**Formation Mean Stack**

Les outils utilisés :

Node.js, MongoDB, robo3T, Postman, Github Desktop, visual studio code.

* **Get :** client recoit des données du serveur.
* **Post :** envoyer des données du client vers serveur.
* Push() : ajouter un tableau , exemple : this.comment.push(this.commentaire)
* Si en utilise [(ngModel)], il faut importer FormsModule => imports [FormsModule]
* **Router :** si en utilise un fichier.js principal et des autres fichiers.js, il y a routage entre ces fichiers.
* **App**: en utilise sous le fichier.js principal

Version Angular : ng -v

Importer notre projet sur github :

* **Créer nouveau repository :**

**create a new repository on the command line**

echo "# Projet-Angular" >> README.md

1. git init
2. git add README.md
3. git commit -m "first commit"
4. git remote add origin <https://github.com/saberjdidi/Projet-Angular.git> si ne fonctionne pas : git remote remove origin
5. git push -u origin master

ajouter sur repository créer des commits :

1. git add .
2. git commit -m "autre commit"
3. git push -u origin master

Création de projet :

1. **Installation Node.js**

Invite de commande : **npm init** (sous notre projet pour créer fichier package.json)

1. **Installation Express**

Cmd> **npm install express --save**

Exemple : créer un fichier **index.js** :

const express = require('express')

const app = express()

app.get('/', function (req, res) {

res.send('Hello World!')

})

app.listen(3000, function () {

console.log('Example app listening on port 3000!')

})

1. **Installation body-parser**

Cmd> **npm install body-parser**

Utiliser dans le fichier index.js :

var bodyParser **=** require('body-parser')

*// parse application/x-www-form-urlencoded*

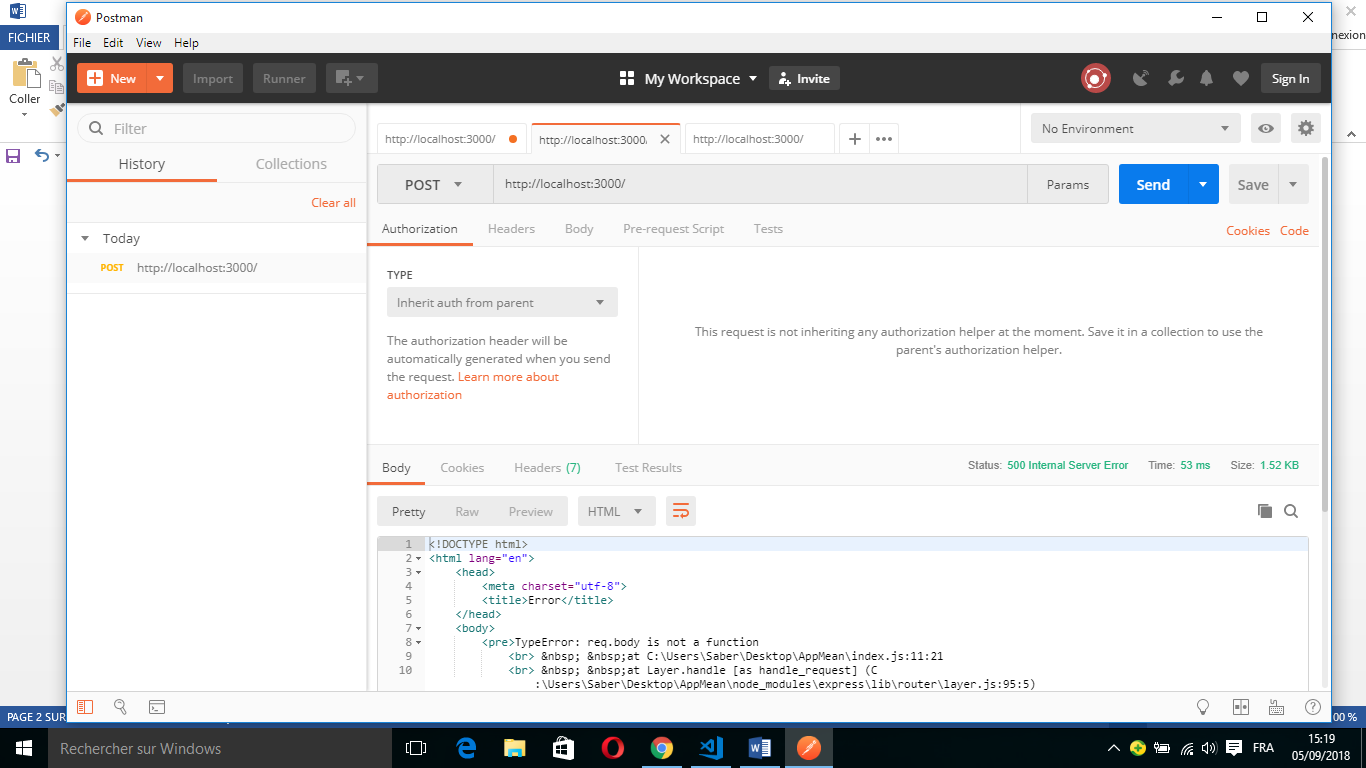
app.use(bodyParser.urlencoded({ extended**:** false }))

*// parse application/json*

app.use(bodyParser.json())

1. **Utilise Postman**

* Ajouter post pour index.js :
* app.post('/', function (req, res) {
* console.log(req.body())
* res.send('Hello World!')
* })
* Postman : permet de tester le code Node.js

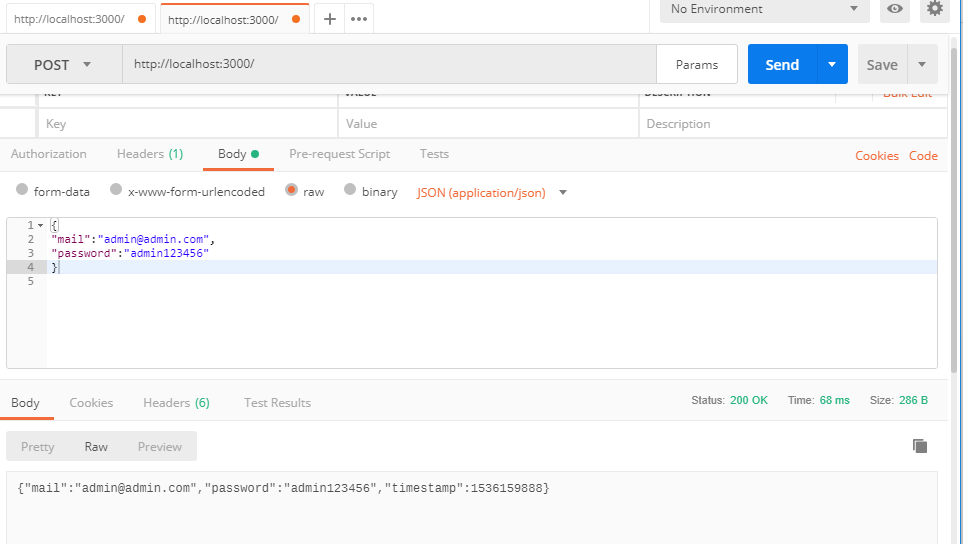


1. **Lancer serveur après chaque enregistrement automatique :**

Cmd> **npm install –g nodemon**

Cmd> **nodemon index.js** (nom de fichier)

1. **Utiliser body parser pour notre application**
2. const express = require('express')
3. const app = express()
4. var bodyParser = require('body-parser')
5. // parse application/x-www-form-urlencoded
6. app.use(bodyParser.urlencoded({ extended: false })) // n’est pas obligé
8. // parse application/json
9. app.use(bodyParser.json())
10. app.get('/', function (req, res) {
11. res.send('Hello World!')
12. })
13. app.post('/', function (req, res) {
14. //console.log(req.body)
15. req.body.timestamp = Math.floor(new Date().getTime()/1000);
16. res.send(req.body)
17. })
18. app.listen(3000, function () {
19. console.log('Example app listening on port 3000!')
20. })



1. **Créer fichier index.js permet d’afficher pseudoname sans espace, date de naissance (majeur ou mineur), vérifier mail et vérifier password :**

const express = require('express')

const app = express()

//Import the mongoose module

var mongoose = require('mongoose');

var bodyParser = require('body-parser')

// parse application/x-www-form-urlencoded

app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json

app.use(bodyParser.json())

mongoose.connect('mongodb://localhost/dbusers', function(err) {

if (err) { throw err; }

console.log("vous etes connectez");

});

app.get('/', function (req, res) {

res.send('Hello World!')

})

app.post('/', function (req, res){

//afficher pseudoname sans espace

var a=" ";

name=req.body.pseudoname;

var pseudo="";

for(var i=0 ; i<name.length ; i++){

if(name.indexOf(a)==-1){

//res.send("name ne contient pas espace")

pseudo=("name ne contient pas espace")

} else{

//res.send("name contient espace")

pseudo=("name contient espace")

}

}

//Date de naissance

var date = new Date(req.body.date);

var D = Date.now();

var dt ="";

com=Math.round(Math.abs(D-date));

sd=com/(1000\*60\*60\*24\*365)

//console.log(sd);

if(sd>18){

//res.send("date est majeur");

dt = ("date est majeur");

}else{

//res.send("date est mineur");

dt = ("date est mineur");

}

// verifier mail

var mail = req.body.mail;

var regex = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

var vide ="mail not valid";

var ml="";

//methode 1 avec fonction

function check\_email(val,regex,res){

if(!val.match(regex)){

console.log('mail invalid')

//vide ="mail not valid"

//res.send("mail invalid")

ml = ("mail invalid");

}

if( val.indexOf(' ')!=-1 || val.indexOf('..')!=-1){

console.log('mail invalid')

//vide ="mail not valid"

//res.send("mail invalid")

ml = ("mail invalid");

}

console.log('mail valid')

// vide ="mail valid"

//res.send("mail valid")

ml = ("mail valid");

}

check\_email(mail , regex ,res);

///methode 2 verifier mail avec test

///if(regex.test(mail)){

///res.send("mail valid")

///} else {

/// res.send("mail invalid")

///}

//methode 3 verifier mail

//var x=0;

//var y=0;

//var z=".";

//var a="@";

//for (var i=0 ; i<mail.length; i++){

//if(mail[i] == a && i>3){

// i>3 càd au moins il ya 4 caractères

//x=i;

// y++;

//} else if(mail[i]==z && i>x+3){

//i>x+3 càd après @ il ya au moins 3 caractères

//x=i;

//y++;

//}else if(y==2 && i>x+1){

//i>x+1 càd après "." il ya au moins 2 caractères

// vide="mail valid"

//}

// }

//res.send(vide)

// verifier password

var csp = ["-","\_","@",".",",",";",":"];

var char = req.body.password;

var incrime=0;

var incrimeMessages = ['not valid','faible','moyenne','fort'];

var pd="";

//var vide = "";

if(char.length < 8){

incrime="0";

//char = "Password is not valid";

}else{

for(var i=0 ; i < char.length ; i++){

if(isNaN(char[i])==false){

incrime++;

console.log('Number');

break;

}

}

for(var i=0 ; i < char.length ; i++){

if(csp.indexOf(char[i]) == -1 && isNaN(char[i]) && char[i].toUpperCase()){

incrime++;

console.log('Uppercase');

break;

}

}

for (var i=0 ; i<char.length ; i++){

if(csp.indexOf(char[i]) > -1){

incrime++

console.log('special');

break;

}

}

}

pd=(incrimeMessages[incrime]);

res.send("Pseudoname est: "+ pseudo + "\n Date de naissance est: "+ dt +"\n Mail est:"+ ml + "\n Password est:"+pd)

})

app.listen(3000, function () {

console.log('Example app listening on port 3000!')

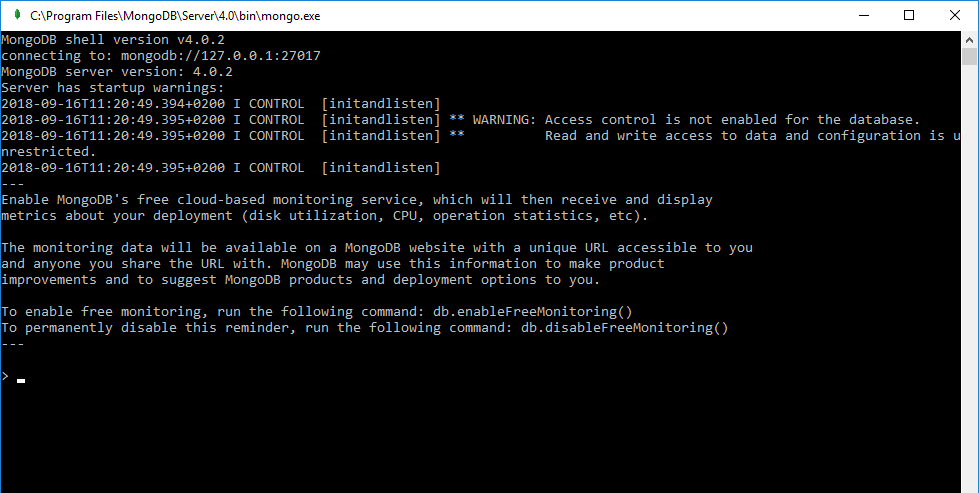
})

1. **Installation Mongo DB**

On va créer dossier (**data**) et sous (**data**) deux dossiers (**db** et **log**). Puis en lance (C:\Program Files\MongoDB\Server\4.0\bin\ **mongod**) pour faire l’installation.

Dans MongoDB on va créer une base de données (DB). Une DB contient une ou plusieurs collections, chaque collection contient un ou plusieurs documents.

On ouvre mongo pour créer notre DB :



Si ce mongo ne fonctionne pas, on va démarrer MongoDB à partir de (gestionnaire des taches / services / MongoDB (démarrer)).

