React/Typescript/GraphQL

Express: A web application framework fro Node.js

MongoDB: A document base open source

Mogoose: Simple, schema-based solution to model application data



Les topics :

* GraphQL : c’est quoi
* Environnement setup

**GraphQL : c’est quoi :**

GraphQL (Graph Query Language ou GQL) est un langage de requête, open source crée par Facebook en 2012 et il est en open source depuis 2015.

L’intérêt du GraphQL permet d’exécuter des requêtes côté serveur en utilisant un seul endpoint contrairement aux API REST qui possèdent plusieurs endpoint. Un point d’entrée pour gérer la récupération et l’ajout des données.

**Prérequis :**

Vous devez connaître Reactjs, Typescript.

Il faut au préalable avoir installé NodeJs sur votre poste de travail.

Si ce n’est pas le cas je vous invite à le faire en suivant le guide sur le site internet.

Il est aussi nécessaire d’avoir git installé sur votre poste.

**Setup :**

Dans ce tutorial, nous allons utiliser express.js pour la partie backend.

Ouvrez votre terminal de commande favorite et naviguez jusqu’au votre espace de travail dédié pour le développement.

Créer un dossier parent « react-express-typescript-graphql » dans votre espace de travail.

$ mkdir react-express-typescript-graphql

$ cd react-express-typescript-graphql

Initialisez le répertoire avec la commande npm (Node Package Manager est inclus dans Nodejs, si ce n’est pas le cas pour x raison alors je vous invite à l’installer manuellement) :

$ npm init -y

Le flag « -y » accepte tout simplement les valeurs par défaut sans poser de question.

Au final nous aurons un fichier « package.json » avec le contenu suivant :

{

  "name": "server",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

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

},

  "keywords": [],

  "author": "",

  "license": "ISC",}

Initialisation git sur le projet :

$ git init

**Préparation des dépendances avec NPM :**

$ npm install express body-parser

$ npm install --save-dev typescript ts-node @types/express @types/node @types/body-parser @types/express-graphql @types/graphql

Créons un fichier de configuration typescript « tsconfig.json »

$ touch tsconfig.json

Autre dépendence :

$ npm install lodash

$ npm install --save-dev @types/lodash

$ npm install cors

$ npm install --save-dev @types/cors

Préparation express.js :

Créer un dossier « server » dans « react-express-typescript-graphql ».

Cette commande crée le « package.json » avec les paramètres par défaut.

$ npm install express

Maintenant créer un fichier « server.js » dans « react-express-typescript-graphql/server »

Ajoutez le code suivant dans ce fichier :

const express = require("express");

const app = express();

const PORT = 5000;

app.get("/", (req, res) => {

  res.send(`Server express is running on port ${PORT}`);

});

app.listen(PORT, () => {

  console.log(`Server express is running on port ${PORT}`);

});

Pour tester :

$ cd server\

$ node server.js



Ce message indique que le server espress fonctionne.

Tester le web service, ouvrez chrome et tapez dans la bare d’adresse « http://localhost:5000 »



Le web service fonctionne.

Installez nodemon :

$ npm install --save-dev nodemon

Nodemon est un outil pour les applications nodejs qui permet de redémarrer l’application lorsqu’il détecte un changement dans le fichier.

Ouvrez le fichier package.json et ajoutez la ligne suivante dans la section scripts après le « test »:

"start": "nodemon server.js"

Pour lancer l’application utilisez la commande suivante :

$ npm run start

GraphQLpour express:

$ npm install express-graphql graphql --save

Nous allons utiliser « graphql » pour monter le serveur GraphQL

Nous pouvons utiliser la bibliothèque « express-graphql » pour monter un serveur API GraphQL.

Ouvrez le fichier « server.js » et importez la librairie après l’import express

var graphqlHTTP = require('express-graphql');

Ensuite, nous allons activer l’interface graphql. Pour cela ajoutez le code suivant avant la ligne « app.listen(…) » :

app.use('/graphql', graphqlHTTP({

   graphiql: true

}));

Relancer la commande :

$ npm run start

Ouvrez chrome puis saissisez l’url suivant <http://localhost:5000/graphql> et vous devriez avoir ceci à l’écran :



Il nous faut déclare un schéma :

Arrêtez le serveur en appuyant « ctrl + c » puis « y » pour confirmer.

