Sawsen fattahi <sawsen.fattahi@gmail.com>

react, graphql, mongoose

I. Creat a react APP

1. Download and install node js:

https://nodejs.org/en/

2. Install yarn

https://yarnpkg.com/lang/en/docs/install/

3. Open cmd

4. Installation web-client : https://reactjs.org/tutorial/tutorial.html

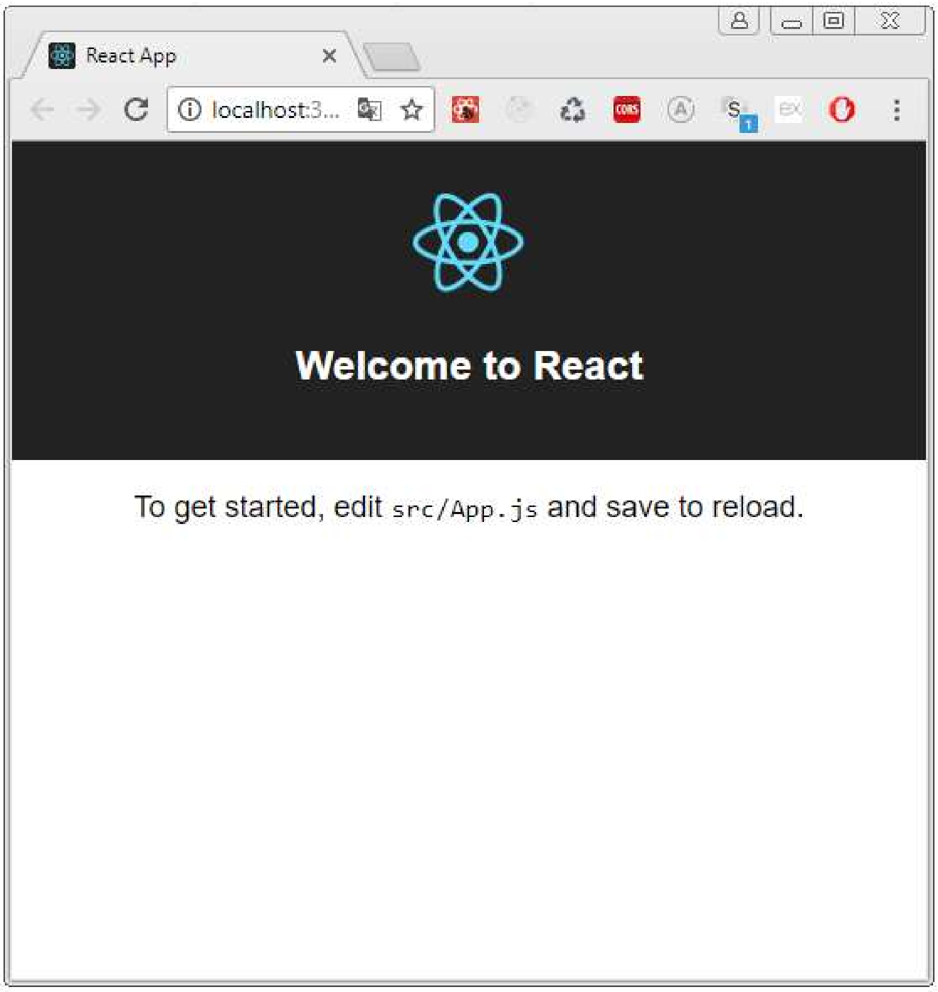
1- npm install -g create-react-app

2- create-react-app front-office

3- cd front-office

4- yarn install

5- yarn start



Welcome to React

To get started, edit `src/App.js` and save to reload.

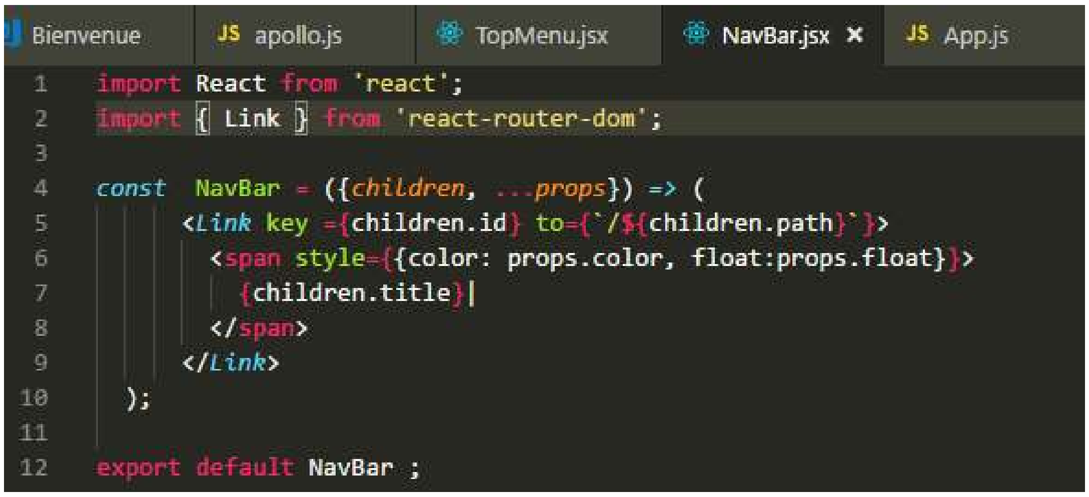
II. Containers, Components:

create a file tree as follows

```
src
 |---components
          |-- NavBar:
                   |--NavBar.jsx
                   |-- index.js
 |---containers
          |-- Menu:
                   |--Menu.jsx
                   |-- index.js
```

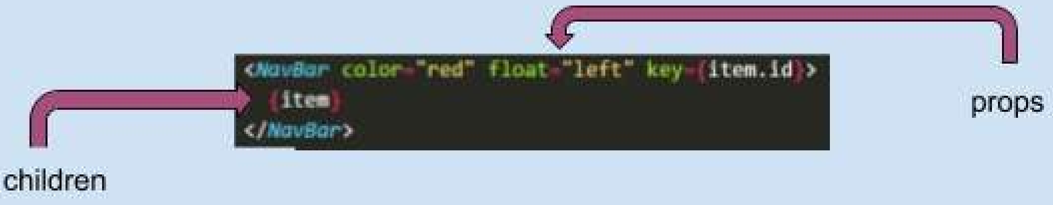
we will put our logic in containers folder and the UI in components, breaking makes our code more usable, cleaner and very easy to test and maintainable.

1. components/NavBar: we will use menu in header menu, footer and perhaps in other places, so we should create a reusable component..
    a. firstable add react-router-dom : yarn add react-router-dom
    to build our links;
    b. add this code to your components/NavBar/NavBar.jsx



```jsx
import React from 'react';
import { Link } from 'react-router-dom';

const  NavBar = ({children, ...props}) => (
        <Link key ={children.id} to={`/${children.path}`}>
          <span style={{color: props.color, float:props.float}}>
            {children.title}
          </span>
        </Link>
    );

export default NavBar ;
```

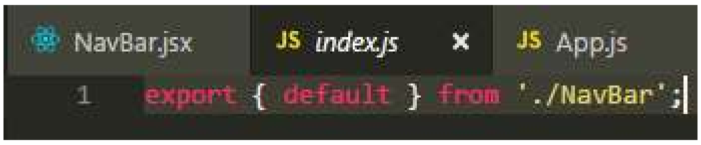
in function args we've use the … operator to destruct coming args, it destruct children from the object passed to the function and all the rest in an object called props we can use another name instead props ;)



```jsx
<NavBar color="red" float="left" key={item.id}>
  {item}
</NavBar>
```

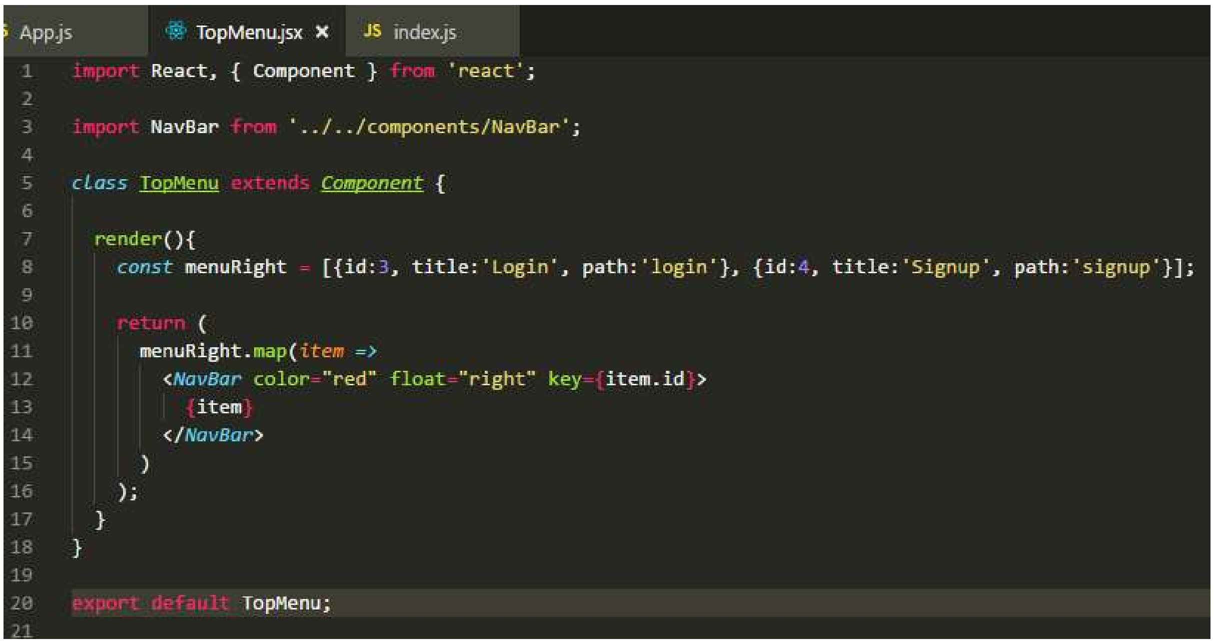
c. and in components/NavBar/index.js:

we use this to export many things from the same folder for example we can add LeftMenu.jsx and replace the export by :
    export { TopMenu } from './NavBar';
    export { LeftMenu } from './NavBar';



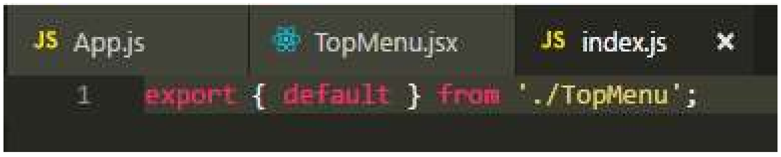
```
export { default } from './NavBar';
```

d. container: containers/TopMenu/TopMenu.jsx: here we put all our logic



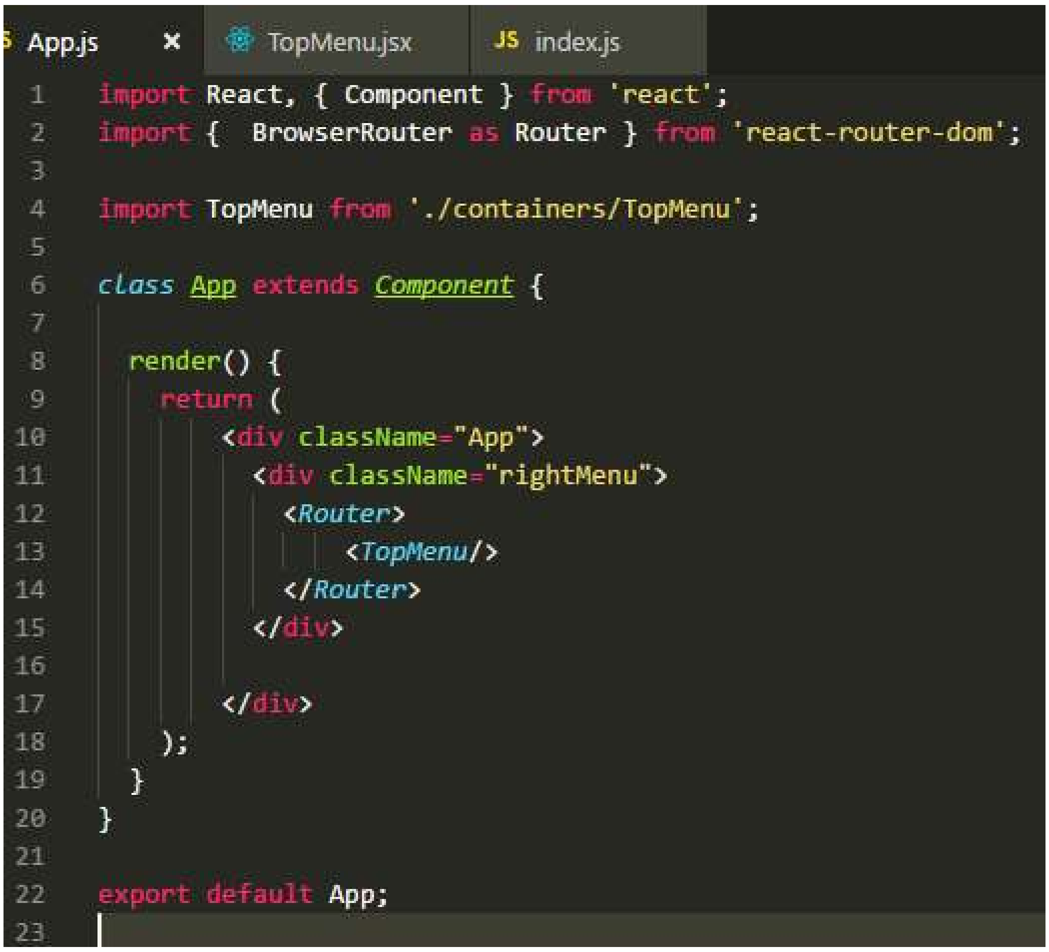
```
import React, { Component } from 'react';

import NavBar from '../../components/NavBar';

class TopMenu extends Component {

  render(){
    const menuRight = [{id:3, title:'Login', path:'login'}, {id:4, title:'Signup', path:'signup'}];

    return (
      menuRight.map(item =>
        <NavBar color="red" float="right" key={item.id}>
          {item}
        </NavBar>
      )
    );
  }
}

export default TopMenu;
```

