**Lab 1.0**

**Backend**: sici esta hecho bajo el estándar de desarrollo MEAN, por lo tanto el backend esta construido en nodejs.

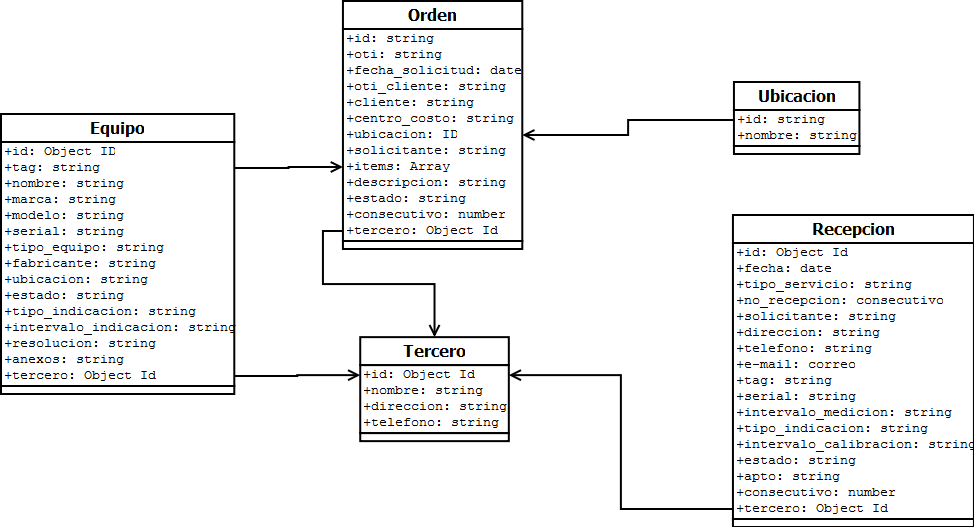
empecemos instalando las dependencias que necesitamos para nuestro backend

1. **Npm install express --save** : librería de node que nos permite estableces un servidor que trabaja con protocolos http, rutas, controladores etc.
2. **Npm install bcrypt-nodejs --save** :librería que se encarga del encriptado de contraseñas
3. **Npm install body-parser –save**: Librería que nos permite convertir los json que nos llegan a la api o al backend a un objeto javascript totalmente usable y funcional
4. **Npm install connect-multiparty --save**  : librería que se encarga del proceso de la subida de los ficheros o : archivos es decir fotos, documentos etc.
5. **Npm install jwt-simple --save:**librería que nos permite cifrar y descifrar los token para la autenticación u otros procesos en los que se requieran
6. **Npm install mongoose:** librería que nos permite trabajar de una manera mas simple con mongodb
7. **Npm install mongoose-**pagination: librería que nos permite paginar los resultados de las busqeudas de mongoose.
8. **Npm install moment --save :** librería para trabajar las fechas, horas y todo lo que tenga que ver con tiempo.

**Npm install nodemon --save-dev** : librería que refresca el servidor cada que guardamos para no estar parando y arrancando el servidor.

**Diseñemos la base de datos**

La base de datos esta construida en mongoDB una base de datos no relacional.



Creemos en la raíz el fichero app.js y lo exportamos este fichero sera donde convergerá todo:

'use strict';

var express = require('express');

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

//inicialziar variables

var app = express();

//cargar middlewares

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

app.use(bodyParser.json());

//escuchar peticiones

*module*.*exports* = app;

ahora creemos el archivo index.js donde conectaremos con mongoDB y donde requeriremos el app.module para inicial el servidor de express. Nuestra aplicación correra su backend en el puerto 3800.

'use strict'

var mongoose = require('mongoose');

var app = require('./app');

var port = 3800;

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost:27017/laboratorio', {useNewUrlParser: true})

.then(()=>{

*console*.log("conectado a la db laboratorio");

app.listen(port, ()=>{

*console*.log("express server corriendo en pueto 3800 online");

})

})

.catch(*err*=>{

*console*.log("Error al conectarse a la db");

})

**Creemos una ruta de prueba en app.js para saber desde el navegador si en realidad nuestro servidor esta corriendo:**

'use strict';

var express = require('express');

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

//inicialziar variables

var app = express();

//cargar middlewares

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

app.use(bodyParser.json());

//cargar rutas

app.post('/', (*req*, *res*)=>{

*console*.log(req.body);

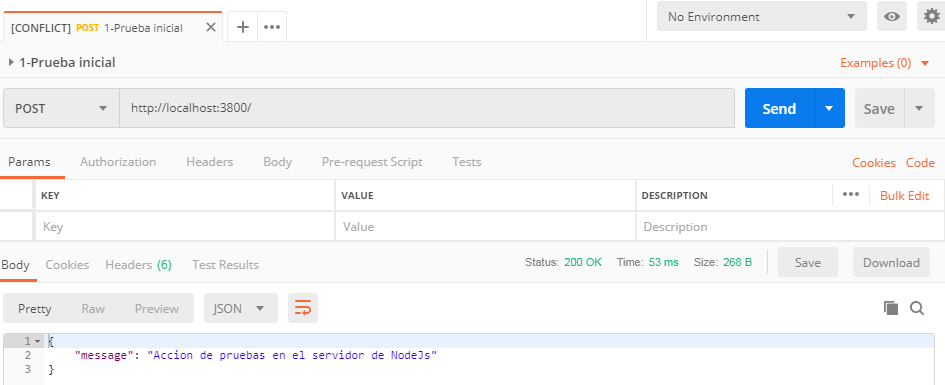
res.status(200).json({

message:'Accion de pruebas en el servidor de NodeJs'

});

});

*module*.*exports* = app;



**Creemos el modelo de las colecciones MongoDb en la base de Datos.**

1-creamos en al raíz la carpeta models

2-dentro creamos el fichero usersModel.js(será el fichero del modelo de usuario)

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

//defino una variable que defina lso roles validos

var rolesValidos = {

values:['CLIENTE', 'PROVEEEDOR', 'SUPERSU'],

message: '{VALUE} no es un rol permitido'

}

UserSchema = **new** *Schema*({

nombre : {type:*String*, required:[true, "el nombre es requerido"]},

correo : {type:*String*, unique:true, required:[true, "el correo es requerido"]},

role : {type:*String*, required: true, default:"CLIENTE", enum:rolesValidos},

password : {type:*String*, required:[true, "la contraseña es requerida"]}

});

*module*.*exports* = mongoose.model('User', UserSchema);

3-seguimos con las otras colecciones del diagrama en mongo que hicimos. **cargoModel.js**.

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

CargoSchema = **new** *Schema*({

nombre:{type:*String*, required:[true, 'el nombre del cargo es requerido']},

vhora\_hombre:{type:*String*, required:[true, 'valor hora del cargo ers requerida']}

})

*module*.*exports* = mongoose.model('Cargo', CargoSchema);

4-terceroModel.js

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var tipoTercero = {

values:['CLIENTE', 'PROVEEDOR'],

message :'{VALUE: no es un tipo tercero permitido}'

}

var TerceroSchema = **new** *Schema*({

nit:{type:*String*, required:[true, "El nit es requerido para el tercero"]},

tipoTercero : {type:*String*, required:true, default:'CLIENTE', enum:tipoTercero},

nombre:{type:*String*, required:[true, 'El nombre es requerido para el tercero']},

direccion : {type:*String*, required:[true, 'La direccion es requerida para el tercero']},

telefono : {type:*String*, required:[true, 'El telefono es requerido para el tercero']},

contacto : {type:*String*, required:[true, 'El contacto es requerido para el tercero']},

descripcion :{type:*String*, required:[true, 'La descripcion en el tercero son datos adicionales']}

})

*module*.*exports* = mongoose.model('Tercero', TerceroSchema);

5-equipoModel.js

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var EquipoSchema = **new** *Schema*({

tag:{type:*String*, unique:true, required:[true, 'El tag para el equipo es necesario']},

nombre\_equipo:{type:*String*, required:[true, 'El nombre del equipo es requerido']},

fecha\_creacion:{type:*String*, required:[true, 'La fecha de creacion es requerida']},

usuario\_creador :{type:*String*, required:[true, 'El usuario que crea el equipo es requerido']},

tercero:{type:*String*, required:[true, 'El tercero dueño del equipo es requerido']}

})

*module*.*exports* = mongoose.model('Equipo', EquipoSchema);

/\*----------------------------------------------------------------------------

tag : Codigo unico que tiene cada equipo

nombre: nombre o descripcion del equipo

fecha\_creacion: fecha en la que se crea el equipo, el sistema la toma en automatico,

usuario\_creador:usuario que crea el equipo, el sistema lo toma en automatico,

tercero:dueño del equipo

------------------------------------------------------------------------------\*/

6-consecutivoModel.js

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var ConsecutivoSchema = **new** *Schema*({

prefijo:{type:*String*, required:[true, 'El prefijo es requerido']},

consecutivo: {type:*Number*, required:[true, 'El consecutivo empieza en cero y va sumando 1 a 1']},

tercero :{type:*String*, required:[true, 'El tercero es el cliente o proveedor a quien le pertenece el consecutivo']}

})

*module*.*exports* = mongoose.model('Consecutivo', ConsecutivoSchema);

/\*---------------------------------------------------------------------------------

prefijo : Caracteres que antecederan al numero en la actividad

consecutivo: empieza en cero y sumara 1 cada que el tercero cree una actividad

tercero : PROVEEDOR O CLIENTE que se asociaran a los consecutivos

-----------------------------------------------------------------------------------\*/

ubicacionFisicaModel.js

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var UbicacionfisicaSchema = **new** *Schema*({

nombre:{type:*String*, required:[true, 'El nombre de la ubicacion sisica es requerida']},

usuario\_creador:{type:*String*, required:[true, 'El usuario creador es requerido']},

fecha\_creacion:{type:*String*, required:[true, 'La fecha de creacion es requerida']},

tercero:{type:*String*, required:[true, 'El tercero es requerido']}

})

*module*.*exports* = mongoose.model('Ubicacionfisica', UbicacionfisicaSchema);

/\*---------------------------------------------------------------------------------

nombre : nombre de la ubicacion fisica

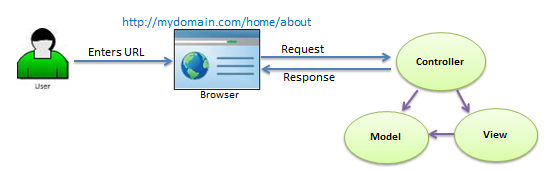
consecutivo: incrementador 1 a 1 por cada actividad

tercero : es el tercero que usara la ubicacion fisica

-----------------------------------------------------------------------------------\*/

**Armando arquitectura MVC(Modelo Vista Controlador)**

Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.



1-ya creamos los modelos

2-creemos ahora los controladores

3-las vistas las crearemos en el frontend con angular 6

**Creando la carpeta de controladores**

En la raíz crearemos esta carpeta donde encontraremos los controladores esta carpeta se llama **controllers**.

1-el primer controlador que creamos el el **userController.js**, aquí ira la lógica del funcionamiento de los usuarios.

* Importamos el modelo de usuarios es decir **userModel.js**.
* Crearemos una función de prueba
* Exportaremos la función creada
* 'use strict'

var UserModel = require('../models/userModel');

*//probando una ruta por get para saber que esta bien*

function prueba(req, res){

console.log(req.body);

res.status(200).send({message:"la prueba ha sido un exito"})

}

*//exporto la funcion creada*

module.exports = {

prueba

}

* Ahora creo la ruta para poder probar el método. Lo ideal es que cada controlador tenga una ruta independiente.
* Lo primero es crear en la raíz la carpeta que contendrá las rutas es decir se llama **routes**.
* Ahora creamos el primer fichero de rutas en este caso se llama **userRoutes.js**.
* Importamos express
* Importamos el controlador de usuarios
* Ejecutamos en una variable el método Router de express.
* Creo la ruta con la variable que contiene el Router de express con los métodos http, el primer parámetro es la ruta, el segundo parámetro es el controlador.
* Exportamos la ruta desde la variable que contiene el router de express

'use strict';

const express = require('express');

var UserController = require('../controllers/userController');

var api = express.Router();

api.get('/prueba', UserController.prueba);

module.exports = api;

* Importo los ficheros de rutas base en el app.js que es el archivo donde converge todo.
* Podemos definir rutas base para que tengamos mas orden, es decir primero vendría la ruta base y luego separado con un / la ruta definida en el fichero de rutas, esto lo hacemos creando un middleware para crear rutas base con el método **use de express**.
* El archivo **app.js** quedaría de la siguiente manera.

'use strict';

var express = require('express');

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

*//inicialziar variables*

var app = express();

*//cargar middlewares*

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

app.use(bodyParser.json());

*//configuracion de Cors(cabeceras http)*

*//carga de ficheros de rutas*

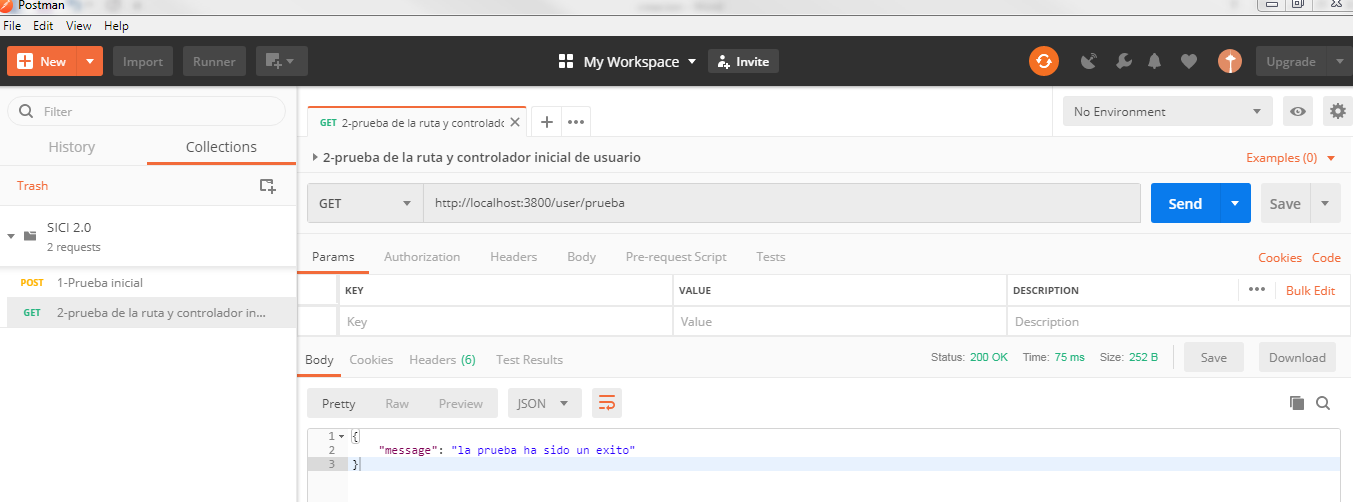
var UserRoutes = require('./routes/userRoutes');

*//middleware para definir la ruta base*

app.use('/user', UserRoutes);

module.exports = app;

Ahora probemos en postman la ruta(antes inicia el servidor de express con npm)



**Función GuardarUser del controlador de usuario**

* Como vamos a encriptar la contraseña usaremos el modulo bcrypt que instalamos desde el inicio del proyecto.

'use strict'

var bcrypt = require('bcrypt-nodejs')

Creamos la function para guardarUser usamos el bcryp para encriptar la contraseña, este modulo funciona de la siguiente manera.