Ajoutez un dossier « schema » à la racine du dossier « server » puis un fichier « schema.js » dans le répertoire « schema »

$ mkdir schema

$ touch schema/schema.js

Ajoute le code suivant :

const graphql = require("graphql");

const {

  GraphQLSchema,

  GraphQLObjectType,

  GraphQLString,

  GraphQLInt,

  GraphQLList

} = graphql;

const contactDatas = [

  {

    firstName: "Coumarane",

    lastName: "COUPPANE"

  },

  {

    firstName: "Helios",

    lastName: "Kumar"

  }

];

const contactType = new GraphQLObjectType({

  name: "Contact",

  fields: {

    firstName: {

      type: GraphQLString

    },

    lastName: {

      type: GraphQLString

    }

  }

});

const queryType = new GraphQLObjectType({

  name: "Query",

  fields: {

    // contact: {

    //     type: contactType,

    //     args: {

    //         id: { type: GraphQLInt}

    //     }

    // },

    contacts: {

      type: new GraphQLList(contactType),

      resolve: () => {

        return contactDatas;

      }

    }

  }

});

const rootSchema = new GraphQLSchema({

  query: queryType

});

module.exports = rootSchema;

dans le fichier « server.js » :

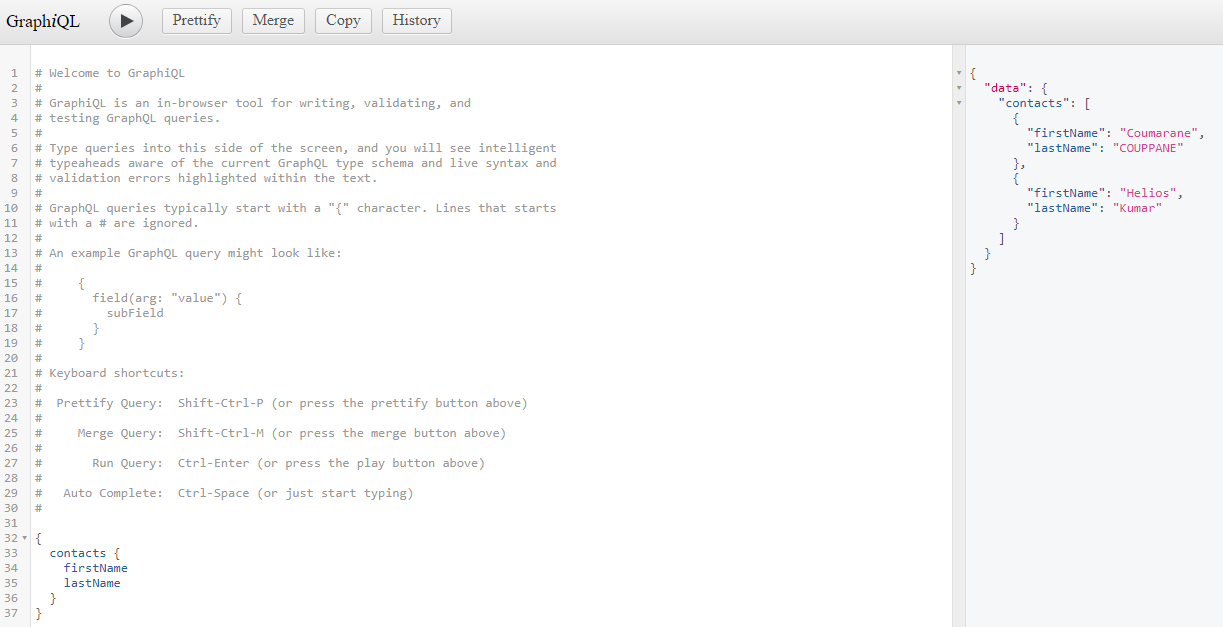
const rootSchema = require('./schema/schema')

app.use('/graphql', graphqlHTTP({

   schema: rootSchema,

   graphiql: true

}));





<https://github.com/mateusconstanzo/express-graphql-typescript>