e. containers/TopMenu/index.js



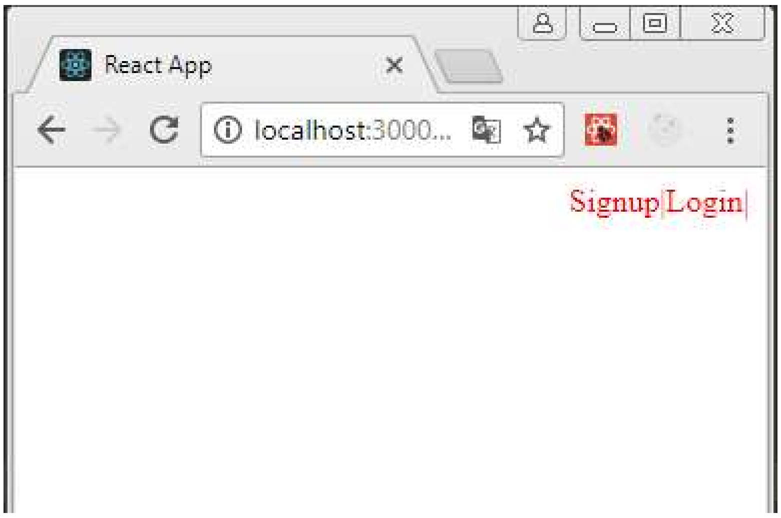
```
export { default } from './TopMenu';
```

f. App.js


App.js TopMenu.jsx index.js

```
import React, { Component } from 'react';
import {  BrowserRouter as Router } from 'react-router-dom';

import TopMenu from './containers/TopMenu';

class App extends Component {

  render() {
    return (
        <div className="App">
          <div className="rightMenu">
            <Router>
                <TopMenu/>
            </Router>
          </div>

        </div>
    );
  }
}

export default App;

```

finally you should have this result

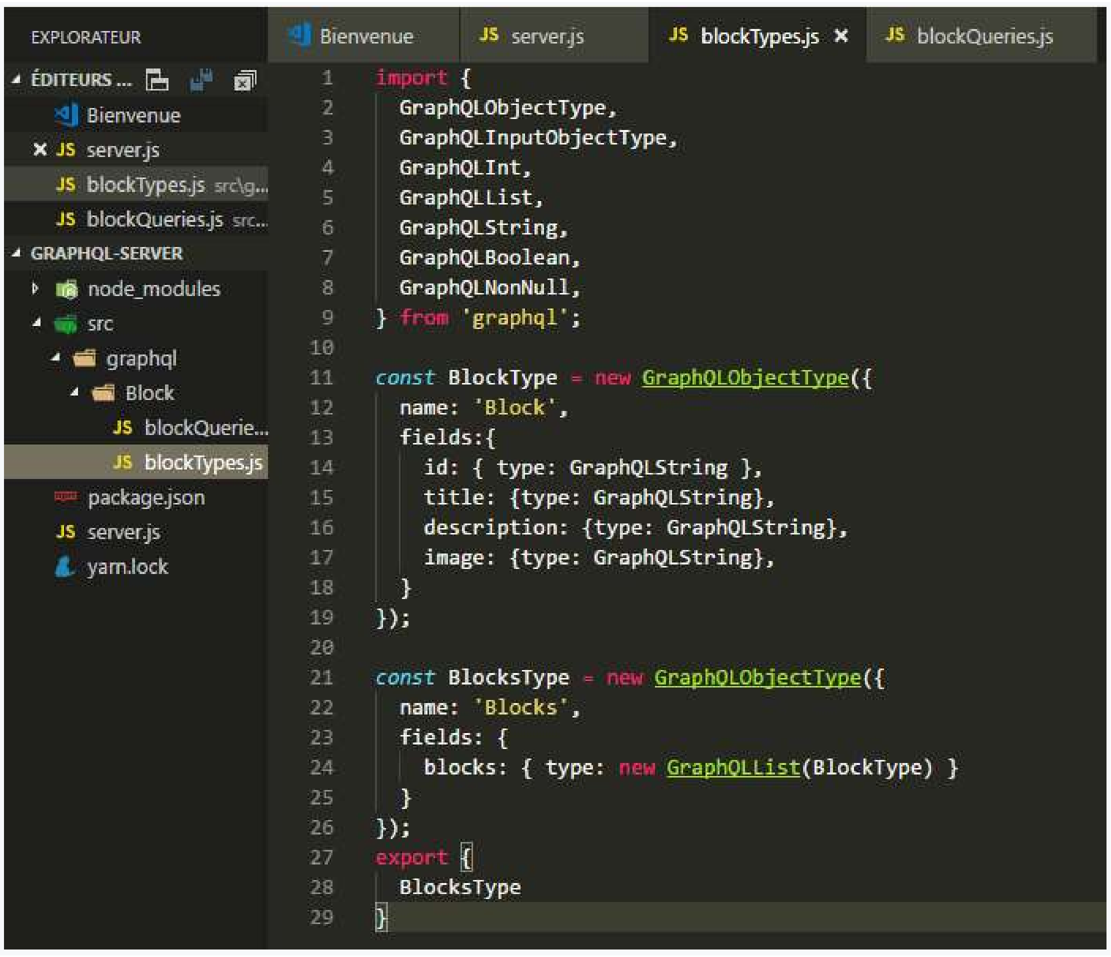


III. Graphql server

1. create new folder "server",

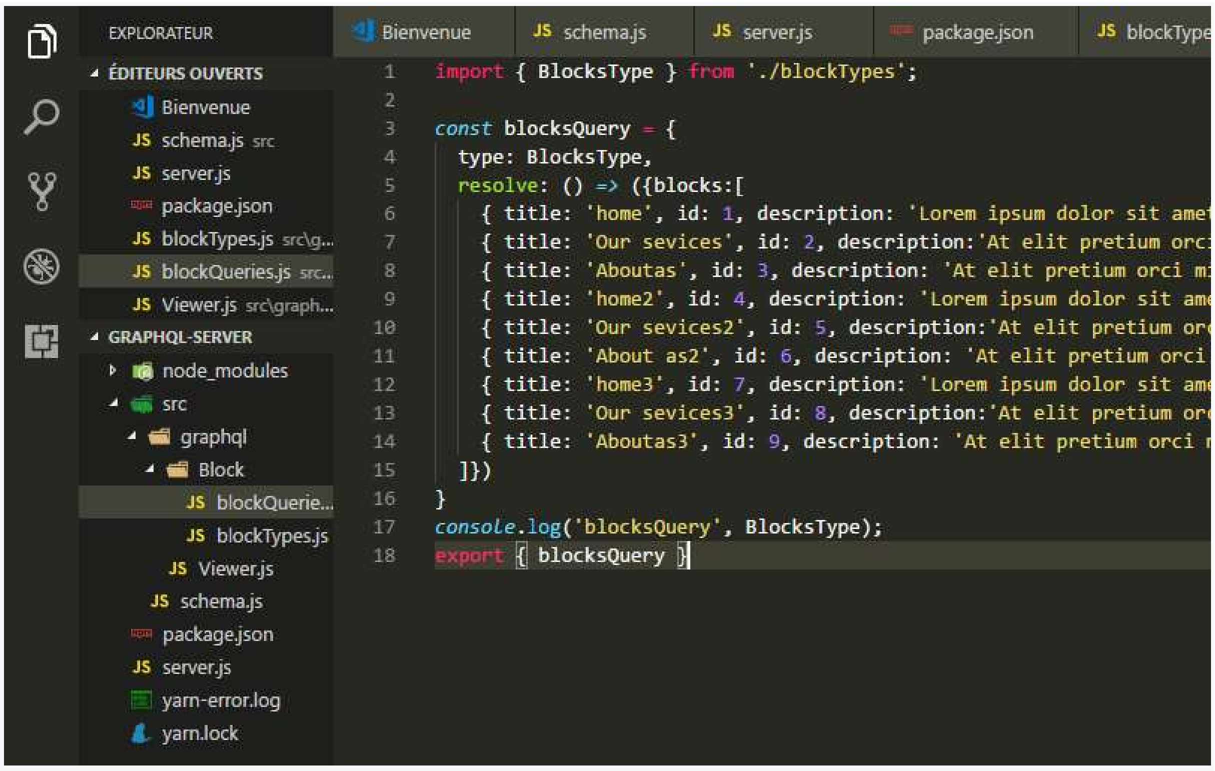
2. cd server
3. yarn add graphql
4. create this files tree
5. add types:

```
src
        |---graphql
                |--Block:
                        |--blocksQueries.js
                        |-- blocksTypes.js
```



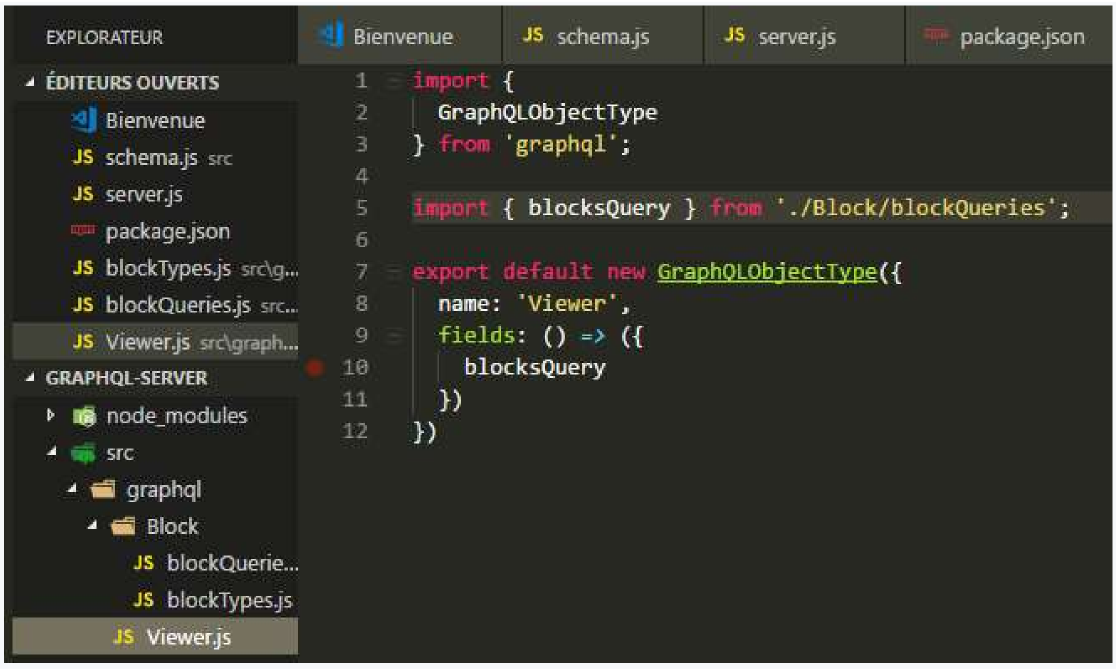
```javascript
import {
  GraphQLObjectType,
  GraphQLInputObjectType,
  GraphQLInt,
  GraphQLList,
  GraphQLString,
  GraphQLBoolean,
  GraphQLNonNull,
} from 'graphql';

const BlockType = new GraphQLObjectType({
  name: 'Block',
  fields:{
    id: { type: GraphQLString },
    title: {type: GraphQLString},
    description: {type: GraphQLString},
    image: {type: GraphQLString},
  }
});

const BlocksType = new GraphQLObjectType({
  name: 'Blocks',
  fields: {
    blocks: { type: new GraphQLList(BlockType) }
  }
});
export {
  BlocksType
}
```

6. in src/graphql/Block/blockQueries.js add your first query: we return a simple object that contain an array of blocks, we'll replace it next time by a get from database