* Debemos buscar primero si el usuario existe, si existe no se debe permitir guardar
* Sino existe permitirá guardar, antes deberemos o encriptar la contraseña usando el método hash de bcryp

el primer parámetro sera la cadena a encriptar que seria el password y existen 3 parametros mas los 2 primeros son para hacer mas encriptación sobre la encriptación y el tercero es una función callback que nos arrojara la encriptación y el error como se muestra a continuación de modo que el user.password lo igualaremos al hash resultante.

function guardarUser(req, res){

var body = req.body;

if(body.nombre && body.correo && body.password1 && body.password2){

if(body.password1 === body.password2){

UserModel.find({correo: body.correo.toLowerCase()})

.exec((err, users)=>{

if(err){

res.status(500).send({message:'Error en la peticion'});

}

if(users && users.length >= 1){*//si encuentra al usuario no permitir guardar*

return res.status(200).send({message:'El usuario que intenta registrar ya existe'});

}else{*//si el usuario no existe podemos guardarlo*

var User = new UserModel();

User.nombre = body.nombre;

User.correo = body.correo;

*//encriptando el password*

bcrypt.hash(body.password1, null, null, (err, hash)=>{

User.password = hash;

*//Guardamos el usuario*

User.save((err, usuarioGuardado)=>{

if(err) return res.status(500).send({message:"error al guardar el usuario"});

if(usuarioGuardado) return res.status(200).send({user:usuarioGuardado});

if(!usuarioGuardado) return res.status(404).send({message: "ha ocurrrido un problema no se guardo el usuario"});

})

});

}

})

}else{

return res.status(200).send({message:"Contraseñas no son iguales"})

}

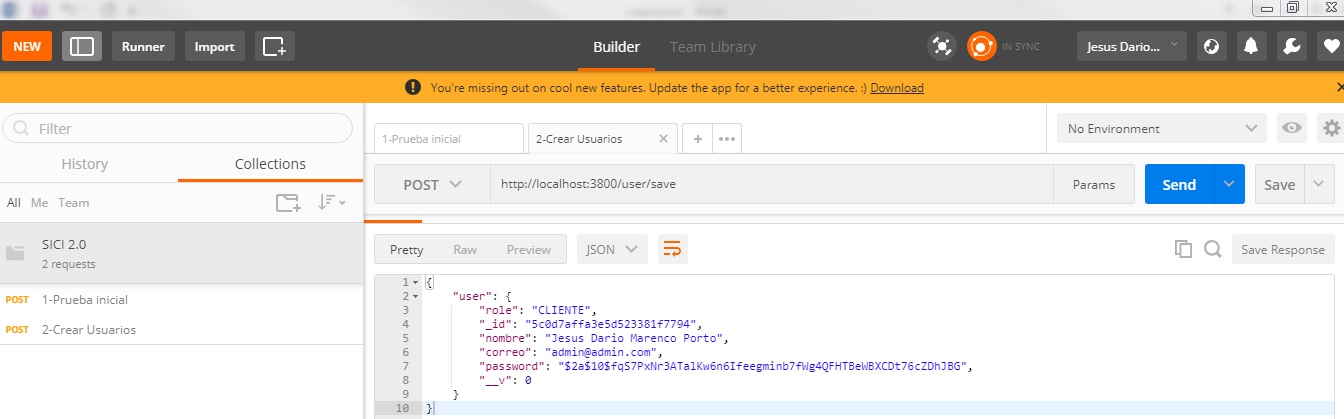
}else{

return res.status(200).send({message:"todos los campos son necesarios"});

}

}

Ya se pueden crear usuarios.



**Haciendo el Login**

Cuando el metodo de login devuelve los datos es decir el login esta correcto, nos devuelve todo incluido el password y todo el hash(encriptación de bcrypt) de la contraseña, esto es mejor no devolverlo y que se quede en el backend, para eliminar estos datos en el código antes de mandar los datos puedo setiar el password que me trae el user para que envie otra cosa.

J **jwt(JAON WEB TOKEN PARA EL LOGIN)**

ahora a la hora de devolver los datos del usuario podremos devolver un token en donde iran todos los datos del usuario pero codificados o encriptados dentro de un hash.

Para esto deberemos crear un servicio dentro de nodejs.

1-creemos una carpeta llamada services en la raíz que contendrá el fichero jwt.js donde construiremos el servicio para devolver un token en el Login y no la información del usuario.

* Requerimos el modulo de jwt para generar el token.
* Requerimos el momulo de momento para configurar los tiempos del token
* Creamos en una variable que llamaremos seccret una clave que nos ayudara a encriptar y desencriptar el token.
* Creamos la function createToken, añadimos fecha inicial y de expirtacion conm momento
* Retornamos usando el método encode al cual le pasamos el objeto creado en la funtion y el secret

'use strict'

var jwt = require('jwt-simple');

var moment = require('moment');

var secret = 'Noraver\_Igora\_es\_$%.una{clave78/paraDR\_xa+loque{´+}Raton\_Think.'*/\*este es un String secreto que solo nosotros*

*como programadores sabremos, el token es casi que imposible de descifrarlo sin esta clave\*/*

exports.createToken = function(user){

var payload = {

sub:user.\_id,

nombre:user.nombre,

email:user.email,

image:user.image,

iat:moment().unix(),*//fecha en la que se crea el token*

exp:moment().add(1, 'days').unix*//fecja en la que expira el token, a la fecha actual le agregamos 1 dias*

}

*//el metodo encode lo codifica todo y genera un hash con ayuda de la clave secret*

return jwt.encode(payload, secret);

}

Ahora utilicemos este servicio jwt.js dentro de nuestro controlador de usuarios userController.js

1-var jwt = require('../services/jwt');

2- ahora lo usamos en nuestra function de login

*//Login de Usuarios*

function loginUser(req, res){

var body = req.body;

var correo = body.correo;

console.log(correo);

var password = body.password;

UserModel.findOne({correo:correo}, (err, user)=>{

console.log(user);

if(err) return res.status(500).send({message:'Error al ejecutar el login'});

if(user){

bcrypt.compare(password, user.password, (er, check)=>{

if(check){

if(body.gettoken=='true'){

return res.status(200).send({toke: jwt.createToken(user)});

}else{

*//devolvemos el usuariuo normal*

user.password =undefined;*//esto para hacerle set a la contraseña y que no la devuelva*

return res.status(200).send({

message:"Usuario Logueado Correctamente",

user:user,

check:check

});

}

}else{

*//si la contraseña es incorrecta*

return res.status(404).send({message:"usuario no pudo identificarse"});

}

})

}else{

*//si el usuario no es encontrado*

return res.status(504).send({message:"el usuario no pudo ser identificado!!"});

}

})

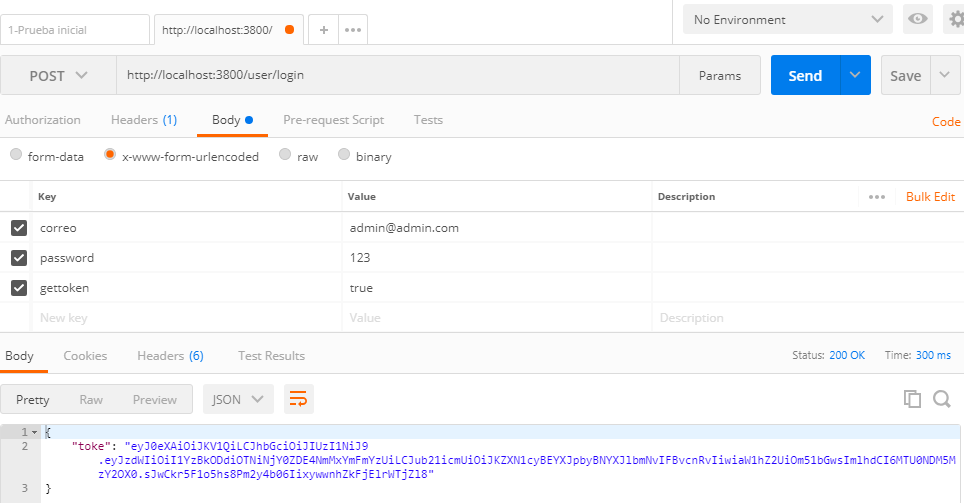
}

Exportamos la función y creamos la ruta:

*//Login de usuario*

api.post('/login', UserController.loginUser)

probamos



**Creemos un middleware de autenticación.**

Este middleware comprobara el token que se envía para ver si es correcto y si es correcto puede entrar a la sesion

1-creamos una carpeta mueva que se llama middlewares y dentro un fichero llamado autenticated.js

'use strict'

var jwt = require('jwt-simple');

var moment = require('moment');

var secret = 'Noraver\_Igora\_es\_$%.una{clave78/paraDR\_xa+loque{+}Raton\_Think.';

exports.ensureAuth = function(req, res, next){

if( !req.headers.autorization){

return res.status(403).send({message:"la peticion no tiene cabecera de autenticacion"});

}

*//aqui cargamos el token*

var token = req.headers.autorization.replace(/['"]+/g,'');*//);//remplaza cualquier comilla doble o simple por vacio*

try{

*//decodificando el token*

var payload = jwt.decode(token, secret);*/\* el payload es sencible a errores y exepciones que*

*causa que la aplicacion pare por eso lo meto en un try cacth\*/*

if(payload.exp <= moment.unix()){

return res.status(401).send({message:"token Expiro"});

}

}catch(ex){

return res.status(404).send({message:"el token no es valido"});

}

*//adjuntamos el payload a la req para tener siempre a la mano dentro de los controladores el objeto del usuario logeado*

req.user = payload;

next();

}

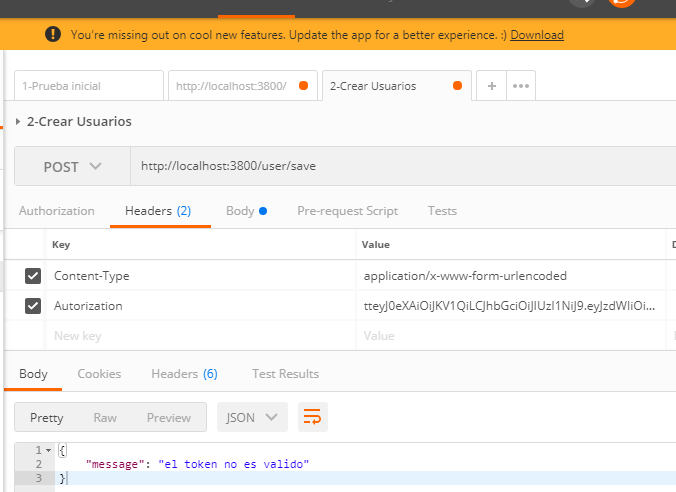
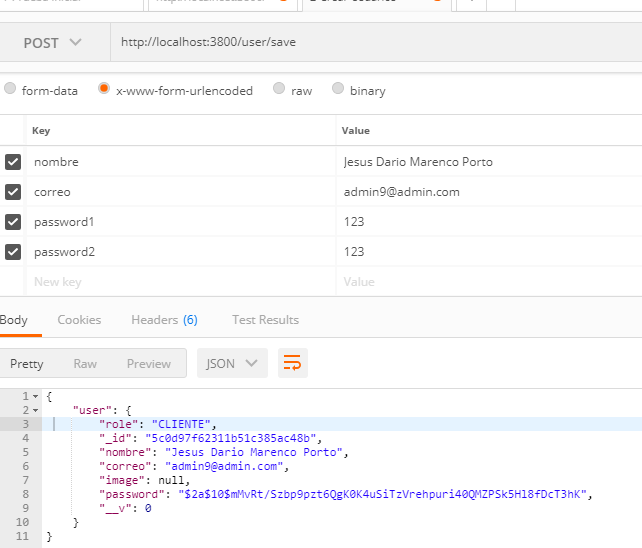
*//Estos middlewares los usaremos en las rutas de modo que deberemos cargarlos allí vamos a userRoutes.js*

Lo importamos y agregamos a la ruta de registro de usuarios

var md\_auth = require('../middlewares/autenticated');

*//Registrar Usuarios*

api.post('/save', md\_auth.ensureAuth, UserController.guardarUser);

Para poder seguir según la ingenieria debemos modificar el modelo de usuario agregando el tercero.

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

*//defino una variable que defina lso roles validos*

var rolesValidos = {

values:['CLIENTE', 'PROVEEEDOR', 'SUPERSU'],

message: '{VALUE} no es un rol permitido'

}

var UserSchema = new Schema({

nombre : { type:String, required:[true, "el nombre es requerido"]},

correo : { type:String, unique:true, required:[true, "el correo es requerido"]},

role : { type:String, required: true, default:"CLIENTE", enum:rolesValidos},

password : {type:String, required:[true, "la contraseña es requerida"]},

tercero :{ type:Schema.ObjectId, ref:'Tercero', required:[true, "Tercero Requerido"]},

image: { type:String}

});

module.exports = mongoose.model('User', UserSchema);

*/\*----------------------------------------------------------------------------*

*nombre: nombre del usuario*

*correo: correo del usuario, este servira para iniciar sesion*

*role:valor que definira el nivel de operacion o restricciones del usuario*

*password:contraseña para poder acceder, debera estar encriiptada*

*------------------------------------------------------------------------------\*/*

Ahora hagamos un crud de terceros para poder añadirlos al usuario.

1-creemos el controlador de terceros llamado terceroController.js

'use strict'

var TerceroModel = require('../models/terceroModel');

function crearTercero(req, res){

var body = req.body;

var Tercero = new TerceroModel();

Tercero.nit = body.nit;

Tercero.nombre = body.nombre;

Tercero.tipo\_tercero = body.tipo\_tercero;

Tercero.direccion = body.direccion;

Tercero.telefono = body.telefono;

Tercero.contacto = body.contacto;

Tercero.descripcion = body.descripcion;

if(body.nit && body.nombre && body.tipo\_tercero ){

Tercero.save((err, TerceroGuardado)=>{

if(err) return res.status(500).send({message:"error al guardar el tercero"});

if(!TerceroGuardado) return res.status(404).send({message:"No se pudo guardad"});

return res.status(200).send({message:"Exito", tercero:TerceroGuardado})

})

}

}

module.exports = {

crearTercero

}

Creamos la ruta terceroRoutes.js

'use strict';

const express = require('express');

var TerceroController = require('../controllers/terceroController');

var md\_auth = require('../middlewares/autenticated');

var api = express.Router();

api.post('/save', TerceroController.crearTercero);

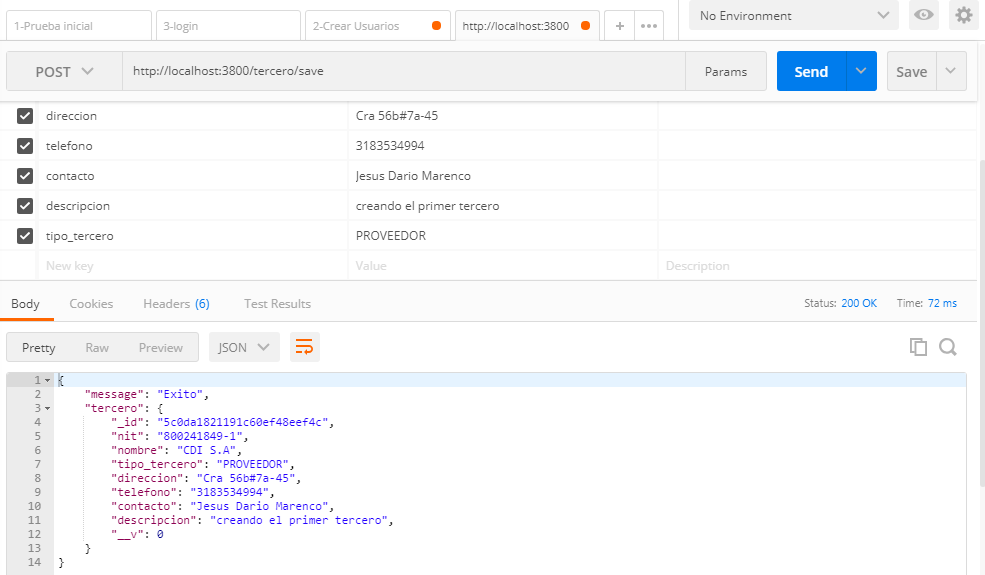
module.exports = api;

añadimos al app.js y creamos la ruta base

var TerceroRoutes = require('./routes/terceroRoutes');

app.use('/tercero', TerceroRoutes);

probamos en postman



Todo ok mas adelante agregaremos el middleware de autenticación, por lo pronto borremos el usuario que tenemos para poder crearlo con un tercero asociado.

