axios.get(`http://localhost:3000/companies/${parentValue.companyId}`)

          .then(res => res.data);

      }

    }

  })

});

6. Root Queries

Tạo RootQuery nếu đưa cho nó id nó sẽ tìm và trả về có kiểu UserType, hàm resolve trả về data

const RootQuery = new GraphQLObjectType({

  name: 'RootQueryType',

  fields: {

    user: {

      type: UserType,

      args: { id: { type: GraphQLString } },

      resolve(parentValue, args) {

        return axios.get(`http://localhost:3000/users/${args.id}`)

          .then(resp => resp.data);

      }

    },

    company: {

      type: CompanyType,

      args: { id: { type: GraphQLString } },

      resolve(parentValue, args) {

        return axios.get(`http://localhost:3000/companies/${args.id}`)

          .then(resp => resp.data);

      }

    }

  }

});

module.exports = new GraphQLSchema({

  mutation,

  query: RootQuery

});

7. Resolving with Data

Schema

const \_ = require("lodash");

const {

  GraphQLObjectType,

  GraphQLString,

  GraphQLInt,

  GraphQLSchema,

  GraphQLList,

  GraphQLNonNull

} = graphql;

const users = [

  { id: 23, firstName: "abc", age: 20 },

  { id: 23, firstName: "def", age: 21 }

];

const RootQuery = new GraphQLObjectType({

  name: 'RootQueryType',

  fields: {

    user: {

      type: UserType,

      args: { id: { type: GraphQLString } },

      resolve(parentValue, args) {

        return \_.find(users, {id: args.id});

      }

    }

  }

});

module.exports = new GraphQLSchema({

  query: RootQuery

});

9. A Realistic Data Source

Vào gg gõ: json server

<https://github.com/typicode/json-server>

npm install --save json-server

Tạo file db.json

Start JSON Server

json-server --watch db.json

hay

vào file package.json thêm

"scripts": {

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

    "json:server": "json-server --watch db.json",

    "dev": "nodemon server.js"

  },

Rồi gõ npm run json:server

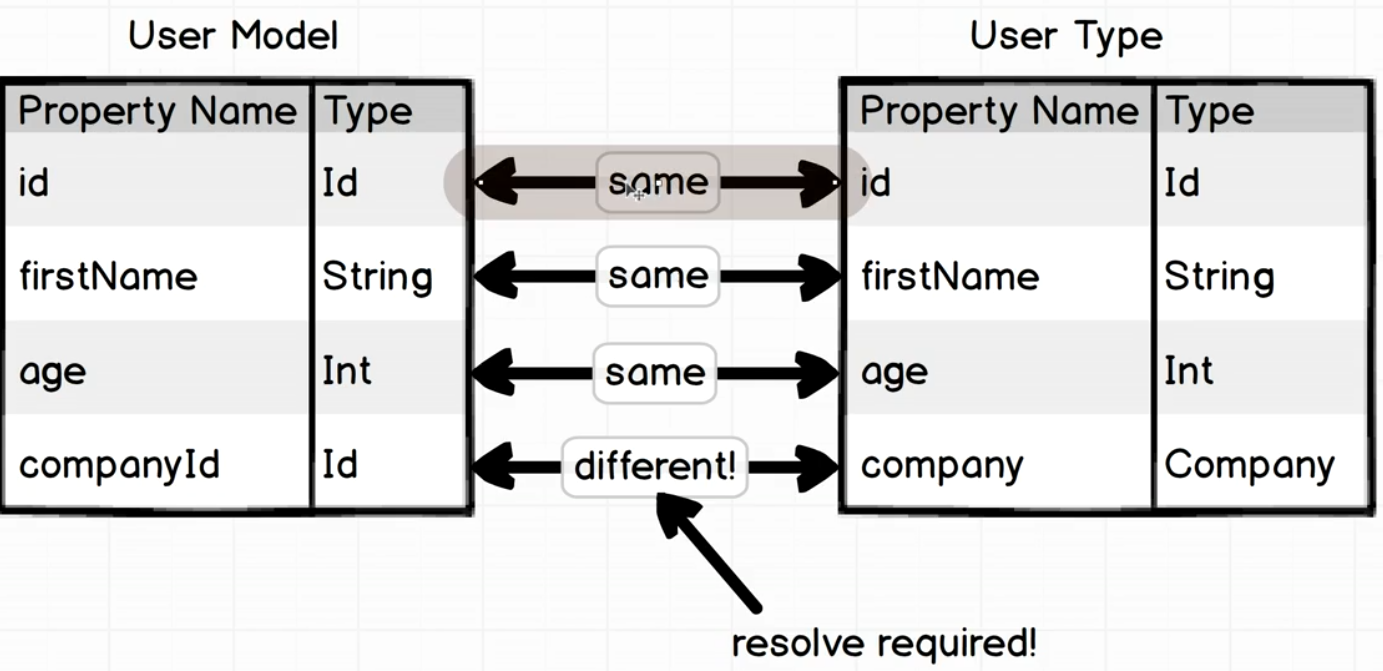
Cài axios, nodemon(auto build)