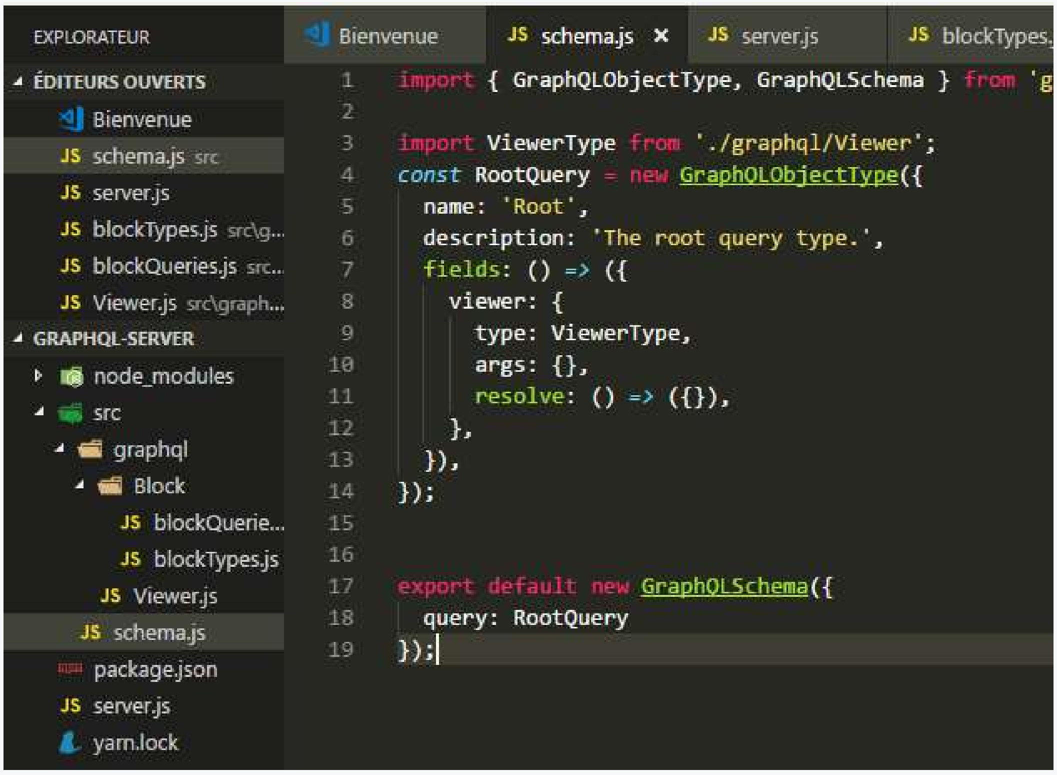
```
import { BlocksType } from './blockTypes';

const blocksQuery = {
  type: BlocksType,
  resolve: () => ({blocks:[
    { title: 'home', id: 1, description: 'Lorem ipsum dolor sit amet
    { title: 'Our sevices', id: 2, description:'At elit pretium orci
    { title: 'Aboutas', id: 3, description: 'At elit pretium orci mi
    { title: 'home2', id: 4, description: 'Lorem ipsum dolor sit ame
    { title: 'Our sevices2', id: 5, description:'At elit pretium orc
    { title: 'About as2', id: 6, description: 'At elit pretium orci
    { title: 'home3', id: 7, description: 'Lorem ipsum dolor sit ame
    { title: 'Our sevices3', id: 8, description:'At elit pretium orc
    { title: 'Aboutas3', id: 9, description: 'At elit pretium orci r
  ]})
}
console.log('blocksQuery', BlocksType);
export { blocksQuery }
```

7. in src/graphql add Viewer.js: here we export an object of all our queries



```
import {
  GraphQLObjectType
} from 'graphql';

import { blocksQuery } from './Block/blockQueries';

export default new GraphQLObjectType({
  name: 'Viewer',
  fields: () => ({
    blocksQuery
  })
})
```

8. in serc/add schema.js: it return the query and the mutation (we will introduce an example of mutation at the end of this tutorial)



EXPLORATEUR

ÉDITEURS OUVERTS
- Bienvenue
- schema.js src
- server.js
- blockTypes.js src\g...
- blockQueries.js src...
- Viewer.js src\graph...

GRAPHQL-SERVER
- node_modules
- src
  - graphql
    - Block
      - blockQuerie...
      - blockTypes.js
    - Viewer.js
  - schema.js
- package.json
- server.js
- yarn.lock

```
import { GraphQLObjectType, GraphQLSchema } from 'g

import ViewerType from './graphql/Viewer';
const RootQuery = new GraphQLObjectType({
  name: 'Root',
  description: 'The root query type.',
  fields: () => ({
    viewer: {
      type: ViewerType,
      args: {},
      resolve: () => ({}),
    },
  }),
});


export default new GraphQLSchema({
  query: RootQuery
});
```

9. create server: create an express server using

a. first you should add those dependencies:

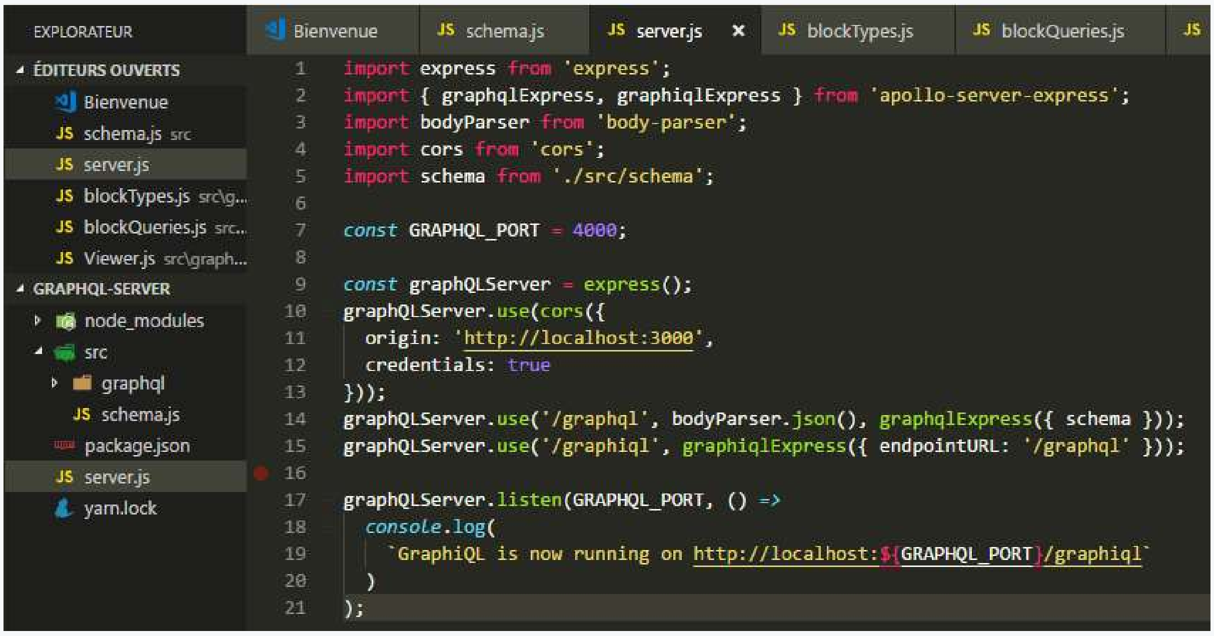
yarn add apollo-server-express

yarn add express

yarn add cors

yarn add body-parser

b. in src create server.js



EXPLORATEUR

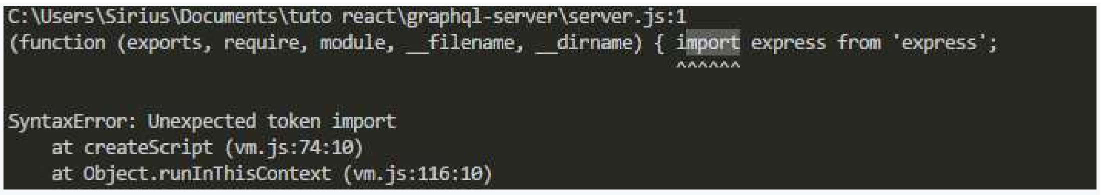
ÉDITEURS OUVERTS
- Bienvenue
- schema.js src
- server.js
- blockTypes.js src\g...
- blockQueries.js src...
- Viewer.js src\graph...

GRAPHQL-SERVER
- node_modules
- src
  - graphql
  - schema.js
- package.json
- server.js
- yarn.lock

```
import express from 'express';
import { graphqlExpress, graphiqlExpress } from 'apollo-server-express';
import bodyParser from 'body-parser';
import cors from 'cors';
import schema from './src/schema';

const GRAPHQL_PORT = 4000;

const graphQLServer = express();
graphQLServer.use(cors({
  origin: 'http://localhost:3000',
  credentials: true
}));
graphQLServer.use('/graphql', bodyParser.json(), graphqlExpress({ schema }));
graphQLServer.use('/graphiql', graphiqlExpress({ endpointURL: '/graphql' }));

graphQLServer.listen(GRAPHQL_PORT, () =>
  console.log(
    `GraphiQL is now running on http://localhost:${GRAPHQL_PORT}/graphiql`
  )
);
```

10. yarn start

you will get this error



```
C:\Users\Sirius\Documents\tuto react\graphql-server\server.js:1
(function (exports, require, module, __filename, __dirname) { import express from 'express';
                                                              ^^^^^^

SyntaxError: Unexpected token import
    at createScript (vm.js:74:10)
    at Object.runInThisContext (vm.js:116:10)
```

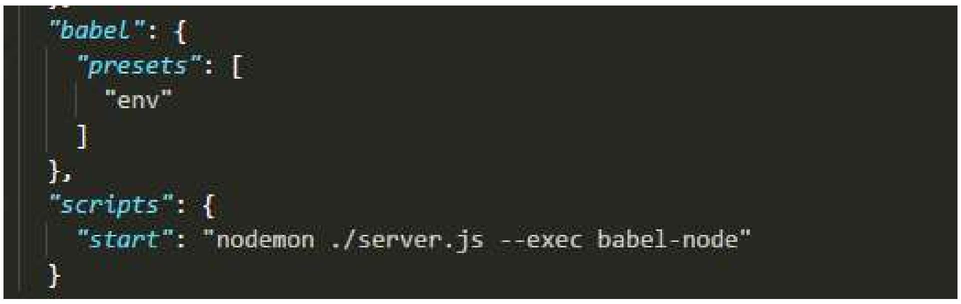
To resolve it add those dependencies:

yarn add babel-cli

yarn add babel-preset-env

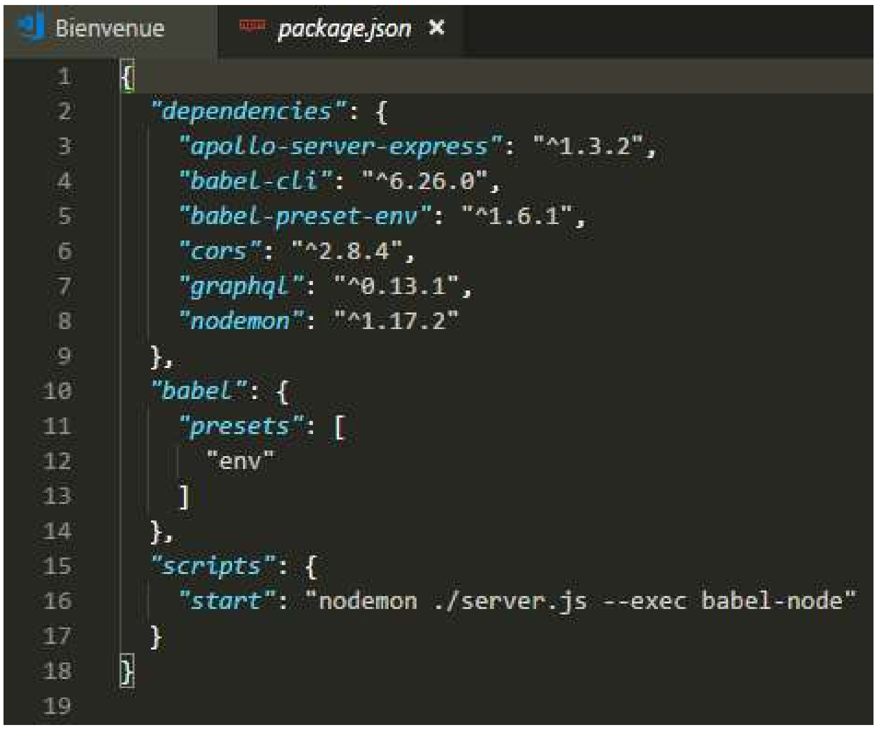
yarn add nodemon

then in your package.json add these



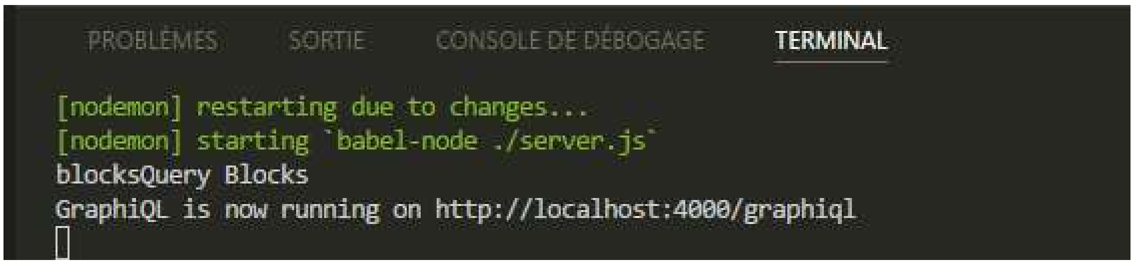
```
"babel": {
  "presets": [
    "env"
  ]
},
"scripts": {
  "start": "nodemon ./server.js --exec babel-node"
}
```

so package.json should look like these:

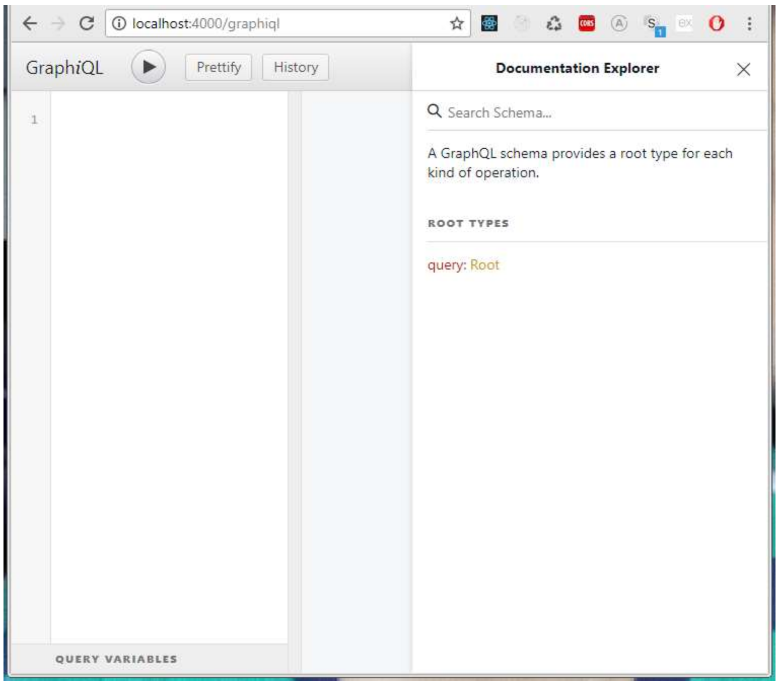


```
{
  "dependencies": {
    "apollo-server-express": "^1.3.2",
    "babel-cli": "^6.26.0",
    "babel-preset-env": "^1.6.1",
    "cors": "^2.8.4",
    "graphql": "^0.13.1",
    "nodemon": "^1.17.2"
  },
  "babel": {
    "presets": [
      "env"
    ]
  },
  "scripts": {
    "start": "nodemon ./server.js --exec babel-node"
  }
}
```