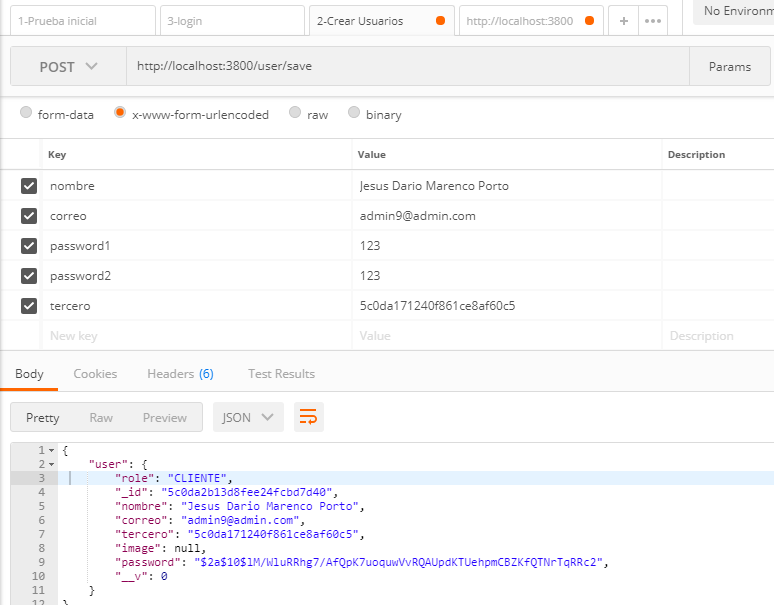
var User = new UserModel();

User.nombre = body.nombre;

User.correo = body.correo;

User.tercero = body.tercero;

User.image = null;



Ahora si ya tenemos un usuario podemos agregar autenticación de token a la creación de terceros en la ruta de terceros y en la de usuarios perfecto

*//creando un tercero*

api.post('/save', md\_auth.ensureAuth, TerceroController.crearTercero);

*//Registrar Usuarios*

api.post('/save', md\_auth.ensureAuth, UserController.guardarUser);

ya tenemos el registrar usuarios y el registrar terceros, hagamos ahora:

1-listar un usuario:

2-listar todos los usuarios

3-borrar un usuario

Empecemos:

**Listar usuario**

*//Listar Un Usuario*

function listUser(req, res){

var id = req.params.id;

console.log(id);

UserModel.findById(id, (err, user)=>{

if(err) return res.status(500).send({message:"error al consultar el usuario"});

if(!user) return res.status(404).send({message: "no se encontro al usuario"});

if(user) return res.status(200).send({user:user});

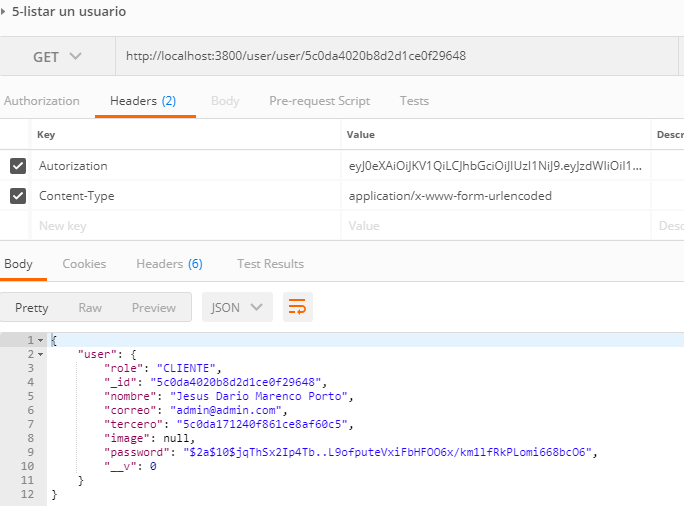
})

}

La ruta

*//Listar un Usuario*

api.get('/user/:id', md\_auth.ensureAuth, UserController.listUser)



**Listemos todos los usuarios**

*//Listar Los Usuarios*

function listUsers(req, res){

UserModel.find({}, (err, users)=>{

if(err) return res.status(500).send({message:"error al consultar los usuarios"});

if(!users) return res.status(404).send({message: "no se encontraron usuarios"});

if(users) return res.status(200).send({users:users});

})

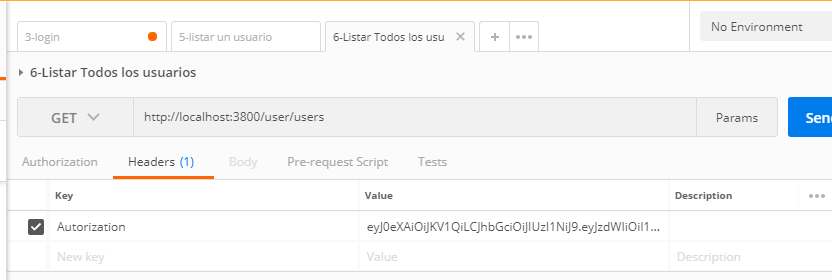
}

La ruta

*//Listar todos los Usuarios*

api.get('/users', md\_auth.ensureAuth, UserController.listUsers)

el resultado



{

"users": [

{

"role": "CLIENTE",

"\_id": "5c0da4020b8d2d1ce0f29648",

"nombre": "Jesus Dario Marenco Porto",

"correo": "admin@admin.com",

"tercero": "5c0da171240f861ce8af60c5",

"image": null,

"password": "$2a$10$jqThSx2Ip4Tb..L9ofputeVxiFbHFOO6x/km1lfRkPLomi668bcO6",

"\_\_v": 0

},

{

"role": "CLIENTE",

"\_id": "5c0da6273964e51058070e07",

"nombre": "Lider Ecopetrol",

"correo": "ecopetrol@ecopetrol.com",

"tercero": "5c0da4f69ca7082594a79a27",

"image": null,

"password": "$2a$10$/1m/27HDjPPWxq0KXFSES.deFbvWq/PTLauanAV8PRc34/jFStRdS",

"\_\_v": 0

},

{

"role": "CLIENTE",

"\_id": "5c0da8ea960a252160989f27",

"nombre": "Usuario para borrar",

"correo": "ecopetrol2@ecopetrol.com",

"tercero": "5c0da4f69ca7082594a79a27",

"image": null,

"password": "$2a$10$X90VMN..vOQF9YlKnq0gu.FmltEk5utzN6MxRmbHcNH3A46dJzYaW",

"\_\_v": 0

}

]

}

**Borremos un usuario: pensemos esto, es mejor no borrar usuarios sino mas bien desactivarlos ya que estos usuarios muy posiblemente estén asociados a otras colecciones entonces en mejor no borrarlos, mas bien cambiémoslos de estado a inactivo**

1-para esto debemos agregar en el modelo de usuario la propiedad estado validando que sea activo o inactivo.

var estadosValidos ={

values:['ACTIVO', 'INACTIVO'],

message:'{VALUES} estado no permitido'

}

var UserSchema = new Schema({

nombre : { type:String, required:[true, "el nombre es requerido"]},

correo : { type:String, unique:true, required:[true, "el correo es requerido"]},

role : { type:String, required: true, default:"CLIENTE", enum:rolesValidos},

password : {type:String, required:[true, "la contraseña es requerida"]},

tercero :{ type:Schema.ObjectId, ref:'Tercero', required:[true, "Tercero Requerido"]},

image: { type:String},

estado:{type:String, enum:estadosValidos}

});

Creamos la función paras actualizar el estado del usuario: userController.js

*//Inactivar Usuario*

function cambioEstadoUsuario(req, res){

var id = req.params.id;

var body = req.body;

UserModel.findById(id, (err, user)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion"});

if(!user) return res.status(404).send({message: "no se encontraro usuario"});

user.estado = body.estado;

user.save((err, usuarioUpdate)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion"});

if(!user) return res.status(404).send({message: "no se encontraro usuario"});

if(user) return res.status(200).send({user:usuarioUpdate});

})

})

}

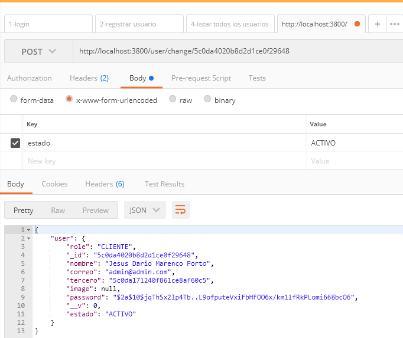
Exportamos la función

Luego creamos la ruta

*//Cambiar el Estado del usuqario*

api.post('/change/:id', md\_auth.ensureAuth, UserController.cambioEstadoUsuario);

probamos

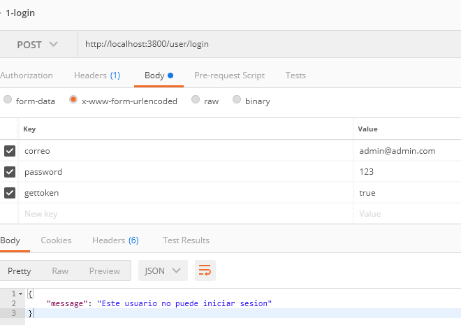


Ahora validamos la función de login para uqe los usuarios inactivos no puedan iniciar sesión

if(user.estado =='INACTIVO'){

return res.status(200).send({message:'Este usuario no puede iniciar sesion'})

probamos EL LOGIN CON UN USUARIO INACTIVO



La validación esta completa.

**Hagamos el update de usuario:**

En el modelo de usuario agregamos un campo mas para identificar al usuario que hizo el cambio

user\_id :{ type:Schema.ObjectId, ref:'Usuario'},

**creamos la function en userController.js**

*//Actualizar un Usuario para el administrador*

function updateUser(req, res){

var id = req.params.id;

var body = req.body;

UserModel.findById(id)

.populate('tercero', 'nombre')

.exec((err, user)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion"});

if(!user) return res.status(404).send({message: "no se encontraro usuario"});

if(user){

user.nombre = body.nombre;

user.correo = body.correo;

user.tercero = body.tercero;

user.role = body.role;

user.user\_id = req.user.sub;

user.save((err, UserActualizado)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion"});

if(!UserActualizado) return res.status(404).send({message: "no se pudo actualizar"});

if(UserActualizado) return res.status(200).send({message: "Exitoo", user:UserActualizado});

});

}

});

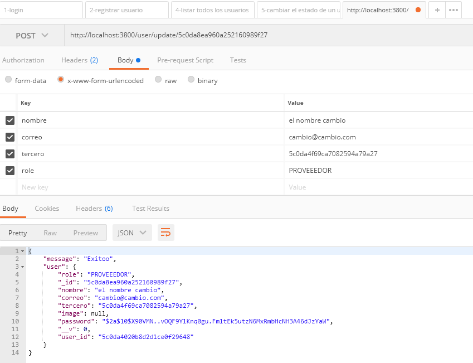
}

Creamos la ruta

*//Actualizar un usuario*

api.post('/update/:id', md\_auth.ensureAuth, UserController.updateUser);

hacemos las pruebas.



Sigamos el controlador de terceros.

1-listemos un tercero

En terceroController.js creemos la función

*//Listar un tercero*

function listTercero(req, res){

var id = req.params.id;

TerceroModel.findById(id, (err, tercero)=>{

if(err) return res.status(500).send({message:"error al realizar operacion"});

if(!tercero) return res.status(404).send({message:"no se encontro tercero"});

return res.status(200).send({message:"Exito", tercero:tercero})

})

}

Exportamos la function

module.exports = {

crearTercero,

listTercero

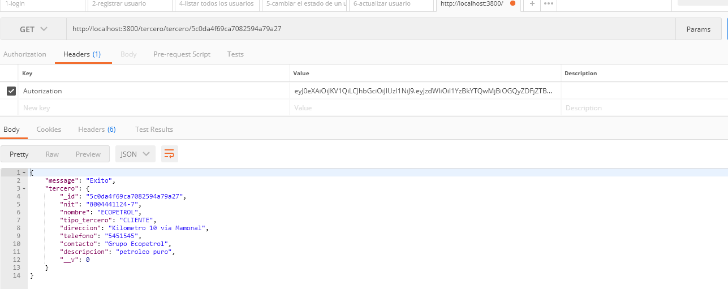
}

Creamos la ruta:

*//lListar un tercero*

api.get('/tercero/:id', md\_auth.ensureAuth, TerceroController.listTercero);

probamos:



Listemos todos los terceros creemos la funcion

*//Listar todos los terceros*

function listarTerceros(req, res){

TerceroModel.find({}).exec((err, terceros)=>{

if(err) return res.status(500).send({message:"error al realizar operacion"});

if(!terceros) return res.status(404).send({message:"no se encontraron terceros"});

return res.status(200).send({message:"Exito", terceros:terceros})

})

}

Exportamos

module.exports = {

crearTercero,

listTercero,

listarTerceros

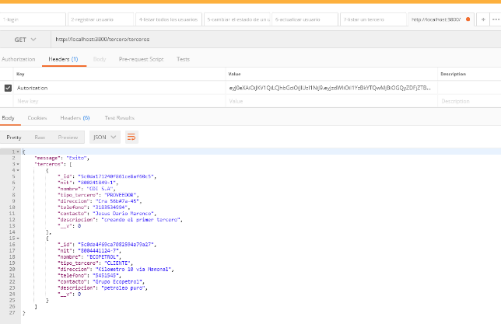
}

Creamos la ruta

*//Listar todos los terceros*

api.get('/terceros', md\_auth.ensureAuth, TerceroController.listarTerceros);

probamos en postman



Cambiemos el estado de los terceros para eso hay que agregar una propiedad en el modelo de terceros llamado estado y otro llamado user\_id para capturar al usuario que lo ha modificado.

user\_id :{ type:Schema.ObjectId, ref:'Usuario'},

estado: {type:String}

creemos la función en el controlador de terceros

*//Cambiar estado de tercero*

function changeEstadoTercero(req, res){

var id = req.params.id;

var body = req.body;

TerceroModel.findById(id, (err, tercero)=>{

if(err) return res.status(500).send({message:"error al realizar operacion"});

if(!tercero) return res.status(404).send({message:"no se encontro tercero"});

tercero.estado = body.estado;

tercero.user\_id = req.user.sub;

tercero.save((err, terceroChange)=>{

if(err) return res.status(500).send({message:"error al realizar operacion"});

if(!terceroChange) return res.status(404).send({message:"no pudo cambiar"});

return res.status(200).send({message:"Exito", terceroChange:terceroChange});

});

});