# afficher toutes les base de données

show dbs

# utiliser la base "madatabase"

**use** madatabase

# voir dans quelle base on est

db

# afficher les collections

show collections

# renommer une collection

db.oldname.renameCollection("newname")

# effacer la collection (contacts)

db.contacts.drop()

# effacer la base dans laquelle on est

db.dropDatabase()

# insertion

db.contacts.insert({ first: 'Quentin', **last**: 'Busuttil' })

# sélection

db.contacts.find()

db.contacts.find({ first: 'quentin' })

db.contacts.find({ first: 'quentin', **last**: 'busuttil' })

# sélectionner les documents dont un array contient une valeur x

db.contacts.find({ tags: 'business' })

# ou un élément parmi plusieurs

db.contacts.find({ tags: { $in: ['business', 'french'] } })

# ou la combinaison de plusieurs

db.contacts.find({ tags: { $all: ['business', 'french'] } })

# sous cette forme l'array correspond exactement à cela (ordre des éléments compris)

db.contacts.find({ tags: ['business', 'french'] })

1. **Installation mongoose :** faire interaction entre node.js et MongoDB :

Cmd> **npm install mongoose --save**

Utilise mongoose dans node :

//Import the mongoose module

var mongoose = require('mongoose');

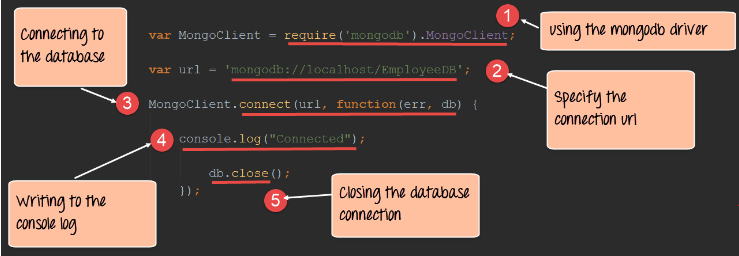
mongoose.connect('mongodb://localhost/userdb', function(err) {

if (err) { throw err; }

console.log("vous etes connectez");

});

1. **CRUD : Create, Read, Update and Delete**

****

**Afficher document : créer fichier find.js**

const express = require('express')

const app = express()

//Import the mongoose module

var mongoose = require('mongoose');

var bodyParser = require('body-parser')

// parse application/x-www-form-urlencoded

app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json

app.use(bodyParser.json())

//connect mongoose vers mongodb

mongoose.connect('mongodb://localhost/userdb', function(err) {

if (err) { throw err; }

console.log("vous etes connectez");

});

app.get('/', function (req, res) {

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

// find one document /afficher un document

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("userdb");

dbo.collection("users").findOne({}, function(err, result) {

if (err) throw err;

db.close();

res.send(result);

});

});

// find all documents /afficher

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("userdb");

dbo.collection("users").find({}).toArray(function(err, result) {

if (err) throw err;

console.log(result);

db.close();

res.send(result);

});

});

});

app.listen(3000, function () {

console.log('Example app listening on port 3000!')

})

**Insertion de document : créer fichier insert.js**

//insérer un document par nodejs

app.post('/', function (req, res){

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("userdb");

var myobj = { name: "saber", lastname: "jdidi", email: "saber@gmail.com", password: "saber1234" , Todo:[{title: "aaa", description:"gfdhvh", done:"false", creationdate:"03/12/2018"}]};

dbo.collection("users").insertOne(myobj, function(err, res) {

if (err) throw err;

console.log("1 document inserted");

db.close();

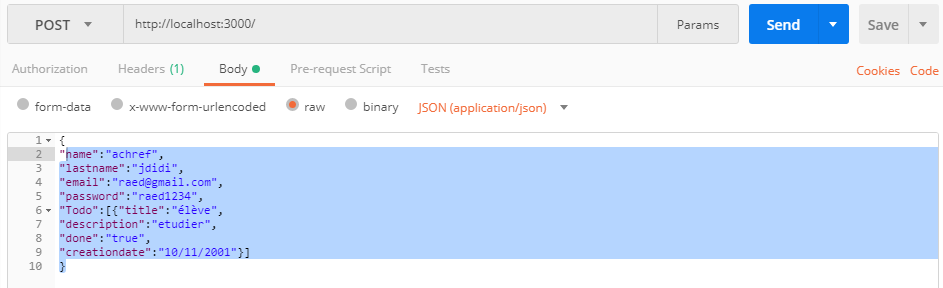
res.send(res);

});

});

})

//insérer un document à partir de postman et nodejs :



app.post('/', function (req, res){

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("userdb");

var myobj = { name: req.body.name, lastname: req.body.lastname, email: req.body.email, password: req.body.password, Todo:[{title: req.body.title, description:req.body.description, done:req.body.done, creationdate:req.body.creationdate}]};

dbo.collection("users").insertOne(myobj, function(err, res) {

if (err) throw err;

console.log("1 document inserted");

db.close();

});

});

res.send("name : " + req.body.name +"\n lastname : " + req.body.lastname);

})

Insert multiple documents in the "customers" collection:

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myobj = [  
    { name: 'John', address: 'Highway 71'},  
    { name: 'Peter', address: 'Lowstreet 4'},  
    { name: 'Amy', address: 'Apple st 652'},  
    { name: 'Viola', address: 'Sideway 1633'}  
  ];  
  dbo.collection("customers").insertMany(myobj, function(err, res) {  
    if (err) throw err;  
    console.log("Number of documents inserted: " + res.insertedCount);  
    db.close();  
  });  
});

**Suppression du document : créer fichier delete.js**

//supprimer un document

app.get('/', function (req, res){

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("userdb");

var myquery = { name: 'achref' };

dbo.collection("users").deleteOne(myquery, function(err, obj) {

if (err) throw err;

console.log("1 document deleted");

db.close();

res.send(obj)

});

});

})

Delete all documents were the address starts with the letter "O":

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
**var myquery = { address: /^O/ };**  dbo.collection("customers").deleteMany(myquery, function(err, obj) {  
    if (err) throw err;  
    console.log(obj.result.n + " document(s) deleted");  
    db.close();  
  });  
});

**Modification du document : créer fichier update.js**

//modifier un document

app.get('/', function (req, res){

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("userdb");

var myquery = { name: "ahmed" };

var newvalues = { $set: {name: "amine" , email: "amin@gmail.com", Todo:[{title: "patisserie", description: "restaurant", done: "true"}] } };

dbo.collection("users").updateOne(myquery, newvalues, function(err, res) {

if (err) throw err;

console.log("1 document updated");

db.close();

});

});

})

Update the document with the address "Valley 345" to name="Mickey" and address="Canyon 123":

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myquery = { address: "Valley 345" };  
  var newvalues = { $set: {name: "Mickey", address: "Canyon 123" } };  
  dbo.collection("customers").updateOne(myquery, newvalues, function(err, res) {  
    if (err) throw err;  
    console.log("1 document updated");  
    db.close();  
  });  
});

Update the address from "Valley 345" to "Canyon 123":

...  
  var myquery = { address: "Valley 345" };  
  var newvalues = { **$set: { address: "Canyon 123" }** };  
  dbo.collection("customers").updateOne(myquery, newvalues, function(err, res) {  
...

Update all documents where the name starts with the letter "S":

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myquery = { address: /^S/ };  
  var newvalues = {$set: {name: "Minnie"} };  
  dbo.collection("customers").updateMany(myquery, newvalues, function(err, res) {  
    if (err) throw err;  
    console.log(res.result.nModified + " document(s) updated");  
    db.close();  
  });  
});

1. **Authentification node.js et MongoDB avec session : Register, Login et Logout :**

**Créer fichier User.js**

const express = require('express')

const app = express()

var mongoose = require('mongoose');

var userSchema = new mongoose.Schema({

username: {type:String, unique: true},

password: {type: String},

firstname: String,

lastname: String

});

var User = mongoose.model('User', userSchema);

module.exports = User; //exporter User vers authentication.js

**Créer fichier authentication.js:**

Installer package express-session : **npm install express-session --save** (permet de lancer notre application par login)

const express = require('express')

const app = express()

//installer express-session: cmd>npm install express-session --save

var session = require('express-session');

//Import the mongoose module

var mongoose = require('mongoose');

//var router = express.Router(); routage entre plusieurs pages

//Import User module

var User = require('./User'); //utilise fichier User.js

var bodyParser = require('body-parser')

app.use(bodyParser.urlencoded({ extended: false }))

app.use(bodyParser.json())

//utilise session

app.use(session({secret:"ashd325bnjh657ezgjdbjdu1",resave:false,saveUninitialized:true}));

mongoose.connect('mongodb://localhost/userdb', function(err) {

if (err) { throw err; }

console.log("vous etes connectez");

});

//authentification

app.get('/', function(req, res, next){

res.render('authentication', { title: 'Express' });

});

app.post('/login', function(req, res){

var username = req.body.username;

var password = req.body.password;

User.findOne({username: username, password: password}, function(err, user){

if(err){

console.log(err);

return res.send('error');

}

if(!user){

return res.send('not user');

}

//ajouter session pour entrer avec login

req.session.user = user;

return res.send(user);

console.log('succès');

})

});

app.get('/home', function(req, res){

if(!req.session.user) {

return res.send('not loggin')

}

return res.send('welcome to home with login !')

});

app.get('/logout', function(req, res){

req.session.destroy();

return res.send('logout');

});

app.post('/register', function(req, res){

var username = req.body.username;

var password = req.body.password;

var firstname = req.body.firstname;

var lastname = req.body.lastname;

var newuser = new User();

newuser.username = username;

newuser.password = password;

newuser.firstname = firstname;

newuser.lastname = lastname;

newuser.save(function(err, savedUser) {

if(err) {

console.log(err);

return res.send('error');

}

return res.send('succes');

})

});

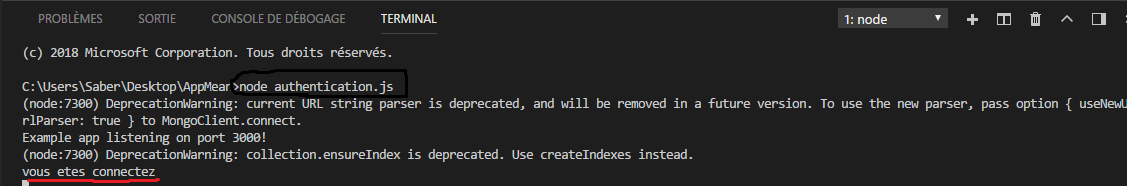
//module.exports = router;

app.listen(3000, function () {

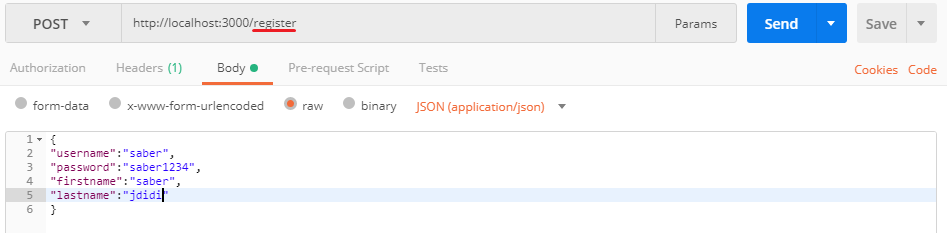
console.log('Example app listening on port 3000!')

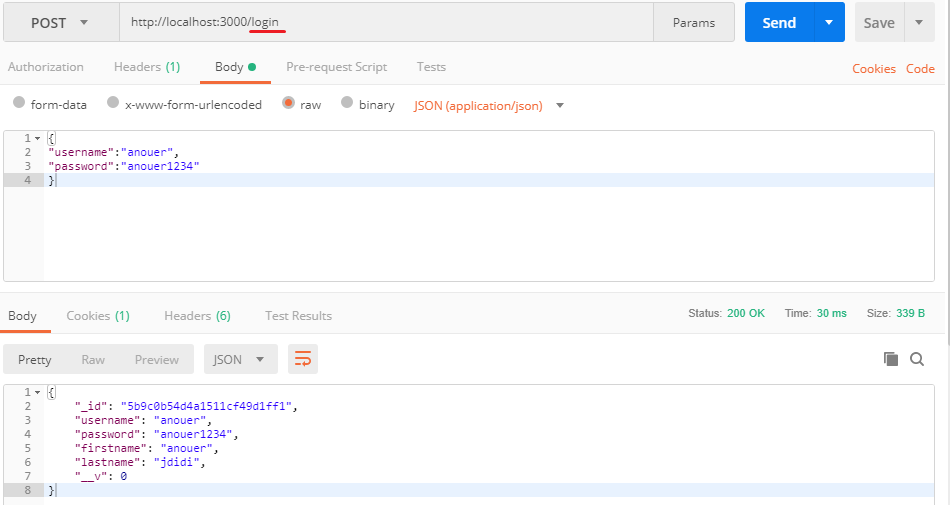
})

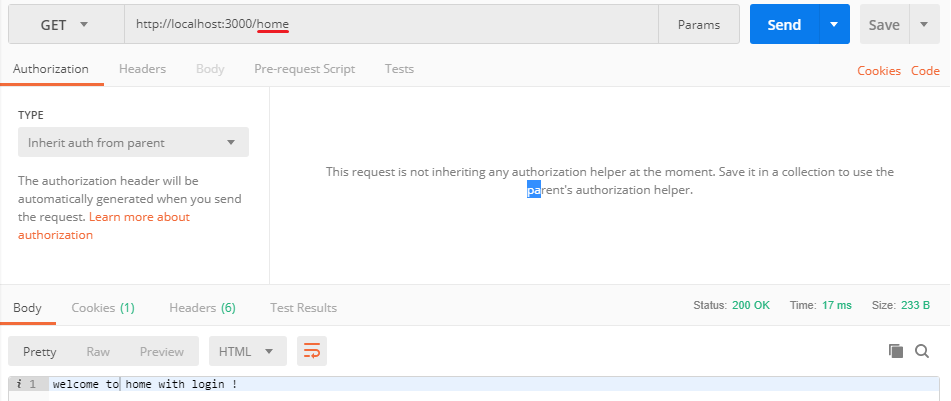
**Lancer serveur :**

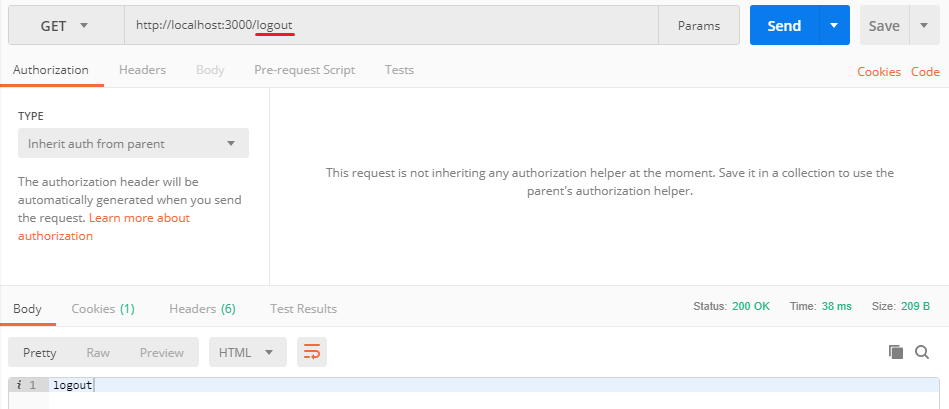
****

**Tester avec Postman :**

****

****

****

****

1. **Application structurée CRUD et authentification :**

**Exemple de document dans une collection :**

{

"name": "saber",

"lastname": "jdidi",

"email": "saber@gmail.com",

"password": "1234saber",

"todo": [

{

"date": "2018-09-20T11:43:39.946Z",

"Done": false,

"title": "Amine"