11. yarn start



```
[nodemon] restarting due to changes...
[nodemon] starting `babel-node ./server.js`
blocksQuery Blocks
GraphiQL is now running on http://localhost:4000/graphiql
```

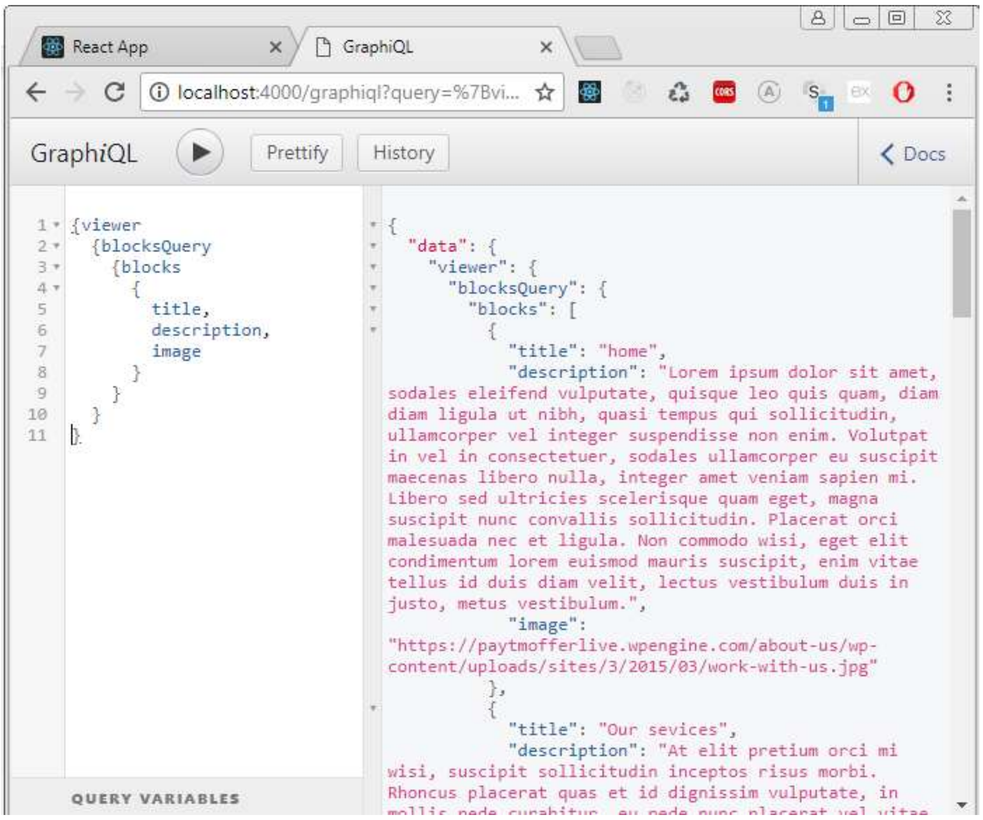


A GraphQL schema provides a root type for each kind of operation.

ROOT TYPES

query: Root

example of use



```
{viewer
  {blocksQuery
    {blocks
      {
        title,
        description,
        image
      }
    }
  }
}
```

```
{
  "data": {
    "viewer": {
      "blocksQuery": {
        "blocks": [
          {
            "title": "home",
            "description": "Lorem ipsum dolor sit amet, sodales eleifend vulputate, quisque leo quis quam, diam diam ligula ut nibh, quasi tempus qui sollicitudin, ullamcorper vel integer suspendisse non enim. Volutpat in vel in consectetuer, sodales ullamcorper eu suscipit maecenas libero nulla, integer amet veniam sapien mi. Libero sed ultricies scelerisque quam eget, magna suscipit nunc convallis sollicitudin. Placerat orci malesuada nec et ligula. Non commodo wisi, eget elit condimentum lorem euismod mauris suscipit, enim vitae tellus id duis diam velit, lectus vestibulum duis in justo, metus vestibulum.",
            "image": "https://paytmofferlive.wpengine.com/about-us/wp-content/uploads/sites/3/2015/03/work-with-us.jpg"
          },
          {
            "title": "Our sevices",
            "description": "At elit pretium orci mi wisi, suscipit sollicitudin inceptos risus morbi. Rhoncus placerat quas et id dignissim vulputate, in
```

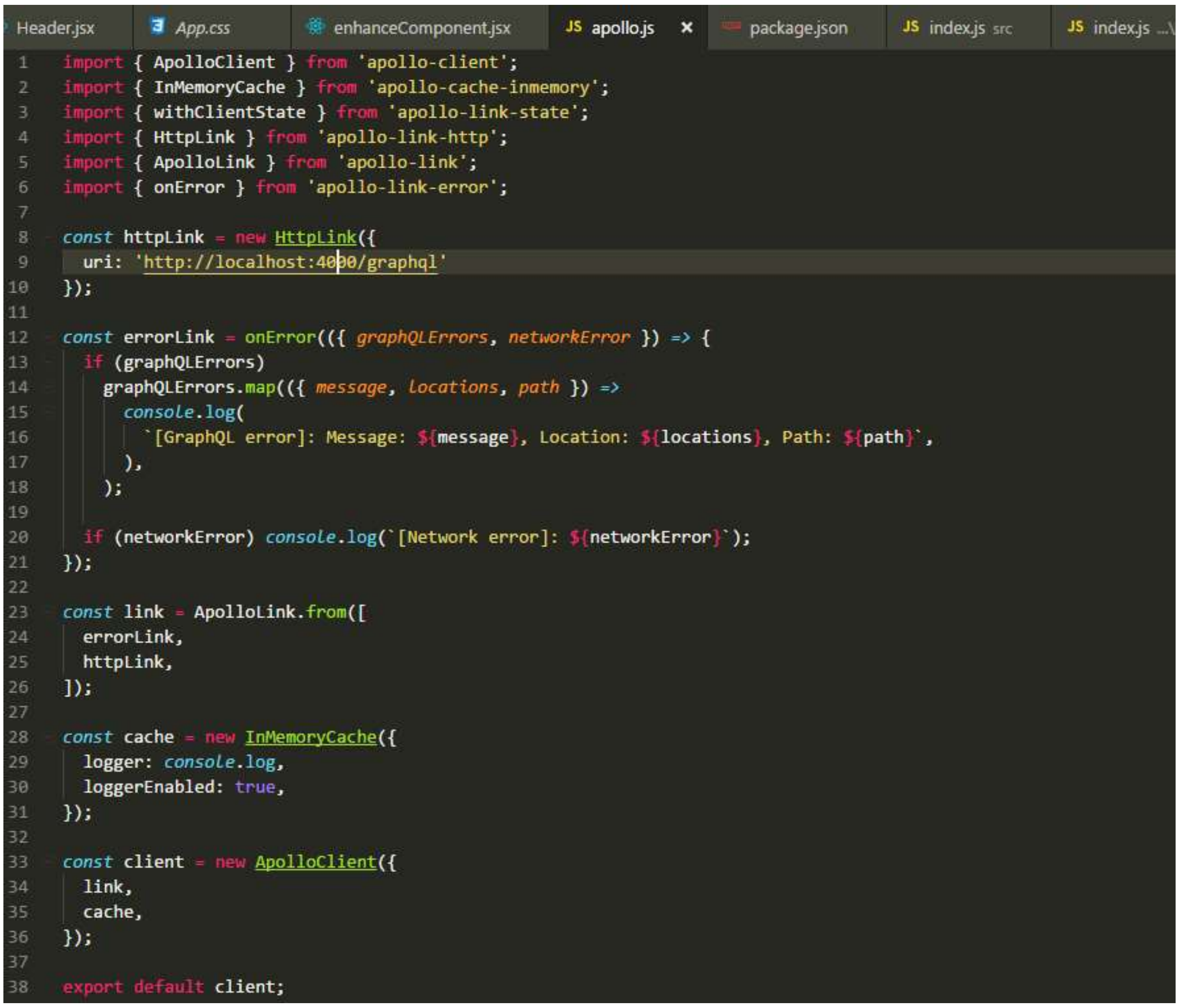
IV. get data from server:

return to front-office folder

1. setup apollo

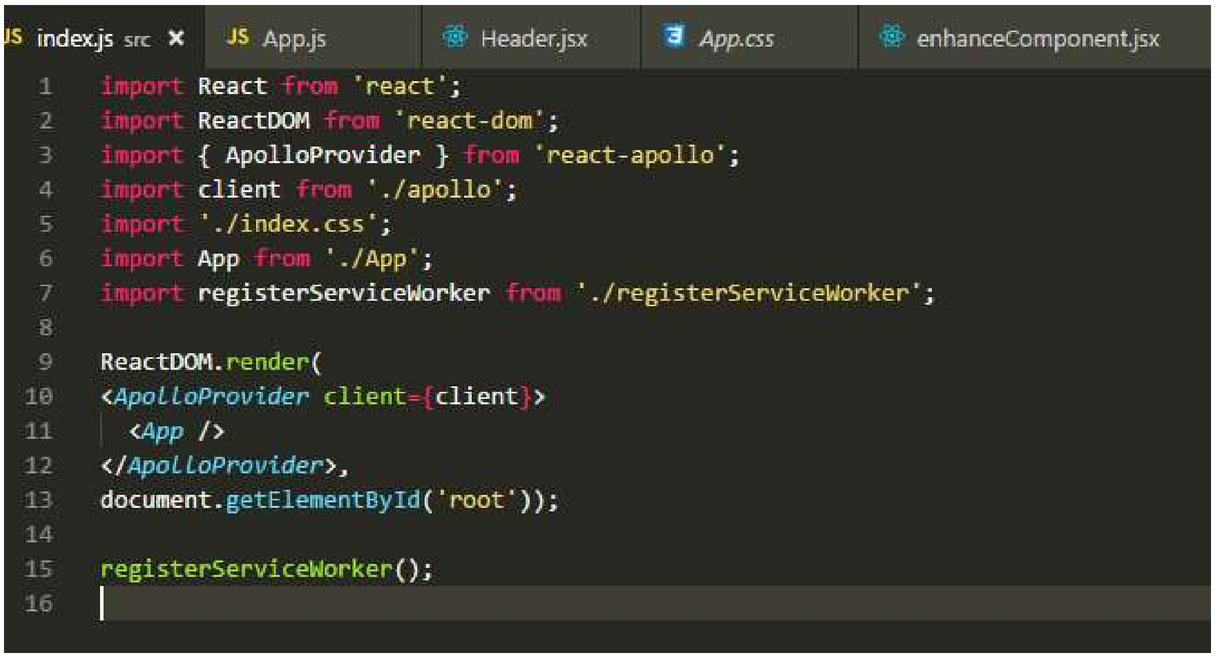
1- yarn add react-dom
2- yarn add apollo-link-state
3- yarn add apollo-link-error
4- yarn add react-apollo
5- yarn add apollo-client-preset
6- yarn add graphql

2. in src add apollo.js:



```js
import { ApolloClient } from 'apollo-client';
import { InMemoryCache } from 'apollo-cache-inmemory';
import { withClientState } from 'apollo-link-state';
import { HttpLink } from 'apollo-link-http';
import { ApolloLink } from 'apollo-link';
import { onError } from 'apollo-link-error';

const httpLink = new HttpLink({
  uri: 'http://localhost:4000/graphql'
});

const errorLink = onError(({ graphQLErrors, networkError }) => {
  if (graphQLErrors)
    graphQLErrors.map(({ message, locations, path }) =>
      console.log(
        `[GraphQL error]: Message: ${message}, Location: ${locations}, Path: ${path}`,
      ),
    );

  if (networkError) console.log(`[Network error]: ${networkError}`);
});

const link = ApolloLink.from([
  errorLink,
  httpLink,
]);

const cache = new InMemoryCache({
  logger: console.log,
  loggerEnabled: true,
});

const client = new ApolloClient({
  link,
  cache,
});

export default client;
```

3. go to App.js to wrapp our App component by the ApolloProvider

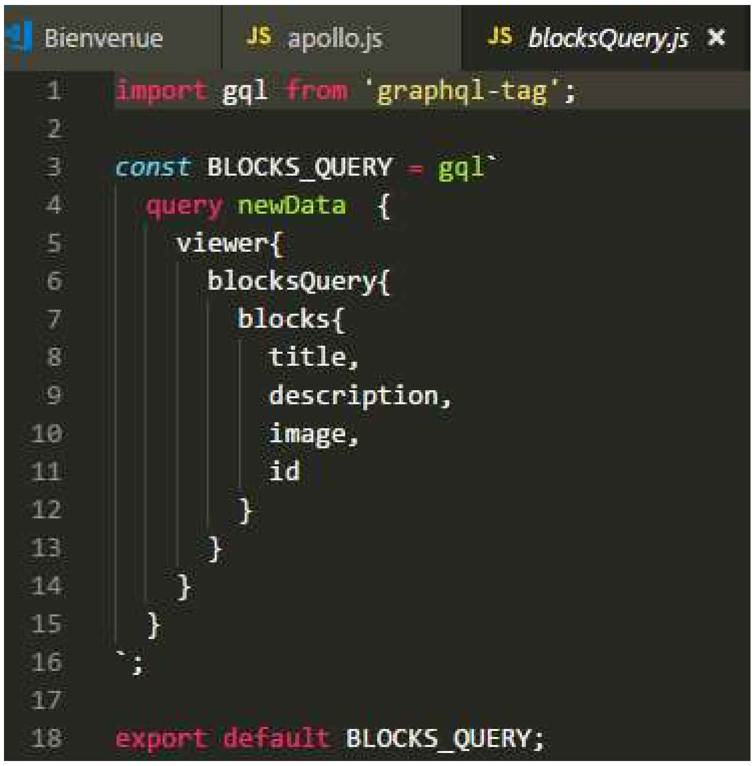


```
import React from 'react';
import ReactDOM from 'react-dom';
import { ApolloProvider } from 'react-apollo';
import client from './apollo';
import './index.css';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(
<ApolloProvider client={client}>
  <App />
</ApolloProvider>,
document.getElementById('root'));

registerServiceWorker();
```

III- Higher-Ordered Components

A Higher Ordered Component is just a React Component that wraps another one.it is a function that takes a component and returns a new component.