}

Exportamos la función

module.exports = {

crearTercero,

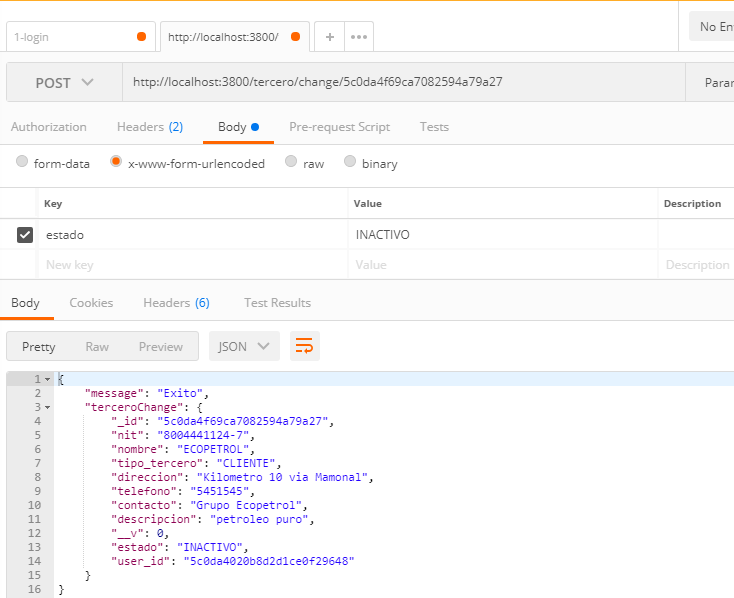
listTercero,

listarTerceros,

changeEstadoTercero

}

Probamos en postman

perfecto sigamos

Hagamos lo mismo con el modelo de cargos.(creamos el controlador cargoController.js)

1-añadamos al modelo de cargos dos propiedades la user\_id y el estado(validado a activo e inactivo).

var estadosValidos ={

values:['ACTIVO', 'INACTIVO'],

message:'{VALUES} estado no permitido'

}

user\_id :{ type:Schema.ObjectId, ref:'Usuario'},

estado:{type:String, enum:estadosValidos}

importamos lo necesario, creamos la función, exportamos la función

'use strict'

var CargoModel = require('../models/cargoModel');

function crearCargo(req, res){

var body = req.body;

if(body.nombre && body.vhora\_hombre){

var cargo = new CargoModel();

cargo.nombre = body.nombre;

cargo.vhora\_hombre = body.vhora\_hombre

cargo.user\_id = req.user.sub;

cargo.estado = body.estado;

cargo.save((err, cargoSave)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!cargoSave) return res.status(404).send({message: "no se pudo actualizar"});

if(cargoSave) return res.status(200).send({message: "Exitoo", cargo:cargoSave});

});

}

}

module.exports = {

crearCargo

}

Ahora creamos la ruta

'use strict'

const express = require('express');

var api = express.Router();

var md\_auth = require('../middlewares/autenticated');

var CargoController = require('../controllers/cargoController');

*//Crear Cargo*

api.post('/save', md\_auth.ensureAuth, CargoController.crearCargo)

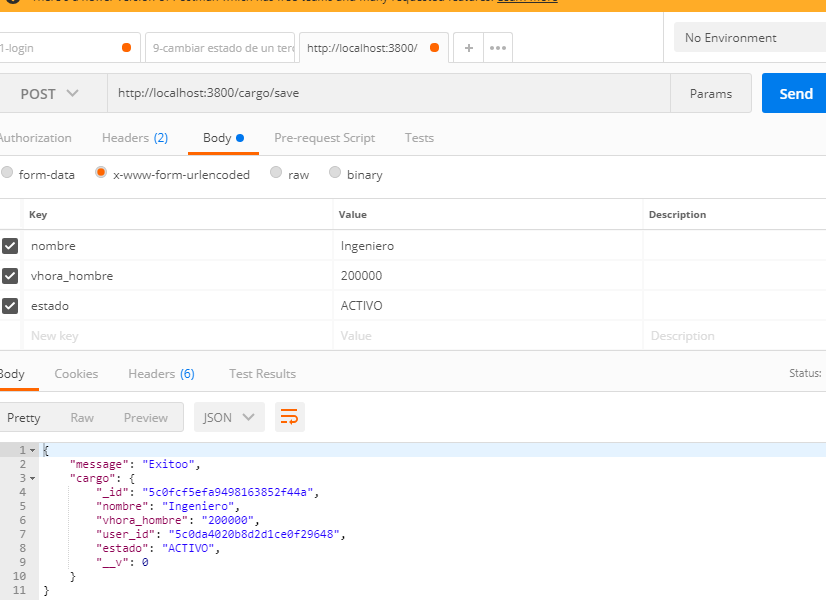
module.exports = api;

añadimos a app.js:

var CargoRoutes = require('./routes/cargoRouter');

app.use('/cargo', CargoRoutes);

probamos

perfecto

Creemos la function update cargos(que incluya cambiarle el estado al cargo) hay que crear los cargos por cliente ya que son ellos los que definen el valor de la hora hombre por tanto ya que agregar el campo tercero al modelo de cargo y añadir el tercero al momento de la creación del cargo.

Listo ya quedo corregido sigamos con el update cargos.

*//Update cargo(incluye la actualziacion del estado*

function updateCargos(req, res){

var id = req.params.id;

var body = req.body;

CargoModel.findById(id, (err, cargo)=>{

cargo.nombre = body.nombre;

cargo.vhora\_hombre = body.vhora\_hombre;

cargo.estado = body.estado;

cargo.user\_id = req.user.sub;

cargo.tercero = body.tercero;

cargo.save((err, cargoUpdate)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!cargoUpdate) return res.status(404).send({message: "no se pudo actualizar"});

if(cargoUpdate) return res.status(200).send({message: "Exitoo", cargoUpdate:cargoUpdate});

});

});

}

Exportemos

module.exports = {

crearCargo,

updateCargos

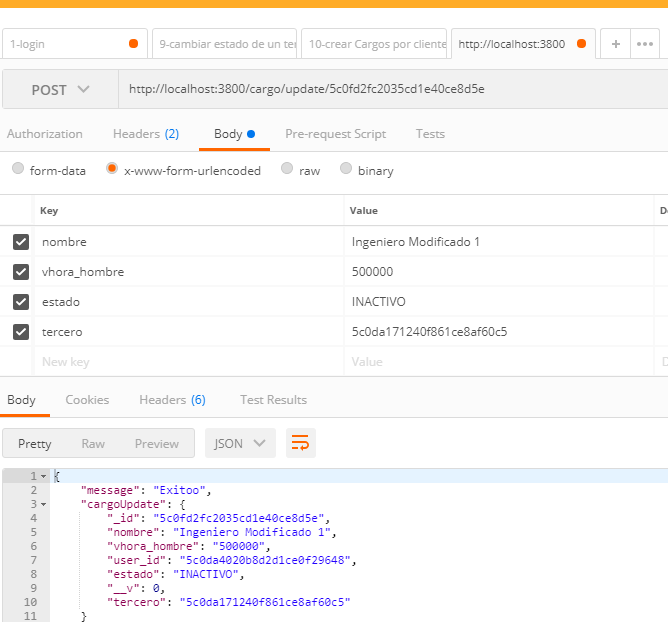
}

La ruta

*//Update Cargo*

api.post('/update/:id', md\_auth.ensureAuth, CargoController.updateCargos)

probemos

perfecto

Ahora listemos los cargos

Creemos la función

*//listar los cargos en orden alfabetico*

function listarCargos(req, res){

CargoModel.find({}).exec((err, cargos)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!cargos) return res.status(404).send({message: "no se pudo generar"});

if(cargos) return res.status(200).send({message: "Exitoo", cargos:cargos});

});

}

Expoertamos la function

module.exports = {

crearCargo,

updateCargos,

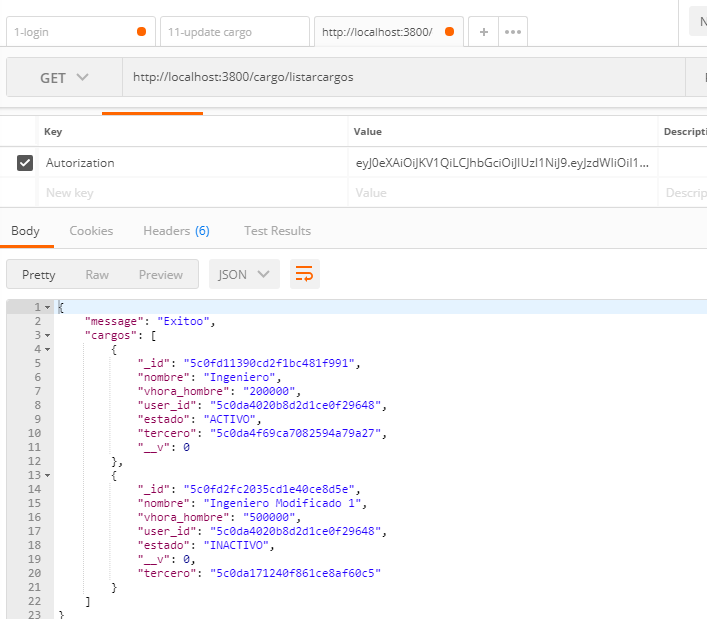
listarCargos

}

Creamos la ruta

api.get('/listarcargos', md\_auth.ensureAuth, CargoController.listarCargos)

probamos

todo ok

Listemos un solo cargo:

1-function

*//Listar un solo cargo*

function listCargo(req, res){

var id = req.params.id

CargoModel.findById(id).exec((err, cargo)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!cargo) return res.status(404).send({message: "no se pudo generar"});

if(cargo) return res.status(200).send({message: "Exitoo", cargo:cargo});

});

}

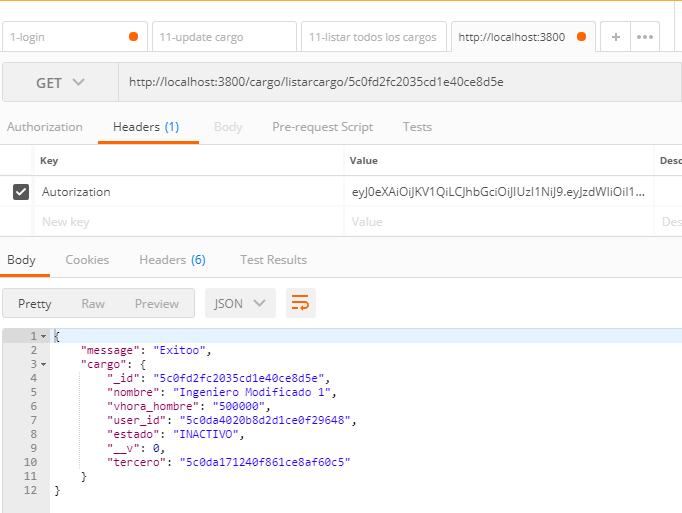
2-Exportamos

3-creamos la ruta

*//Listar un solo cargo*

api.get('/listarcargo/:id', md\_auth.ensureAuth, CargoController.listCargo);

4-probamos

exitooo

**Creemos el controlador y rutas de ubicaciones físicas**

1-crear el controlador ubicacionController.js

2-importar lo necesario

3-crear la función

4-exportar la función

5-crear la ruta

6-incluir en el app.js

7-probar

**Desarrollo**

1-modificare el modelo un poco para capturar los datos de quien modifique las ubicaciones físicas

user\_modificador :{ type:Schema.ObjectId, ref:'Usuario'}

viene la function:

'use strict'

var UbicacionFisicaModel = require('../models/ubicacionFisicaModel');

var moment = require('moment');

*//Crear cargo*

function crearUbicacionFisica(req, res){

var body = req.body;

if(body.nombre){

var UbicacionFisica = new UbicacionFisicaModel();

UbicacionFisica.nombre = body.nombre;

UbicacionFisica.tercero = body.tercero;

UbicacionFisica.fecha\_creacion = moment().unix()

UbicacionFisica.save((err, ubicacion)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!ubicacion) return res.status(404).send({message: "no se pudo crear"});

if(ubicacion) return res.status(200).send({message: "Exitoo", ubicacion:ubicacion});

});

}

}

Ruta

'use strict'

const express = require('express');

var api = express.Router();

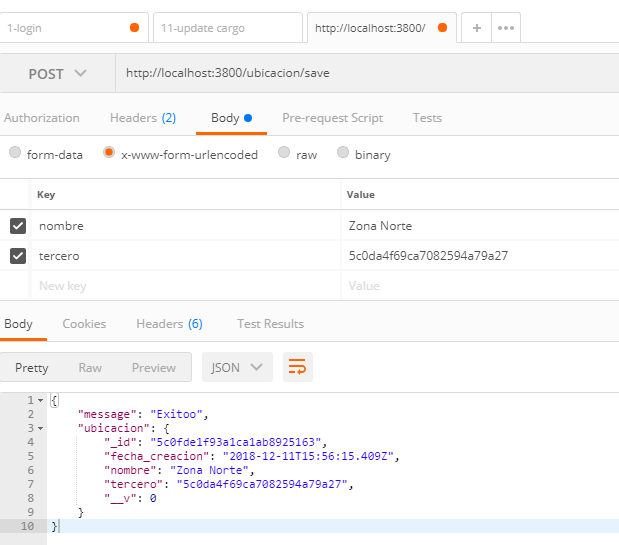
var md\_auth = require('../middlewares/autenticated');

var UbicacionController = require('../controllers/ubicacionFisica');

*//Crear ubicacion fisica*

api.post('/save', md\_auth.ensureAuth, UbicacionController.crearUbicacionFisica);

prueba

éxito

Vamos a listar todas las ubicaciones físicas en orden alfabetico

*//Listar las ubicaciones fisicas en orden alfabetico*

function listarUbicaciones(req, res){

UbicacionFisicaModel.find({}).sort({nombre: 1}).exec((err, ubicaciones)=>{

if(err) return res.status(500).send({message:"error al procesar la peticion", error:err});

if(!ubicaciones) return res.status(404).send({message: "no se pudo generar"});

if(ubicaciones) return res.status(200).send({message: "Exitoo", ubicaciones:ubicaciones});

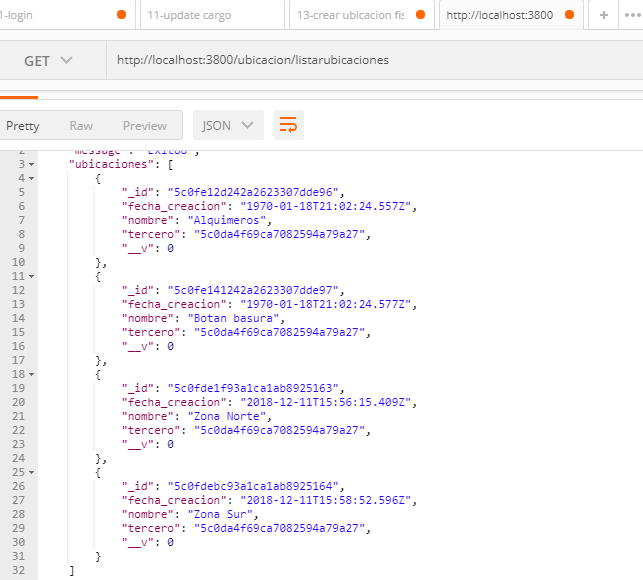
});

}

La ruta

api.get('/listarubicaciones', md\_auth.ensureAuth, UbicacionController.listarUbicaciones);

la prueba:



Función para mostrar una sola ubicación física

*//Listar una sola ubicacion fisica*

function listarUbicacion(req, res){

var id = req.params.id

UbicacionFisicaModel.findById(id).exec((err, ubicacion)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!ubicacion) return res.status(404).send({message: "no se pudo generar"});

if(ubicacion) return res.status(200).send({message: "Exitoo", ubicacion:ubicacion});

});