},

{

"date": "2018-09-20T15:31:45.686Z",

"Done": false,

"title": "informatique"

}

]

}

**Créer server.js**

const express = require("express");

const mongoClient = require("mongodb").MongoClient;

const ObjectId = require("mongodb").ObjectId;

const bodyparser = require("body-parser");

const app = express();

var authRouter = require('./server/routes/auth')

var todoRouter = require('./server/routes/todo')

app.use(bodyparser.json());

app.use('/auth', authRouter)

app.use('/todo', todoRouter)

const connection = (closure) => {

mongoClient.connect("mongodb://localhost:27017/appTodoDB", (err,client)=>{

if(err) throw err;

let db = client.db("appTodoDb");

closure(db);

})

}

//find documents

app.get("/users",(req,res)=>{

connection(async(db) => {

const result = await db.collection("users").find().toArray();

res.send(result);

})

})

//find document by id

app.get("/users/:id",(req,res)=>{

connection(async(db) => {

const result = await db.collection("users").findOne({\_id:ObjectId(req.params.id)});

res.send(result);

})

})

//insert document

app.post("/users", (req,res)=>{

connection(async(db)=>{

const result = await db.collection("users").insert(req.body);

res.send(result);

})

})

//update document

app.put("/users/:id",(req,res)=>{

connection(async(db) => {

const result = await db.collection("users").update({\_id:ObjectId(req.params.id)},{$set:req.body} );

res.send(result);

})

})

//delete document

app.delete("/users/:id",(req,res)=>{

connection(async(db) => {

const result = await db.collection("users").deleteOne({\_id:ObjectId(req.params.id)});

res.send(result);

})

})

//update by name

app.put("/users/:name",(req,res)=>{

connection(async(db) => {

const result = await db.collection("users").update({name:req.params.name},{$set:req.body} );

res.send(result);

})

})

app.listen(3000);

**Créer modèle user avec mongoose :**

**server (dossier) / models (dossier) /user.js :**

const mongoose = require('mongoose');

const user = new mongoose.Schema({

name: String,

lastname: String,

email: {

type: String,

unique: true, //utilise un email une seule fois

required: true,

},

password: String,

todo: [{

title: String,

date: {

type: Date,

default: Date.now(),

},

Done: {type: Boolean, default: false}

}]

});

module.exports = user;

* **Créer authentification pour l’application avec jwt :**

**Installer jwt (json web token) : npm install jsonwebtoken**

**Installer bcrypt : npm install bcrypt**

**S’il y a erreur installer : npm install bcrypt-nodejs**

**Site jwt : https://www.npmjs.com/package/jsonwebtoken**

**server / routes/ auth.js**

const router = require('express').Router();

const mongoose = require('mongoose');

const jwt = require('jsonwebtoken'); //utilise pour authentification

const bcrypt = require('bcrypt'); //faire hashage de mot de passe

mongoose.connect('mongodb://localhost:27017/appTodoDb', {

useNewUrlParser: true,

useCreateIndex: true,

});

const userSchema = require('../models/user')

const userModel = mongoose.model('users', userSchema); // import model

//Register user

router.post('/register', async (req, res) => {

req.body.password = bcrypt.hashSync(req.body.password, 10);

const result = await userModel.create(req.body).then().catch(err => {

res.send(err);

return;

});

res.send(result);

});

// login with jsonwebtoken (jwt)

router.post('/login', async (req, res) => {

resultLogin = await userModel.findOne({email:req.body.email});

if(!resultLogin){

res.send({message:'user not found'})

}

if(!bcrypt.compareSync(req.body.password,resultLogin.password ) ){

res.send({message:'bad password'})

}

// resultLogin.password = '';// pour masquer password

const token = jwt.sign({data:resultLogin},'secret\_code');

res.send({message:'ok', usertoken : token});

});

// just for testing

router.get('/register', async (req, res) => {

const result = await userModel.find();

res.send(result);

});

module.exports = router;

* **Create CRUD array (todo) in document (user) :**

**server / routes / todo.js :**

const router = require('express').Router();

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/appTodoDb', {

useNewUrlParser: true,

useCreateIndex: true,

});

const userSchema = require('../models/user')

const userModel = mongoose.model('users', userSchema); // import model

// find todo by user id

router.get('/:id', async (req, res) => {

const result = await userModel.findOne({\_id : req.params.id});

res.send(result.todo);

})

// inserer todo by user id

router.post('/:id', async (req, res)=>{

const result = await userModel.update({\_id: req.params.id},

{ $push:{

"todo": req.body

} } );

res.send(result);

})

// update todo by user id and index

router.put('/:id/:indextodo', async (req, res) => {

const result = await userModel.updateOne({\_id : req.params.id}, {$set : {["todo."+req.params.indextodo] : req.body}} );

res.send(result);

})

// delete todo by user id and index

router.delete('/:id/:indextodo', async (req, res) => {

/// methode 1

const todos = await userModel.findOne({\_id : req.params.id});

const result = await userModel.update({\_id: req.params.id}, {$pull: {todo : todos.todo[req.params.indextodo]}});

///methode 2

//const result = await userModel.update({\_id: req.params.id}, {$unset: {["todo."+req.params.indextodo]: 0000}});

//const result2 = await userModel.update({\_id: req.params.id}, {$pull: {"todo": null}}); //supprimer un element avec une condition

res.send(result);

})

module.exports = router;

**Application blog CRUD et authentification partie Backend**

Installer les packages :

Node : **npm init** (installer package.json)

**npm install express body-parser mongoose jsonwebtoken bcrypt --save**

3 collections : user, article et commentaire

**server.js** (fichier principale)

const express = require('express');

const app = express();

const bodyparser = require('body-parser');

const port = 3000;

app.use(bodyparser.json());

const authRouter = require('./Server/Routes/auth');

app.use('/auth', authRouter);

const blogRouter = require('./Server/Routes/blog');

app.use('/blog', blogRouter);

app.listen(port, err => {

console.log(`Connect with port ${port}`)

})

server / models / **user.js**

const mongoose = require('mongoose');

const user = new mongoose.Schema({

name : String,

lastname : String,

email : {

type : String,

unique : true,

required : true

},

password : String

});

module.exports = user;

server / models / **article.js**

const mongoose = require('mongoose');

const article = new mongoose.Schema({

title: String,

content: String,

creationDate: {

type: Date,

default: Date.now()

},

author: {

type: mongoose.Schema.Types.ObjectId,

ref: 'user'

}, // author est un objet dans article

comments: [{

type: mongoose.Schema.Types.ObjectId,

ref: 'comment'

}] // comments de type array parce que est un array dans article

});

module.exports = article ;

server / models / **comment.js**

const mongoose = require('mongoose');

const comment = new mongoose.Schema({

content: String,

creationDate: {

type: Date,

default: Date.now()

},

author: {

type: mongoose.Schema.Types.ObjectId,

ref: 'user',

required: true //author est un element obliger dans comment

} // ce commentaire pour quel auteur

});

module.exports = comment;

server / routes / **auth.js**

const router = require ('express').Router();

const mongoose = require ('mongoose');

const jwt = require ('jsonwebtoken');

const bcrypt = require ('bcrypt'); // si bcrypt ne fonctionne pas utilise bcrypt-nodejs

mongoose.connect('mongodb://localhost:27017/blogdb',{

useNewUrlParser: true,

useCreateIndex: true,

});

const userSchema = require('../models/user');

const userModel = mongoose.model('user', userSchema);

router.post('/register', async(req, res) =>{

req.body.password = await bcrypt.hashSync(req.body.password);

const result = await userModel.create(req.body);

res.send(result);

})

// login with jsonwebtoken (jwt)

router.post('/login', async(req, res) => {

const resultlogin = await userModel.findOne({email: req.body.email});

if(!resultlogin){res.send('address mail not existing');}

if(!bcrypt.compareSync(req.body.password, resultlogin.password)){res.send('wrong password');}

const token = jwt.sign({data:resultlogin}, 'shhhhhhhh');

res.send({message:'ok', usertoken : token});

})

// find users

router.get('/registered', async(req, res) =>{

const result = await userModel.find();

res.send(result);

})

module.exports = router;

server / routes / **blog.js**

const router = require ('express').Router();

const mongoose = require ('mongoose');

mongoose.connect('mongodb://localhost:27017/blogdb',{

useNewUrlParser: true,

useCreateIndex: true,

});

const articleSchema = require('../models/article');

const articleModel = mongoose.model('article', articleSchema);

const commentSchema = require('../models/comment');

const commentModel = mongoose.model('comment', commentSchema);

// add article

router.post('/article', async(req,res)=>{

const result = await articleModel.create(req.body);

res.send(result);

});

// get all articles

router.get('/article', async(req,res)=>{

const result = await articleModel.find().populate({path: 'author'}).populate({path: 'comments'}); //(populate) retourner

les donneés de l'user et de comment par l’id

res.send(result);

});

// get article by id

router.get('/article/:id', async(req,res)=>{

const result = await articleModel.findById({\_id: req.params.id}).populate({path: 'author'}).populate({path: 'comments'});

res.send(result);

});

// get aricles by userId

router.get('/user/:userid', async(req,res)=>{

const result = await articleModel.findById({auther : req.params.userid}).populate({path: 'author'}).populate({path: 'comments'});

res.send(result);

});

// update article by id

router.put('/article/:id', async(req,res)=>{

const result = await articleModel.updateOne({\_id : req.params.id}, {$set : req.body});

res.send(result);

});

// remove article by id

router.delete('/article/:id', async(req,res)=>{

const result = await articleModel.remove({\_id: req.params.id});

res.send(result);

});

// add commentaire a un article par id article

router.post('/comment/:idArticle', async(req,res)=>{

const result = await commentModel.create(req.body);

const result2 = await articleModel.updateOne({\_id:req.params.idArticle}, {$push:{comments: result.\_id}})

res.send(result2);

});

// delete commentaire

router.delete('/comment/:id', async(req,res)=>{

const result = await commentModel.remove({\_id:req.params.id});

res.send(result);

});

module.exports = router;

**Angular : la partie Frontend**

**Installation Angular :**

npm uninstall -g @angular/cli //désinstaller

npm cache clean

npm cache verify

npm install -g @angular/cli@latest //la dernière version

ou npm install -g @angular/cli

**Créer application : ng new nom d’application**

**Lancer serveur de notre application : ng serve**

**Application Todo (MEAN Stack) Backend et Frontend :**

* Utiliser notre partie Frontend par Angular :
* Copier partie Backend sous le projet et **npm install** par notre projet.
* Index.html : metter css et js.
* Creation 4 components (home, menu, login, register) et un service
* Il faut ouvrir les 2 serveurs (l’un pour angular (ng serve) et l’autre pour backend (node server.js))
* Ajouter à **server.js** cors pour faire communication entre les 2 serveurs :

const express = require('express');

const port = 3000;

const app = express();

//faire communication entre serveur angular (frontend) et serveur backend

app.use(function(req, res, next) {

res.header("Access-Control-Allow-Origin", "\*");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

res.header("Access-Control-Allow-Methods", "PUT,POST,GET,DELETE,OPTIONS"); //pour résoudre problème des méthodes

next();

});

const bodyparser = require('body-parser');

app.use(bodyparser.json());

const auth = require('./Server/Routes/auth');

app.use('/auth', auth);

const todo = require('./Server/Routes/todo');

app.use('/todo',todo);

app.listen(port, err => {

console.log(`hani namsa3 fik 3al ${port}`)

})

**server (dossier) / models (dossier) /user.js :**

const mongoose = require('mongoose');

const user = new mongoose.Schema({

name: String,

lastname: String,

email: {

type: String,

unique: true, //utilise un email une seule fois

required: true,

},

password: String,

todo: [{

title: String,

date: {

type: Date,

default: Date.now(),

},

Done: {type: Boolean, default: false}

}]

});

module.exports = user;

**server / routes / todo.js :**

const router = require('express').Router();

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/appTodoDb', {

useNewUrlParser: true,

useCreateIndex: true,

});

const userSchema = require('../models/user')

const userModel = mongoose.model('users', userSchema); // import model

// find todo by user id

router.get('/:id', async (req, res) => {

const result = await userModel.findOne({\_id : req.params.id});

res.send(result.todo);

})

// inserer todo by user id

router.post('/:id', async (req, res)=>{

const result = await userModel.update({\_id: req.params.id},

{ $push:{

"todo": req.body

} } );

res.send(result);

})

// update todo by user id and index

router.put('/:id/:indextodo', async (req, res) => {

const result = await userModel.updateOne({\_id : req.params.id}, {$set : {["todo."+req.params.indextodo] : req.body}} );

res.send(result);

})

// delete todo by user id and index

router.delete('/:id/:indextodo', async (req, res) => {

/// methode 1

const todos = await userModel.findOne({\_id : req.params.id});

const result = await userModel.update({\_id: req.params.id}, {$pull: {todo : todos.todo[req.params.indextodo]}});

