Initialisation du projet

- Création d'un nouveau repo git "graphql-articles"
- *git clone*
- `npm init -y`
- `package.json` => `"start": "nodemon --exec babel-node server/index.js"`
- `mkdir server`

Installation des dépendences

- `npm i body-parser cors express express-graphql graphql graphql-tools graphql-yoga jsonwebtoken merge-graphql-schemas mongoose mongoose-unique-validator slug`

- `npm i -D @babel/cli @babel/core @babel/preset-env concurrently nodemon`

Déclaration des *models*

Les *models* sont les déclaration des schemas *mongo*

- Copier le dossier *models* dans le dossier *server*

- Créer le *model* "Comment" :

    - body: String,
    - author: ref: 'User'
    - article: ref: 'Article'

Déclaration de *graphql*

Le dossier graphql contient la configuration GraphQL :

- *resolvers* : contient les *queries* et *mutations* (https://graphql.org/learn/queries/)

- *types* : contient les *schemas* (https://graphql.org/learn/schema/)

- Copier le dossier *graphql* dans le dossier *server*

- Créer les *resolvers* et le *type* pour le *model* “Comment”

Implémentation du serveur

```
import { GraphQLServer } from "graphql-yoga";
import mongoose from "mongoose";
import User from "./models/User";
import Article from "./models/Article";
import Comment from "./models/Comment";
import schema from "../graphql";

const options = {
  port: process.env.PORT || "4000",
  endpoint: "/graphql",
  playground: "/playground"
};

const models = {
    Article,
    Comment,
    User
}

const context = {
  models,
};

// Connect to MongoDB with Mongoose.
mongoose
  .connect(
    'mongodb://localhost/conduit',
    {
      useCreateIndex: true,
      useNewUrlParser: true
    }
  )
```

```
  .then(() => console.log("MongoDB connected"))
  .catch(err => console.log(err));

const server = new GraphQLServer({
  schema,
  context
});

server.start(options, ({ port }) => {
  console.log(`🚀  Server is running on http://localhost:${port}`);
});
```

Test du serveur :

- `npm start`
- http://localhost:4000/playground

Créer un utilisateur

```
mutation {
  createUser(user: { username: "testuser", email: "test@tesl.com", password: "TestPass
word"}) {
    username
    email
  }
}
```

Récupérer la liste des utilisateurs

```
query {
  users {
    _id
    username
    email
  }
}
```

Créer un article

Créer un commentaire

Récupérer la liste des articles et leur commentaire