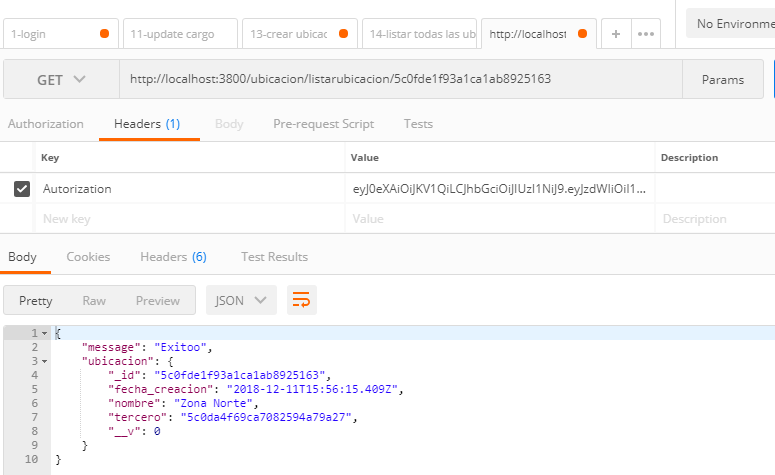
}

La ruta:

*//Listar una sola ubicacion*

api.get('/listarubicacion/:id', md\_auth.ensureAuth, UbicacionController.listarUbicacion);

la prueba

exitoo

Creemos la función de actualizar ubicación física: agregue unna nueva propiedad en el modelo llamada estado y esta validada

*//Update cargo(incluye la actualziacion del estado*

function updateUbicacion(req, res){

var id = req.params.id;

var body = req.body;

UbicacionFisicaModel.findById(id, (err, ubicacion)=>{

ubicacion.nombre = body.nombre;

ubicacion.user\_modificador = req.user.sub;

ubicacion.estado = body.estado;

ubicacion.save((err, ubicacionUpdate)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!ubicacionUpdate) return res.status(404).send({message: "no se pudo actualizar"});

if(ubicacionUpdate) return res.status(200).send({message: "Exitoo", ubicacionUpdate:ubicacionUpdate});

});

});

}

RUTA

*//Update Cargo*

api.post('/update/:id', md\_auth.ensureAuth, UbicacionController.updateUbicacion);

Creemos el controlador, y la ruta de Equipos

1-crear el controlador equipoController.js

2-importar lo necesario

3-crear la función

4-exportar la función

5-crear la ruta

6-incluir en el app.js

7-probar

**Desarrollo**

Lo primero es modificar el modelo de equipo y agregar también la propiedad estado(ACTIVO E INACTIVO) y ubicación(para saber donde se encuentra)

tag:{type:String, unique:true, required:[true, 'El tag para el equipo es necesario']},

nombre\_equipo:{type:String, required:[true, 'El nombre del equipo es requerido']},

fecha\_creacion:{type:String, required:[true, 'La fecha de creacion es requerida']},

usuario\_creador :{type:String, required:[true, 'El usuario que crea el equipo es requerido']},

tercero:{type:String, required:[true, 'El tercero dueño del equipo es requerido']},

estado:{type:String, enum:estadosValidos},

ubicacion :{ type:Schema.ObjectId, ref:'Ubicacionfisica'}

la función crear en el controlador

*//Crear un equipo*

function saveEquipo(req, res){

var body = req.body;

var Equipo = new EquipoModel();

if(body.tag && body.nombre){

Equipo.tag = body.tag;

Equipo.nombre\_equipo = body.nombre;

Equipo.usuario\_creador = req.user.sub;

Equipo.tercero = body.tercero;

Equipo.ubicacion = body.ubicacion;

Equipo.save((err, equipoGuardado)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!equipoGuardado) return res.status(404).send({message: "no se pudo actualizar"});

if(equipoGuardado) return res.status(200).send({message: "Exitoo", equiponew:equipoGuardado});

});

}

}

La ruta

'use strict'

const express = require('express');

var api = express.Router();

var md\_auth = require('../middlewares/autenticated');

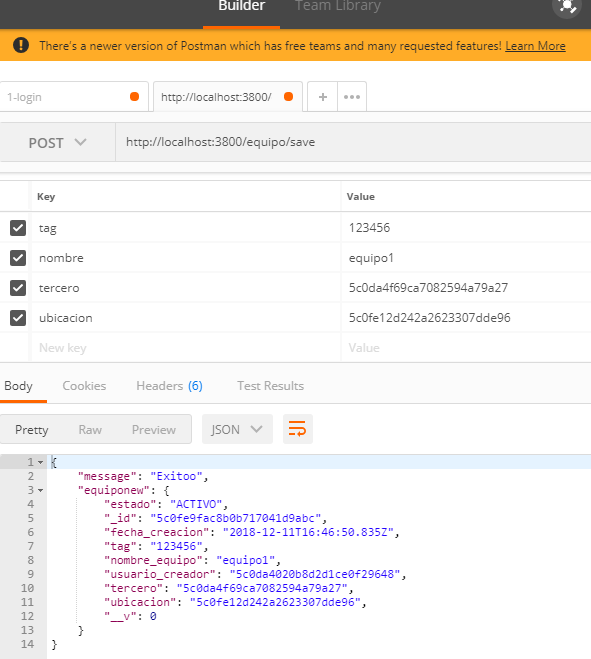
var EquipoController = require('../controllers/equipoController');

*//Crear Equipos*

api.post('/save', md\_auth.ensureAuth, EquipoController.saveEquipo);

module.exports = api;

probamos:

exitoooo

Ahora listemos todos los equipos

La función

*//Listar todos los equiposen orden alfabetico*

function listarEquipos(req, res){

EquipoModel.find({}).sort({nombre\_equipo: 1}).exec((err, equipos)=>{

if(err) return res.status(500).send({message:"error al procesar la peticion", error:err});

if(!equipos) return res.status(404).send({message: "no se pudo generar"});

if(equipos) return res.status(200).send({message: "Exitoo", equipos:equipos});

});

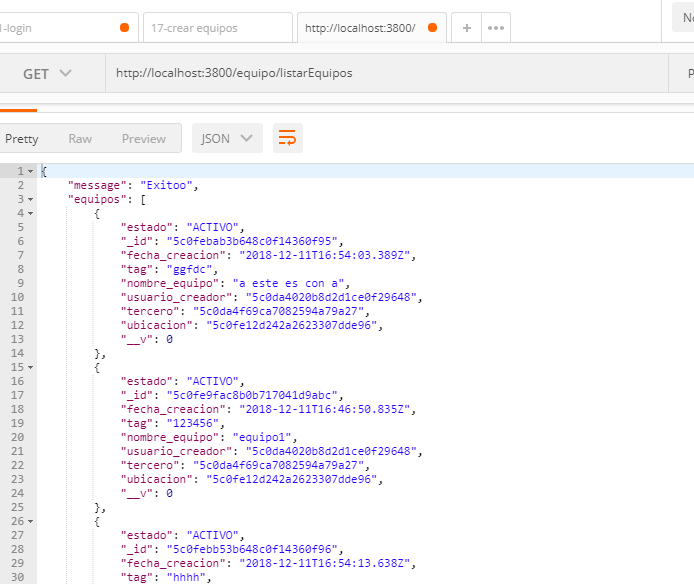
}

La ruta

*//Crear Equipos*

api.get('/listarEquipos', md\_auth.ensureAuth, EquipoController.listarEquipos);

la prueba

Exitooo

Ahora listemos solo un equipo

La function

*//Listar un solo equipo*

function listarUnEquipo(req, res){

var id = req.params.id

EquipoModel.findById(id).exec((err, equipo)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!equipo) return res.status(404).send({message: "no se pudo generar"});

if(equipo) return res.status(200).send({message: "Exitoo", equipo:equipo});

});

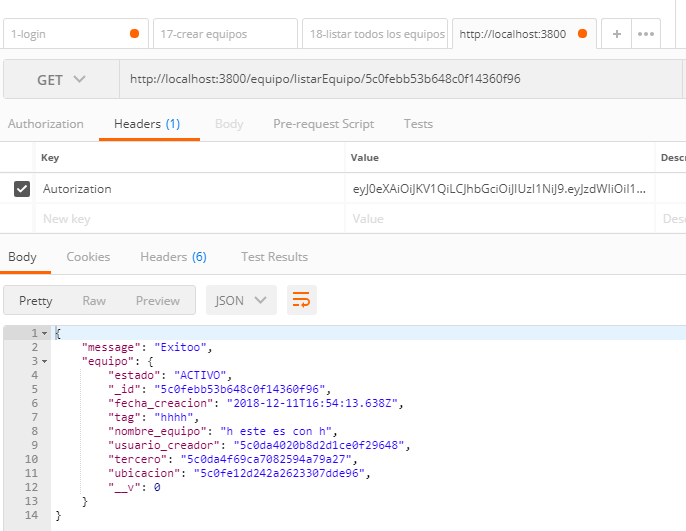
}

La ruta

*//Listar un Equipo*

api.get('/listarEquipo/:id', md\_auth.ensureAuth, EquipoController.listarUnEquipo);

la prueba

Exitooo

Ahora la function de actualizar Equipo

*//Actualziar los datos de un equipo*

function updateEquipo(req, res){

var id = req.params.id;

var body = req.body;

EquipoModel.findById(id, (err, equipo)=>{

equipo.nombre\_equipo = body.nombre\_equipo;

equipo.tercero = body.tercero;

equipo.estado = body.estado;

equipo.save((err, equipoUpdate)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!equipoUpdate) return res.status(404).send({message: "no se pudo actualizar"});

if(equipoUpdate) return res.status(200).send({message: "Exitoo", equipoUpdate:equipoUpdate});

});

});

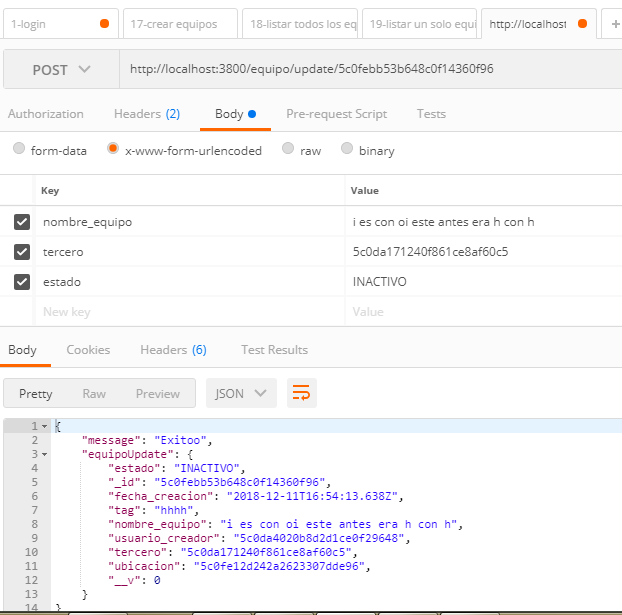
}

LA RUTA

*//Update Equipo*

api.post('/update/:id', md\_auth.ensureAuth, EquipoController.updateEquipo);

la prueba

Exitooooo

**Trabajemos creando el controlador y las rutas del Modelo de Consecutivos.**

Creemos la función para crear consecutivos.

'use strict'

var ConsecutivoModel = require('../models/consecutivoModel');;

*//Crear Consecutivos*

function crearConsecuticos(req, res){

var body = req.body;

var Consecutivo = new ConsecutivoModel();

if(prefijo && tercero){

Consecutivo.prefijo = body.prefijo;

Consecutivo.tercero = body.tercero;

Consecutivo.save((err, ConsecutivoCreado)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion"});

if(!ConsecutivoCreado) return res.status(404).send({message: "no se pudo actualizar"});

if(ConsecutivoCreado) return res.status(200).send({message: "Exitoo", consecutivo:ConsecutivoCreado});

});

}else{

return res.status(200).send({message: "Ingrese lso campos requeridos"});

}

}

module.exports = {

crearConsecuticos

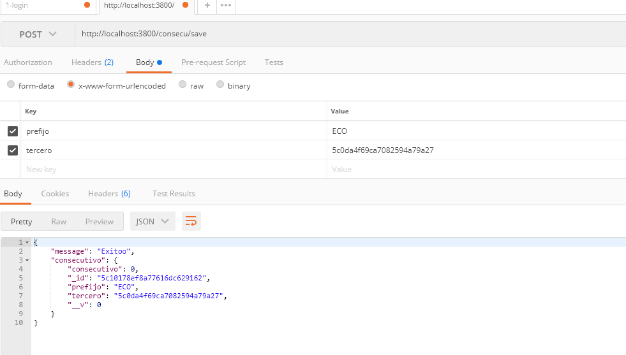
}

La ruta

*//Crear Consecutivos*

api.post('/save', md\_auth.ensureAuth, ConsecutivoController.crearConsecutivos);

la prueba

Exitooo

Hagamos la function para listar todos los consecutivos

*//listar todos los consecutivos en order alfabetico*

function listarConsecutivos(req, res){

ConsecutivoModel.find({}).sort({prefijo: 1}).exec((err, consecutivos)=>{

if(err) return res.status(500).send({message:"error al procesar la peticion", error:err});

if(!consecutivos) return res.status(404).send({message: "no se pudo generar"});

if(consecutivos) return res.status(200).send({message: "Exitoo", consecutivos:consecutivos});

});

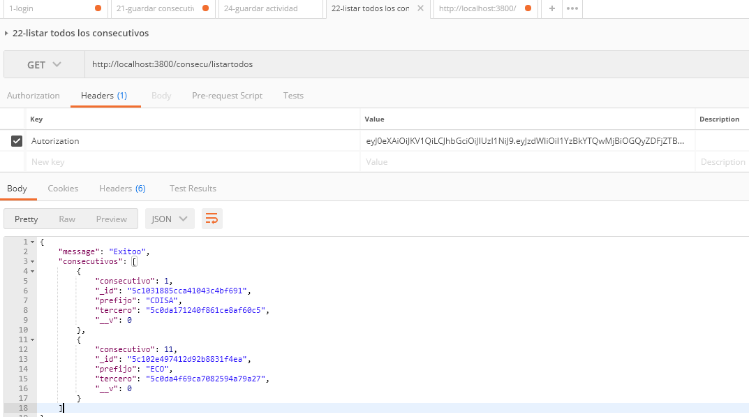
}

La ruta

*//Listar todos los consecutivos*

api.get('/listartodos', md\_auth.ensureAuth, ConsecutivoController.listarConsecutivos);

la prueba

exitooo

Creemos la function para listar un solo consecutivo

*//listar un solo consecutivo*

function listarUnConsecutivo(req, res){

var id = req.params.id

ConsecutivoModel.findById(id).exec((err, consecutivo)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!consecutivo) return res.status(404).send({message: "no se pudo generar"});

if(consecutivo) return res.status(200).send({message: "Exitoo", consecutivo:consecutivo});

});

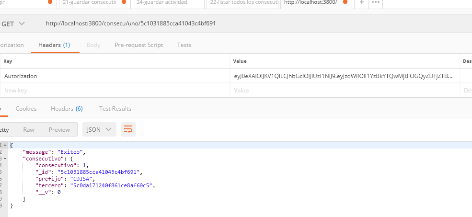
}

La ruta

*//Listar un solo consecutivo*

api.get('/uno/:id', md\_auth.ensureAuth, ConsecutivoController.listarUnConsecutivo);

la prueba

exitooooo

**Creemos el controlador de Actividad**

1-crear archivo controlador actividadController.js y archive de ruta **activitadRouter.js**.

2-se me olvidaba hay que crear también el modelo de actividad ya que aun no esta creado.

El modelo:

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var estadosValidos ={

values:['ACTIVA', 'DETENIDA', 'EJECUCION', 'COMPLETADA', 'ELIMINADA'],

message:'{VALUES} estado no permitido'

}

var ActividadSchema = new Schema({

consecutivo : {type:Number, required:[true, "el consecutivo es requerido"]},

nombre : {type:String, required:[true, "el nombre es requerido"]},

equipo : {type:Schema.ObjectId, ref:'Equipo', required:[true, "el equipo es requerido"] },

ubicacion:{type:Schema.ObjectId, ref: 'Ubicacionfisica', required:[true, "la ubicacion es requerida"]},

solicitante :{type:String}, *//persona que solicita la actividad*

tercero :{type:Schema.ObjectId, ref:'Tercero', required:[true, "el tercero es requerido"]},

usuario\_creador :{type:Schema.ObjectId, ref:'Usuario', required:[true, "el usuario creador es requerido"]},

fecha\_creacion :{type:Date, default:Date.now},

fecha\_requerida:{type:Date},

*//cargos:{type:Array, required:true},*

*//horas\_hombre :{type:Array, required:true},*

valor : {type:Number},

estado : {type:String, default:'ACTIVA', enum:estadosValidos, required:[true, "el estado es requerido"]}

});

module.exports = mongoose.model('Actividad', ActividadSchema);

la function de crear actividad

*//Crear Actividad*

function saveActividad(req, res){

var body = req.body;

if(body.nombre && body.equipo && body.ubicacion && body.solicitante && body.tercero){

var Actividad = new ActividadModel();

Actividad.nombre = body.nombre;

Actividad.equipo = body.equipo;

Actividad.ubicacion = body.ubicacion;

Actividad.solicitante = body.solicitante;

Actividad.tercero = body.tercero;

Actividad.usuario\_creador = req.user.sub;

*//Actividad.fecha\_requerida = new Date(body.fecha\_requerida)//dia/mes/año*

*//Actividad.cargos.push(body.cargo);*

*//Actividad.horas\_hombre.push(body.horas);*

ConsecutivoModel.findOne({tercero:body.tercero})

.exec((err, consecutivo)=>{

if(err) return res.status(500).send({message:"error al consultar el consecutivo", error:err});

if(!consecutivo) return res.status(404).send({message: "no se encontro el consecutivo"});

console.log("aqui viene el consecutivo");

Actividad.consecutivo = consecutivo.consecutivo;

consecutivo.consecutivo++;

Actividad.save((err, ActividadGuardada)=>{

if(err) return res.status(500).send({message:"error al guardar la actividad", error:err});

if(!ActividadGuardada) return res.status(404).send({message: "no se guardo la Actividad"});

consecutivo.save((err, actConsecutivo)=>{

if(err) return res.status(500).send({message:"error al actualizar la consecutivo", error:err});

if(!actConsecutivo) return res.status(404).send({message: "no se actualizo el consecutivo"});

return res.status(200).send({

message: "Exito",

prefijo :consecutivo.prefijo,

actividad : Actividad

});

})

});

});

}else{

return res.status(200).send({message:"Suministre todos los campos requeridos"});

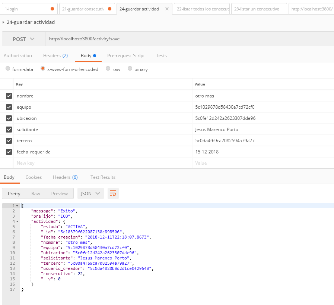
}

La ruta

*//Crear Actividad*

api.post('/save', md\_auth.ensureAuth, ActividadController.saveActividad);

la prueba:

Exitoooo

La función de listar todas las actividades

*//Listar todas las actividades por consecutivo*

function listarActividades(req, res){

ActividadModel.find({}).sort({consecutivo: 1}).exec((err, actividades)=>{

if(err) return res.status(500).send({message:"error al procesar la peticion", error:err});

if(!actividades) return res.status(404).send({message: "no se pudo generar"});

if(actividades) return res.status(200).send({message: "Exitoo", actividades:actividades});

});

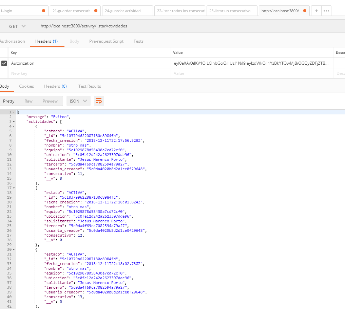
}

La ruta:

*//Listar todas las actividades*

api.get('/listarActividades', md\_auth.ensureAuth, ActividadController.listarActividades);

la prueba

todo ok

La function de listar solo una actividad

*//listar solo una actividad*

function listarActividad(req, res){

var id = req.params.id

ActividadModel.findById(id).exec((err, acti)=>{

if(err) return res.status(500).send({message:"error aprocesar la peticion", error:err});

if(!acti) return res.status(404).send({message: "no se pudo generar"});

if(acti) return res.status(200).send({message: "Exitoo", actividad:acti});

});

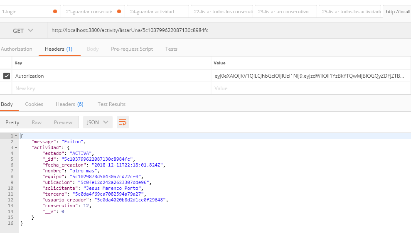
}

La ruta

*//Listar una actividad*

api.get('/listarUna/:id', md\_auth.ensureAuth, ActividadController.listarActividad);

la prueba

Exitoooooo

**Estamos listos para el Frontend**

1-Ng new sicifront = crea el proyecto frontend con ayuda de angular cli

2-vamos al package.json y agregamos jquery 3.2.1 y bootstrap 4.0.0

3-ejecutamos npm update para uqe se instalen estas dos dependencias

4-vamos a trabajar con angular material asi que debemos instalarlo

npm install --save @angular/material @angular/cdk @angular/animations

5-una vez instalamos procedemos a configurarlo en el app.module.ts (importamos lo siguiente)

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

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

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

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

import {MatButtonModule, MatCheckboxModule, MatFormFieldModule, MatInputModule, MatAutocompleteModule} from '@angular/material';

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

los agregamos a los imports

imports: [

BrowserModule,

BrowserAnimationsModule,

MatButtonModule,

MatCheckboxModule,

MatFormFieldModule,

MatInputModule,

MatAutocompleteModule,

ReactiveFormsModule,

FormsModule

],

Incluimos el tema uno de los temas de angularMaterial nuestro css principal

*/\* You can add global styles to this file, and also import other style files \*/*

@import "~@angular/material/prebuilt-themes/indigo-pink.css";

Instalamos hammer.js para otros plugins que podriamos usar después

npm install --save hammerjs

Incluimos la fuente e iconos en el index.html

*<*link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet"*>*

*<*link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500" rel="stylesheet"*>*

Ahora configuremos jquery , bootstrap, y hammersjs en el archivo **angular.json**

"styles": [

"src/assets/css/styles.css",

"node\_modules/bootstrap/dist/css/bootstrap.min.css"

],

"scripts": [

"node\_modules/jquery/dist/jquery.min.js",

"node\_modules/hammerjs/hammer.min.js",

"node\_modules/bootstrap/dist/js/bootstrap.min.js"

]

**Estamos listos**

1-en el app.component.html agregaremos un navbar simple, lasa demás opciones estarán en el body.(cuando el usuario este logueado aparecerá la barra y cuando no solo la imagen de fondo y el body de presentacion)

*<*nav class="navbar "*>*

*<*div class="navbar-header"*>*

*<*img class="logo" src="assets/image/logo/logo.png" alt=""*>*

*</*div*>*

*<*ul class="nav justify-content-end"*>*

*<*li class="nav-item"*>*

*<*a class="nav-link" href="#"*>*Active*</*a*>*

*</*li*>*

*<*li class="nav-item dropdown"*>*

*<*a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false"*>*nombre de usuario*</*a*>*

*<*div class="dropdown-menu"*>*

*<*a class="dropdown-item" href="#"*>*Action*</*a*>*

*<*a class="dropdown-item" href="#"*>*Another action*</*a*>*

*<*a class="dropdown-item" href="#"*>*Something else here*</*a*>*

*<*div class="dropdown-divider"*></*div*>*

*<*a class="dropdown-item" href="#"*>*Separated link*</*a*>*

*</*div*>*

*</*li*>*

*</*ul*>*

*</*nav*>*

El css

.logo{

height: 30px;

}

.nav{

border: 1px solid transparent;

border-top-left-radius: .25rem;

border-top-right-radius: .25rem;

color: black !important;

}

.nav-item:hover{

border: 1px solid black;

}

a{

color: black !important;

}

.navbar{

background-color: #fff !important;

border: 1px solid rgba(0, 0, 0, 0.1);

}

**Creando el archivo de rutas:**

creemos el routing de la aplicacion para utilizar las rutas y en function de la url se cargue un component u otro. Dentro de app vamos a crear un fichero llamado app.routing.ts que se encargara de esto.

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

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

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

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

import { InicioComponent } from './components/inicio/inicio.component';

const appRoutes : Routes = [

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

{path: 'login', component: LoginComponent}

];

export const appRoutingProviders : any[]=[];

export const routing : ModuleWithProviders = RouterModule.forRoot(appRoutes)

importamos las constantes creadas en el routing al app.module.ts(el routing a los imports ya que es un modulo, y el appRoutingProvider en los prividers ya que es un servicio )

imports: [

HttpClientModule,

BrowserModule,

BrowserAnimationsModule,

MatButtonModule,

MatCheckboxModule,

MatFormFieldModule,

MatInputModule,

MatAutocompleteModule,

ReactiveFormsModule,

FormsModule,

routing

],

providers: [

appRoutingProviders

],

Agreguemos el selector en el app.component.html

*<*router-outlet*></*router-outlet*>*

Con esto se activa el routing y podemos navegar en la aplicación

Creemos el componente de inicio es decir InicioComponent

En el html

*<*div class="fondo col-lg-12"*>*

*</*div*>*

*<*div*>*

*<*div class="col-md-12 contenido"*>*

*<*div class="row"*>*

*<*div class="col-md-8 offset-md-2 col-sm-12"*>*

*<*section*>*

*<*div class="row"*>*

*<*div class="col-md-6 col-sm-12"*>*

*<*p*>*

*<*img class="img-fluid" src="../assets/image/bi2.png" alt=""*>*

*</*p*>*

*<*button [routerLink]="['/login']" mat-flat-button color="primary"*>*Ingresar*</*button*>*

*</*div*>*

*<*div class="col-md-6 col-sm-12"*>*

*<*h1*>*Bienvenido a SICI*</*h1*>*

*<*p*>*

SICI que se define como un Sistema Integrado de Confiabilidad Industrial,basado en una plataforma totalmente cloud de alta disponibilidad y rendimiento. SICI se encarga de administrar y controlar ordenes de servicio, analizar y procesar la informacion, informar a los distintos departamentos de la empresa y generar informacion de primera mano en tiempo real.

*</*p*>*

*</*div*>*

*</*div*>*

*</*section*>*

*</*div*>*

*</*div*>*

*</*div*>*

El css

*/\*fondo del componente principal\*/*

.fondo{

background-image: url(../../../assets/image/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 100% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

*/\*contenido\*/*

.contenido{

z-index: 2;

position: absolute;

}

section{

margin-top: 30vh;

}

p{

text-align: justify;

}

button{

width: 100%;

}

@media (max-width: 767px) {

.fondo{

background-image: url(../../../assets/image/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size:cover;

height: 100% !important;

opacity: 0.2px;

z-index: 1;

position: absolute;

}

section{

margin-top: 0px !important;

}

}

La ruta

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

La prueba



Creemos el componente de loginComponent

El html

*<*div class="fondo col-lg-12"*>*

*</*div*>*

*<*div class="wrapper fadeInDown contenido"*>*

*<*div id="formContent"*>*

*<!-- Tabs Titles -->*

*<!-- Icon -->*

*<*div class="fadeIn first"*>*

*<*img src="../../../assets/img/logo/logo.png" id="icon" alt="User Icon" */>*

*<*h1*>*Ingresar*</*h1*>*

*</*div*>*

*<!-- Login Form -->*

*<*form*>*

*<*input type="text" id="login" class="fadeIn second" name="login" placeholder="username"*>*

*<*input type="text" id="password" class="fadeIn third" name="login" placeholder="password"*>*

*<*input type="submit" class="fadeIn fourth" value="Log In"*>*

*</*form*>*

*<!-- Remind Passowrd -->*

*<*div id="formFooter"*>*

*</*div*>*

*</*div*>*

*</*div*>*

El css

*/\* BASIC \*/*

html {

background-color: #56baed;

}

body {

font-family: "Poppins", sans-serif;

height: 100vh;

}

a {

color: #92badd;

display:inline-block;

text-decoration: none;

font-weight: 400;

}

h2 {

text-align: center;

font-size: 16px;

font-weight: 600;

text-transform: uppercase;

display:inline-block;

margin: 40px 8px 10px 8px;

color: #cccccc;

}

*/\*contenido\*/*

.contenido{

z-index: 2;

position: absolute;

}

*/\*fondo del componente principal\*/*

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 100% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

*/\* STRUCTURE \*/*

.wrapper {

display: flex;

align-items: center;

flex-direction: column;

justify-content: center;

width: 100%;

min-height: 100%;

padding: 20px;

}

#formContent {

-webkit-border-radius: 10px 10px 10px 10px;

border-radius: 10px 10px 10px 10px;

background: #fff;

padding: 30px;

width: 90%;

max-width: 450px;

position: relative;

padding: 0px;

-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

text-align: center;

}

#formFooter {

background-color: #f6f6f6;

border-top: 1px solid #dce8f1;

padding: 25px;

text-align: center;

-webkit-border-radius: 0 0 10px 10px;

border-radius: 0 0 10px 10px;

}

*/\* TABS \*/*

h2.inactive {

color: #cccccc;

}

h2.active {

color: #0d0d0d;

border-bottom: 2px solid #5fbae9;

}

*/\* FORM TYPOGRAPHY\*/*

input[type=button], input[type=submit], input[type=reset] {

background-color: #56baed;

border: none;

color: white;

padding: 15px 80px;

text-align: center;

text-decoration: none;

display: inline-block;

text-transform: uppercase;

font-size: 13px;

-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

margin: 5px 20px 40px 20px;

-webkit-transition: all 0.3s ease-in-out;

-moz-transition: all 0.3s ease-in-out;

-ms-transition: all 0.3s ease-in-out;

-o-transition: all 0.3s ease-in-out;

transition: all 0.3s ease-in-out;

}

input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {

background-color: #39ace7;

}

input[type=button]:active, input[type=submit]:active, input[type=reset]:active {

-moz-transform: scale(0.95);

-webkit-transform: scale(0.95);

-o-transform: scale(0.95);

-ms-transform: scale(0.95);

transform: scale(0.95);

}

input[type=text] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=text]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=text]:placeholder {

color: #cccccc;

}

*/\* ANIMATIONS \*/*

*/\* Simple CSS3 Fade-in-down Animation \*/*

.fadeInDown {

-webkit-animation-name: fadeInDown;

animation-name: fadeInDown;

-webkit-animation-duration: 1s;

animation-duration: 1s;

-webkit-animation-fill-mode: both;

animation-fill-mode: both;

}

@-webkit-keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

@keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

*/\* Simple CSS3 Fade-in Animation \*/*

@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

.fadeIn {

opacity:0;

-webkit-animation:fadeIn ease-in 1;

-moz-animation:fadeIn ease-in 1;

animation:fadeIn ease-in 1;

-webkit-animation-fill-mode:forwards;

-moz-animation-fill-mode:forwards;

animation-fill-mode:forwards;

-webkit-animation-duration:1s;

-moz-animation-duration:1s;

animation-duration:1s;

}

.fadeIn.first {

-webkit-animation-delay: 0.4s;

-moz-animation-delay: 0.4s;

animation-delay: 0.4s;

}

.fadeIn.second {

-webkit-animation-delay: 0.6s;

-moz-animation-delay: 0.6s;

animation-delay: 0.6s;

}

.fadeIn.third {

-webkit-animation-delay: 0.8s;

-moz-animation-delay: 0.8s;

animation-delay: 0.8s;

}

.fadeIn.fourth {

-webkit-animation-delay: 1s;

-moz-animation-delay: 1s;

animation-delay: 1s;