Npm run dev

## 4. Fetching Data with Queries

2. More on Nested Queries

Id của company trong obj user sẽ tham chiếu đến company obj



In log ra để thấy parentValue

company: {

      type: CompanyType,

      resolve(parentValue, args) {

        return axios.get(`http://localhost:3000/companies/${parentValue.companyId}`)

          .then(res => res.data);

      }

    }

4. Multiple RootQuery Entry Points

Thêm company vào root để có thể access trực tiếp

7. Resolving Circular References

Khi company không nhận ra UserType ta fix bằng cách:

fields: () => ({

... return something here

  })

8. Query Fragments

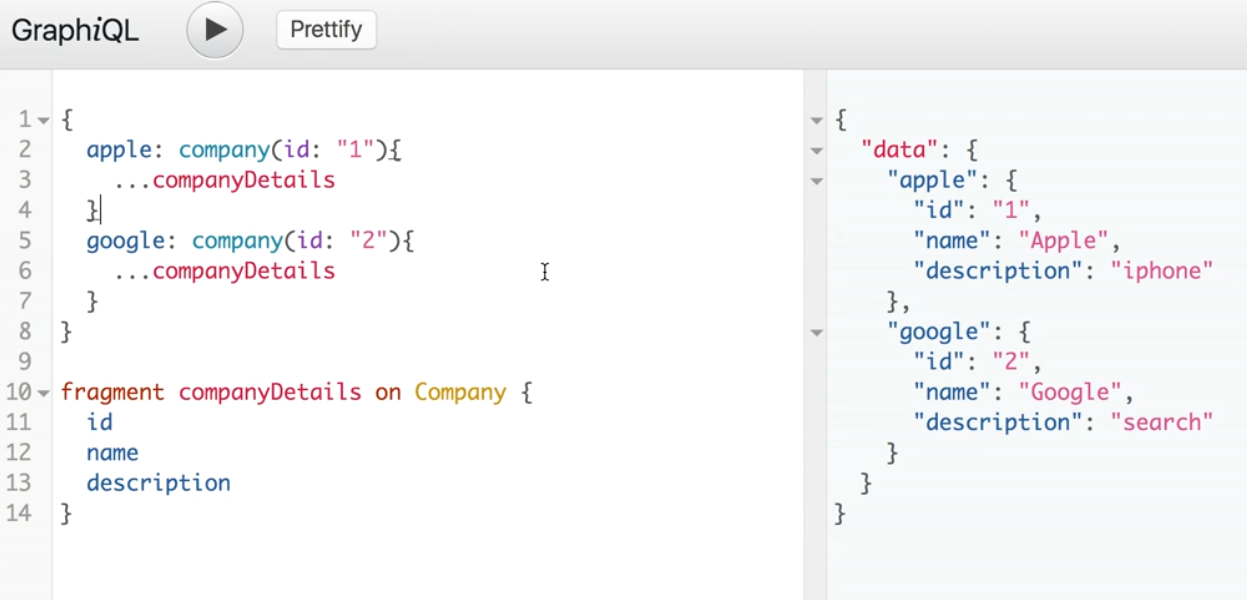
ở giao diện có thể gọi 2 cái get company cùng lúc