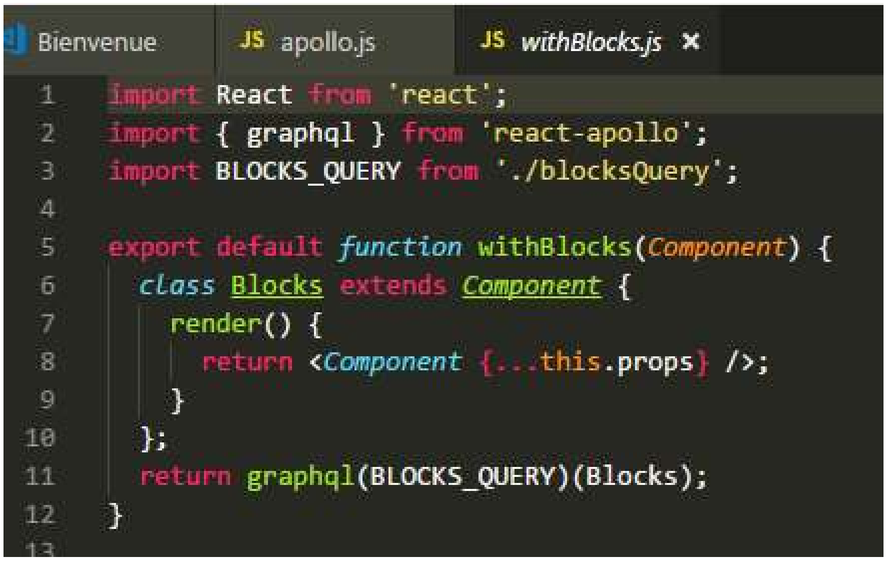
1. in src create a new folder hocs
2. in hocks create blocksQuery.js: usually isolate your queries to make them reusable



```
import gql from 'graphql-tag';

const BLOCKS_QUERY = gql`
  query newData  {
    viewer{
      blocksQuery{
        blocks{
          title,
          description,
          image,
          id
        }
      }
    }
  }
`;

export default BLOCKS_QUERY;
```

you should add graphql tag dependency: yarn add graphql-tag

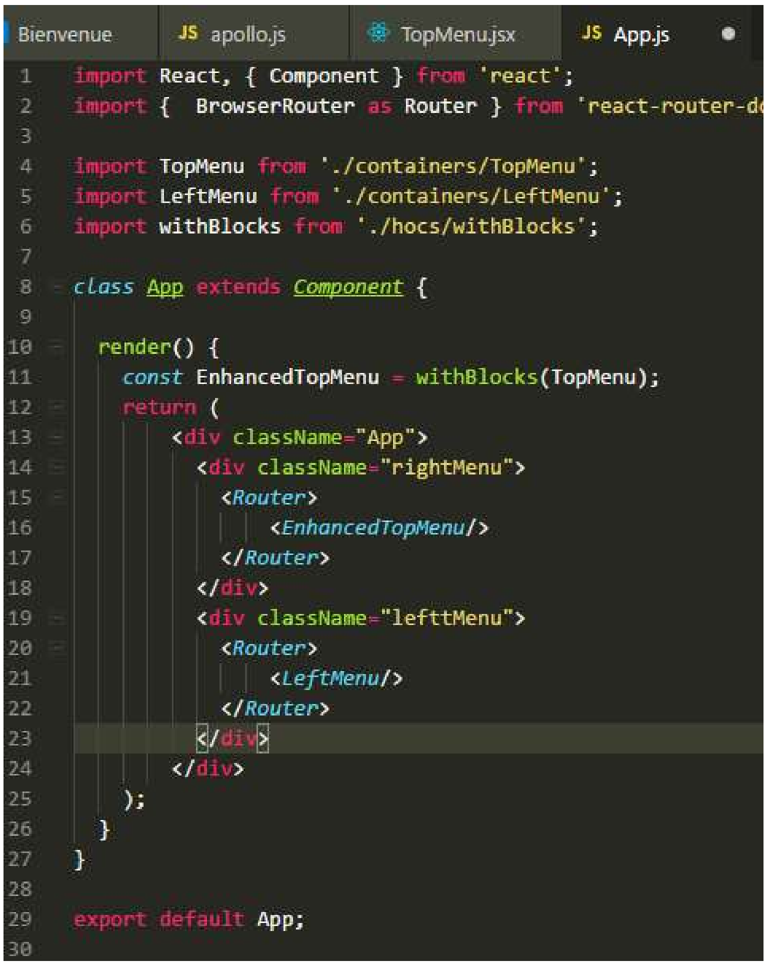
3. always in hocs add new file withBloks.js: this is our hoc, it will receive a component and return a new component that has the list of blocks (queried from server) as props.



```
import React from 'react';
import { graphql } from 'react-apollo';
import BLOCKS_QUERY from './blocksQuery';

export default function withBlocks(Component) {
  class Blocks extends Component {
    render() {
      return <Component {...this.props} />;
    }
  };
  return graphql(BLOCKS_QUERY)(Blocks);
}
```

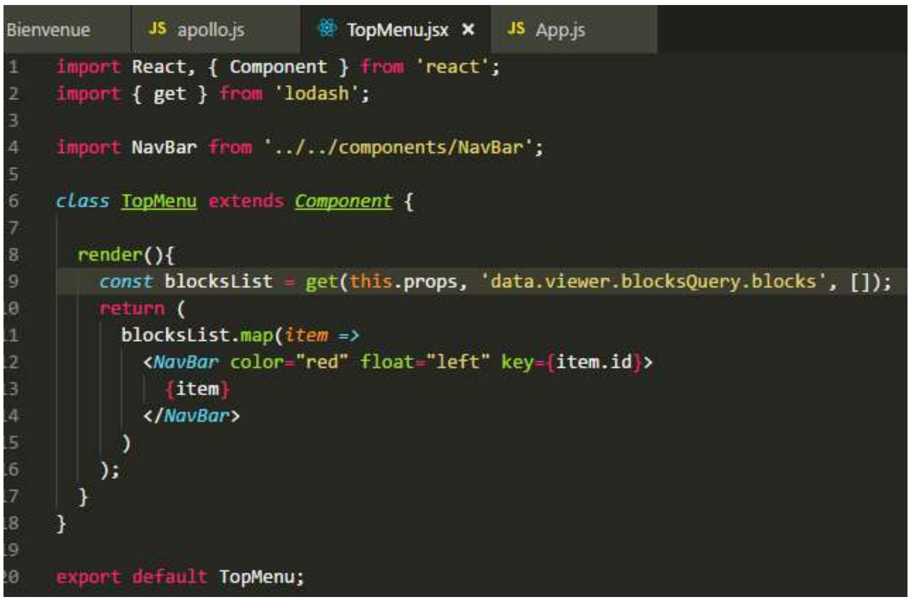
4. wrap the TopMenu container withBlocks:

go to App.js



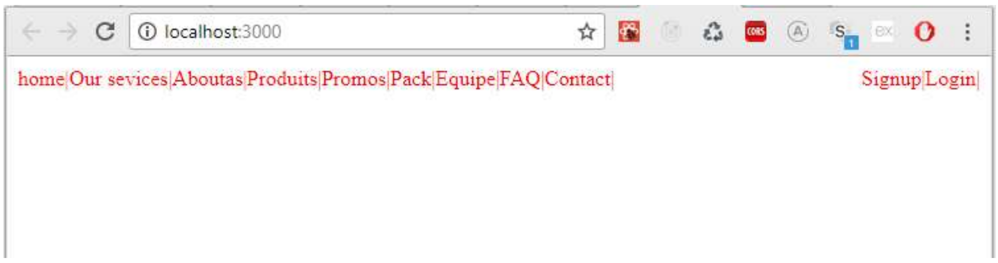
```
import React, { Component } from 'react';
import {  BrowserRouter as Router } from 'react-router-d

import TopMenu from './containers/TopMenu';
import LeftMenu from './containers/LeftMenu';
import withBlocks from './hocs/withBlocks';

class App extends Component {

  render() {
    const EnhancedTopMenu = withBlocks(TopMenu);
    return (
        <div className="App">
          <div className="rightMenu">
            <Router>
                <EnhancedTopMenu/>
            </Router>
          </div>
          <div className="lefttMenu">
            <Router>
                <LeftMenu/>
            </Router>
          </div>
        </div>
    );
  }
}

export default App;
```

5. explore our data upcoming from server :
   add lodash library : yarn add lodash



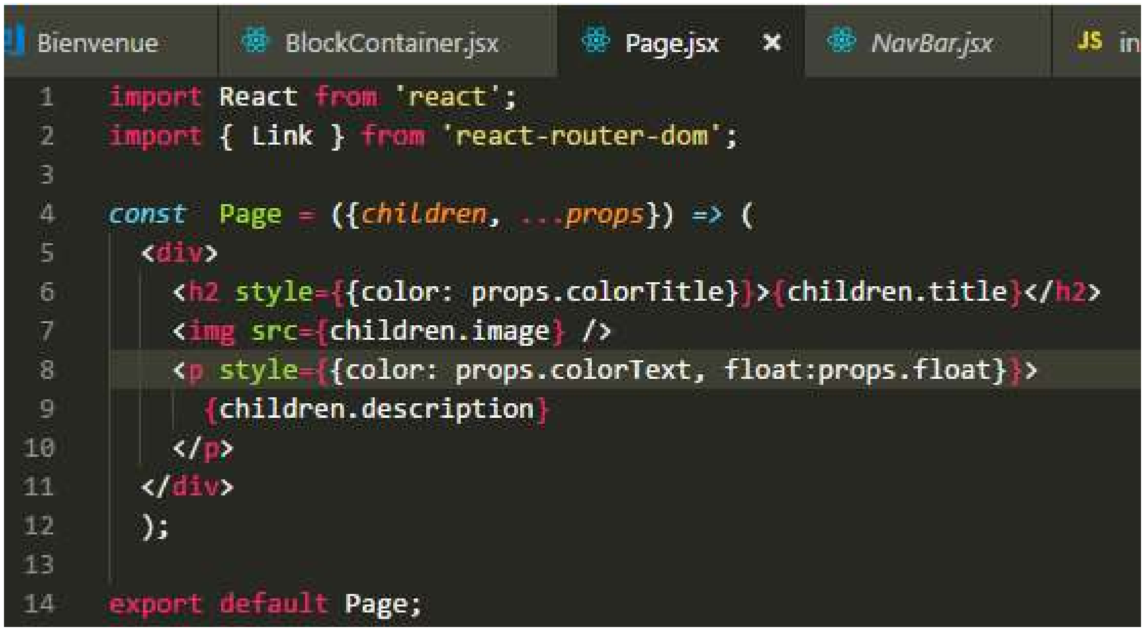
```
import React, { Component } from 'react';
import { get } from 'lodash';

import NavBar from '../../components/NavBar';

class TopMenu extends Component {

  render(){
    const blocksList = get(this.props, 'data.viewer.blocksQuery.blocks', []);
    return (
      blocksList.map(item =>
        <NavBar color="red" float="left" key={item.id}>
          {item}
        </NavBar>
      )
    );
  }
}

export default TopMenu;
```

back to navigator:



