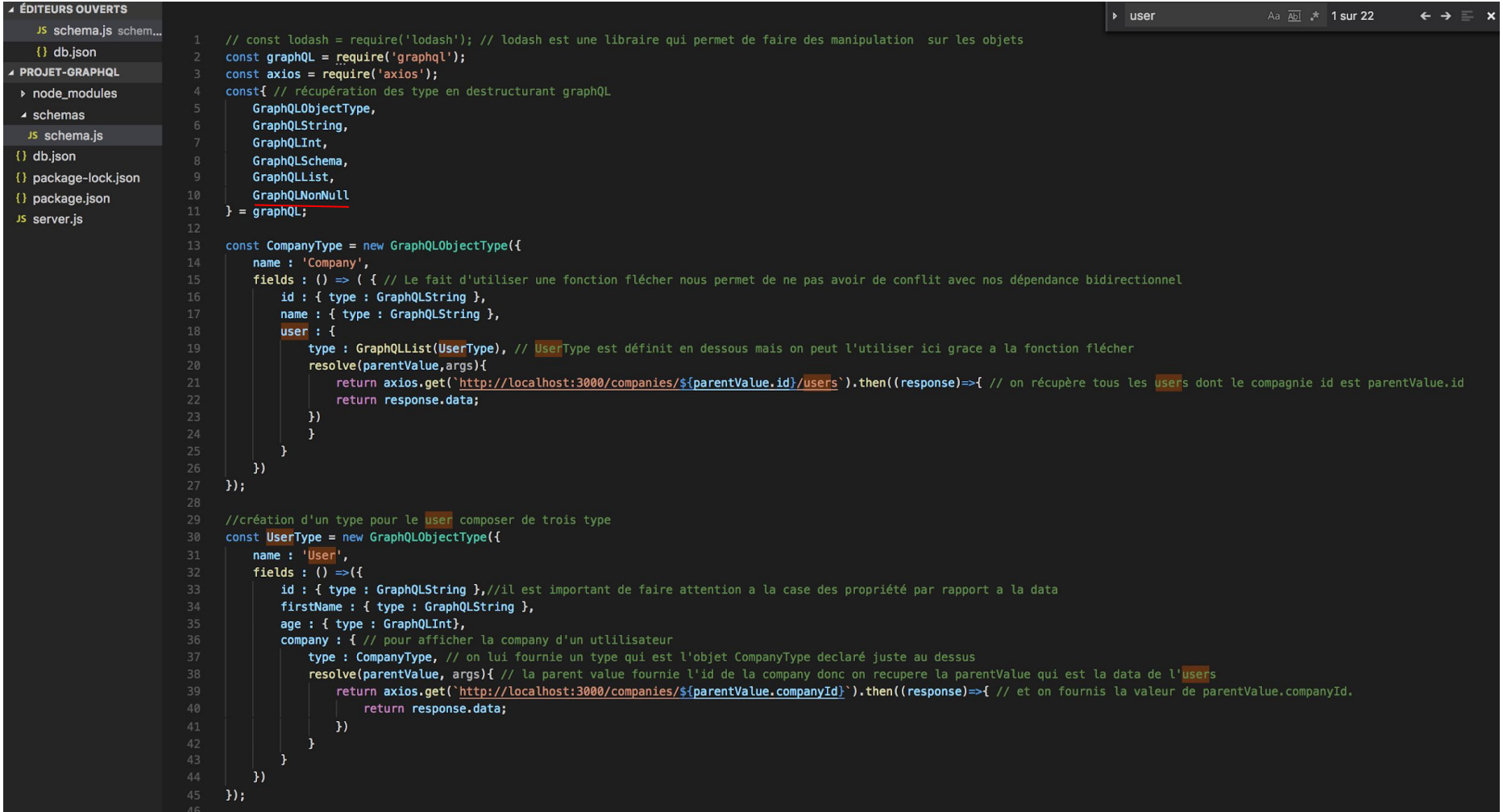
schema.js 1er partie




```
// 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,
    GraphQLList,
    GraphQLNonNull
} = graphQL;

const CompanyType = new GraphQLObjectType({
    name : 'Company',
    fields : () => ( { // Le fait d'utiliser une fonction flécher nous permet de ne pas avoir de conflit avec nos dépendance bidirectionnel
        id : { type : GraphQLString },
        name : { type : GraphQLString },
        user : {
            type : GraphQLList(UserType), // UserType est définit en dessous mais on peut l'utiliser ici grace a la fonction flécher
            resolve(parentValue,args){
                return axios.get(`http://localhost:3000/companies/${parentValue.id}/users`).then((response)=>{ // on récupère tous les users dont le compagnie id est parentValue.id
                return response.data;
            })
            }
        }
    })
});

//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;
                })
            }
        }
    })
});
```

schema.js 2eme partie

```
const MutationType = new GraphQLObjectType({ // ajout d'un utilisateur
    name : 'Mutation',
    fields : {
        addUser :{ //nom de la fonction
            type : UserType, //elle retourne un UserType
            args : {
                firstName : { type: new GraphQLNonNull(GraphQLString) },// GraphQLNonNull permet de verifier que le champs est bien renseigner
                age : { type : new GraphQLNonNull(GraphQLInt)},
                companyId : { type: GraphQLString }
            },
            resolve(parentValue, args){
                return axios.post(`http://localhost:3000/users/`,{firstName: args.firstName, age: args.age, companyId: args.companyId}).then((response)=>{
                    return response.data; // la fonction retourne donc un UserType
                })
            }
        }
    }
});

//création d'une Root Query qui est notre point d'entrée
const RootQueryType = 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/${args.id}`).then((response)=>{ // la requête est une promesse, .then = quand tu a fini.
                    return response.data;
                })
            }
        },
        company : {
            type : CompanyType,
            args : {id : { type : GraphQLString }},
            resolve(parentValue,args){
                return axios.get(`http://localhost:3000/companies/${args.id}`).then((response)=>{
                    return response.data;
                })
            }
        }
    }
});
module.exports = new GraphQLSchema({ //permet d'exporter tout le schema
    query : RootQueryType, // a partir de ce moment là Rootquery contient tout les informations de UserType
    mutation : MutationType
})
```

Le retour de la requête post attendu

```
query{
  user(id:"5NwE0GC"){
    firstName
  }
}
```

Le User a bien été ajouter.