{

Apple: company…

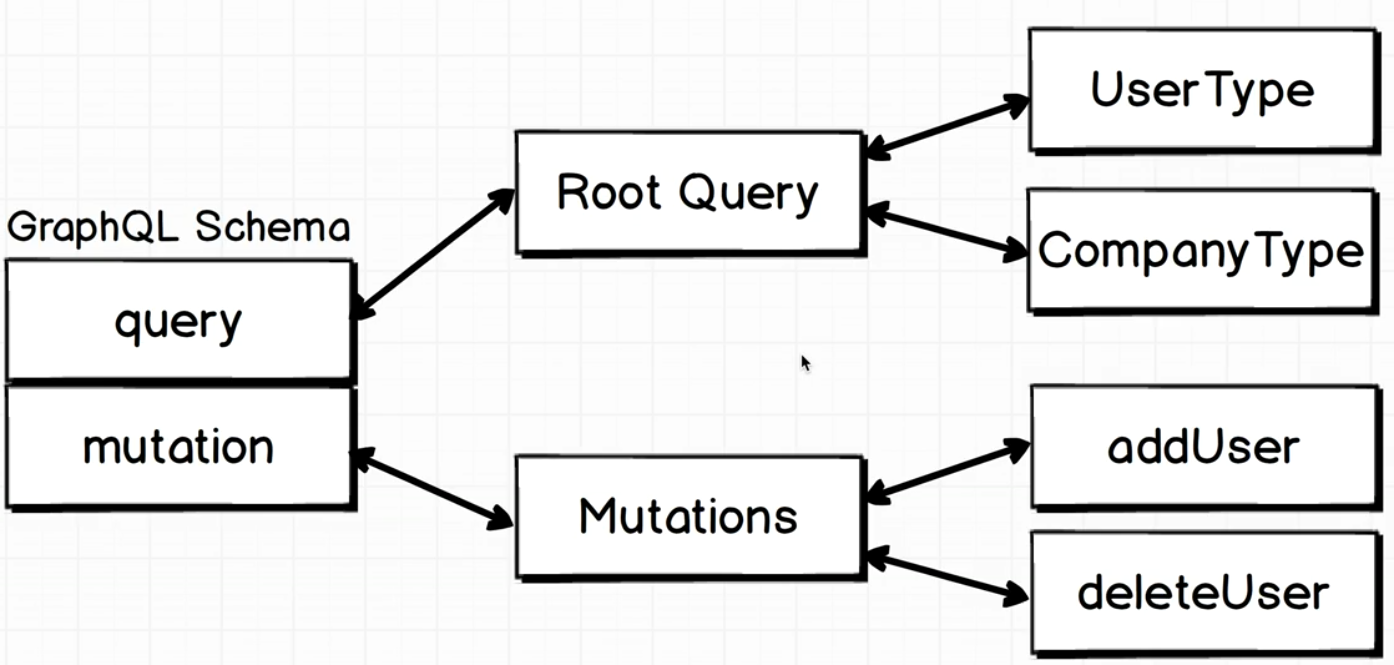
Google: company…

}



Tránh lặp lại, on dựa vào name khai báo

9. Introduction to Mutations



const mutation = new GraphQLObjectType({

  name: 'Mutation',

  fields: {

    addUser: {

      type: UserType,

      args: {

        firstName: { type: new GraphQLNonNull(GraphQLString) },

        age: { type: GraphQLInt },

        companyId: { type: GraphQLString }

      },

      resolve(parentValue, { firstName, age, companyId }) {

        return axios.post('http://localhost:3000/users', { firstName, age, companyId })

          .then(res => res.data);

      }

    },

    deleteUser: {

      type: UserType,

      args: {

        id: { type: new GraphQLNonNull(GraphQLString) }

      },

      resolve(parentValue, { id }) {

        return axios.delete(`http://localhost:3000/users/${id}`)

          .then(res => res.data);