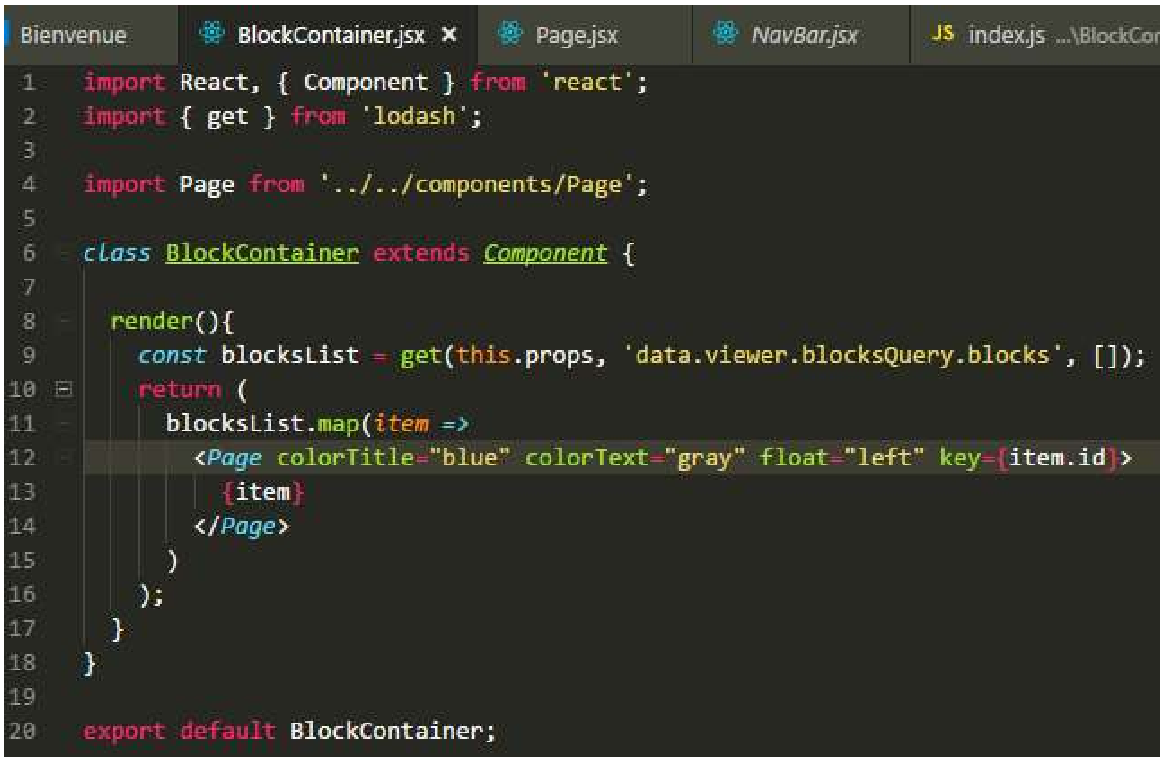
home|Our sevices|Aboutas|Produits|Promos|Pack|Equipe|FAQ|Contact| Signup|Login|

=> use the hoc “withBlocks” to wrap another container
so create a container BlockContainer and a component Page
Page.jsx:



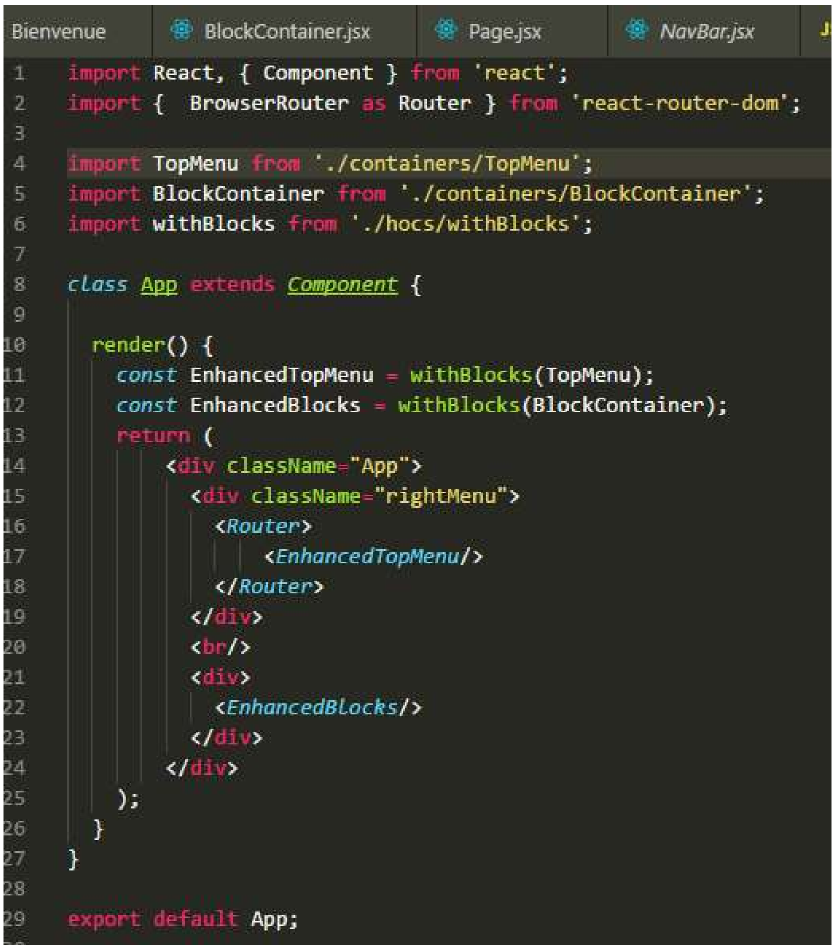
```jsx
import React from 'react';
import { Link } from 'react-router-dom';

const  Page = ({children, ...props}) => (
  <div>
    <h2 style={{color: props.colorTitle}}>{children.title}</h2>
    <img src={children.image} />
    <p style={{color: props.colorText, float:props.float}}>
      {children.description}
    </p>
  </div>
  );

export default Page;
```

BlockContainer.jsx



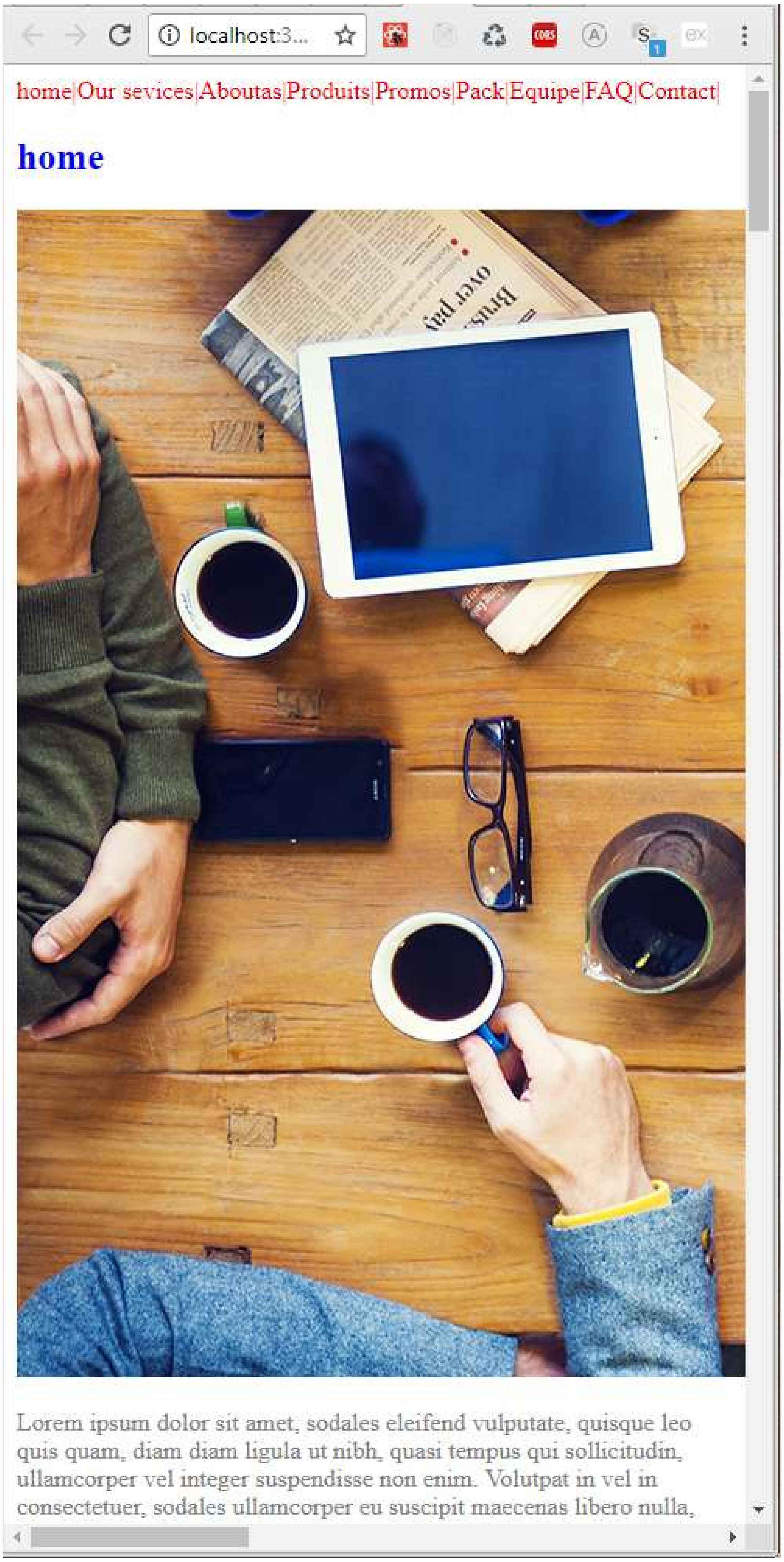
```jsx
import React, { Component } from 'react';
import { get } from 'lodash';

import Page from '../../components/Page';

class BlockContainer extends Component {

  render(){
    const blocksList = get(this.props, 'data.viewer.blocksQuery.blocks', []);
    return (
      blocksList.map(item =>
        <Page colorTitle="blue" colorText="gray" float="left" key={item.id}>
          {item}
        </Page>
      )
    );
  }
}

export default BlockContainer;
```

in App.js



```
import React, { Component } from 'react';
import {  BrowserRouter as Router } from 'react-router-dom';

import TopMenu from './containers/TopMenu';
import BlockContainer from './containers/BlockContainer';
import withBlocks from './hocs/withBlocks';

class App extends Component {

  render() {
    const EnhancedTopMenu = withBlocks(TopMenu);
    const EnhancedBlocks = withBlocks(BlockContainer);
    return (
        <div className="App">
          <div className="rightMenu">
            <Router>
                <EnhancedTopMenu/>
            </Router>
          </div>
          <br/>
          <div>
            <EnhancedBlocks/>
          </div>
        </div>
    );
  }
}

export default App;
```

the result



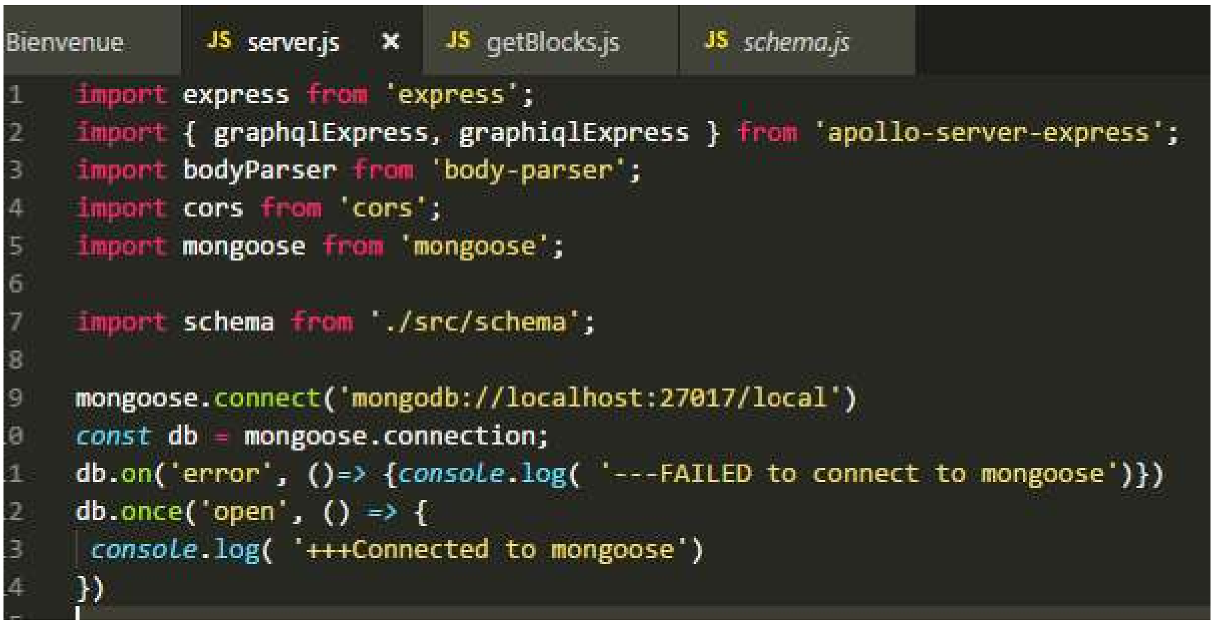
home|Our sevices|Aboutas|Produits|Promos|Pack|Equipe|FAQ|Contact|

home

Lorem ipsum dolor sit amet, sodales eleifend vulputate, quisque leo quis quam, diam diam ligula ut nibh, quasi tempus qui sollicitudin, ullamcorper vel integer suspendisse non enim. Volutpat in vel in consectetuer, sodales ullamcorper eu suscipit maecenas libero nulla,

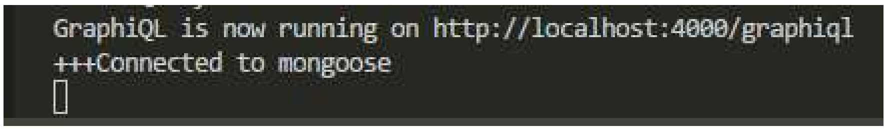
V. Mongoose

1. first, install mongodb: https://www.mongodb.com/download-center?jmp=nav#community
2. then open cmd and type mongod.
3. return to our server folder,
4. open cmd and add mongoose dependency using; yarn add mongoose
5. go to src/server.js import mongose and connect to mongo database server



```
import express from 'express';
import { graphqlExpress, graphiqlExpress } from 'apollo-server-express';
import bodyParser from 'body-parser';
import cors from 'cors';
import mongoose from 'mongoose';

import schema from './src/schema';

mongoose.connect('mongodb://localhost:27017/local')
const db = mongoose.connection;
db.on('error', ()=> {console.log( '---FAILED to connect to mongoose')})
db.once('open', () => {
  console.log( '+++Connected to mongoose')
})
```