///methode 2

//const result = await userModel.update({\_id: req.params.id}, {$unset: {["todo."+req.params.indextodo]: 0000}});

//const result2 = await userModel.update({\_id: req.params.id}, {$pull: {"todo": null}}); //supprimer un element avec une condition

res.send(result);

})

module.exports = router;

**index.html :**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>TodoApp</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link href="https://blackrockdigital.github.io/startbootstrap-logo-nav/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- <link href="https://blackrockdigital.github.io/startbootstrap-logo-nav/css/logo-nav.css" rel="stylesheet"> -->

<!-- https://blackrockdigital.github.io/startbootstrap-logo-nav/ -->

</head>

<body>

<app-root></app-root>

<!-- here -->

</body>

</html>

**app.Module.ts :**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { HomeComponent } from './home/home.component';

import { MenuComponent } from './menu/menu.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { RouterModule, Routes } from '@angular/router';

import { LoginComponent } from './login/login.component';

import { HttpModule } from '@angular/http';

import { RegisterComponent } from './register/register.component';

const routes: Routes = [

{ path: 'home', component: HomeComponent },

{ path: 'login', component: LoginComponent },

{ path: 'register', component: RegisterComponent },

{ path: '', pathMatch: 'full', redirectTo: 'login' }

];

// decorator

@NgModule({ // metadata

declarations: [

AppComponent,

HomeComponent,

MenuComponent,

LoginComponent,

RegisterComponent

],

imports: [

BrowserModule,

FormsModule,

ReactiveFormsModule,

HttpModule,

RouterModule.forRoot(routes),

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.html**

<router-outlet></router-outlet>

<!-- <app-menu></app-menu> -->

<!-- appel component -->

<!-- <app-home></app-home> -->

Src/app/**api.service.ts** (créer service)

import { Injectable } from '@angular/core';

import { Http } from '@angular/http';

@Injectable({

providedIn: 'root'

})

export class ApiService {

constructor(private http: Http) { }

loginApi(form) {

return this.http.post('http://localhost:3000/auth/login', form);

}

registerApi(form) {

return this.http.post('http://localhost:3000/auth/register', form);

}

getTodos(id) {

return this.http.get('http://localhost:3000/todo/' + id);

}

postTodos(id, form) {

return this.http.post('http://localhost:3000/todo/' + id, form);

}

deleteTodo(id, i) {

return this.http.delete('http://localhost:3000/todo/' + id + '/' + i);

}

editTodo(id, i, todo) {

return this.http.put(`http://localhost:3000/todo/${id}/${i}`, todo)

}

}

**menu.component.html**

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav ml-auto">

<li class="nav-item active">

<a class="nav-link" routerLink="/home">Home

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" routerLink="/about">About</a>

</li>

<li class="nav-item active">

<a class="nav-link" href="javascript:void(0)" (click)="logoutBtn()">Logout</a>

</li>

</ul>

</div>

</div>

</nav>

**menu.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

@Component({

selector: 'app-menu',

templateUrl: './menu.component.html',

styleUrls: ['./menu.component.css']

})

export class MenuComponent implements OnInit {

constructor(private router: Router) { }

ngOnInit() {

console.log('hello from menu')

}

logoutBtn() {

localStorage.clear();

this.router.navigateByUrl('/login');

}

}

src/app/home : [**home.component.html**](http://home.component.html)

<app-menu></app-menu> //utilise component menu sous component home

<div class="container">

<h3 class="pb-3 mb-4 font-italic border-bottom">

Card Styles

</h3>

<br>

<br>

<br>

<br>

<form [formGroup]="addForm">

<input type="text" formControlName="title" placeholder="Ajout Todo"> <button type="button" (click)="onAddClick()"

class="btn btn-outline- btn-sm"><i class="fa fa-plus" aria-hidden="true"></i></button>

</form>

<br>

<br>

<br>

<br>

<div class="row">

<div class="col-md-6">

<div class="card flex-md-row mb-4 shadow-sm h-md-250" \*ngFor="let todo of todos; let i = index">

<div class="card-body d-flex flex-column align-items-start">

<strong \*ngIf="toUpdate !== i" class="d-inline-block mb-2 text-primary" (click)="toUpdate = i">{{i}} -

{{todo.title}}</strong>

<input \*ngIf="toUpdate === i" type="text" [(ngModel)]="todo.title">

<h6 class="mb-0">

<a class="text-dark" href="#"></a>

</h6>

<div class="mb-1 text-muted small"></div>

<div>

Done <input type="checkbox" [checked]="todo.Done" (change)="doneChecked(i)" />

</div>

<i \*ngIf="todo.Done" class="fas fa-check"></i><!-- font awesome-->

<p class="card-text mb-auto"> Date : {{todo.date}} </p>

<br>

<button class="btn btn-outline-danger btn-sm" role="button" href="#" (click)="deleteTodoBtn(i)"><i class="fa fa-trash" aria-hidden="true"></i></button>

<button class="btn btn-outline-primary btn-sm" \*ngIf="toUpdate === i" (click)="updateTitle(i, todo.title);toUpdate=-1 "><i class="fa fa-upload" aria-hidden="true"></i></button>

</div>

</div>

</div>

</div>

</div>

[**home.component.ts**](http://home.component.ts)

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import \* as jwt\_decode from "jwt-decode"; // installer jwt-decode : // npm install jwt-decode

@Component({

selector: 'app-home',

templateUrl: './home.component.html',

styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

addForm: FormGroup;

todos = [];

userId = '';

toUpdate = -1;

constructor(private router: Router, private apiService: ApiService) {

const token = localStorage.getItem('token');

this.userId = jwt\_decode(token).data.\_id;

}

ngOnInit() {

console.log(this.userId);

this.addForm = new FormGroup({

title: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

});

this.apiService.getTodos(this.userId).subscribe(res => {

console.log(res.json());

this.todos = res.json();

})

}

onAddClick() {

if (this.addForm.valid) {

this.apiService.postTodos(this.userId, this.addForm.value).subscribe(res => {

this.ngOnInit();

});

}

}

deleteTodoBtn(i) {

this.apiService.deleteTodo(this.userId, i).subscribe(res => {

this.ngOnInit();

})

}

doneChecked(i) {

const todo = this.todos[i];

todo.Done = !todo.Done;

this.apiService.editTodo(this.userId, i, todo).subscribe(res => {

this.ngOnInit();

})

}

updateTitle(i, title) {

const todo = this.todos[i];

todo.title = title;

this.apiService.editTodo(this.userId, i, todo).subscribe(res => {

this.ngOnInit();

})

}

}

**login.component.css**

.container-body {

background-image: url("https://hdwallsource.com/img/2014/9/blur-26347-27038-hd-wallpapers.jpg");

background-repeat: no-repeat;

background-position: center;

background-size: cover;

padding: 10px;

height: 100vh;

}

.form-heading {

color: #fff;

font-size: 23px;

}

.panel h2 {

color: #444444;

font-size: 18px;

margin: 0 0 8px 0;

}

.panel p {

color: #777777;

font-size: 14px;

margin-bottom: 30px;

line-height: 24px;

}

.login-form .form-control {

background: #f7f7f7 none repeat scroll 0 0;

border: 1px solid #d4d4d4;

border-radius: 4px;

font-size: 14px;

height: 50px;

line-height: 50px;

}

.main-div {

background: #ffffff none repeat scroll 0 0;

border-radius: 2px;

margin: 10px auto 30px;

max-width: 38%;

padding: 50px 70px 70px 71px;

}

.login-form .form-group {

margin-bottom: 10px;

}

.login-form {

text-align: center;

}

.forgot a {

color: #777777;

font-size: 14px;

text-decoration: underline;

}

.login-form .btn.btn-primary {

background: #f0ad4e none repeat scroll 0 0;

border-color: #f0ad4e;

color: #ffffff;

font-size: 14px;

width: 100%;

height: 50px;

line-height: 50px;

padding: 0;

}

.forgot {

text-align: left;

margin-bottom: 30px;

}

.botto-text {

color: #ffffff;

font-size: 14px;

margin: auto;

}

.login-form .btn.btn-primary.reset {

background: #ff9900 none repeat scroll 0 0;

}

.back {

text-align: left;

margin-top: 10px;

}

.back a {

color: #444444;

font-size: 13px;

text-decoration: none;

}

body {

padding-top: 0px;

}

@media (min-width: 992px) {

body {

padding-top: 0px;

}

}

**login.component.html**

<div class="container-body">

<div class="container">

<h1 class="form-heading"></h1>

<div class="login-form">

<div class="main-div">

<div class="panel">

<h2>Admin Login</h2>

<p>Please enter your email and password</p>

</div>

<form [formGroup]="loginForm">

<div class="form-group">

<input type="email" formControlName="email" class="form-control" id="inputEmail" placeholder="Email Address">

</div>

<div class="form-group">

<input type="password" formControlName="password" class="form-control" id="inputPassword" placeholder="Password">

</div>

<div class="forgot">

<!-- <a href="reset.html">Forgot password?</a> -->

</div>

<button type="submit" class="btn btn-primary" (click)="loginBtn()">Login</button>

<div class="alert alert-danger" \*ngIf="message">

<strong>Error !</strong> {{message}}

</div>

{{ loginForm.valid | json}}

</form>

</div>

<!-- <p class="botto-text"> Designed by Sunil Rajput</p> -->

</div>

</div>

</div>

**login.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

// recuperation login forms

// {email: 'aaa@bb.cc', password: '123'}

email = '';

password = '';

reg = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

constructor(private router: Router, private apiService: ApiService) { }

message = '';

ngOnInit() {

}

loginBtn() {

// console.log('{ email:' + ' ' + this.email + '\n' + 'password:' + ' ' + this.password + '}');

let chaine1;

if (this.reg.test(this.email)) {

chaine1 = 'test mail valide';

} else {

chaine1 = 'test non valide';

}

console.log(chaine1);

const myObj = { email: this.email, password: this.password };

console.log(myObj);

this.message = '';

this.apiService.loginApi(myObj).subscribe(res => {

console.log(res.json());

if (res.json().message === 'ok') {

this.router.navigateByUrl('/home');

} else {

this.message = res.json().message;

}

});

// send to backend for verification

// if OKAY then go to home

}

}

La méthode la plus sécurisé et simple **login.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

loginForm: FormGroup;

message = '';

constructor(private router: Router, private apiService: ApiService) { }

ngOnInit() {

this.loginForm = new FormGroup({

email: new FormControl('', [Validators.email, Validators.required]),

password: new FormControl('', [Validators.required, Validators.minLength(8)]),

});

}

loginBtn() {

if (this.loginForm.valid) {

this.message = '';

this.apiService.loginApi(this.loginForm.value).subscribe(res => {

console.log(res.json());

if (res.json().message === 'ok') {

localStorage.setItem('token', res.json().usertoken)

this.router.navigateByUrl('/home');

} else {

this.message = res.json().message;

}

});

}

}

}

**register.component.html**

<div class="container-body">

<div class="container">

<h1 class="form-heading"></h1>

<div class="login-form">

<div class="main-div">

<div class="panel">

<h2>Admin Login</h2>

<p>Please enter your email and password</p>

</div>

<form [formGroup]="registerForm">

<div class="form-group">

<input type="name" formControlName="name" class="form-control" id="inputname" placeholder="name">

</div>

<div class="form-group">

<input type="lastname" formControlName="lastname" class="form-control" id="inputlastname" placeholder="lastname">

</div>

<div class="form-group">

<input type="email" formControlName="email" class="form-control" id="inputEmail" placeholder="Email Address">

</div>

<div class="form-group">

<input type="password" formControlName="password" class="form-control" id="inputPassword" placeholder="Password">

</div>

<div class="forgot">

<!-- <a href="reset.html">Forgot password?</a> -->

</div>

<button type="submit" class="btn btn-primary" (click)="registerBtn()">Register</button>

<div class="alert alert-danger" \*ngIf="message">

<strong>Error !</strong> {{message}}

</div>

<!-- {{ registerForm.value | json}} -->

{{ registerForm.valid | json}}

</form>

</div>

<!-- <p class="botto-text"> Designed by Sunil Rajput</p> -->

</div>

</div>

</div>

**register.component.css**

.container-body {

background-image: url("https://hdwallsource.com/img/2014/9/blur-26347-27038-hd-wallpapers.jpg");

background-repeat: no-repeat;

background-position: center;

background-size: cover;

padding: 10px;

height: 100vh;

}

.form-heading {

color: #fff;

font-size: 23px;

}

.panel h2 {

color: #444444;

font-size: 18px;

margin: 0 0 8px 0;

}

.panel p {

color: #777777;

font-size: 14px;

margin-bottom: 30px;

line-height: 24px;

}

.login-form .form-control {

background: #f7f7f7 none repeat scroll 0 0;

border: 1px solid #d4d4d4;

border-radius: 4px;

font-size: 14px;

height: 50px;

line-height: 50px;

}

.main-div {

background: #ffffff none repeat scroll 0 0;

border-radius: 2px;

margin: 10px auto 30px;

max-width: 38%;

padding: 50px 70px 70px 71px;

}

.login-form .form-group {

margin-bottom: 10px;

}

.login-form {

text-align: center;

}

.forgot a {

color: #777777;

font-size: 14px;

text-decoration: underline;

}

.login-form .btn.btn-primary {

background: #f0ad4e none repeat scroll 0 0;

border-color: #f0ad4e;

color: #ffffff;

font-size: 14px;

width: 100%;

height: 50px;

line-height: 50px;

padding: 0;

}

.forgot {

text-align: left;

margin-bottom: 30px;

}

.botto-text {

color: #ffffff;

font-size: 14px;

margin: auto;

}

.login-form .btn.btn-primary.reset {

background: #ff9900 none repeat scroll 0 0;

}

.back {

text-align: left;

margin-top: 10px;

}

.back a {

color: #444444;

font-size: 13px;

text-decoration: none;

}

body {

padding-top: 0px;

}

@media (min-width: 992px) {

body {

padding-top: 0px;

}

}

**register.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

@Component({

selector: 'app-register',

templateUrl: './register.component.html',

styleUrls: ['./register.component.css']

})

export class RegisterComponent implements OnInit {

registerForm: FormGroup;

message = '';

constructor(private router: Router, private apiService: ApiService) { }

ngOnInit() {

// angular reactive form

//controle de saisie

this.registerForm = new FormGroup({

name: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

lastname: new FormControl('', [Validators.required, Validators.maxLength(20)]),

email: new FormControl('', [Validators.email, Validators.required]),

password: new FormControl('', [Validators.required, Validators.minLength(8)]),

});

}

registerBtn() {

if (this.registerForm.valid) {

this.message = '';

this.apiService.registerApi(this.registerForm.value).subscribe(res => {

console.log(res.json());

if (res.json().\_id) {

this.router.navigateByUrl('/login');

} else {

this.message = res.json().message;

}

});

}

}

}

* Si en utilise jwt-decode, il faut installer : **npm install jwt-decode**

import \* as jwt\_decode from "jwt-decode";

importer dans component qu’utilise jwt-decode

Jwt\_decode utilisé juste pour l’utilisateur pour recupérer leur id

token se trouve dans locale storage.