      }

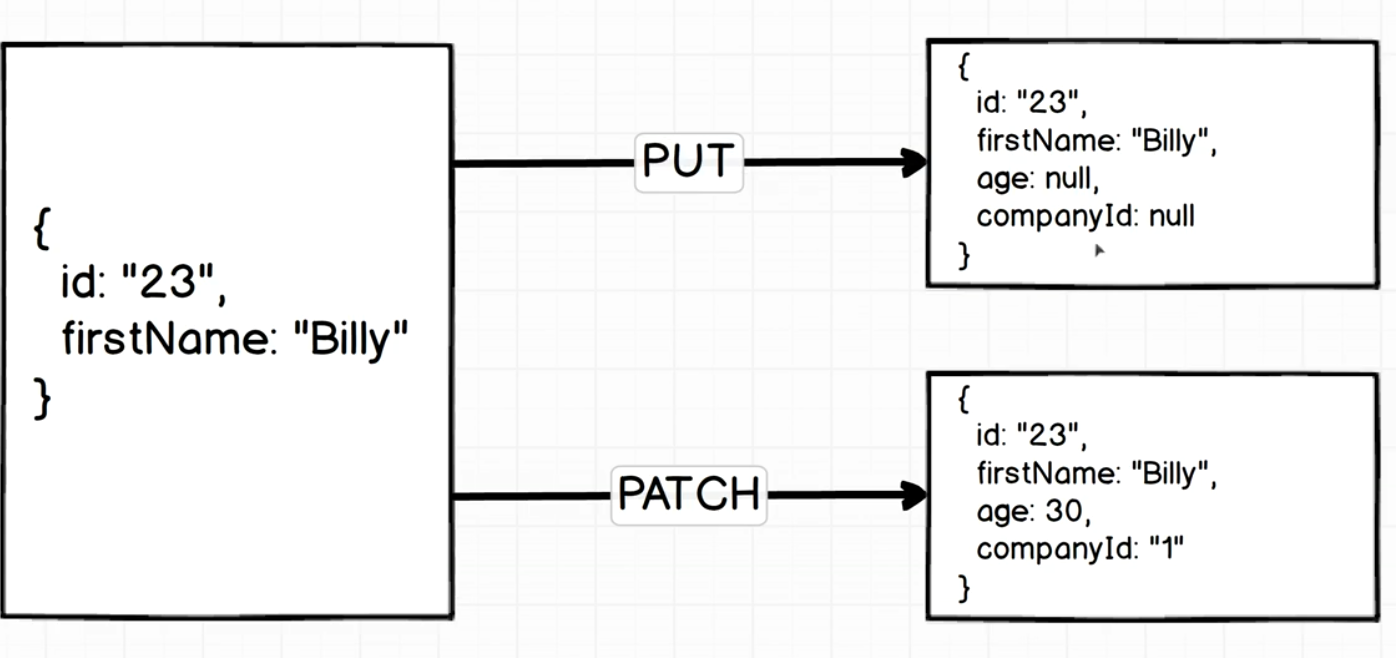
    }

  }

});

Nếu delete thì nó sẽ trả về id null

Put will override all props => SD patch

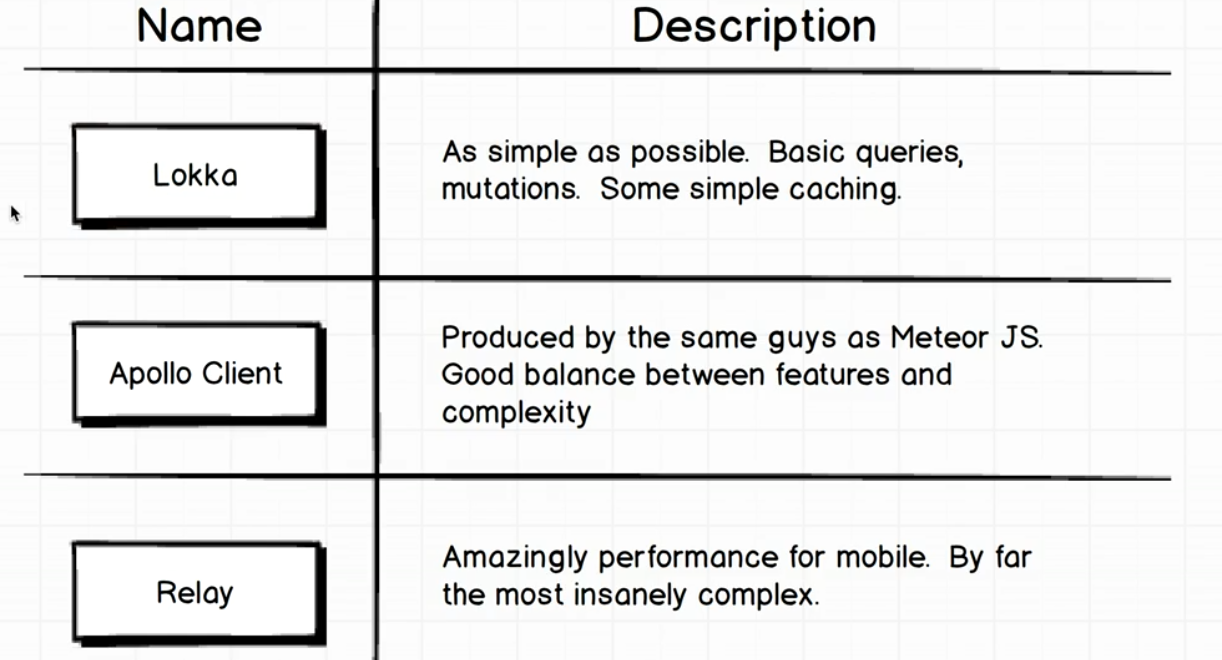


axios.patch(`http://localhost:3000/users/${args.id}`, args)