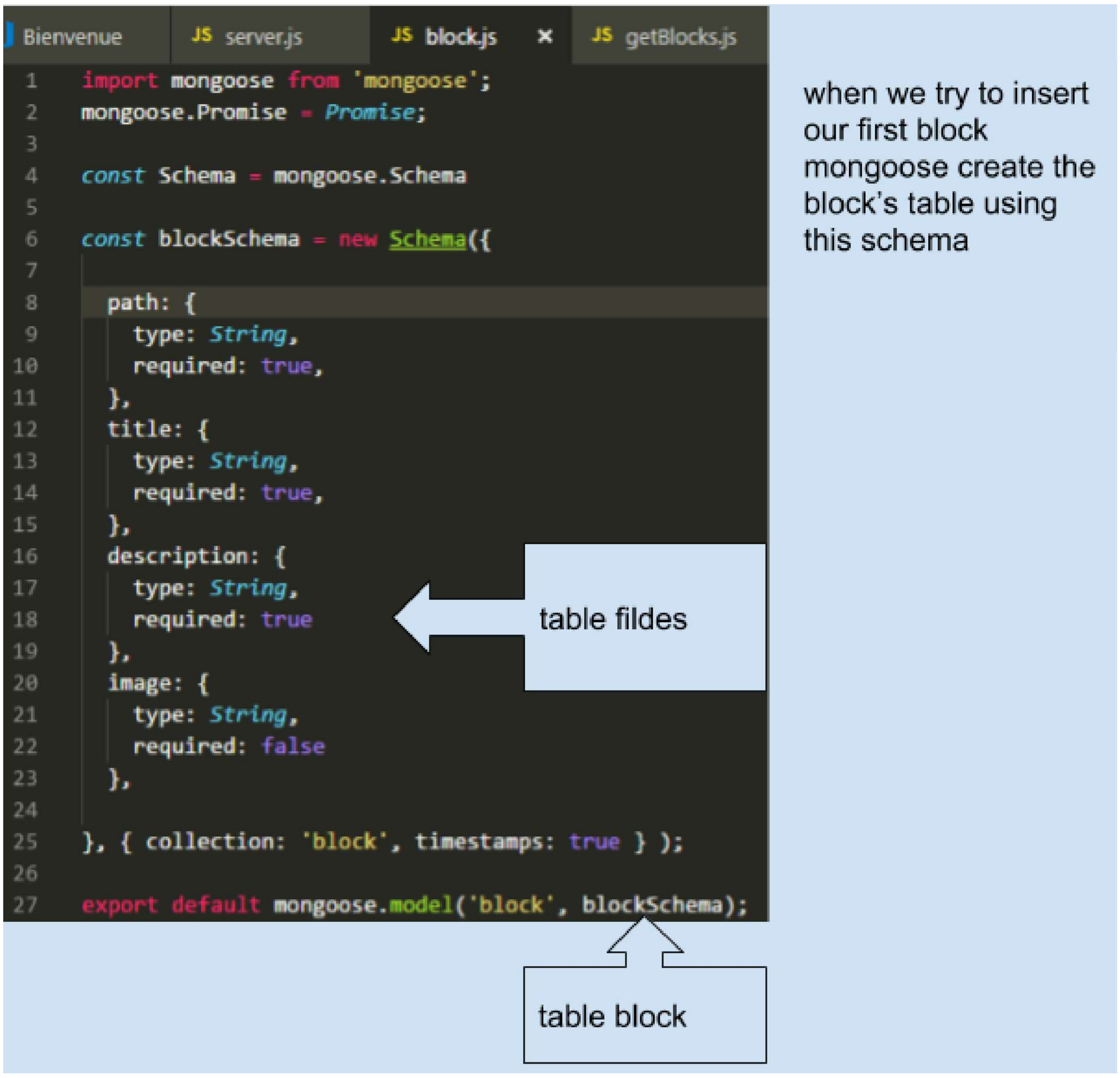
in cmd console we get this message



```
GraphiQL is now running on http://localhost:4000/graphiql
+++Connected to mongoose
```

so our server is connected to mongoDB.

6. under src create new folder “service”, in service create folder “blockServices” in this folder create folder named “models”
7. under models create block.js: lets define the schema of block

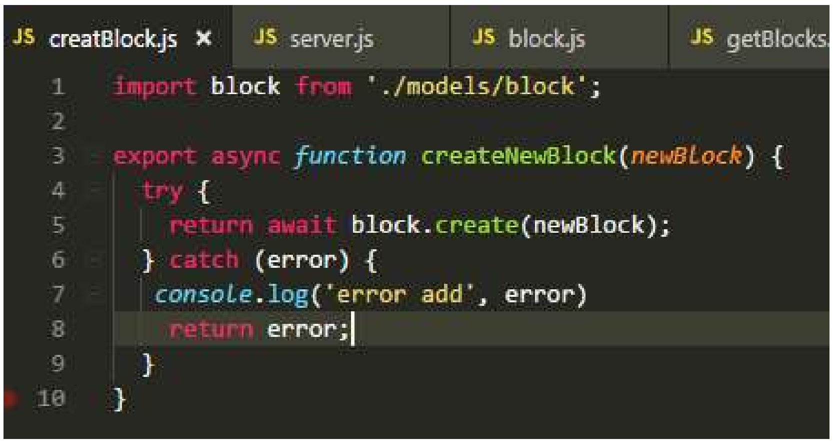


```
import mongoose from 'mongoose';
mongoose.Promise = Promise;

const Schema = mongoose.Schema

const blockSchema = new Schema({

  path: {
    type: String,
    required: true,
  },
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true
  },
  image: {
    type: String,
    required: false
  },

}, { collection: 'block', timestamps: true } );

export default mongoose.model('block', blockSchema);
```

when we try to insert our first block mongoose create the block's table using this schema

8. Constructing documents(used to insert blocks into database):

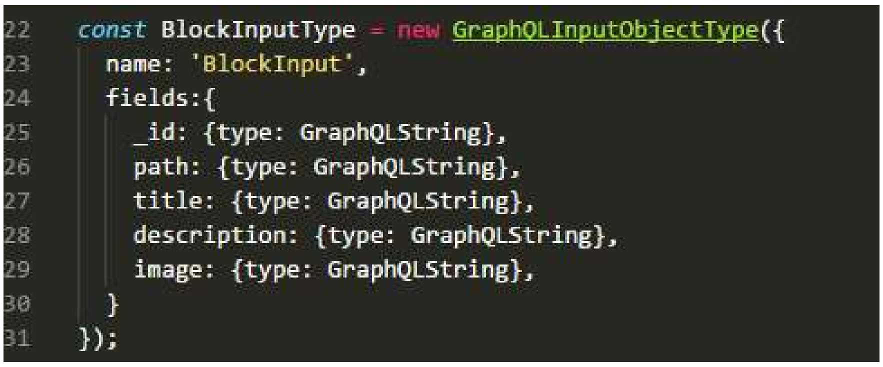
go back to blockServices and add creatBlock.js: here we put the logic of creating documents and saving to the database. this function will be called by the graphql mutation



```
JS creatBlock.js ×   JS server.js   JS block.js   JS getBlocks.
import block from './models/block';

export async function createNewBlock(newBlock) {
  try {
    return await block.create(newBlock);
  } catch (error) {
   console.log('error add', error)
    return error;
  }
}
```

9. Create block's mutation

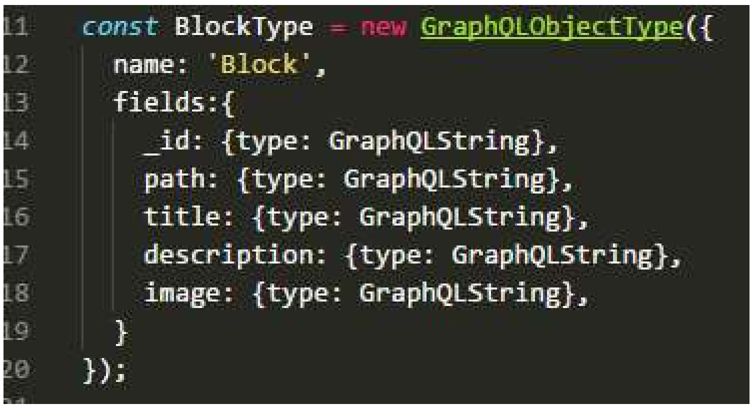
before creating the mutation go under src/graphql/Block/blockTypes.js add **GraphQLInputObjectType,** into import , than add the block input type:



```
const BlockInputType = new GraphQLInputObjectType({
  name: 'BlockInput',
  fields:{
    _id: {type: GraphQLString},
    path: {type: GraphQLString},
    title: {type: GraphQLString},
    description: {type: GraphQLString},
    image: {type: GraphQLString},
  }
});
```

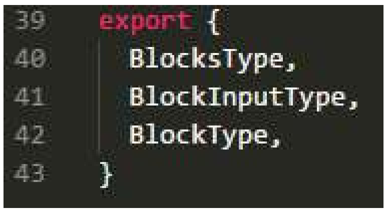
we need to modify the BlockType, replace the "id" field by "_id" the autogenerated id by mongo DB.

so the new BlockType should look like this:



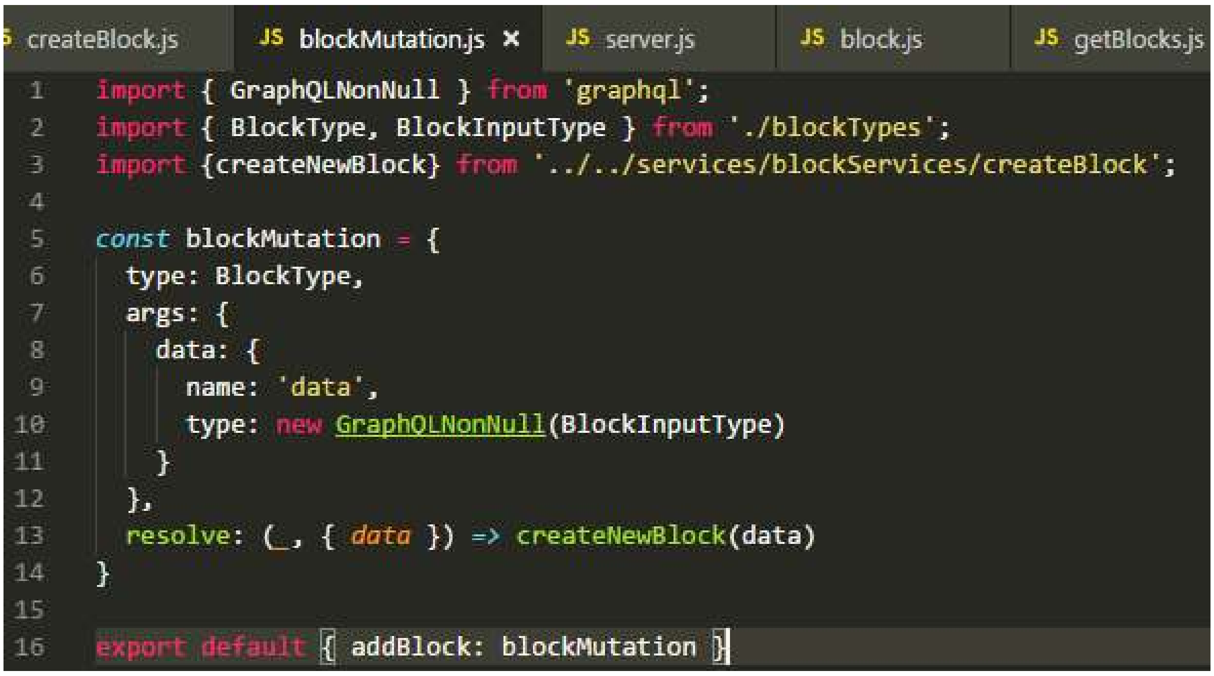
```
const BlockType = new GraphQLObjectType({
  name: 'Block',
  fields:{
    _id: {type: GraphQLString},
    path: {type: GraphQLString},
    title: {type: GraphQLString},
    description: {type: GraphQLString},
    image: {type: GraphQLString},
  }
});
```

Don't forgot to export new types :



```
export {
  BlocksType,
  BlockInputType,
  BlockType,
}
```

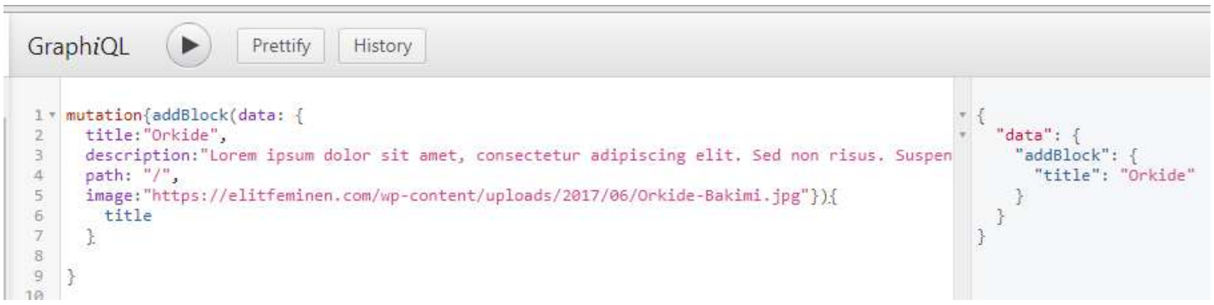
In src/Graphql/Block create blockMutation.js



createBlock.js | blockMutation.js × | server.js | block.js | getBlocks.js

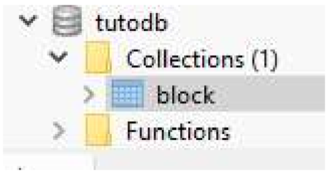
```
import { GraphQLNonNull } from 'graphql';
import { BlockType, BlockInputType } from './blockTypes';
import {createNewBlock} from '../../services/blockServices/createBlock';

const blockMutation = {
  type: BlockType,
  args: {
    data: {
      name: 'data',
      type: new GraphQLNonNull(BlockInputType)
    }
  },
  resolve: (_, { data }) => createNewBlock(data)
}

export default { addBlock: blockMutation }
```