**Application Blog :**

**Créer projet MEAN Stack : ng new ApplicationBlog**

**Partie Backend :** utilise partie backend dans Angular

**Installer express : npm install express**

**Installer body-parser : npm install body-parser**

**Installer mongoose : npm install mongoose**

**Installer nodemon : npm install nodemon**

**Installer jwt (json web token) : npm install jsonwebtoken**

**Installer bcrypt : npm install bcrypt ou npm install bcrypt-nodejs**

**Installer jwt-decode : npm install jwt-decode**

**Si en utilise socket : installer socket : npm install socket.io**

**Server.js**

const express = require('express');

const app = express();

const bodyparser = require('body-parser');

const port = 3000;

// utilise http si en utilise socket

var http = require('http').Server(app);

var io = require('socket.io')(http);

//socket afficher users connecter dans console

io.on('connection', function(socket) {

console.log('user connect');

})

//donne l'accès au socket par router express

app.use((req, res, next) => {

req.io = io

next()

})

// interconnexion entre angular et node.js par cors

app.use(function(req, res, next) {

res.header("Access-Control-Allow-Origin", "\*");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

res.header("Access-Control-Allow-Methods", "PUT,POST,GET,DELETE,OPTIONS"); //pour résoudre prb des méthodes

next();

});

app.use(bodyparser.json());

const auth = require('./server/routes/auth');

app.use('/auth', auth);

const blog = require('./server/routes/blog');

app.use('/blog', blog);

// si en utilise express (app)

/\* app.listen(port, err => {

console.log(`connect with port ${port}`)

}) \*/

// si en utilise express et socket (http)

http.listen(port, err => {

console.log(`connect with port ${port}`)

})

**server/models/user.js**

const mongoose = require('mongoose');

const user = new mongoose.Schema({

name: String,

lastname: String,

email: {

type: String,

unique: true, //email utilise une seule fois

required: true

},

password: String,

access: { type: Boolean, default: false }, // access control, use here and

// in auth.js and user.component.html

role: { type: String, enum: ['user', 'admin'], default: 'user' } // role of user

});

module.exports = user;

**server/models/article.js**

const mongoose = require('mongoose');

const article = new mongoose.Schema({

title : String,

content : String,

creationdate : {

type : Date,

default : Date.now()

},

author : { //user créer cet article

type : mongoose.Schema.Types.ObjectId,

ref : 'users' // meme nom utilise par modèle mongoose

},

comments : [{ //commentaire de l'article il y a un ou plusieurs comments []

type : mongoose.Schema.Types.ObjectId,

ref : 'comment'

}]

});

module.exports = article;

**server/models/comment.js**

const mongoose = require('mongoose');

const comment = new mongoose.Schema({

content : String,

creationdate : {

type: Date,

default: Date.now()

},

author : { // user céer cette commentaire

type : mongoose.Schema.Types.ObjectId,

ref : 'users', // users utilisé par model mongoose

required : true

}

});

module.exports = comment ;

**server/routes/auth.js**

const router = require('express').Router();

const mongoose = require('mongoose');

const jwt = require('jsonwebtoken');

const bcrypt = require('bcrypt-nodejs')

mongoose.connect('mongodb://localhost:27017/blogAngular', {

useNewUrlParser: true,

useCreateIndex: true,

});

const userSchema = require('../models/user')

const userModel = mongoose.model('users', userSchema); // import model

router.post('/register', async (req, res) => {

req.body.password = await bcrypt.hashSync(req.body.password); // hashage de password

console.log(req.body);

const result = await userModel.create(req.body).then().catch(err => {

res.send(err);

return;

});

res.send(result);

});

// login with jsonwebtoken (jwt) faire cryptage et décryptage

router.post('/login', async (req, res) => {

resultLogin = await userModel.findOne({ email: req.body.email });

if (!resultLogin) {

res.send({ message: 'user not found' })

}

if (!bcrypt.compareSync(req.body.password, resultLogin.password)) {

res.send({ message: 'bad password' })

}

// resultLogin.password = '';// pour masquer pwd

//utiliser l'accès par true ou false, telque l'admin donne l'accès au user pour connecter ou non à l'application

if (!resultLogin.access) {

res.send({message: 'access denied !'})

}

//utilise token de jwt

const token = jwt.sign({ data: resultLogin }, 'secret\_code');

res.send({ message: 'ok', usertoken: token });

});

// get users

router.get('/register', async (req, res) => {

const result = await userModel.find();

res.send(result);

});

//delete user

router.delete('/register/:id', async (req, res) => {

const result = await userModel.remove({ \_id: req.params.id })

res.send(result)

});

//update user

router.put('/register/:id', async (req, res) => {

const result = await userModel.update({ \_id: req.params.id }, { $set: req.body });

res.send(result);

});

module.exports = router;

**server/routes/blog.js**

const router = require('express').Router(); //exporter blog

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/blogAngular',{

useNewUrlParser : true,

useCreateIndex : true,

});

const userSchema = require('../models/user');

const userModel = mongoose.model('users', userSchema);

const articleSchema = require('../models/article');

const articleModel = mongoose.model('article', articleSchema);

const commentSchema = require('../models/comment');

const commentModel = mongoose.model('comment', commentSchema);

// add article // on ajoute socket

router.post('/article', async(req, res) => {

const result = await articleModel.create(req.body);

req.io.emit('newarticle', result) // on utilise socket pour ajouter article dans ce ligne

res.send(result);

});

// find article

router.get('/article', async(req, res) => {

const result = await articleModel.find().populate({path : 'author'}).populate({path : 'comments'});

res.send(result)

});

// find article by id

router.get('/article/:id', async(req, res) => {

const result = await articleModel.findById({\_id : req.params.id}).populate({path : 'author'}).populate({path : 'comments'});

res.send(result)

});

// find article by id user

router.get('/users/:iduser', async(req, res) => {

const result = await articleModel.findById({author: req.params.iduser}).populate({path : 'author'}).populate({path : 'comments'});

res.send(result)

})

//update article with id

router.put('/article/:id', async(req, res) => {

const result = await articleModel.update({\_id: req.params.id}, {$set: req.body});

req.io.emit('newarticle', result) // on utilise socket pour update article dans ce ligne

res.send(result);

});

//remove article by id

router.delete('/article/:id', async(req, res) => {

const result = await articleModel.remove({\_id : req.params.id})

req.io.emit('newarticle', result) // on utilise socket pour delete article dans ce ligne

res.send(result)

});

//add commentaire

router.post('/comment/:idarticle', async(req, res) =>{

const result = await commentModel.create(req.body);

const result2 = await articleModel.updateOne({\_id: req.params.idarticle}, {$push: {comments : result.\_id }});

res.send(result2);

});

//delete commentaire

router.delete('/comment/:id', async(req, res) =>{

const result = await commentModel.remove({\_id : req.params.id});

res.send(result)

});

//find comment

router.get('/comment', async(req, res) => {

const result = await articleModel.find().populate({path : 'comments', populate:{path: 'author'}});

res.send(result);

});

module.exports = router;

**Partie Frontend :**

**installer core pour faire communication entre angular et node : npm install core-js**

**Si en utilise socket : installer socket : npm install ngx-socket-io**

**installer compat : npm install rxjs-compat**

**ajouter à polyfills.ts :**

(window as any).global = window;

**index.html**

<! doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>AngularBlog</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.3.1/css/all.css" integrity="sha384-mzrmE5qonljUremFsqc01SB46JvROS7bZs3IO2EmfFsd15uHvIt+Y8vEf7N7fWAU"

crossorigin="anonymous">

</head>

<body>

<app-root></app-root>

<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>

<script src="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"></script>

<!-- Bootstrap core JavaScript

================================================== -->

<!-- Placed at the end of the document so the pages load faster -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script>window.jQuery || document.write('<script src="../../assets/js/vendor/jquery-slim.min.js"><\/script>')</script>

<!-- Icons -->

<script src="https://unpkg.com/feather-icons/dist/feather.min.js"></script>

<script>

feather.replace()

</script>

</body>

</html>

**app.component.html**

<router-outlet></router-outlet>

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

import { LoginComponent } from './login/login.component';

import { RegisterComponent } from './register/register.component';

import { HomeComponent } from './home/home.component';

import { DashboardComponent } from './dashboard/dashboard.component';

import { RouterModule, Routes } from '@angular/router';

import { HttpModule } from '@angular/http';

import { MenuComponent } from './menu/menu.component';

import { AddArticleComponent } from './add-article/add-article.component';

import { ListArticleComponent } from './list-article/list-article.component';

import { UsersComponent } from './users/users.component';

import { AuthGuard } from './auth.guard'; // garde d’application

import { RoleGuard } from './role.guard'; // role des users

import { CommentComponent } from './comment/comment.component';

import { SocketIoModule, SocketIoConfig } from 'ngx-socket-io'; // import socket // config socket

const config: SocketIoConfig = { url: 'http://localhost:3000', options: {} };

// router utilisé

const routes: Routes = [

{ path: 'home', component: HomeComponent, canActivate: [AuthGuard] },

{ path: 'login', component: LoginComponent },

{ path: 'register', component: RegisterComponent },

{ path: 'comment', component: CommentComponent },

{ path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard, RoleGuard], data: { roles: ['admin'] } }, // garde et role sur dashboard

{ path: 'add', component: AddArticleComponent, canActivate: [AuthGuard] },

{ path: 'users', component: UsersComponent, canActivate: [AuthGuard] },

{ path: 'article/:idArticle', component: ListArticleComponent, canActivate: [AuthGuard] },

{ path: '', pathMatch: 'full', redirectTo: 'login' }

];

@NgModule({

declarations: [

AppComponent,

LoginComponent,

RegisterComponent,

HomeComponent,

DashboardComponent,

MenuComponent,

AddArticleComponent,

ListArticleComponent,

NavbarComponent,

UsersComponent,

CommentComponent

],

imports: [

BrowserModule,

FormsModule,

ReactiveFormsModule,

HttpModule,

RouterModule.forRoot(routes),

SocketIoModule.forRoot(config) //socket

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Les services** faire appel au **APIs** consommées dans Backend : **ng g s api** **:**

**api.service.ts**

import { Injectable } from '@angular/core';

import { Http } from '@angular/http';

import {Socket} from 'ngx-socket-io'; //importer socket

@Injectable({

providedIn: 'root'

})

export class ApiService {

constructor(private http: Http, private socket: Socket ) { }

loginApi(form) {

return this.http.post('http://localhost:3000/auth/login', form);

}

registerApi(form) {

return this.http.post('http://localhost:3000/auth/register', form);

}

getArticle() {

return this.http.get('http://localhost:3000/blog/article');

}

postArticle(form) {

return this.http.post('http://localhost:3000/blog/article', form);

}

deleteArticle(id) {

return this.http.delete('http://localhost:3000/blog/article/' + id);

}

editArticle(id, article) {

return this.http.put('http://localhost:3000/blog/article/' + id, article);

}

getarticleById(id) {

return this.http.get(`http://localhost:3000/blog/article/${id}`)

}

getUser() {

return this.http.get('http://localhost:3000/auth/register');

}

postUser(form) {

return this.http.post('http://localhost:3000/auth/register', form);

}

deleteUser(id) {

return this.http.delete('http://localhost:3000/auth/register/' + id);

}

updateUser(id, user) {

return this.http.put('http://localhost:3000/auth/register/' + id, user);

}

postcomment(id, form) { // id of article

return this.http.post('http://localhost:3000/blog/comment/' + id, form);

}

getcomment() {

return this.http.get('http://localhost:3000/blog/comment');

}

deletecomment(id) {

return this.http.delete('http://localhost:3000/blog/comment/' + id);

}

//API de socket tel que admin post article to all users

newArticle() {

return this.socket

.fromEvent("newarticle")

}

}

Component register : **ng g c register** **:**

**register.component.css**

.container-body {

background-image: url("https://hdwallsource.com/img/2014/9/blur-26347-27038-hd-wallpapers.jpg");

background-repeat: no-repeat;

background-position: center;

background-size: cover;

padding: 10px;

height: 100vh;

}

.form-heading {

color: #fff;

font-size: 23px;

}

.panel h2 {

color: #444444;

font-size: 18px;

margin: 0 0 8px 0;

}

.panel p {

color: #777777;

font-size: 14px;

margin-bottom: 30px;

line-height: 24px;

}

.login-form .form-control {

background: #f7f7f7 none repeat scroll 0 0;

border: 1px solid #d4d4d4;

border-radius: 4px;

font-size: 14px;

height: 50px;

line-height: 50px;

}

.main-div {

background: #ffffff none repeat scroll 0 0;

border-radius: 2px;

margin: 10px auto 30px;

max-width: 38%;

padding: 50px 70px 70px 71px;

}

.login-form .form-group {

margin-bottom: 10px;

}

.login-form {

text-align: center;

}

.forgot a {

color: #777777;

font-size: 14px;

text-decoration: underline;

}

.login-form .btn.btn-primary {

background: #f0ad4e none repeat scroll 0 0;

border-color: #f0ad4e;

color: #ffffff;

font-size: 14px;

width: 100%;

height: 50px;

line-height: 50px;

padding: 0;