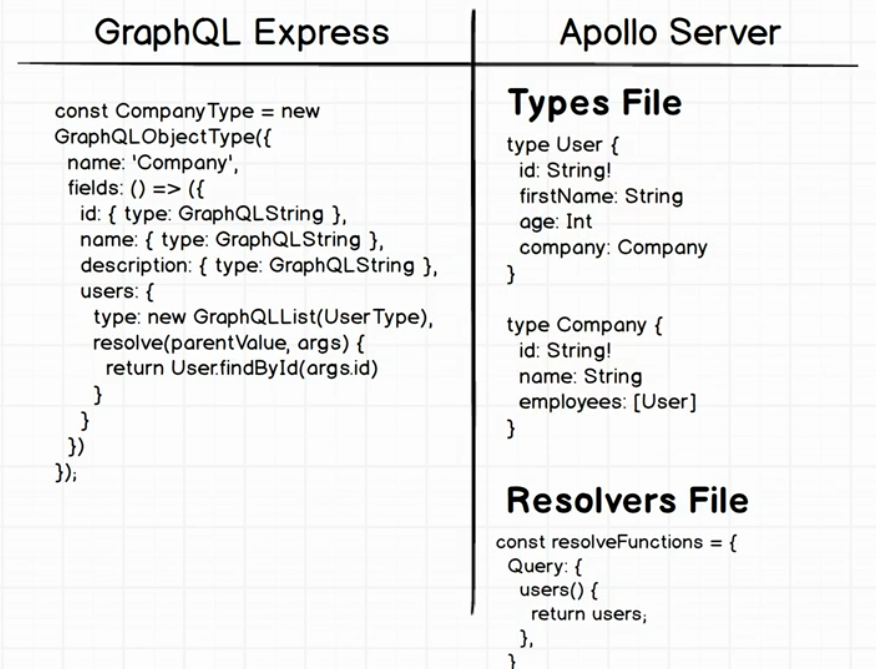
## 5. The GraphQL Ecosystem

1. GraphQL Clients - Apollo vs Relay

Graphql client



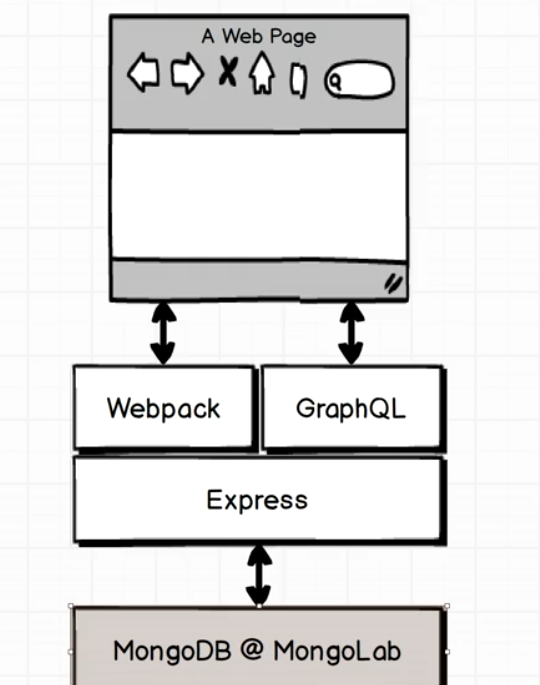
2. Sidenote - Apollo Server vs GraphQL Server



## 6. Clientside GraphQL

<https://github.com/StephenGrider>

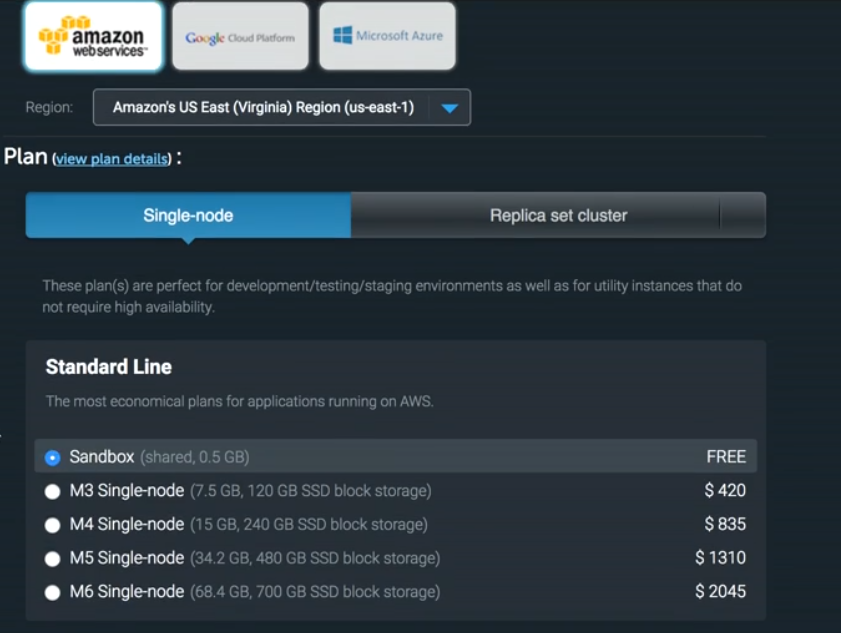
<https://github.com/StephenGrider/lyrical-graphql>