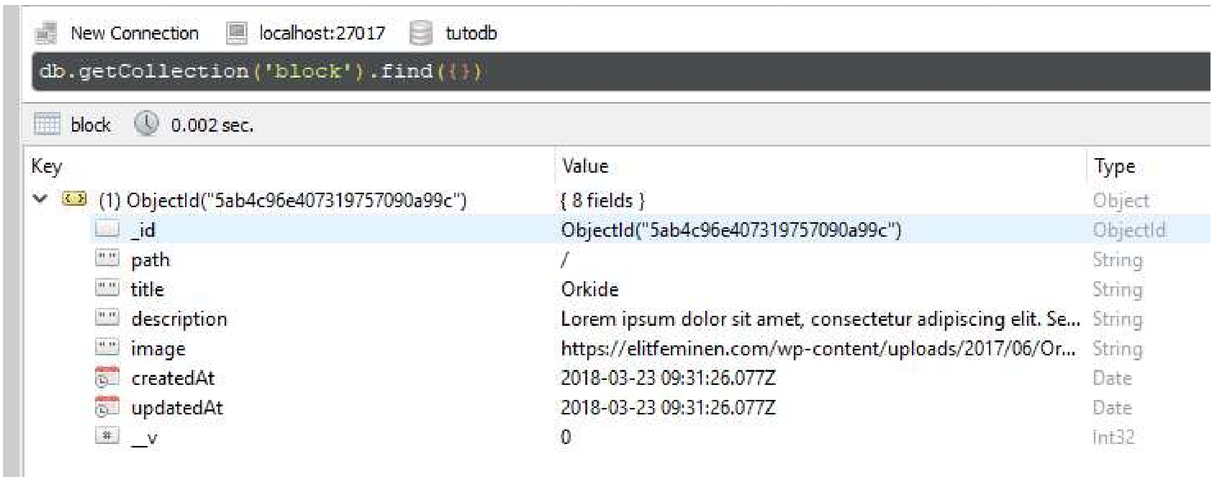
now test this mutation



```
mutation{addBlock(data: {
  title:"Orkide",
  description:"Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspen
  path: "/",
  image:"https://elitfeminen.com/wp-content/uploads/2017/06/Orkide-Bakimi.jpg"}){
    title
  }

}
```

```
{
  "data": {
    "addBlock": {
      "title": "Orkide"
    }
  }
}
```

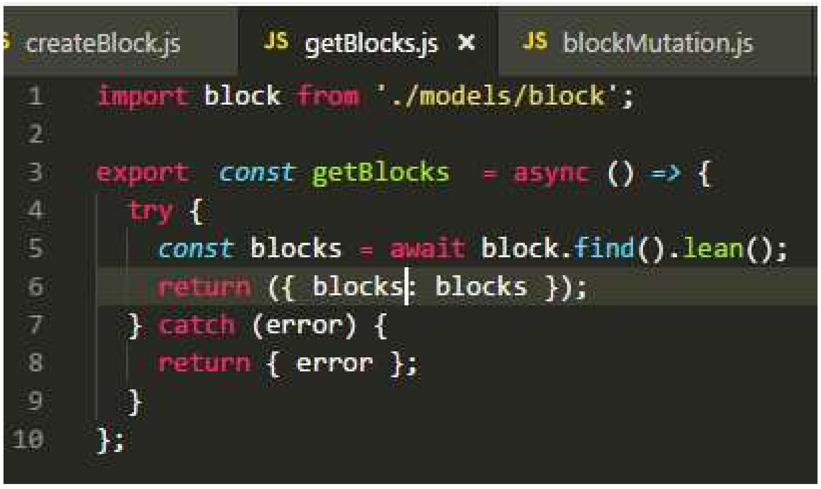




```
db.getCollection('block').find({})
```

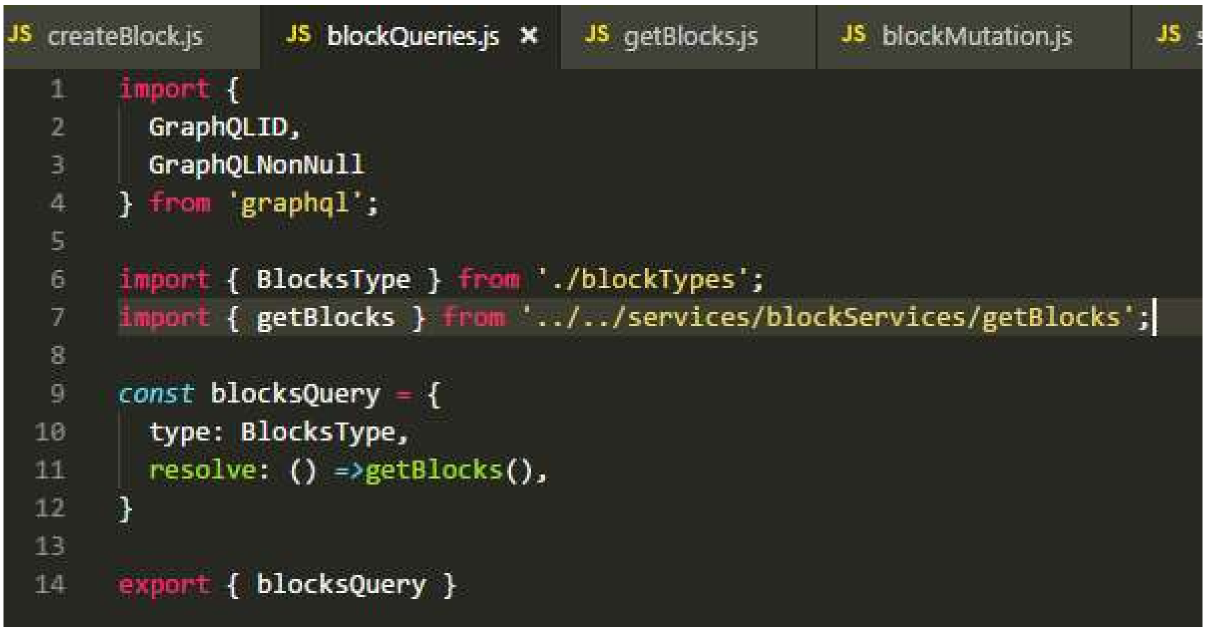
Key	Value	Type
(1) ObjectId("5ab4c96e407319757090a99c")	{ 8 fields }	Object
_id	ObjectId("5ab4c96e407319757090a99c")	ObjectId
path	/	String
title	Orkide	String
description	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Se...	String
image	https://elitfeminen.com/wp-content/uploads/2017/06/Or...	String
createdAt	2018-03-23 09:31:26.077Z	Date
updatedAt	2018-03-23 09:31:26.077Z	Date
__v	0	Int32

10. Querying data : under blockServices create getBlock.js



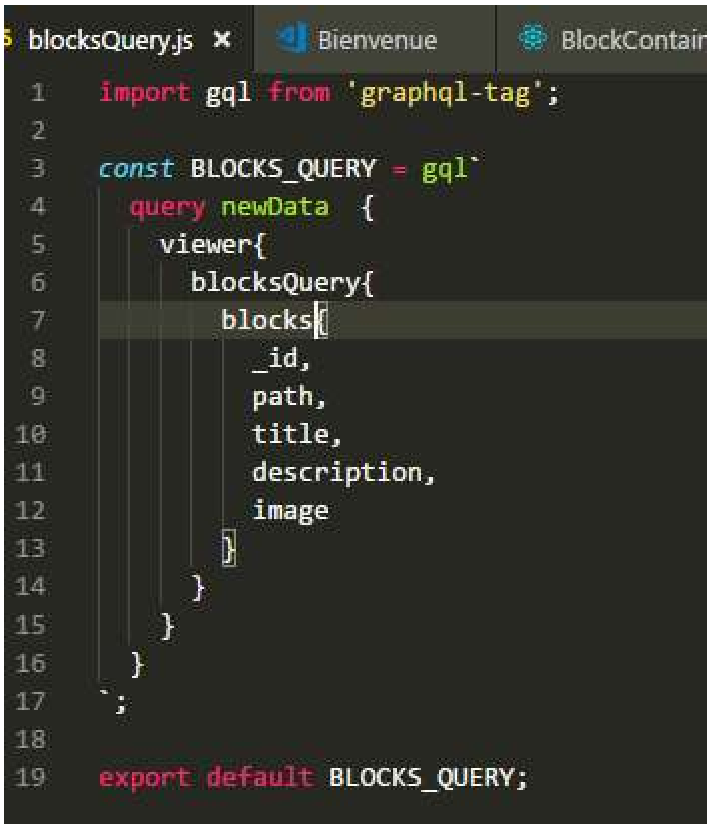
```
import block from './models/block';

export  const getBlocks  = async () => {
  try {
    const blocks = await block.find().lean();
    return ({ blocks: blocks });
  } catch (error) {
    return { error };
  }
};
```

go to blocksType and call getBlocks instead of the static object :



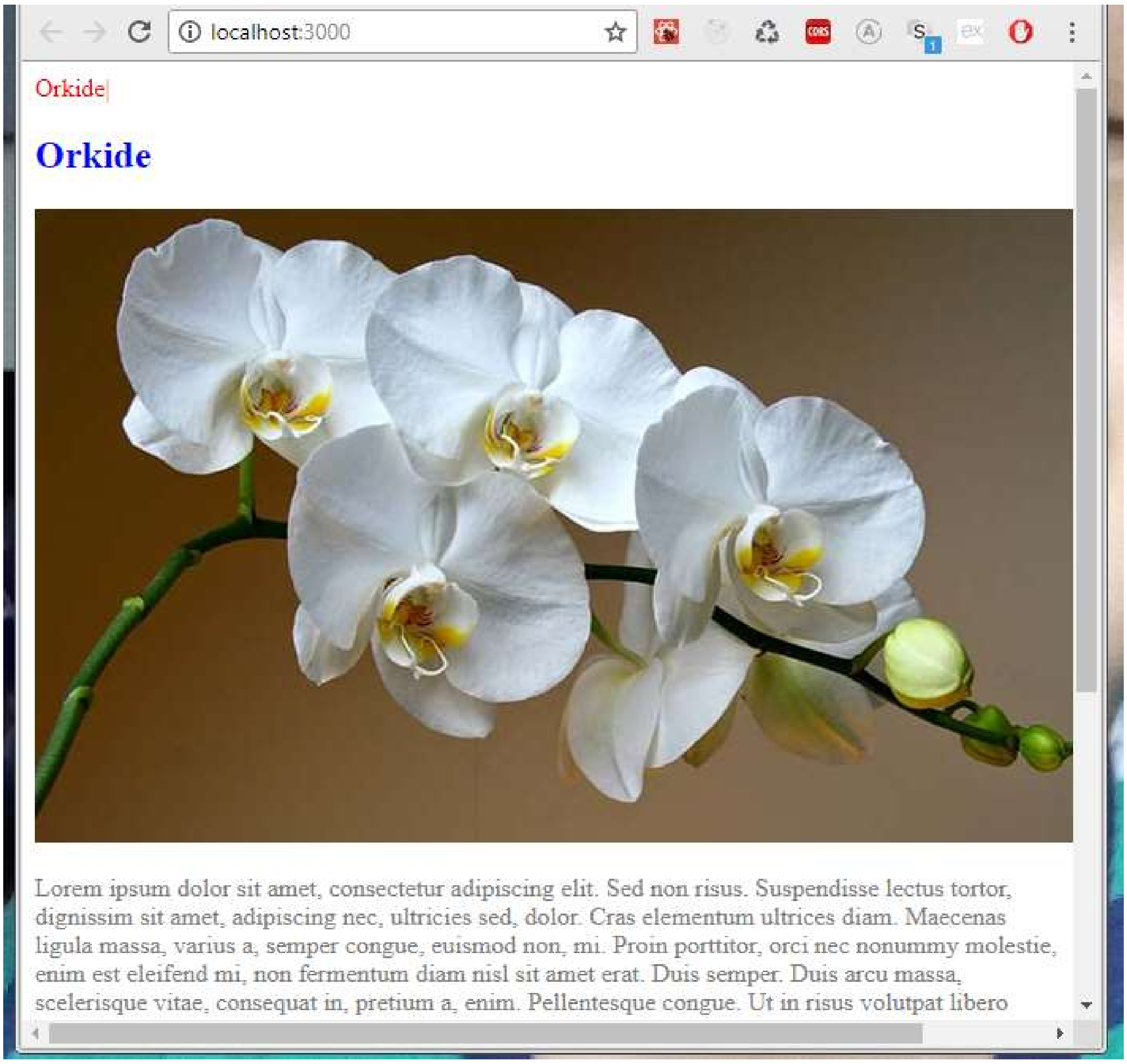
```
import {
  GraphQLID,
  GraphQLNonNull
} from 'graphql';

import { BlocksType } from './blockTypes';
import { getBlocks } from '../../services/blockServices/getBlocks';

const blocksQuery = {
  type: BlocksType,
  resolve: () =>getBlocks(),
}

export { blocksQuery }
```

back to the web client (front-office) and modify the blocksQuery.js



```
import gql from 'graphql-tag';

const BLOCKS_QUERY = gql`
  query newData  {
    viewer{
      blocksQuery{
        blocks{
          _id,
          path,
          title,
          description,
          image
        }
      }
    }
  }
`;

export default BLOCKS_QUERY;
```

open the web client in navigator (http://localhost:3000/)



Orkide

Orkide

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero