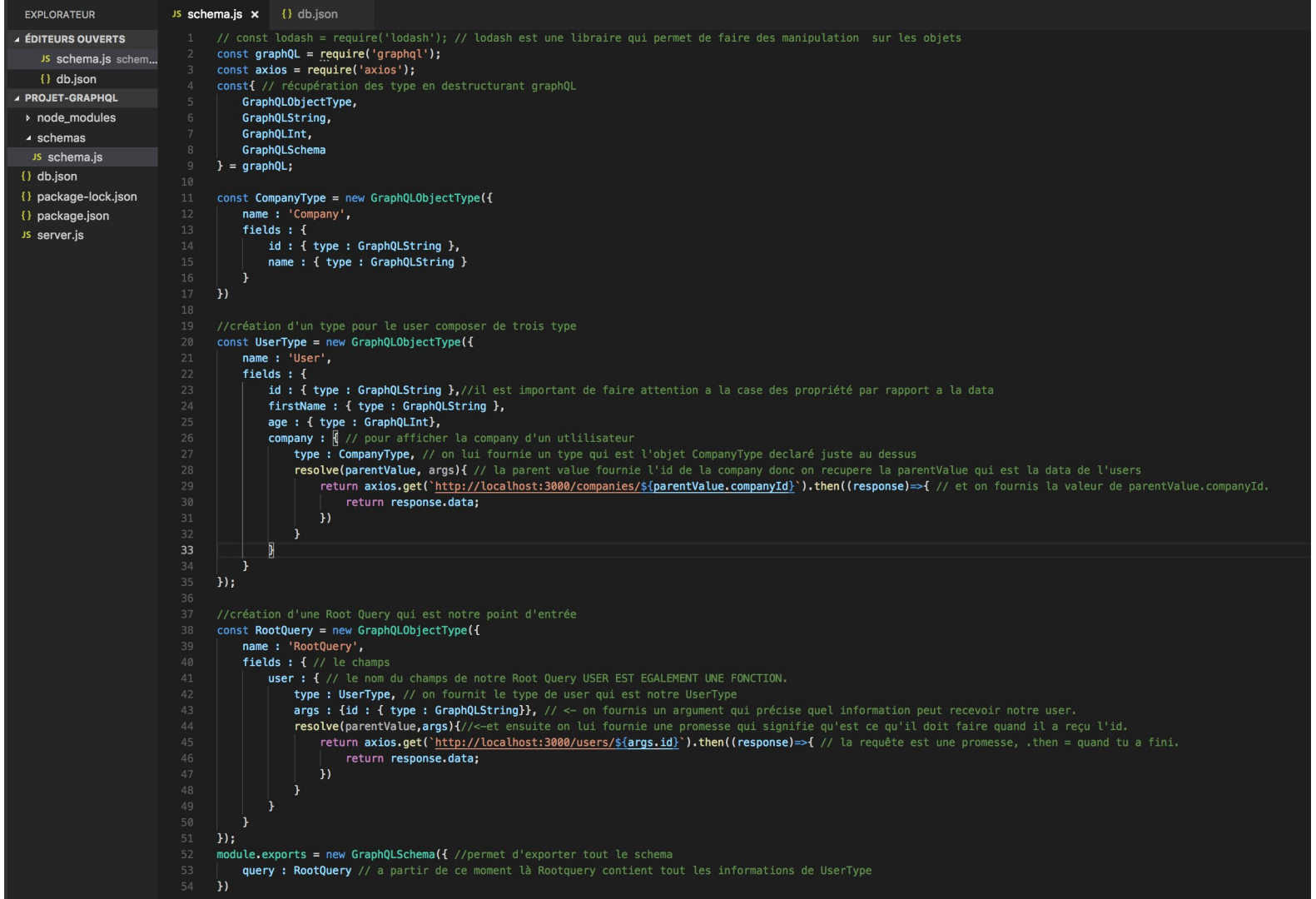
EXPLORATEUR
ÉDITEURS OUVERTS
schema.js schem...
db.json
PROJET-GRAPHQL
node_modules
schemas
schema.js
db.json
package-lock.json
package.json
server.js
schema.js
db.json
// const lodash = require('lodash'); // lodash est une libraire qui permet de faire des manipulation sur les objets
const graphQL = require('graphql');
const axios = require('axios');
const{ // récupération des type en destructurant graphQL
GraphQLObjectType,
GraphQLString,
GraphQLInt,
GraphQLSchema
} = graphQL;
const CompanyType = new GraphQLObjectType({
name : 'Company',
fields : {
id : { type : GraphQLString },
name : { type : GraphQLString }
}
})
//création d'un type pour le user composer de trois type
const UserType = new GraphQLObjectType({
name : 'User',
fields : {
id : { type : GraphQLString },//il est important de faire attention a la case des propriété par rapport a la data
firstName : { type : GraphQLString },
age : { type : GraphQLInt},
company : { // pour afficher la company d'un utlilisateur
type : CompanyType, // on lui fournie un type qui est l'objet CompanyType declaré juste au dessus
resolve(parentValue, args){ // la parent value fournie l'id de la company donc on recupere la parentValue qui est la data de l'users
return axios.get(`http://localhost:3000/companies/${parentValue.companyId}`).then((response)=>{ // et on fournis la valeur de parentValue.companyId.
return response.data;
})
}
}
}
});
//création d'une Root Query qui est notre point d'entrée
const RootQuery = new GraphQLObjectType({
name : 'RootQuery',
fields : { // le champs
user : { // le nom du champs de notre Root Query USER EST EGALEMENT UNE FONCTION.
type : UserType, // on fournit le type de user qui est notre UserType
args : {id : { type : GraphQLString}}, // <- on fournis un argument qui précise quel information peut recevoir notre user.
resolve(parentValue,args){//<-et ensuite on lui fournie une promesse qui signifie qu'est ce qu'il doit faire quand il a reçu l'id.
return axios.get(`http://localhost:3000/users/${args.id}`).then((response)=>{ // la requête est une promesse, .then = quand tu a fini.
return response.data;
})
}
}
}
});
module.exports = new GraphQLSchema({ //permet d'exporter tout le schema
query : RootQuery // a partir de ce moment là Rootquery contient tout les informations de UserType
})


GraphiQL
Prettify
History
query{
user(id:"1") {
firstName,
age,
company{
name
}
}
}
{
"data": {
"user": {
"firstName": "Robin",
"age": 24,
"company": {
"name": "Microsoft"
}
}
}
}
Documentation Explorer
Search Schema...
A GraphQL schema provides a root type for each kind of operation.
ROOT TYPES
query: RootQuery