**React JS Lab**

**Step-by-Step Instructions**

**Step 1: Update the Service to Include All Operations**

1. Update employeeService.js to include operations for departments and designations:

// src/services/employeeService.js

import { gql } from '@apollo/client';

import client from '../apolloClient';

// Employee Queries and Mutations

const GET\_EMPLOYEES = gql`

query GetEmployees {

employees {

id

name

email

designation {

name

}

department {

name

}

manager {

name

}

subordinates {

name

}

}

}

`;

const GET\_EMPLOYEE = gql`

query GetEmployee($id: Int!) {

employee(id: $id) {

id

name

email

designation {

id

name

}

department {

id

name

}

manager {

id

name

}

}

}

`;

const ADD\_EMPLOYEE = gql`

mutation AddEmployee($name: String!, $email: String!, $designationId: Int!, $departmentId: Int!, $managerId: Int) {

addEmployee(name: $name, email: $email, designationId: $designationId, departmentId: $departmentId, managerId: $managerId) {

id

name

email

designation {

name

}

department {

name

}

manager {

name

}

}

}

`;

const UPDATE\_EMPLOYEE = gql`

mutation UpdateEmployee($id: Int!, $name: String!, $email: String!, $designationId: Int!, $departmentId: Int!, $managerId: Int) {

updateEmployee(id: $id, name: $name, email: $email, designationId: $designationId, departmentId: $departmentId, managerId: $managerId) {

id

name

email

designation {

name

}

department {

name

}

manager {

name

}

}

}

`;

const DELETE\_EMPLOYEE = gql`

mutation DeleteEmployee($id: Int!) {

deleteEmployee(id: $id) {

id

}

}

`;

// Department Queries and Mutations

const GET\_DEPARTMENTS = gql`

query GetDepartments {

departments {

id

name

}

}

`;

const GET\_DEPARTMENT = gql`

query GetDepartment($id: Int!) {

department(id: $id) {

id

name

}

}

`;

const ADD\_DEPARTMENT = gql`

mutation AddDepartment($name: String!) {

addDepartment(name: $name) {

id

name

}

}

`;

const UPDATE\_DEPARTMENT = gql`

mutation UpdateDepartment($id: Int!, $name: String!) {

updateDepartment(id: $id, name: $name) {

id

name

}

}

`;

const DELETE\_DEPARTMENT = gql`

mutation DeleteDepartment($id: Int!) {

deleteDepartment(id: $id) {

id

}

}

`;

// Designation Queries and Mutations

const GET\_DESIGNATIONS = gql`

query GetDesignations {

designations {

id

name

}

}

`;

const GET\_DESIGNATION = gql`

query GetDesignation($id: Int!) {

designation(id: $id) {

id

name

}

}

`;

const ADD\_DESIGNATION = gql`

mutation AddDesignation($name: String!) {

addDesignation(name: $name) {

id

name

}

}

`;

const UPDATE\_DESIGNATION = gql`

mutation UpdateDesignation($id: Int!, $name: String!) {

updateDesignation(id: $id, name: $name) {

id

name

}

}

`;

const DELETE\_DESIGNATION = gql`

mutation DeleteDesignation($id: Int!) {

deleteDesignation(id: $id) {

id

}

}

`;

const employeeService = {

// Employee services

getEmployees: async () => {

const response = await client.query({ query: GET\_EMPLOYEES });

return response.data.employees;

},

getEmployee: async (id) => {

const response = await client.query({ query: GET\_EMPLOYEE, variables: { id } });

return response.data.employee;

},

addEmployee: async (employee) => {

const response = await client.mutate({ mutation: ADD\_EMPLOYEE, variables: employee });

return response.data.addEmployee;

},

updateEmployee: async (employee) => {

const response = await client.mutate({ mutation: UPDATE\_EMPLOYEE, variables: employee });

return response.data.updateEmployee;

},

deleteEmployee: async (id) => {

const response = await client.mutate({ mutation: DELETE\_EMPLOYEE, variables: { id } });

return response.data.deleteEmployee;

},

// Department services

getDepartments: async () => {

const response = await client.query({ query: GET\_DEPARTMENTS });

return response.data.departments;

},

getDepartment: async (id) => {

const response = await client.query({ query: GET\_DEPARTMENT, variables: { id } });

return response.data.department;