}

.forgot {

text-align: left;

margin-bottom: 30px;

}

.botto-text {

color: #ffffff;

font-size: 14px;

margin: auto;

}

.login-form .btn.btn-primary.reset {

background: #ff9900 none repeat scroll 0 0;

}

.back {

text-align: left;

margin-top: 10px;

}

.back a {

color: #444444;

font-size: 13px;

text-decoration: none;

}

body {

padding-top: 0px;

}

@media (min-width: 992px) {

body {

padding-top: 0px;

}

}

**register.component.html**

<div class="container-body">

<div class="container">

<h1 class="form-heading"></h1>

<div class="login-form">

<div class="main-div">

<div class="panel">

<h2>Admin Login</h2>

<p>Please enter your email and password</p>

</div>

<form [formGroup]="registerForm">

<div class="form-group">

<input type="name" formControlName="name" class="form-control" id="inputname" placeholder="name">

</div>

<div class="form-group">

<input type="lastname" formControlName="lastname" class="form-control" id="inputlastname" placeholder="lastname">

</div>

<div class="form-group">

<input type="email" formControlName="email" class="form-control" id="inputEmail" placeholder="Email Address">

</div>

<div class="form-group">

<input type="password" formControlName="password" class="form-control" id="inputPassword" placeholder="Password">

</div>

<div class="forgot">

<!-- <a href="reset.html">Forgot password?</a> -->

</div>

<button type="submit" class="btn btn-primary" (click)="registerBtn()">Register</button>

<div class="alert alert-danger" \*ngIf="message">

<strong>Error !</strong> {{message}}

</div>

<!-- {{ registerForm.value | json}} -->

{{ registerForm.valid | json}}

</form>

</div>

<!-- <p class="botto-text"> Designed by Sunil Rajput</p> -->

</div>

</div>

</div>

**register.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms'; //import ReactiveFormsModule in app.module

@Component({

selector: 'app-register',

templateUrl: './register.component.html',

styleUrls: ['./register.component.css']

})

export class RegisterComponent implements OnInit {

registerForm: FormGroup;

constructor(private router: Router, private apiService: ApiService) { }

message = '';

ngOnInit() {

// angular reactive form

//controle de saisie

this.registerForm = new FormGroup({

name: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

lastname: new FormControl('', [Validators.required, Validators.maxLength(20)]),

email: new FormControl('', [Validators.email, Validators.required]),

password: new FormControl('', [Validators.required, Validators.minLength(8)]),

});

}

registerBtn() {

if (this.registerForm.valid) {

this.message = '';

this.apiService.registerApi(this.registerForm.value).subscribe(res => {

console.log(res.json());

if (res.json().\_id) {

this.router.navigateByUrl('/login');

} else {

this.message = res.json().message;

}

});

}

}

}

Component login :

**login.component.css**

.container-body {

background-image: url("https://hdwallsource.com/img/2014/9/blur-26347-27038-hd-wallpapers.jpg");

background-repeat: no-repeat;

background-position: center;

background-size: cover;

padding: 10px;

height: 100vh;

}

.form-heading {

color: #fff;

font-size: 23px;

}

.panel h2 {

color: #444444;

font-size: 18px;

margin: 0 0 8px 0;

}

.panel p {

color: #777777;

font-size: 14px;

margin-bottom: 30px;

line-height: 24px;

}

.login-form .form-control {

background: #f7f7f7 none repeat scroll 0 0;

border: 1px solid #d4d4d4;

border-radius: 4px;

font-size: 14px;

height: 50px;

line-height: 50px;

}

.main-div {

background: #ffffff none repeat scroll 0 0;

border-radius: 2px;

margin: 10px auto 30px;

max-width: 38%;

padding: 50px 70px 70px 71px;

}

.login-form .form-group {

margin-bottom: 10px;

}

.login-form {

text-align: center;

}

.forgot a {

color: #777777;

font-size: 14px;

text-decoration: underline;

}

.login-form .btn.btn-primary {

background: #f0ad4e none repeat scroll 0 0;

border-color: #f0ad4e;

color: #ffffff;

font-size: 14px;

width: 100%;

height: 50px;

line-height: 50px;

padding: 0;

}

.forgot {

text-align: left;

margin-bottom: 30px;

}

.botto-text {

color: #ffffff;

font-size: 14px;

margin: auto;

}

.login-form .btn.btn-primary.reset {

background: #ff9900 none repeat scroll 0 0;

}

.back {

text-align: left;

margin-top: 10px;

}

.back a {

color: #444444;

font-size: 13px;

text-decoration: none;

}

body {

padding-top: 0px;

}

@media (min-width: 992px) {

body {

padding-top: 0px;

}

}

**login.component.html**

<div class="container-body">

<div class="container">

<h1 class="form-heading"></h1>

<div class="login-form">

<div class="main-div">

<div class="panel">

<h2>Admin Login</h2>

<p>Please enter your email and password</p>

</div>

<form [formGroup]="loginForm">

<div class="form-group">

<input type="email" formControlName="email" class="form-control" id="inputEmail" placeholder="Email Address">

</div>

<div class="form-group">

<input type="password" formControlName="password" class="form-control" id="inputPassword" placeholder="Password">

</div>

<div class="forgot">

<!-- <a href="reset.html">Forgot password?</a> -->

</div>

<button type="submit" class="btn btn-primary" (click)="loginBtn()">Login</button>

<div class="alert alert-danger" \*ngIf="message">

<strong>Error !</strong> {{message}}

</div>

{{ loginForm.valid | json}}

</form>

</div>

<!-- <p class="botto-text"> Designed by Sunil Rajput</p> -->

</div>

</div>

</div>

**login.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import \* as jwt\_decode from "jwt-decode";

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

loginForm: FormGroup;

constructor(private router: Router, private apiService: ApiService) { }

message = '';

ngOnInit() {

this.loginForm = new FormGroup({

email: new FormControl('', [Validators.email, Validators.required]),

password: new FormControl('', [Validators.required, Validators.minLength(8)]),

});

}

loginBtn() {

if (this.loginForm.valid) {

this.message = '';

this.apiService.loginApi(this.loginForm.value).subscribe(res => {

console.log(res.json());

if (res.json().message === 'ok') {

localStorage.setItem('token', res.json().usertoken) //sauvegarder token dans local storage

//ajouter au role de user

const token = localStorage.getItem('token');

const roles = jwt\_decode(token).data.role;

if (roles == 'admin' ) {

this.router.navigateByUrl('/dashboard');

} else {

this.router.navigateByUrl('/home'); //use avec role et sans role

}

} else {

this.message = res.json().message;

}

});

}

}

}

Component menu :

**menu.component.html**

<nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarCollapse">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" routerLink="/home">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" routerLink="/add">Add Article</a>

</li>

<li class="nav-item">

<a class="nav-link disabled" routerLink="/home">About</a>

</li>

</ul>

<form class="form-inline mt-2 mt-md-0">

<input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<a class="nav-link" >{{user.name}}</a>

<a class="navbar-brand" href="javascript:void(0)" (click)="logoutBtn()"><i class="fas fa-sign-out-alt"></i> &nbsp; logout</a>

</form>

</div>

</nav>

**menu.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import \* as jwt\_decode from "jwt-decode";

import { ApiService } from '../api.service';

@Component({

selector: 'app-menu',

templateUrl: './menu.component.html',

styleUrls: ['./menu.component.css']

})

export class MenuComponent implements OnInit {

user =[]; //généralement en utilise document dans un array de type objet json

constructor(private router: Router, private apiService: ApiService) { }

ngOnInit() {

const token = localStorage.getItem('token');

let Data = jwt\_decode(token).data;

console.log( Data)

this.user=Data

console.log(this.user)

}

logoutBtn() {

localStorage.clear();

this.router.navigateByUrl('/login');

}

AddBtn() {

this.router.navigateByUrl('/Add');

}

}

Component home :

**home.component.css**

.social-card-header{

position: relative;

display: -ms-flexbox;

display: flex;

-ms-flex-align: center;

align-items: center;

-ms-flex-pack: center;

justify-content: center;

height: 96px;

}

.social-card-header i {

font-size: 32px;

color:#FFF;

}

.bg-facebook {

background-color:#3b5998;

}

.text-facebook {

color:#3b5998;

}

.bg-google-plus{

background-color:#dd4b39;

}

.text-google-plus {

color:#dd4b39;

}

.bg-twitter {

background-color:#1da1f2;

}

.text-twitter {

color:#1da1f2;

}

.bg-pinterest {

background-color:#bd081c;

}

.text-pinterest {

color:#bd081c;

}

.share:hover {

text-decoration: none;

opacity: 0.8;

}

.card-deck .card {

min-width: 220px;

}

**home.component.html**

<app-menu></app-menu>

<br>

<br>

<br>

<br>

<br>

<br>

<div class="container">

<div class="row">

<div class="card-deck col-sm-6 col-md-offset-2 col-md-4 text-center" \*ngFor="let art of articles">

<div class="card mb-4 shadow-sm">

<div class="card-header">

<h4 class="my-0 font-weight-normal">{{art.title}}</h4>

</div>

<div class="card-body">

<h1 class="card-title pricing-card-title">{{art.content}} </h1>

<button type="button" class="btn btn-lg btn-block btn-primary" routerLink="/article/{{art.\_id}}">Voir Article

</button>

</div>

</div>

</div>

</div>

</div>

**home.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

@Component({

selector: 'app-home',

templateUrl: './home.component.html',

styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

addForm: FormGroup;

articles;

content='';

constructor(private router: Router, private apiService: ApiService) {

/// ajout de socket

//1ere méthode avec post

// this.apiService.newArticle().subscribe(data => {

// this.articles.push(data) //get les article par socket

//})

///2ème méthode avec post, delete, put

this.apiService.newArticle().subscribe(data => {

this.ngOnInit()

})

}

ngOnInit() {

this.apiService.getArticle().subscribe(res => {

this.articles = res.json();

//afficher juste 4 caractères de content of article

for(let i = 0; i < this.articles.length; i++){

if(this.articles[i].content.length > 5){

this.articles[i].content = this.articles[i].content.substring(0,4) + '... ';

}

}

console.log(res.json());

console.log(this.articles);

})

}

}

Component add-article :

**add-article.component.html**

<app-menu></app-menu>

<br>

<br>

<br>

<br>

<br>

<div class="container">

<div class="row">

<div class="span12">

<form class="form-horizontal span6" [formGroup]="addForm">

<fieldset>

<legend>Article</legend>

<div class="form-group">

<label class="control-label">Title &nbsp;</label>

<input type="text" formControlName="title" class="form-control">

</div>

<br>

<div class="form-group">

<label class="control-label">Content &nbsp;</label>

<input type="text" formControlName="content" class="form-control">

</div>

<!--<p class="card-text mb-auto"> creationDate </p>-->

<div class="form-actions">

<button type="submit" class="btn btn-primary" (click)=addArticle() routerLink="/home">Submit</button>

<button type="button" class="btn">Cancel</button>

</div>

</fieldset>

</form>

</div>

</div>

</div>

**add-article.component.ts**

import { Component, OnInit } from '@angular/core';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import \* as jwt\_decode from "jwt-decode";

@Component({

selector: 'app-add-article',

templateUrl: './add-article.component.html',

styleUrls: ['./add-article.component.css']

})

export class AddArticleComponent implements OnInit {

addForm: FormGroup

article = {};

userId='';

constructor(private router: Router, private apiService: ApiService) { }

ngOnInit() {

this.addForm = new FormGroup({

title: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

content: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

});

}

addArticle() {

console.log(this.addForm.value);

console.log(this.addForm.valid);

if (this.addForm.valid) {

//ajouter user de l'article

const token = localStorage.getItem('token');

const userId = jwt\_decode(token).data.\_id;

this.addForm.value['author'] = userId;

this.apiService.postArticle(this.addForm.value).subscribe(res => {

console.log(res.json());

//this.ngOnInit();

});

}

}

}

**list-article.component.html**

<app-menu></app-menu>

<br>

<br>

<br>

<br>

<br>

<br>

<div class="container">

<div class="card-deck mb-3 text-center">

<div class="card mb-4 shadow-sm">

<div class="card-header">

<h4 class="my-0 font-weight-normal">{{article.title}}</h4>

</div>

<div class="card-body">

<h1 class="card-title pricing-card-title">{{article.content}} </h1>

<form [formGroup]="addForm">

<div class="form-group">

<input type="text" class="form-control" formControlName="content" aria-describedby="emailHelp"

placeholder="commentaire">

</div>

<input type="submit" value="Add comment" (click)="addCommentbtn()" class="btn btn-primary ">

<div class="clearfix"></div>

</form>

<!-- <h1 class="card-title pricing-card-title">{{article.comment}} </h1> -->

</div>

</div>

</div>

</div>

<ul class="list-group">

<li class="list-group-item" \*ngFor="let com of article.comments"><strong>{{com.content}}</strong> </li>

</ul>

**list-article.component.ts**

import { Component, OnInit } from '@angular/core';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import { Router, ActivatedRoute } from '@angular/router'; //ActivatedRoute pour recupérer l'id of article

import { ApiService } from '../api.service';

import \* as jwt\_decode from "jwt-decode";

@Component({

selector: 'app-list-article',

templateUrl: './list-article.component.html',

styleUrls: ['./list-article.component.css']

})

export class ListArticleComponent implements OnInit {

addForm: FormGroup

article;

articleId = '';

constructor(private router: Router, private apiService: ApiService, private route: ActivatedRoute) {

this.route.params.subscribe(params => {

this.articleId = params.idArticle; //idArticle se trouve dans path (app.module.ts)

console.log(this.articleId);

})

}

ngOnInit() {

// 2 get article by id

console.log(this.articleId);

this.apiService.getarticleById(this.articleId).subscribe(res => {

console.log(res.json());

this.article = res.json(); //get article by id

});

// ajouter content of comment

this.addForm = new FormGroup({

content: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

});

// 1 get tout les articles

this.apiService.getArticle().subscribe(res => {

console.log(res.json());