}

*/\* Simple CSS3 Fade-in Animation \*/*

.underlineHover:after {

display: block;

left: 0;

bottom: -10px;

width: 0;

height: 2px;

background-color: #56baed;

content: "";

transition: width 0.2s;

}

.underlineHover:hover {

color: #0d0d0d;

}

.underlineHover:hover:after{

width: 100%;

}

h1{

color:#60a0ff;

}

*/\* OTHERS \*/*

\*:focus {

outline: none;

}

#icon {

width:30%;

}

@media (max-width: 767px) {

.boton{

width: 50vw;

padding: 5px !important;

margin: 5px !important;

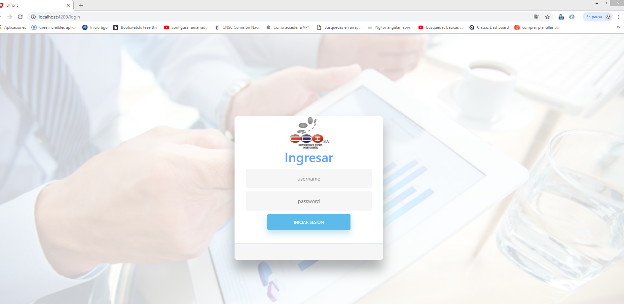
}

}

Creamos la ruta

{path: login, component: InicioComponent},

La prueba



Creemos el componente dashboard(aquí estará el menú de la aplicacion);

El html

*<*div class="col-sm-12 fondo"*>*

*</*div*>*

*<*div class="container"*>*

*<*div class="row espacios"*>*

*<*div class="col-sm-12" style="text-align:center"*>*

*<*h1*>*Menu de opciones*</*h1*>*

*</*div*>*

*<*div class="col-lg-6 col-md-6 col-sx-12 "*>*

*<*button type="button" class="btn btn-info tamanoBotones btn-block efectoBotones1 efectoBotones2"*>*

*<*i class="fas fa-shopping-cart fa-4x fa-lg"*></*i*>*

*<*br*>*Ordenes

*</*button*>*

*</*div*>*

*<*div class="col-lg-6 col-md-6 col-sx-12"*>*

*<*button type="button" class="btn btn-success tamanoBotones btn-block efectoBotones1 efectoBotones2"*>*

*<*i class="fas fa-chart-pie fa-4x fa-lg"*></*i*>*&nbsp;*<*i class="fas fa-chart-line fa-4x fa-lg"*></*i*>*

*<*br*>*Avance

*</*button*>*

*</*div*>*

*<*div class="col-lg-12 col-md-12 col-sx-12" *>*

*<*button type="button" class="btn btn-warning text-white tamanoBotones btn-block efectoBotones1 efectoBotones2"*>*

*<*i class="fas fa-user-shield fa-4x fa-lg"*></*i*>*

*<*br*>* Configuracion

*</*button*>*

*</*div*>*

*<*div class="col-lg-12 col-md-12 col-sx-12"*>*

*<*button type="button" class="btn btn-danger tamanoBotones btn-block efectoBotones1 efectoBotones2"*>*

*<*i class="fas fa-cog fa-4x fa-lg"*></*i*>*

*<*br*>*Super Configuracion

*</*button*>*

*</*div*>*

*</*div*>*

*</*div*>*

El css

*/\*fondo del componente principal\*/*

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 90.2% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

.container{

z-index: 2;

position: relative;

}

.espacios{

margin: 1vh;

}

.tamanoBotones{

height: 20vh !important;

margin: 5px;

font-weight: bold;

}

.efectoBotones1:hover{

background: rgba(0,0,0,0);

box-shadow: inset 0 0 0 3px #3a7999;

color : black !important;

font-weight: bold;

}

.efectoBotones2:before{

content:”;

position: absolute;

top: 0px;

left: 0px;

width: 0px;

height: 42px;

background: rgba(255,255,255,0.3);

border-radius: 5px;

transition: all 2s ease;

color: white;

}

.blanco:hover{

background: white !important;

color: black;

}

.borde:hover{

border : 3px solid #3a7999;

}

.botonletrablanca p{

color: white !important;

}

@media (max-width: 767px) {

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size:cover;

height: 100% !important;

opacity: 0.2px;

z-index: 1;

position: absolute;

}

section{

margin-top: 0px !important;

}

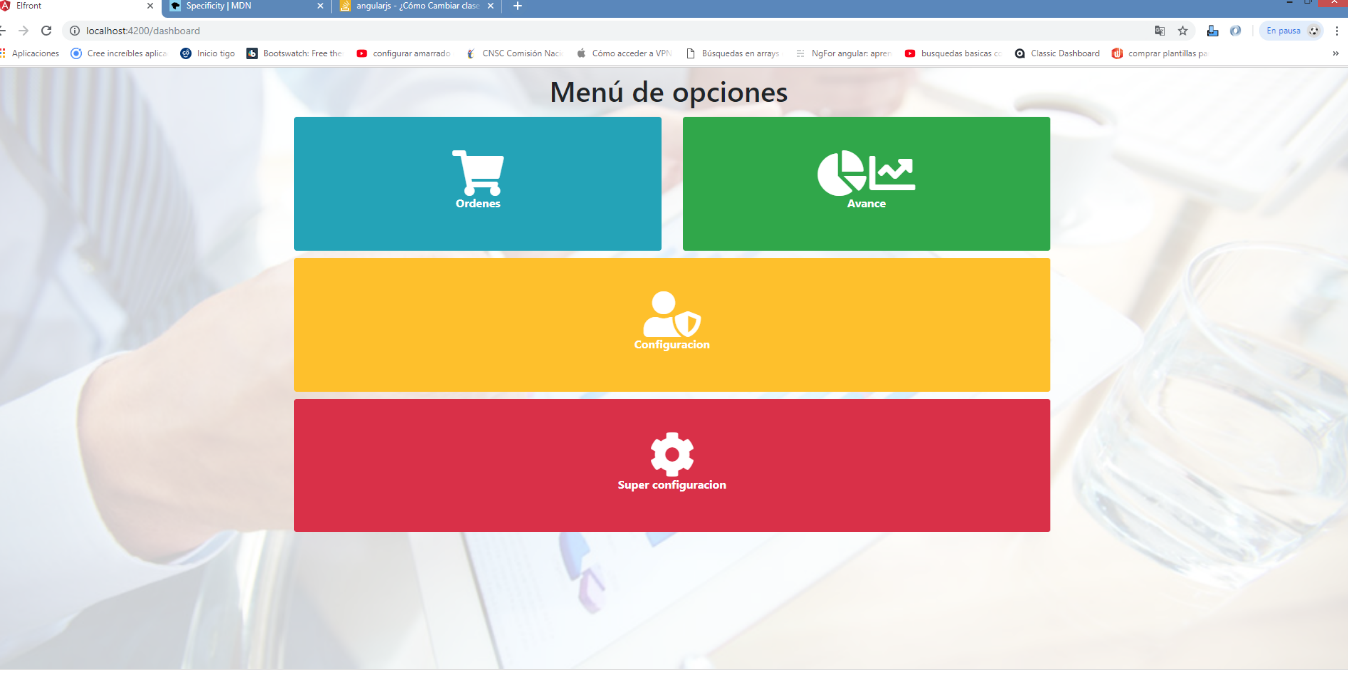
h1{

font-size: 2rem;

}

}

En ejecucion(hay que resaltar que la barra de navegacion aun no se ve porque no se ha iniciado sesion)



Ahora en el modelo del backend y en el diagraga agregare a la actividad 2 propiedades mas

**Fecha\_ejecucion** = fecha en que se inicia la actividad

**Fecha\_final** = fecha en que se termina la actividad

Estas fechas no las tratare en el backend en los registros las dejare comentadas hasta que pueda definir que me carga el formulario del frontend.

(asi quedo el modelo en el backend).

var ActividadSchema = new Schema({

consecutivo : {type:Number, required:[true, "el consecutivo es requerido"]},

nombre : {type:String, required:[true, "el nombre es requerido"]},

equipo : {type:Schema.ObjectId, ref:'Equipo', required:[true, "el equipo es requerido"] },

ubicacion:{type:Schema.ObjectId, ref: 'Ubicacionfisica', required:[true, "la ubicacion es requerida"]},

solicitante :{type:String}, *//persona que solicita la actividad*

tercero :{type:Schema.ObjectId, ref:'Tercero', required:[true, "el tercero es requerido"]},

usuario\_creador :{type:Schema.ObjectId, ref:'Usuario', required:[true, "el usuario creador es requerido"]},

fecha\_creacion :{type:Date, default:Date.now},

fecha\_requerida:{type:Date},

fecha\_ejecucion :{type:Date},

fecha\_final :{type:Date},

*//cargos:{type:Array, required:true},*

*//horas\_hombre :{type:Array, required:true},*

valor : {type:Number},

estado : {type:String, default:'ACTIVA', enum:estadosValidos, required:[true, "el estado es requerido"]}

});

**Ahora si cremos los modelos en el frontend**

Iniciemos con el modelo de usuario:

1-dentro de app creemos una carpeta llamada models y allí dentro crearemos nuestros ficheros de modelos. El primer modelo que crearemos sera el de usuario, el fichero se llamara userModel.ts

export class UserModel{

constructor(

public nombre :string,

public correo : string,

public role : string,

public password : string,

public tercero : string,

public image : string,

public user\_id :string,

public estado : string,

public \_id: string,

){}

}

Sigamos con cargoModel.ts

export class CargoModel{

constructor(

public nombre :string,

public vhora\_hombre : string,

public user\_id :string,

public estado : string,

public \_id: string,

){}

}

Ahora el consecutivoModel.ts

export class ConsecutivoModel{

constructor(

public prefijo :string,

public consecutivo : number,

public tercero :string,

public estado : string,

public \_id: string,

){}

}

Ahora el ubicacionModel.ts

export class ConsecutivoModel{

constructor(

public nombre :string,

public usuario\_creador : string,

public fecha\_creacion: string,

public tercero :string,

public user\_modificador : string,

public estado : string,

public \_id: string,

){}

}

Ahora el tercero

export class TerceroModel{

constructor(

public nit :string,

public tipo\_tercero : string,

public nombre : string,

public direccion : string,

public telefono : string,

public contacto : string,

public descripcion :string,

public user\_id : string,

public estado : string,

public \_id: string,

){}

}

Ahora el equipo

export class EquipoModel{

constructor(

public tag :string,

public descripcion : string,

public fecha\_creacion : string,

public Usuario\_creador : string,

public tercero : string,

public estado : string,

public user\_modificador :string,

public \_id: string,

){}

}

Sigamos con el de actividadesModel.ts

export class ActividadModel{

constructor(

public consecutivo :string,

public nombre : string,

public equipo : string,

public ubicacion\_fisica : string,

public solicitante : string,

public tercero : string,

public usuario\_creador :string,

public cargos : string,

public horas\_hombre :string,

public valor :number,

public fecha\_creacion : string,

public fecha\_requerida : string,

public fecha\_ejecucion : string,

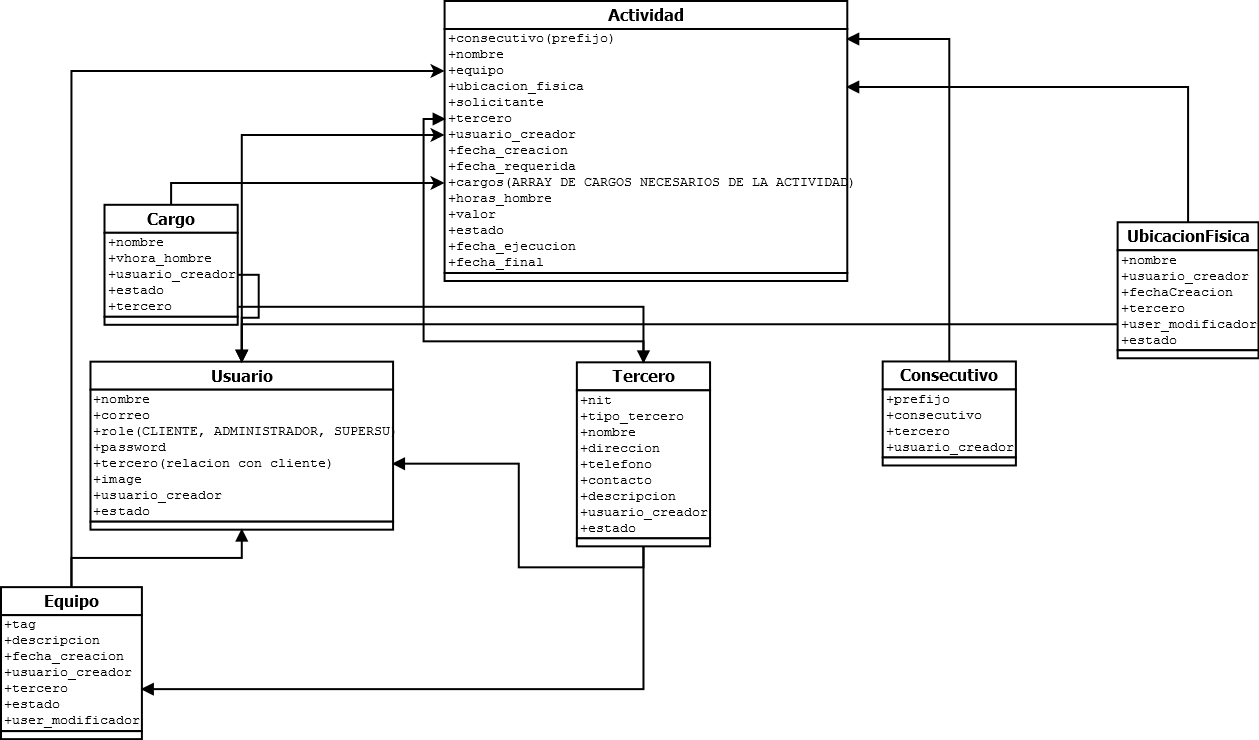
public fecha\_final : string,

public \_id: string,

){}

}

Vamos a estandarizar todo tanto en el backend como en el frontend, el campo user\_id se llamara usuario\_creador.



Vamos al modelo del frontend y corrijamos todo

Vayamos al backend y corrijamos todo también.