},

addDepartment: async (department) => {

const response = await client.mutate({ mutation: ADD\_DEPARTMENT, variables: department });

return response.data.addDepartment;

},

updateDepartment: async (department) => {

const response = await client.mutate({ mutation: UPDATE\_DEPARTMENT, variables: department });

return response.data.updateDepartment;

},

deleteDepartment: async (id) => {

const response = await client.mutate({ mutation: DELETE\_DEPARTMENT, variables: { id } });

return response.data.deleteDepartment;

},

// Designation services

getDesignations: async () => {

const response = await client.query({ query: GET\_DESIGNATIONS });

return response.data.designations;

},

getDesignation: async (id) => {

const response = await client.query({ query: GET\_DESIGNATION, variables: { id } });

return response.data.designation;

},

addDesignation: async (designation) => {

const response = await client.mutate({ mutation: ADD\_DESIGNATION, variables: designation });

return response.data.addDesignation;

},

updateDesignation: async (designation) => {

const response = await client.mutate({ mutation: UPDATE\_DESIGNATION, variables: designation });

return response.data.updateDesignation;

},

deleteDesignation: async (id) => {

const response = await client.mutate({ mutation: DELETE\_DESIGNATION, variables: { id } });

return response.data.deleteDesignation;

},

};

export default employeeService;

**Step 2: Create Components for Departments and Designations**

Create form components to handle create, read, update, and delete operations for departments and designations.

**DepartmentForm.js**

// src/components/DepartmentForm.js

import React, { useEffect, useState } from 'react';

import { useForm } from 'react-hook-form';

import { useHistory, useParams } from 'react-router-dom';

import employeeService from '../services/employeeService';

const DepartmentForm = () => {

const { id } = useParams();

const isEditing = !!id;

const history = useHistory();

const { register, handleSubmit, setValue } = useForm();

const [loading, setLoading] = useState(true);

useEffect(() => {

const fetchData = async () => {

if (isEditing) {

const departmentData = await employeeService.getDepartment(parseInt(id));

setValue('name', departmentData.name);

}

setLoading(false);

};

fetchData();

}, [id, isEditing, setValue]);

const onSubmit = async (formData) => {

if (isEditing) {

await employeeService.updateDepartment({ id: parseInt(id), name: formData.name });

} else {

await employeeService.addDepartment({ name: formData.name });

}

history.push('/departments');

};

if (loading) return <p>Loading...</p>;

return (

<div>

<h2>{isEditing ? 'Edit Department' : 'Add Department'}</h2>

<form onSubmit={handleSubmit(onSubmit)}>

<div className="form-group">

<label>Name:</label>

<input className="form-control" {...register('name', { required: true })} />

</div>

<button type="submit" className="btn btn-primary mt-3">{isEditing ? 'Update Department' : 'Add Department'}</button>

</form>

</div>

);

};

export default DepartmentForm;

**DesignationForm.js**

// src/components/DesignationForm.js

import React, { useEffect, useState } from 'react';

import { useForm } from 'react-hook-form';

import { useHistory, useParams } from 'react-router-dom';

import employeeService from '../services/employeeService';

const DesignationForm = () => {

const { id } = useParams();

const isEditing = !!id;

const history = useHistory();

const { register, handleSubmit, setValue } = useForm();

const [loading, setLoading] = useState(true);

useEffect(() => {

const fetchData = async () => {

if (isEditing) {

const designationData = await employeeService.getDesignation(parseInt(id));

setValue('name', designationData.name);

}

setLoading(false);

};

fetchData();

}, [id, isEditing, setValue]);

const onSubmit = async (formData) => {

if (isEditing) {

await employeeService.updateDesignation({ id: parseInt(id), name: formData.name });

} else {

await employeeService.addDesignation({ name: formData.name });

}

history.push('/designations');

};

if (loading) return <p>Loading...</p>;

return (

<div>

<h2>{isEditing ? 'Edit Designation' : 'Add Designation'}</h2>

<form onSubmit={handleSubmit(onSubmit)}>

<div className="form-group">

<label>Name:</label>

<input className="form-control" {...register('name', { required: true })} />

</div>

<button type="submit" className="btn btn-primary mt-3">{isEditing ? 'Update Designation' : 'Add Designation'}</button>

</form>

</div>

);

};

export default DesignationForm;