// this.article = res.json(); //get tous les articles

// console.log(this.articles);

})

}

addCommentbtn() {

if (this.addForm.valid) {

const token = localStorage.getItem('token');

const userId = jwt\_decode(token).data.\_id;

const commentObj = {

"content": this.addForm.value.content,

"author": userId

}

this.apiService.postcomment(this.articleId, commentObj).subscribe(res => {

console.log(res.json());

this.ngOnInit();

});

}

}

}

**dashboard.component.css**

.w-100 {

width: 30%!important;

}

**dashboard.component.html**

<nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">

<a class="navbar-brand col-sm-3 col-md-2 mr-0" routerLink="/home">Home</a>

<input class="form-control form-control-dark w-100" type="text" placeholder="Search" aria-label="Search">

<ul class="navbar-nav px-3">

<li class="nav-item text-nowrap">

<a class="nav-link" (click)="logoutBtn()"><i class="fas fa-sign-out-alt"></i> &nbsp; logout</a>

</li>

</ul>

</nav>

<br>

<br>

<br>

<div class="container-fluid">

<div class="row">

<nav class="col-md-2 d-none d-md-block bg-light sidebar">

<div class="sidebar-sticky">

<ul class="nav flex-column">

<li class="nav-item">

<a class="nav-link active" href="#">

<span data-feather="home"></span> <i class="fas fa-tachometer-alt"></i>

<strong> Dashboard </strong><span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" routerLink="/users">

<span data-feather="file"></span> <i class="fas fa-users"></i> &nbsp;

<strong>Users</strong>

</a>

</li>

<li class="nav-item">

<a class="nav-link" routerLink="/comment">

<span data-feather="file"></span> <i class="fas fa-comment-dots"></i> &nbsp;

<strong>Comment</strong>

</a>

</li>

</ul>

</div>

<br>

<br>

</nav>

<br>

<br>

<br>

<br>

<main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

<div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">

<!-- <h1 class="h2">Dashboard</h1> -->

<div class="container">

<div class="container">

<div class="">

<h2>Article</h2>

<div class="row">

<div class="col-4">

<form [formGroup]="addForm">

<div class="form-group">

<label> <strong>Title</strong></label>

<input type="text" class="form-control" formControlName="title" aria-describedby="emailHelp"

placeholder="title">

</div>

<div class="form-group">

<label><strong>Content</strong></label>

<input type="text" class="form-control" formControlName="content" placeholder="content">

</div>

<!--<label>creation date</label>

<input type="text" formControlName="creationdate" placeholder="Date" class="span3"> -->

<input type="submit" value="Submit" (click)="onAddClick()" class="btn btn-primary ">

<div class="clearfix"></div>

</form>

</div>

</div>

</div>

<br>

<br>

<div class="widget stacked widget-table action-table">

<div class="widget-content">

<table class="table table-striped table-bordered">

<thead>

<tr>

<th>title</th>

<th>content</th>

<th>creation date</th>

<th class="td-actions"></th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let art of article">

<td>{{art.title}}</td>

<td>{{art.content}}</td>

<td>{{art.creationdate}}</td>

<td class="td-actions">

<a href="javascript:;" class="btn btn-small btn-primary" (click)="articleToUpdate = art">

<i class="fa fa-upload"></i>

</a>

&nbsp; &nbsp;

<button class="btn btn-danger" role="button" href="#" (click)="deleteArticle(art.\_id)">

<i class="fa fa-trash" aria-hidden="true"></i>

</button>

</td>

</tr>

</tbody>

</table>

<div class="container">

<div class="control-group" \*ngIf="articleToUpdate">

<label class="control-label">title</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="articleToUpdate.title" class="input-block-level">

</div>

</div>

</div>

<label class="control-label">Content</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="articleToUpdate.content" class="input-block-level">

</div>

</div>

</div>

<label class="control-label">Date</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="articleToUpdate.creationdate" class="input-block-level">

</div>

</div>

</div>

<input type="submit" value="update" (click)="updateArticle()" class="btn btn-success pull-right">

<div class="clearfix"></div>

</div>

</div>

</div>

<!-- /widget-content -->

</div>

<!-- /widget -->

</div>

</div>

</div>

</main>

</div>

</div>

**dashboard.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import \* as jwt\_decode from "jwt-decode";

@Component({

selector: 'app-dashboard',

templateUrl: './dashboard.component.html',

styleUrls: ['./dashboard.component.css']

})

export class DashboardComponent implements OnInit {

addForm: FormGroup;

article = {};

userId = '';

articleToUpdate = null;

constructor(private router: Router, private apiService: ApiService) {

const token = localStorage.getItem('token');

this.userId = jwt\_decode(token).data.\_id;

}

ngOnInit() {

console.log(this.userId);

this.addForm = new FormGroup({

title: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

content: new FormControl('', [Validators.required, Validators.maxLength(30), Validators.minLength(3)]),

author: new FormControl(''),

});

this.apiService.getArticle().subscribe(res => {

console.log(res.json());

this.article = res.json();

})

}

onAddClick() {

if (this.addForm.valid) {

this.addForm.value.author = this.userId

this.apiService.postArticle(this.addForm.value).subscribe(res => {

this.ngOnInit();

});

}

}

deleteArticle(id) {

if(confirm('Are you sur to delete this article ?') == true){

this.apiService.deleteArticle(id).subscribe(res => {

this.ngOnInit();

})

}

}

updateArticle() {

console.log(this.articleToUpdate);

this.apiService.editArticle(this.articleToUpdate.\_id , this.articleToUpdate).subscribe(res => {

this.articleToUpdate = null;

this.ngOnInit();

})

}

logoutBtn() {

localStorage.clear();

this.router.navigateByUrl('/login');

}

}

**auth.guard.ts : (ng g g auth) : use in app.module.ts and auth.guard.ts**

import { Injectable } from '@angular/core';

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class AuthGuard implements CanActivate {

// add this

constructor(private router: Router) { }

canActivate(

next: ActivatedRouteSnapshot,

state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

//add this

if (localStorage.getItem('token')) {

return true;

}

this.router.navigateByUrl('/login');

return false;

}

}

**role.guard.ts (ng g g auth) : use in app.module.ts, role.guard.ts, user Model, login.component.ts and user.component.html**

import { Injectable } from '@angular/core';

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';

import { Observable } from 'rxjs';

import \* as jwt\_decode from "jwt-decode";

@Injectable({

providedIn: 'root'

})

export class RoleGuard implements CanActivate {

constructor(private router: Router) { }

canActivate(

next: ActivatedRouteSnapshot,

state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

let roles = next.data["roles"] as Array<string>;

const token = localStorage.getItem('token');

const userRole = jwt\_decode(token).data.role;

if (!roles.indexOf(userRole)) {

return true;

}

this.router.navigateByUrl('/home');

return false;

}

}

**user.component.html**

<nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">

<a class="navbar-brand col-sm-3 col-md-2 mr-0" >Application Blog</a>

<input class="form-control form-control-dark w-100" type="text" placeholder="Search" aria-label="Search">

<ul class="navbar-nav px-3">

<li class="nav-item text-nowrap">

<a class="nav-link" (click)="logoutBtn()"><i class="fas fa-sign-out-alt"></i> &nbsp; logout</a>

</li>

</ul>

</nav>

<br>

<br>

<br>

<div class="container-fluid">

<div class="row">

<nav class="col-md-2 d-none d-md-block bg-light sidebar">

<div class="sidebar-sticky">

<ul class="nav flex-column">

<li class="nav-item">

<a class="nav-link active" routerLink="/dashboard">

<span data-feather="home"></span><i class="fas fa-tachometer-alt"></i>

<strong> Dashboard </strong><span class="sr-only">(current)</span>

</a>

</ul>

</div>

<br>

<br>

</nav>

<br>

<br>

<br>

<br>

<main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

<div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">

<!-- <h1 class="h2">Dashboard</h1> -->

<div class="container">

<div class="container">

<div class="">

<h2>Users</h2>

<div class="row">

<div class="col-4">

<form [formGroup]="addForm">

<div class="form-group">

<label> <strong>name</strong></label>

<input type="text" class="form-control" formControlName="name" aria-describedby="emailHelp"

placeholder="name">

</div>

<div class="form-group">

<label> <strong>lastname</strong></label>

<input type="text" class="form-control" formControlName="lastname" aria-describedby="emailHelp"

placeholder="lastname">

</div>

<div class="form-group">

<label> <strong>email</strong></label>

<input type="text" class="form-control" formControlName="email" aria-describedby="emailHelp"

placeholder="email">

</div>

<div class="form-group">

<label> <strong>password</strong></label>

<input type="text" class="form-control" formControlName="password" aria-describedby="emailHelp"

placeholder="password">

</div>

<!--<label>creation date</label>

<input type="text" formControlName="creationdate" placeholder="Date" class="span3"> -->

<input type="submit" value="Submit" (click)="addUser()" class="btn btn-primary ">

<div class="clearfix"></div>

</form>

</div>

</div>

</div>

<br>

<br>

<div class="widget stacked widget-table action-table">

<div class="widget-content">

<table class="table table-striped table-bordered">

<thead>

<tr>

<th>name</th>

<th>lastname</th>

<th>email</th>

<!-- <th>password</th> -->

<th>Role</th>

<th class="td-actions"></th>

</tr>

</thead>

<tbody>

<!-- {{user | json}} -->

<tr \*ngFor="let usr of user">

<td>{{usr.name}}</td>

<td>{{usr.lastname}}</td>

<td>{{usr.email}}</td>

<!-- <td>{{usr.password}}</td> -->

<td>{{usr.role}}</td>

<td class="td-actions">

<a href="javascript:;" class="btn btn-small btn-primary" (click)="updateUser = usr">

<!--boutton modif-->

<i class="fa fa-upload"></i>

</a>

&nbsp; &nbsp;

<button class="btn btn-danger" role="button" href="#" (click)="deleteUser(usr.\_id)">

<!--bouton delet-->

<i class="fa fa-trash" aria-hidden="true"></i>

</button>

</td>

</tr>

</tbody>

</table>

<div class="container">

<div class="control-group" \*ngIf="updateUser">

<label class="control-label">Name</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="updateUser.name" class="input-block-level">

</div>

</div>

</div>

<label class="control-label">Lastname</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="updateUser.lastname" class="input-block-level">

</div>

</div>

</div>

<label class="control-label">Email</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="updateUser.email" class="input-block-level">

</div>

</div>

</div>

<label class="control-label">Password</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<input type="text" [(ngModel)]="updateUser.password" class="input-block-level">

</div>

</div>

</div>

<label class="control-label">Access</label>

<div class="controls">

<div class="row-fluid">

<div class="span3">

<!-- donner l'accès au user par l'admin -->

<input type="checkbox" [(ngModel)]="updateUser.access" class="input-block-level">

</div>

</div>

</div>

<!—- utiliser roles -->

<div class="form-group">

<label for="exampleFormControlSelect1">Role</label>

<select class="form-control" id="exampleFormControlSelect1" [(ngModel)]="updateUser.role">

<option>admin</option>

<option>user</option>

</select>

</div>

<input type="submit" value="update" (click)="userToUpdate()" class="btn btn-success pull-right">

<div class="clearfix"></div>

</div>

</div>

</div>

<!-- /widget-content -->

</div>

<!-- /widget -->

</div>

</div>

</div>

</main>

</div>

</div>

**user.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router, ActivatedRoute } from '@angular/router';

import { ApiService } from '../api.service';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import \* as jwt\_decode from "jwt-decode";

@Component({

selector: 'app-users',

templateUrl: './users.component.html',

styleUrls: ['./users.component.css']

})

export class UsersComponent implements OnInit {

addForm: FormGroup;

user = [];

userId = '';

updateUser = null;

constructor(private router: Router, private apiService: ApiService) {

}

ngOnInit() {

console.log(this.userId);

this.addForm = new FormGroup({

name: new FormControl('', [Validators.required, Validators.maxLength(20), Validators.minLength(3)]),

lastname: new FormControl('', [Validators.required, Validators.maxLength(30), Validators.minLength(3)]),

email: new FormControl('', [Validators.email, Validators.required]),

password: new FormControl('', [Validators.required, Validators.minLength(8)]),

});

this.apiService.getUser().subscribe(res => {

console.log(res.json());

this.user = res.json();

})

}

addUser() {

if (this.addForm.valid) {

this.apiService.postUser(this.addForm.value).subscribe(res => {

this.ngOnInit();

});

}

}

deleteUser(id) {

if (confirm('Are you sur to delete this user ?') == true) {

this.apiService.deleteUser(id).subscribe(res => {

this.ngOnInit();

})

}

}

userToUpdate() {

this.apiService.updateUser(this.updateUser.\_id, this.updateUser).subscribe(res => {

console.log(this.updateUser);

this.updateUser = null;

this.ngOnInit();

})

}

logoutBtn() {

localStorage.clear();

this.router.navigateByUrl('/login');

}

}

**comment.component.html**

<nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">

<a class="navbar-brand col-sm-3 col-md-2 mr-0" >Application Blog</a>

<input class="form-control form-control-dark w-100" type="text" placeholder="Search" aria-label="Search">

<ul class="navbar-nav px-3">

<li class="nav-item text-nowrap">

<a class="nav-link" href="#"><i class="fas fa-sign-out-alt"></i> &nbsp; logout</a>

</li>

</ul>

</nav>

<br>

<br>

<br>

<div class="container-fluid">

<div class="row">

<nav class="col-md-2 d-none d-md-block bg-light sidebar">

<div class="sidebar-sticky">

<ul class="nav flex-column">

<li class="nav-item">

<a class="nav-link active" routerLink="/dashboard">

<span data-feather="home"></span>

<i class="fas fa-tachometer-alt"></i>

<strong> Dashboard </strong><span class="sr-only">(current)</span>

</a>

</ul>

</div>

<br>

<br>

</nav>

<br>

<br>

<br>

<br>

<main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

<div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">

<!-- <h1 class="h2">Dashboard</h1> -->

<div class="container">

<div class="container">

<div class="">

<h2>Comments</h2>

</div>

<br>

<br>

<div class="widget stacked widget-table action-table">

<div class="widget-content">

<table class="table table-striped table-bordered">

<thead>

<tr>

<th>Title</th>

<th>Content</th>

<th>Date de création</th>

<th>User</th>

<th class="td-actions"></th>

</tr>

</thead>

<tbody \*ngFor="let art of comments;">

<tr \*ngFor="let comment of art.comments">

<td>{{art.title}}</td>

<td>{{comment.content}}</td>

<td>{{comment.creationdate }}</td>

<td>{{comment.author.name }}</td>

<td class="td-actions">

<button class="btn btn-danger" role="button" href="#" (click)="deleteComment(comment.\_id)">