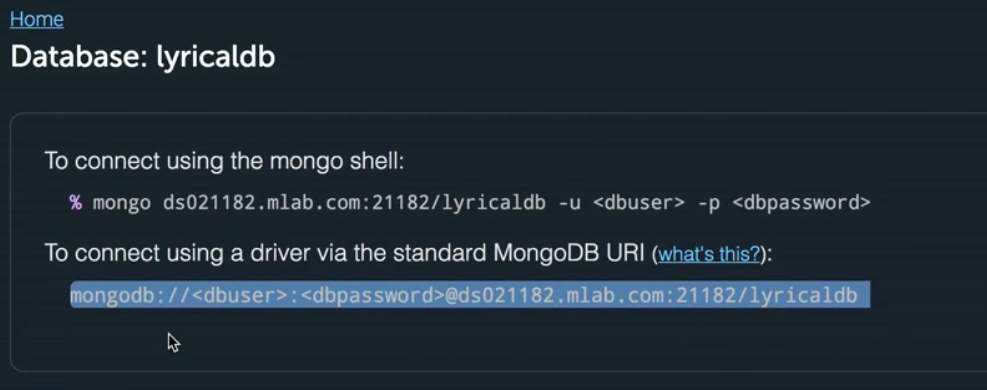
3. MongoLab Setup

<https://mlab.com/>

Sau khi đăng nhập ấn vào nút create new



Chọn amazon/single node/sandbox/ ấn nút tạo



Sau đó vào tab user bên dưới để tạo user và pass cho url

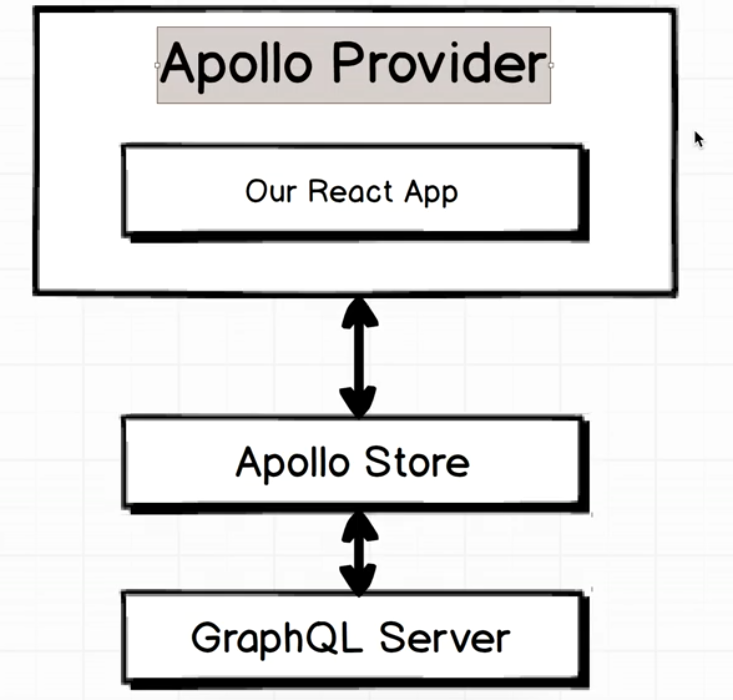
Vào file server.js sửa lại link

*// Replace with your mongoLab URI*

const MONGO\_URI = 'mongodb://stephen:stephen@ds021182.mlab.com:21182/lyricaldb';

Gõ **npm run dev**

5. Apollo Client Setup



Client/Index.js

import ApolloClient from 'apollo-client';

import { ApolloProvider } from 'react-apollo';

const client = new ApolloClient({

  dataIdFromObject: o => o.id

});

const Root = () => {

  return (

    <ApolloProvider client={client}>

      <div>

        Lyrical

      </div>

    </ApolloProvider>

  );

};

ReactDOM.render(

  <Root />,

  document.querySelector('#root')

);

ApolloProvider là react component, ref tới apoloclient

6. React Component Design

Tạo component SongList sửa export default SongList thành:

import gql from 'graphql-tag';

import { graphql } from 'react-apollo';

const query = gql`

  {

    songs {

      id

      title

    }

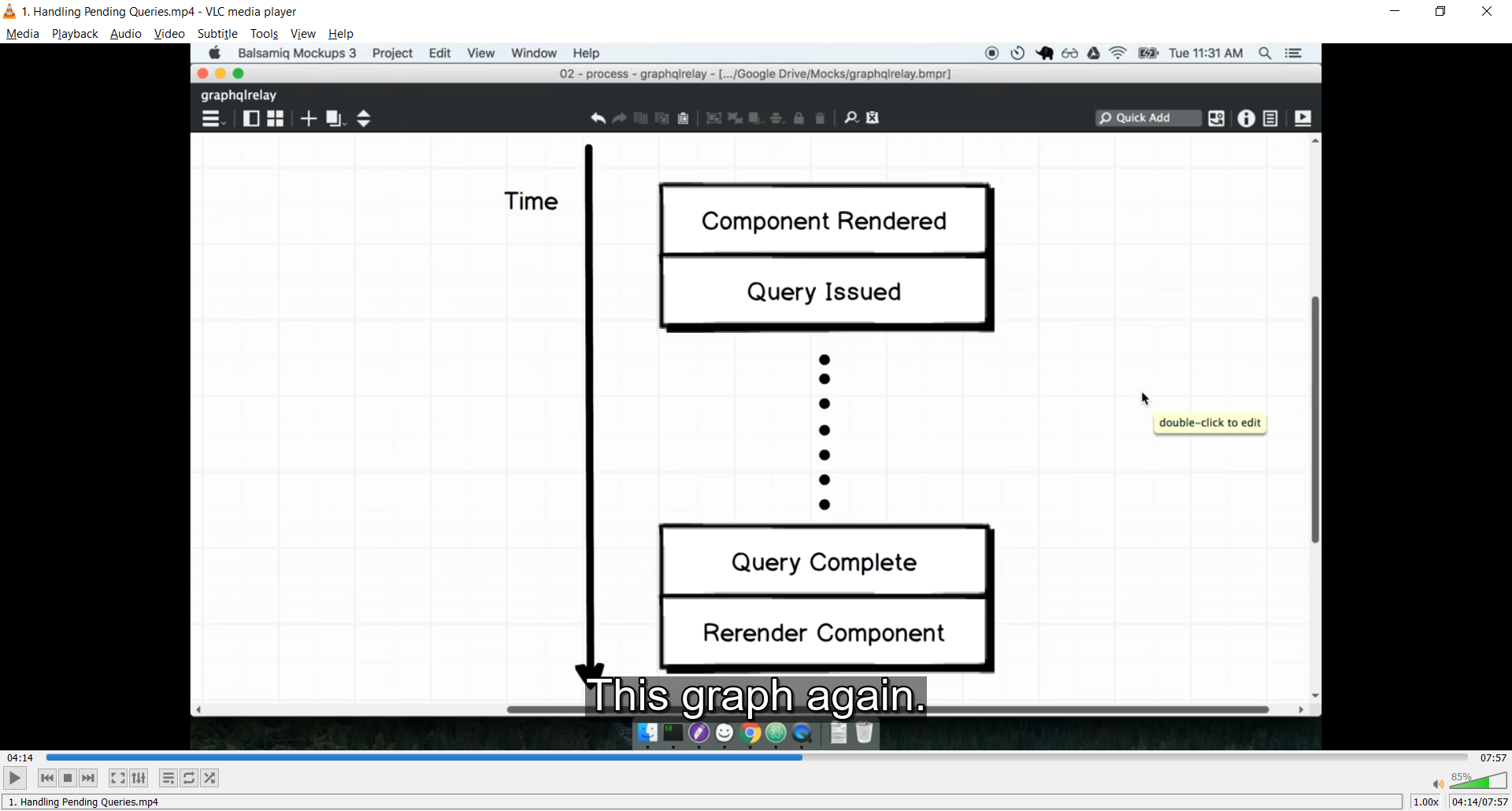
  }

`;

export default graphql(query)(SongList);

## 7. Gotchas with Queries in React

1. Handling Pending Queries



renderSongs() {

    return this.props.data.songs.map(({ id, title }) => {

      return (

        <li key={id} className="collection-item">

          <Link to={`/songs/${id}`}>

            {title}

          </Link>

          <i

            className="material-icons"

            onClick={() => this.onSongDelete(id)}

          >

            delete

          </i>

        </li>

      );

    });

  }

render() {

// nếu chưa load xong dữ liệu thì không render tránh lỗi

    if (this.props.data.loading) { return <div>Loading...</div>; }

    return (

      <div>

        <ul className="collection">

          {this.renderSongs()}

        </ul>

        <Link

          to="/songs/new"

          className="btn-floating btn-large red right"

        >

          <i className="material-icons">add</i>

        </Link>

      </div>

    );

  }

2. Fixing Key Warnings

Sử dụng key lưu id

4. Adding React Router

import { Link } from 'react-router';

index.js

import { Router, Route, hashHistory, IndexRoute } from 'react-router';

const Root = () => {

  return (

    <ApolloProvider client={client}>

      <Router history={hashHistory}>

        <Route path="/" component={App}>

          <IndexRoute component={SongList} />

          <Route path="songs/new" component={SongCreate} />

          <Route path="songs/:id" component={SongDetail} />

        </Route>

      </Router>

    </ApolloProvider>

  );

};

App.js

import React from 'react';

export default ({ children }) => {

  return <div className="container">{children}</div>;

};