**Step 3: Create List Components for Departments and Designations**

**Departments.js**

// src/components/Departments.js

import React, { useEffect, useState } from 'react';

import { Link } from 'react-router-dom';

import employeeService from '../services/employeeService';

const Departments = () => {

const [departments, setDepartments] = useState([]);

const [loading, setLoading] = useState(true);

useEffect(() => {

const fetchDepartments = async () => {

const data = await employeeService.getDepartments();

setDepartments(data);

setLoading(false);

};

fetchDepartments();

}, []);

const handleDelete = async (id) => {

await employeeService.deleteDepartment(id);

setDepartments(departments.filter(department => department.id !== id));

};

if (loading) return <p>Loading...</p>;

return (

<div>

<h2>Departments</h2>

<table className="table table-striped">

<thead>

<tr>

<th>Name</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{departments.map((department) => (

<tr key={department.id}>

<td>{department.name}</td>

<td>

<Link className="btn btn-primary btn-sm mr-2" to={`/edit-department/${department.id}`}>Edit</Link>

<button className="btn btn-danger btn-sm" onClick={() => handleDelete(department.id)}>Delete</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

export default Departments;

**Designations.js**

// src/components/Designations.js

import React, { useEffect, useState } from 'react';

import { Link } from 'react-router-dom';

import employeeService from '../services/employeeService';

const Designations = () => {

const [designations, setDesignations] = useState([]);

const [loading, setLoading] = useState(true);

useEffect(() => {

const fetchDesignations = async () => {

const data = await employeeService.getDesignations();

setDesignations(data);

setLoading(false);

};

fetchDesignations();

}, []);

const handleDelete = async (id) => {

await employeeService.deleteDesignation(id);

setDesignations(designations.filter(designation => designation.id !== id));

};

if (loading) return <p>Loading...</p>;

return (

<div>

<h2>Designations</h2>

<table className="table table-striped">

<thead>

<tr>

<th>Name</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{designations.map((designation) => (

<tr key={designation.id}>

<td>{designation.name}</td>

<td>

<Link className="btn btn-primary btn-sm mr-2" to={`/edit-designation/${designation.id}`}>Edit</Link>

<button className="btn btn-danger btn-sm" onClick={() => handleDelete(designation.id)}>Delete</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

export default Designations;

**Step 4: Update App.js for Routing**

Update App.js to include routes for departments and designations:

// src/App.js

import React from 'react';

import { BrowserRouter as Router, Route, Switch, Link } from 'react-router-dom';

import Employees from './components/Employees';

import EmployeeForm from './components/EmployeeForm';

import Departments from './components/Departments';

import DepartmentForm from './components/DepartmentForm';

import Designations from './components/Designations';

import DesignationForm from './components/DesignationForm';

import './App.css';

function App() {

return (

<Router>

<div className="App">

<header className="App-header">

<h1>GraphQL Client</h1>

<nav className="navbar navbar-expand-lg navbar-dark bg-dark">

<div className="container-fluid">

<ul className="navbar-nav me-auto mb-2 mb-lg-0">

<li className="nav-item">

<Link className="nav-link" to="/">Home</Link>

</li>

<li className="nav-item">

<Link className="nav-link" to="/employees">Employees</Link>

</li>

<li className="nav-item">

<Link className="nav-link" to="/add-employee">Add Employee</Link>

</li>

<li className="nav-item">

<Link className="nav-link" to="/departments">Departments</Link>

</li>

<li className="nav-item">

<Link className="nav-link" to="/add-department">Add Department</Link>

</li>

<li className="nav-item">

<Link className="nav-link" to="/designations">Designations</Link>

</li>

<li className="nav-item">

<Link className="nav-link" to="/add-designation">Add Designation</Link>

</li>

</ul>

</div>

</nav>

</header>

<main className="container mt-3">

<Switch>

<Route path="/" exact component={() => <div>Welcome to GraphQL Client</div>} />

<Route path="/employees" component={Employees} />

<Route path="/add-employee" component={EmployeeForm} />

<Route path="/edit-employee/:id" component={EmployeeForm} />

<Route path="/departments" component={Departments} />

<Route path="/add-department" component={DepartmentForm} />

<Route path="/edit-department/:id" component={DepartmentForm} />

<Route path="/designations" component={Designations} />

<Route path="/add-designation" component={DesignationForm} />

<Route path="/edit-designation/:id" component={DesignationForm} />

</Switch>

</main>

</div>

</Router>

);

}

export default App;

**Step 5: Run the React Application**

1. Start the React application:

npm start

1. Open your browser and navigate to http://localhost:3000/ to see the application in action.

**Final Thoughts**

By adding components and services for CRUD operations on departments and designations, and updating the routing in App.js, you now have a full CRUD application for employees, departments, and designations. This application supports creating, reading, updating, and deleting records for all three entities.

To create an Apollo Server with Sequelize for a MySQL database that matches the endpoints used in the React application, follow these steps:

**Step-by-Step Instructions**

**Step 1: Set Up the Project**

1. Create a new directory for your project and navigate into it:

mkdir graphql-server

cd graphql-server

1. Initialize a new Node.js project:

npm init -y

1. Install the necessary dependencies:

npm install apollo-server graphql sequelize mysql2

**Step 2: Set Up Sequelize Configuration**

1. Create a config directory and add a config.json file for the Sequelize configuration:

mkdir config

touch config/config.json

1. Add the following content to config/config.json:

{

"development": {

"username": "root",

"password": "root1234",

"database": "employeedb",

"host": "127.0.0.1",

"dialect": "mysql"

},

"test": {

"username": "root",

"password": "root1234",

"database": "employeedb",

"host": "127.0.0.1",

"dialect": "mysql"

},

"production": {

"username": "root",

"password": "root1234",

"database": "employeedb",

"host": "127.0.0.1",

"dialect": "mysql"

}

}

**Step 3: Define Sequelize Models**

1. Create a models directory and add the following files for your models:

**index.js**

// models/index.js

const Sequelize = require('sequelize');

const config = require('../config/config.json');

const env = process.env.NODE\_ENV || 'development';

const dbConfig = config[env];

const sequelize = new Sequelize(dbConfig.database, dbConfig.username, dbConfig.password, dbConfig);

const db = {};

db.Sequelize = Sequelize;

db.sequelize = sequelize;

db.Department = require('./department')(sequelize, Sequelize);

db.Designation = require('./designation')(sequelize, Sequelize);

db.Employee = require('./employee')(sequelize, Sequelize);

// Define associations

db.Department.hasMany(db.Employee, { foreignKey: 'department\_id' });

db.Employee.belongsTo(db.Department, { foreignKey: 'department\_id' });

db.Designation.hasMany(db.Employee, { foreignKey: 'designation\_id' });

db.Employee.belongsTo(db.Designation, { foreignKey: 'designation\_id' });

db.Employee.belongsTo(db.Employee, { as: 'Manager', foreignKey: 'manager\_id' });

db.Employee.hasMany(db.Employee, { as: 'Subordinates', foreignKey: 'manager\_id' });

module.exports = db;

**department.js**

// models/department.js

module.exports = (sequelize, DataTypes) => {

const Department = sequelize.define('Department', {

id: {

type: DataTypes.INTEGER,

autoIncrement: true,

primaryKey: true,

},

name: {

type: DataTypes.STRING,

allowNull: false,

},

});

return Department;

};

**designation.js**

// models/designation.js

module.exports = (sequelize, DataTypes) => {

const Designation = sequelize.define('Designation', {

id: {

type: DataTypes.INTEGER,

autoIncrement: true,

primaryKey: true,

},

name: {

type: DataTypes.STRING,

allowNull: false,

},

});

return Designation;

};

**employee.js**

// models/employee.js

module.exports = (sequelize, DataTypes) => {

const Employee = sequelize.define('Employee', {

id: {

type: DataTypes.INTEGER,

autoIncrement: true,

primaryKey: true,

},

name: {

type: DataTypes.STRING,

allowNull: false,

},

email: {

type: DataTypes.STRING,

allowNull: false,

},

designation\_id: {

type: DataTypes.INTEGER,

references: {

model: 'Designations',

key: 'id',

},

},

department\_id: {

type: DataTypes.INTEGER,

references: {

model: 'Departments',

key: 'id',

},

},

manager\_id: {

type: DataTypes.INTEGER,

references: {

model: 'Employees',

key: 'id',

},

},

});

return Employee;

};

**Step 4: Define GraphQL Schema and Resolvers**

1. Create a graphql directory and add the following files:

**typeDefs.js**

// graphql/typeDefs.js

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

const typeDefs = gql`

type Department {

id: Int!

name: String!

}

type Designation {

id: Int!

name: String!

}

type Employee {

id: Int!

name: String!

email: String!

designation: Designation

department: Department

manager: Employee

subordinates: [Employee]

}

type Query {

departments: [Department]

department(id: Int!): Department

designations: [Designation]

designation(id: Int!): Designation

employees: [Employee]

employee(id: Int!): Employee

}

type Mutation {

addDepartment(name: String!): Department

updateDepartment(id: Int!, name: String!): Department

deleteDepartment(id: Int!): Department

addDesignation(name: String!): Designation

updateDesignation(id: Int!, name: String!): Designation

deleteDesignation(id: Int!): Designation

addEmployee(name: String!, email: String!, designationId: Int!, departmentId: Int!, managerId: Int): Employee

updateEmployee(id: Int!, name: String!, email: String!, designationId: Int!, departmentId: Int!, managerId: Int): Employee

deleteEmployee(id: Int!): Employee

}

`;

module.exports = typeDefs;

**resolvers.js**

// graphql/resolvers.js

const { Department, Designation, Employee } = require('../models');

const resolvers = {

Query: {

departments: async () => await Department.findAll(),

department: async (\_, { id }) => await Department.findByPk(id),

designations: async () => await Designation.findAll(),

designation: async (\_, { id }) => await Designation.findByPk(id),

employees: async () => await Employee.findAll({

include: [

{ model: Designation },

{ model: Department },

{ model: Employee, as: 'Manager' },

{ model: Employee, as: 'Subordinates' }

]

}),

employee: async (\_, { id }) => await Employee.findByPk(id, {

include: [

{ model: Designation },

{ model: Department },

{ model: Employee, as: 'Manager' },

{ model: Employee, as: 'Subordinates' }

]

})

},

Mutation: {

addDepartment: async (\_, { name }) => await Department.create({ name }),

updateDepartment: async (\_, { id, name }) => {

const department = await Department.findByPk(id);

department.name = name;

await department.save();

return department;

},

deleteDepartment: async (\_, { id }) => {

const department = await Department.findByPk(id);

await department.destroy();

return department;

},

addDesignation: async (\_, { name }) => await Designation.create({ name }),

updateDesignation: async (\_, { id, name }) => {

const designation = await Designation.findByPk(id);

designation.name = name;

await designation.save();

return designation;

},

deleteDesignation: async (\_, { id }) => {

const designation = await Designation.findByPk(id);

await designation.destroy();

return designation;

},

addEmployee: async (\_, { name, email, designationId, departmentId, managerId }) => await Employee.create({

name,

email,

designation\_id: designationId,

department\_id: departmentId,

manager\_id: managerId

}),

updateEmployee: async (\_, { id, name, email, designationId, departmentId, managerId }) => {

const employee = await Employee.findByPk(id);

employee.name = name;

employee.email = email;

employee.designation\_id = designationId;

employee.department\_id = departmentId;

employee.manager\_id = managerId;

await employee.save();

return employee;

},

deleteEmployee: async (\_, { id }) => {

const employee = await Employee.findByPk(id);

await employee.destroy();

return employee;

}

},

Employee: {

designation: async (employee) => await Designation.findByPk(employee.designation\_id),

department: async (employee) => await Department.findByPk(employee.department\_id),

manager: async (employee) => await Employee.findByPk(employee.manager\_id),

subordinates: async (employee) => await Employee.findAll({ where: { manager\_id: employee.id } })

}

};

module.exports = resolvers;

**Step 5: Set Up Apollo Server**

1. Create a server.js file:

// server.js

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

const typeDefs = require('./graphql/typeDefs');

const resolvers = require('./graphql/resolvers');

const db = require('./models');

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

db.sequelize.sync().then(() => {

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

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

});

});

**Step 6: Initialize the Database**

1. Start your MySQL server and create the employeedb database:

sql

CREATE DATABASE employeedb;

**Step 7: Run the Apollo Server**

1. Start your Apollo Server:

node server.js

1. You should see a message indicating that the server is running:

arduino

🚀 Server ready at <http://localhost:4000/>

**Final Thoughts**

With these steps, you have set up an Apollo Server with Sequelize and a MySQL database that matches the endpoints used in your React application. This setup supports full CRUD operations for employees, departments, and designations.