<!--bouton delet-->

<i class="fa fa-trash" aria-hidden="true"></i>

</button>

</td>

</tr>

</tbody>

</table>

</div>

<!-- /widget-content -->

</div>

<!-- /widget -->

</div>

</div>

</div>

</main>

</div>

</div>

**comment.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router} from '@angular/router';

import { ApiService } from '../api.service';

import \* as jwt\_decode from "jwt-decode";

@Component({

selector: 'app-comment',

templateUrl: './comment.component.html',

styleUrls: ['./comment.component.css']

})

export class CommentComponent implements OnInit {

user = [];

userId = '';

updateUser = null;

articleId='';

article=[];

comments=[];

constructor(private router: Router, private apiService: ApiService) {

}

ngOnInit() {

this.apiService.getArticle().subscribe(res => {

console.log(res.json());

this.article = res.json(); //get article by id

});

//console.log(this.articleId);

this.apiService.getcomment().subscribe(res => {

console.log(res.json());

this.comments = res.json(); //get article by id

});

}

deleteComment(id) {

if (confirm('Are you sur to delete this comment ?') == true) {

this.apiService.deletecomment(id).subscribe(res => {

this.ngOnInit();

})

}

}

logoutBtn() {

localStorage.clear();

this.router.navigateByUrl('/login');

}

}

**Ajouter toastr pour notre application :**

[**https://www.npmjs.com/package/ngx-toastr:Angular**](https://www.npmjs.com/package/ngx-toastr:Angular) **6**

[**https://www.npmjs.com/package/ng6-toastr**](https://www.npmjs.com/package/ng6-toastr)**,** [**https://www.npmjs.com/package/ng2-toastr**](https://www.npmjs.com/package/ng2-toastr) **: Angular 2, 4, 5**

npm install ngx-toastr --save

npm install @angular/animations –-save

**index.html :**

<link rel="stylesheet" href="node\_modules/ngx-toastr/toastr.css">

**App.module.ts :**

import { CommonModule } from '@angular/common';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { ToastrModule } from 'ngx-toastr';

imports: [

//ngx-toastr

CommonModule,

BrowserAnimationsModule, // required animations module

ToastrModule.forRoot() // ToastrModule added

],

**App.component.ts :**

import { ToastrService } from 'ngx-toastr';

constructor(private toastr: ToastrService)

categorieBtn(){

this.toastr.success('success!', 'Toastr fun!');

}

showSuccess() {

        this.toastr.success('You are awesome!', 'Success!');

      }

      showError() {

        this.toastr.error('This is not good!', 'Oops!');

      }

      showWarning() {

        this.toastr.warning('You are being warned.', 'Alert!');

      }

      showInfo() {

        this.toastr.info('Just some information for you.');

      }

      showCustom() {

        this.toastr.custom('<span style="color: red">Message in red.</span>', null, {enableHTML: true});

      }

**Chart.js in Angular 6 :**

**Installer Bibliothèque : npm install chart.js --save**

**App.component.ts :**

import { Chart } from 'chart.js';

export class DashboardComponent implements OnInit {

LineChart = [];

BarChart = [];

ngOnInit() {

// line Chart

this.LineChart = new Chart('lineChart', {

type: 'line',

data: {

labels: ["Jan", "Fev", "March", "April"],

datasets: [

{

label: 'Number of items in Months',

data: [1, 3, 4, 5, 10, 22],

borderColor: '#00AEFF',

fill: false

}

]

},

options: {

title: {

text:"Line Chart",

display: true

},

legend: {

display: false

},

scales: {

xAxes: [{

display: true

}],

yAxes: [{

display: true

}],

}

}

});

// barChart

this.BarChart = new Chart('barChart', {

type: 'bar',

data: {

labels: ["Jan", "Fev", "March", "April"], // your labels array

datasets: [

{

label: 'Number of items in Months',

data: [1, 3, 4, 5, 10, 22], // your data array

backgroundColor: [

'rgba(255, 99, 132, 0.2)',

'rgba(54, 162, 235, 0.2)',

'rgba(255, 206, 86, 0.2)',

'#00AEFF'

],

fill: false

}

]

},

options: {

title: {

text:"Bar Chart",

display: true

},

legend: {

display: false

},

scales: {

xAxes: [{

display: true

}],

yAxes: [{

display: true

}],

}

}

});

}

}

**App.component.html :**

<table>

<tr>

<td style="width:700px; height:50%"><canvas id="lineChart"></canvas></td>

<td style="width:700px; height:50%"><canvas id="barChart"></canvas></td>

</tr>

</table>

**Data Table :**

[**https://l-lin.github.io/angular-datatables/#/getting-started**](https://l-lin.github.io/angular-datatables/#/getting-started)

[**https://www.npmjs.com/package/angular-6-datatable**](https://www.npmjs.com/package/angular-6-datatable)

[**https://datatables.net/**](https://datatables.net/)

npm install jquery --save

npm install datatables.net --save

npm install datatables.net-dt --save

npm install angular-datatables --save

npm install @types/jquery --save-dev

npm install @types/datatables.net --save-dev

**index.js**

<link rel="stylesheet" href="node\_modules/datatables.net-dt/css/jquery.dataTables.css">

<script type="text/javascript" src="node\_modules/jquery/dist/jquery.js"></script>

  <script type="text/javascript" src="node\_modules/datatables.net/js/jquery.dataTables.js"></script>

**App.module.ts**

import { DataTablesModule } from 'angular-datatables';

imports: [

BrowserModule,

DataTablesModule,

],

**Api google maps in Angular :**

**Site des APIs :**

[**https://console.developers.google.com/apis/dashboard?project=my-project-1541707805215&duration=PT1H**](https://console.developers.google.com/apis/dashboard?project=my-project-1541707805215&duration=PT1H)

1. **install angular google maps : npm install @agm/core --save**
2. **app.module.ts**

//google maps

import {AgmCoreModule} from '@agm/core';

imports: [

BrowserModule,

AgmCoreModule.forRoot({

apiKey : 'AIzaSyAfJTVKnpLl0ULuuwDuix-9ANpyQhP6mfc' //AIzaSyDQnP63a5jyiquz-sNYdbc1jT54lbdBiIo

}),

],

1. **Contact.component.ts**

export class ContactComponent implements OnInit {

//latitude et langitude de Tunis

lat: number = 36.8189700;

lng: number = 10.1657900;

constructor() { }

ngOnInit() {

}

}

1. **Contact.component.html**

<agm-map [latitude]="lat" [longitude]="lng">

<agm-marker [latitude]="lat" [longitude]="lng"></agm-marker>

</agm-map>

1. **Contact.component.css**

agm-map {

height: 400px;

}

**API facebook et google + :**

[**https://www.npmjs.com/package/angular-6-social-login**](https://www.npmjs.com/package/angular-6-social-login)

**Changer langage de l’application avec ngx-translate :**

[**https://www.npmjs.com/package/@ngx-translate/core**](https://www.npmjs.com/package/@ngx-translate/core) **avec exemple: https://stackblitz.com/github/ngx-translate/example?file=package.json**

npm install @ngx-translate/core

npm install @ngx-translate/http-loader

**app.module.ts**

//translate

import {HttpClient, HttpClientModule} from '@angular/common/http';

import {TranslateModule, TranslateLoader} from '@ngx-translate/core';

import {TranslateHttpLoader} from '@ngx-translate/http-loader';

// AoT requires an exported function for factories

export function HttpLoaderFactory(httpClient: HttpClient) {

return new TranslateHttpLoader(httpClient);

}

imports: [

//translate

HttpClientModule,

TranslateModule.forRoot({

loader: {

provide: TranslateLoader,

useFactory: HttpLoaderFactory,

deps: [HttpClient]

}

})

],

**component.ts**

import { Component, OnInit } from '@angular/core';

import {TranslateService} from '@ngx-translate/core';

constructor(public translate: TranslateService) {

translate.addLangs(['en', 'fr']);

translate.setDefaultLang('en');

const browserLang = translate.getBrowserLang();

translate.use(browserLang.match(/en|fr/) ? browserLang : 'en');

}

**component.html**

<div>

<h2>{{ 'HOME.TITLE' | translate }}</h2>

<label>

{{ 'HOME.SELECT' | translate }}

<select #langSelect (change)="translate.use(langSelect.value)">

<option \*ngFor="let lang of translate.getLangs()" [value]="lang" [selected]="lang === translate.currentLang">{{ lang }}</option>

</select>

</label>

</div>

**assets/i18n/en.json :**

{

"HOME": {

"TITLE": "Hello Angular with ngx-translate!",

"SELECT": "Change language"

}

}

**assets/i18n/fr.json :**

{

"HOME": {

"TITLE": "Bonjour Angular avec ngx-translate !",

"SELECT": "Changer la langue"

}

}

**Methode search :**

[**https://www.npmjs.com/package/ng2-search-filter**](https://www.npmjs.com/package/ng2-search-filter)

**app.module.ts :**

import { Ng2SearchPipeModule } from 'ng2-search-filter';

imports: [

BrowserModule,

Ng2SearchPipeModule,

],

**Component.html :**

<input [(ngModel)]="term" aria-label="Search" style="border-radius:5%;" type="search" placeholder="search">

<tr \*ngFor="let pat of patient |filter:term | paginate: { itemsPerPage: 4, currentPage: p }">

<td>{{pat.first\_name}}</td>

<td>{{pat.last\_name}}</td>

<td>{{pat.address}}</td>

<td>{{pat.phone}}</td>

<td>{{pat.age}}</td>

<td>{{pat.sexe}}</td>

<td>{{pat.date | date: 'dd/MM/yyyy'}}</td>

<td class="td-actions">

<!--<button class="btn btn-primary" >

<i class="fa fa-upload" (click)="updatePatient = pat" ></i>

</button> -->

<!-- Button trigger modal -->

<button type="button" class="btn btn-primary" (click)="updatePatient = pat" data-toggle="modal" data-target="#exampleModal">

<i class="fa fa-upload" aria-hidden="true"></i>

</button>

&nbsp;

<button type="button" class="btn btn-danger" data-toggle="modal" data-target=".bd-example-modal-sm" (click)="deletePatient = pat"><i class="fa fa-trash" aria-hidden="true"></i></button>

</td>

</tr>

**Pagination :**

[**https://www.npmjs.com/package/ngx-pagination**](https://www.npmjs.com/package/ngx-pagination)

**npm install ngx-pagination --save**

**app.module.ts :**

import {NgxPaginationModule} from 'ngx-pagination';

imports: [

BrowserModule,

NgxPaginationModule ],

**Component.html :**

<tbody>

<tr \*ngFor="let pat of patient |filter:term | paginate: { itemsPerPage: 4, currentPage: p }">

<td>{{pat.first\_name}}</td>

<td>{{pat.last\_name}}</td>

<td>{{pat.address}}</td>

<td>{{pat.phone}}</td>

<td>{{pat.age}}</td>

<td>{{pat.sexe}}</td>

<td>{{pat.date | date: 'dd/MM/yyyy'}}</td>

<td class="td-actions">

<!--<button class="btn btn-primary" >

<i class="fa fa-upload" (click)="updatePatient = pat" ></i>

</button> -->

<!-- Button trigger modal -->

<button type="button" class="btn btn-primary" (click)="updatePatient = pat" data-toggle="modal" data-target="#exampleModal">

<i class="fa fa-upload" aria-hidden="true"></i>

</button>

&nbsp;

<button type="button" class="btn btn-danger" data-toggle="modal" data-target=".bd-example-modal-sm" (click)="deletePatient = pat"><i class="fa fa-trash" aria-hidden="true"></i></button>

</td>

</tr>

</tbody>

</table>

<pagination-controls (pageChange)="p = $event" maxSize="9"

directionLinks="true"

autoHide="true"

responsive="true"></pagination-controls>

**Ajouter ngBootstrap/DatePicker pour le type date :**

[**https://valor-software.com/ngx-bootstrap/?gclid=EAIaIQobChMIteeJm-CN3wIVh7\_tCh1\_PwLIEAAYASAAEgKV3vD\_BwE#/datepicker**](https://valor-software.com/ngx-bootstrap/?gclid=EAIaIQobChMIteeJm-CN3wIVh7_tCh1_PwLIEAAYASAAEgKV3vD_BwE#/datepicker) **: pour ng-bootstrap**

[**https://stackblitz.com/edit/ngx-bootstrap-datepicker?file=main.ts**](https://stackblitz.com/edit/ngx-bootstrap-datepicker?file=main.ts) **: exemple de bootstrap-datepicker**

**installer : npm install ngx-bootstrap --save**

**app.module.ts :**

import { DatepickerModule, BsDatepickerModule } from 'ngx-bootstrap/datepicker';

imports: [

BrowserModule,

BsDatepickerModule.forRoot(),

DatepickerModule.forRoot() ,

],

**Index.html :**

<link rel="stylesheet" href="https://unpkg.com/ngx-bootstrap/datepicker/bs-datepicker.css">

**component.html :**

<input type="text" class="form-control" formControlName="date" [minDate]="minDate"

[maxDate]="maxDate"

#dp="bsDatepicker"

bsDatepicker [(bsValue)]="myDateValue" placeholder="date">

**component.ts :**

export class AddPatientComponent implements OnInit {

myDateValue: Date;

addForm : FormGroup;

constructor(private patientService:PatientService) { }

ngOnInit() {

this.myDateValue = new Date();

}

onDateChange(newDate: Date) {

console.log(newDate);

}

}

**Create model class Angular :**

**ng g class customer –type=model**

**eliminer spec.ts :**

**ng g c customer –spec=false**