Ahora corrijamos los controladores del backend también

Las peticiones en postman quedaron corregidos también

**Creando Servicio de Usuario.**

0-**Ng g s services/user --spec=false (nos crear la carpeta services donde agregaremos nuestros servicios)**

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

@Injectable({

providedIn: 'root'

})

export class UserService {

constructor() { }

}

1-importemos el httpClientModule en el app.module.ts e ingresemoslo en los imports, este modulo nos servirá para realizar peticiones http

import { HttpClientModule } from '@angular/common/http';

imports: [

BrowserModule,

BrowserAnimationsModule,

MatButtonModule,

MatCheckboxModule,

MatFormFieldModule,

MatInputModule,

MatAutocompleteModule,

FormsModule,

ReactiveFormsModule,

routing,

HttpClientModule

],

Vamos a usar el modulo HttpClient y el HttpHeaders para poder hacer las peticiones Ajax y enviar las cabeceras. También importaremos el objeto observable para poder recoger las respuesta que devuelve el api, e importamos el modelo de usuario también. También en services creemos un fichero llamado **global.ts** donde tendre una variable con la url base, la cual utilizare en todos los modulos.

export var GLOBAL = {

url : 'http://localhost/3800/'

}

Ahora la importo el userServices.ts

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable, Subject, pipe } from 'rxjs';

import { UserModel } from '../models/userModel';

import { GLOBAL } from './global';

@Injectable({

providedIn: 'root'

})

export class UserService {

public url : string;

constructor(

public \_http:HttpClient

) {

this.url = GLOBAL.url;

}

login(){

console.log("yujuuuuuu");

}

}

Probémoslo agregándolo al componente de login

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

import { UserService } from 'src/app/services/user.service';

@Component({

selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

constructor(

private \_userServices : UserService

) { }

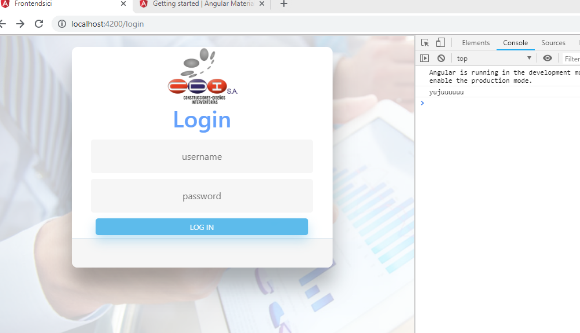
ngOnInit() {

this.\_userServices.login();

}

}

Asi se ve



Perfecto iniciemos creando el servicio de login

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable, Subject, pipe } from 'rxjs';

import { GLOBAL } from './global';

import { map, catchError } from "rxjs/operators";

import { throwError } from "rxjs/internal/observable/throwError";

import swal from 'sweetalert';

import { UserModel } from '../models/userModel';

@Injectable({

providedIn: 'root'

})

export class UserService {

public url : string;

public identity;

public token;

constructor(

public \_http:HttpClient

) {

this.url = GLOBAL.url;

}

//servicio de login

login(user:UserModel, gettoken=null){

if(gettoken !=null){

user.gettoken = gettoken;

}

let params = JSON.stringify(user);

let headers = new HttpHeaders().set('Content-Type', 'application/json');

return this.\_http.post(this.url+'login', params, {headers:headers}).pipe(

map((resp:any)=>{

return resp

}),

catchError(err=>{

swal('!Ya Existe la Cuenta', err.error.message, 'error');

return throwError(err) //nos retornra un observable

})

);

}

//convierte el json string del localstrorage del user en un json

getIdentity(){

let identity = JSON.parse(localStorage.getItem('identity'));

if(identity != undefined){

this.identity = identity;

}else{

this.identity = null;

}

console.log(this.identity);

return this.identity;

}

//convierte el json string del localstrorage del token en un json

getToken(){

let token = JSON.parse(localStorage.getItem('token'));

if(token != undefined){

this.token = token;

}else{

this.token = null;

}

return this.token;

}

}

Ahora el componente

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

import { UserService } from 'src/app/services/user.service';

import { UserModel } from '../../models/userModel';

@Component({

selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

public user : UserModel;

public status :string;

public identity :any;

public token:string;

constructor(

private \_userServices : UserService

) {

this.user = new UserModel("", "", "", "", "", "", "", "", "", "");

}

ngOnInit() {

}

//login del usuario

onSubmit(loginForm){

this.\_userServices.login(this.user).subscribe(

response=>{

this.identity = response.user;

this.status = "success";

//persistir datos del usuario

localStorage.setItem('identity', JSON.stringify(this.identity));

//conseguir el token

this.getToken();

},

error=>{

this.status = "error"

}

)

}

//obtener token del backend

getToken(){

this.\_userServices.login(this.user, 'true' ).subscribe(

response=>{

this.token = response.token;

if(this.token.length<=0){

this.status ='error'

}else{

this.status = "success";

localStorage.setItem('token', JSON.stringify(this.token));

}

//persistir el token del usuario

//conseguir los contadores del usuario es decir seguidos y me siguen

},

error=>{

this.status = "error"

}

)

}

}

Ahora para que todo se carge desde el inicio llamaremos al servicio getidentity en el app.component

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

import {FormControl} from '@angular/forms';

import {Observable} from 'rxjs';

import {map, startWith} from 'rxjs/operators';

import { UserService } from './services/user.service';

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

public identity:any = false;

constructor(

private \_userServices : UserService

) { }

ngOnInit() {

this.identity = this.\_userServices.getIdentity();

console.log(this.identity);

}

}

Ahora en el html mostraresmo el menú superior solo si identity tiene algo dentro es decir solo si identity tiene un usuario

<div class="col-sm-12 contenido">

<div \*ngIf="identity == null">

<nav class="navbar ">

<div class="navbar-header">

<img class="logo" src="assets/img/logo/logo.png" alt="">

</div>

<ul class="nav justify-content-end">

<li class="nav-item">

<a class="nav-link" href="#">Active</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">nombre de usuario</a>

<div class="dropdown-menu">

<a class="dropdown-item" href="#">Action</a>

<a class="dropdown-item" href="#">Another action</a>

<a class="dropdown-item" href="#">Something else here</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Separated link</a>

</div>

</li>

</ul>

</nav>

</div>

</div>

<router-outlet></router-outlet>

Ahora veamos el html del componente login

<div class="col-sm-12 contenido">

<div \*ngIf="identity != null">

<nav class="navbar ">

<div class="navbar-header">

<img class="logo" src="assets/img/logo/logo.png" alt="">

</div>

<ul class="nav justify-content-end">

<li class="nav-item">

<a class="nav-link" href="#">Active</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">nombre de usuario</a>

<div class="dropdown-menu">

<a class="dropdown-item" href="#">Action</a>

<a class="dropdown-item" href="#">Another action</a>

<a class="dropdown-item" href="#">Something else here</a>

<div class="dropdown-divider"></div>

<a (click)="logout()" class="dropdown-item" href="#">Cerrar Sesion</a>

</div>

</li>

</ul>

</nav>

</div>

</div>

<router-outlet></router-outlet>

El componente login

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

import { UserService } from 'src/app/services/user.service';

import { UserModel } from '../../models/userModel';

import { Router, ActivatedRoute, Params } from '@angular/router';

import { containerRefreshStart } from '@angular/core/src/render3/instructions';

@Component({

selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

public user : UserModel;

public status :string;

public identity :any;

public token:string;

constructor(

private \_userServices : UserService,

private \_router : Router

) {

this.user = new UserModel("", "", "", "", "", "", "", "", "", "");

}

ngOnInit() {

}

//login del usuario

onSubmit(loginForm){

this.\_userServices.login(this.user).subscribe(

response=>{

this.identity = response.user;

this.status = "success";

//persistir datos del usuario

localStorage.setItem('identity', JSON.stringify(this.identity));

//conseguir el token

this.getToken();

this.\_router.navigate(['/dashboard']);

},

error=>{

this.status = "error"

}

)

}

//obtener token del backend

getToken(){

this.\_userServices.login(this.user, 'true' ).subscribe(

response=>{

this.token = response.token;

if(this.token.length<=0){

this.status ='error'

}else{

this.status = "success";

localStorage.setItem('token', JSON.stringify(this.token));

}

//persistir el token del usuario

//conseguir los contadores del usuario es decir seguidos y me siguen

},

error=>{

this.status = "error"

}

)

}

}

El css del componente login

/\*fondo del componente principal\*/

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 99.80% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

/\*contenido\*/

.contenido{

z-index: 2;

position: absolute;

}

/\* STRUCTURE \*/

.wrapper {

display: flex;

align-items: center;

flex-direction: column;

justify-content: center;

width: 100%;

min-height: 100%;

padding: 20px;

}

#formContent {

-webkit-border-radius: 10px 10px 10px 10px;

border-radius: 10px 10px 10px 10px;

background: #fff;

padding: 30px;

width: 90%;

max-width: 450px;

position: relative;

padding: 0px;

-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

text-align: center;

}

#formFooter {

background-color: #f6f6f6;

border-top: 1px solid #dce8f1;

padding: 25px;

text-align: center;

-webkit-border-radius: 0 0 10px 10px;

border-radius: 0 0 10px 10px;

}

/\* TABS \*/

h2.inactive {

color: #cccccc;

}

h2.active {

color: #0d0d0d;

border-bottom: 2px solid #5fbae9;

}

/\* FORM TYPOGRAPHY\*/

input[type=button], input[type=submit], input[type=reset] {

background-color: #56baed;

border: none;

color: white;

padding: 15px 80px;

text-align: center;

text-decoration: none;

display: inline-block;

text-transform: uppercase;

font-size: 13px;

-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

margin: 5px 20px 40px 20px;

-webkit-transition: all 0.3s ease-in-out;

-moz-transition: all 0.3s ease-in-out;

-ms-transition: all 0.3s ease-in-out;

-o-transition: all 0.3s ease-in-out;

transition: all 0.3s ease-in-out;

}

input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {

background-color: #39ace7;

}

input[type=button]:active, input[type=submit]:active, input[type=reset]:active {

-moz-transform: scale(0.95);

-webkit-transform: scale(0.95);

-o-transform: scale(0.95);

-ms-transform: scale(0.95);

transform: scale(0.95);

}

input[type=email] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=email]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=email]:placeholder {

color: #cccccc;

}

input[type=password] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=password]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=password]:placeholder {

color: #cccccc;

}

/\* ANIMATIONS \*/

/\* Simple CSS3 Fade-in-down Animation \*/

.fadeInDown {

-webkit-animation-name: fadeInDown;

animation-name: fadeInDown;

-webkit-animation-duration: 1s;

animation-duration: 1s;

-webkit-animation-fill-mode: both;

animation-fill-mode: both;

}

@-webkit-keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

@keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

/\* Simple CSS3 Fade-in Animation \*/

@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

.fadeIn {

opacity:0;

-webkit-animation:fadeIn ease-in 1;

-moz-animation:fadeIn ease-in 1;

animation:fadeIn ease-in 1;

-webkit-animation-fill-mode:forwards;

-moz-animation-fill-mode:forwards;

animation-fill-mode:forwards;

-webkit-animation-duration:1s;

-moz-animation-duration:1s;

animation-duration:1s;

}

.fadeIn.first {

-webkit-animation-delay: 0.4s;

-moz-animation-delay: 0.4s;

animation-delay: 0.4s;

}

.fadeIn.second {

-webkit-animation-delay: 0.6s;

-moz-animation-delay: 0.6s;

animation-delay: 0.6s;

}

.fadeIn.third {

-webkit-animation-delay: 0.8s;

-moz-animation-delay: 0.8s;

animation-delay: 0.8s;

}

.fadeIn.fourth {

-webkit-animation-delay: 1s;

-moz-animation-delay: 1s;

animation-delay: 1s;

}

/\* Simple CSS3 Fade-in Animation \*/

.underlineHover:after {

display: block;

left: 0;

bottom: -10px;

width: 0;

height: 2px;

background-color: #56baed;

content: "";

transition: width 0.2s;

}

.underlineHover:hover {

color: #0d0d0d;

}

.underlineHover:hover:after{

width: 100%;

}

h1{

color:#60a0ff;

}

/\* OTHERS \*/

\*:focus {

outline: none;

}

#icon {

width:30%;

}

@media (max-width: 767px) {

.boton{

width: 50vw;

padding: 5px !important;

margin: 5px !important;

}

}

Ahora rgresamos al app.component.ts y usamos el modulo DoCheck para que cuando escuche algún cambio en el componente app.component.ts krecargue el componente, esto con el fin de que cuando hagamos login y la propiedad identity del componente app.component.ts se de cuenta que cambio, recargue inmediatamente y nos muestre la barra de menú superior.

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

import {FormControl} from '@angular/forms';

import {Observable} from 'rxjs';

import {map, startWith} from 'rxjs/operators';

import { UserService } from './services/user.service';

import { Router, ActivatedRoute, Params } from '@angular/router';

@Component({

selector: 'app-root',

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

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

})

export class AppComponent implements OnInit, DoCheck {

public identity:any = false;

constructor(

private \_userServices : UserService,

private \_router : Router

) { }

ngOnInit() {

this.identity = this.\_userServices.getIdentity();

}

ngDoCheck(){

this.identity = this.\_userServices.getIdentity();

}

//cerrar sesion

logout(){

//borra todo lo que hay en el localStorage

localStorage.clear();

this.identity = null;

this.\_router.navigate(['/login']);

}

}

**Haciendo Uso de Guards para que las rutas que deben verse solo cuando se hace login sean las que se vean en ese caso y no antes.**

Dentro de services creare un nuevo fichero llamado **user.guard.ts**,

**Importaremos** el **Injectable**, **Router**, **CanActivate**(el un modulo que permite que hagamos el guard para que compruebe si se puede acceder a la ruta o no) y **UserService**.

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

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

import { UserService } from './user.service';

@Injectable()

export class UserGuard implements CanActivate{

constructor(

private \_router: Router,

private \_userServices : UserService

){

}

canActivate(){

let identity = this.\_userServices.getIdentity();

if(identity && (identity.role == "CLIENTE" || identity.role == "PROVEEEDOR" || identity.role == "SUPERSU" )){

return true;

}else{

this.\_router.navigate(['/login']);

return false;

}

}

}

Ahora importamos en el app.module.ts el servicio de usuarios y el guard creado. Y cargarlos dentro de los providers

//Servicios

import { UserService } from './services/user.service';

import { UserGuard } from './services/user.guard';

providers: [appRoutingProviders,

UserService,

UserGuard

],

De esta manera ahora puedo coger el guard y aplicarlo a cualquiera de mis rutas, de modo que deberemos importar el CanActivate en el archivo de app.routing.ts y de ese modo aplicarlo a las rutas.

//-------------------------------servicios--------------------------------//

import { UserGuard } from './services/user.guard';

Apliquémoslo al dashboard que es la única ruta que tenemos para después del login por ahora.

{path: 'dashboard', component: DashboardComponent, canActivate:[**UserGuard]}**

Ahora probemos saliendo de la sesión y tratando de entrar al dahsboard.

Ahora hagamos un ajuste en el **app.component.ts** para que ya estando logueados no se permita acceder al login o al componente inicio.

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

import {FormControl} from '@angular/forms';

import {Observable} from 'rxjs';

import {map, startWith} from 'rxjs/operators';

import { UserService } from './services/user.service';

import { Router, ActivatedRoute, Params } from '@angular/router';

@Component({

selector: 'app-root',

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

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

})

export class AppComponent implements OnInit, DoCheck {

public identity:any = false;

constructor(

private \_userServices : UserService,

private \_router : Router

) { }

ngOnInit() {

this.identity = this.\_userServices.getIdentity();

}

ngDoCheck(){

let URLactual = window.location.href;

console.log(URLactual);

if(URLactual == 'http://localhost:4200/login' && this.identity != null || URLactual == 'http://localhost:4200/inicio' && this.identity != null || URLactual == 'http://localhost:4200/' && this.identity != null){

console.log("quieres entrar a donde no debes");

localStorage.clear();//limpiamos el localStorage para que se cierre la sesion

this.identity = null;

}else{

this.identity = this.\_userServices.getIdentity();

console.log(this.identity);

}

}

//cerrar sesion

logout(){

//borra todo lo que hay en el localStorage

localStorage.clear();

this.identity = null;

this.\_router.navigate(['/login']);

}

}

Creemos el componente del dashboard:

El html

<div class="col-sm-12 fondo">

</div>

<div class="container">

<div class="row espacios">

<div class="col-sm-12" style="text-align:center">

<h1>Menu de opciones</h1>

</div>

<div class="col-lg-6 col-md-6 col-sx-12 ">

<button type="button" class="btn btn-info tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-list-alt fa-4x fa-lg"></i>

<br><br> Ordenes

</button>

</div>

<div class="col-lg-6 col-md-6 col-sx-12">

<button type="button" class="btn btn-success tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-chart-pie fa-4x fa-lg"></i>&nbsp;<i class="fas fa-chart-line fa-4x fa-lg"></i>

<br><br> Avance

</button>

</div>

<div class="col-lg-12 col-md-12 col-sx-12" >

<button type="button" [routerLink]="['/config']" class="btn btn-warning text-white tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-user-shield fa-4x fa-lg"></i>

<br><br> Configuracion

</button>

</div>

<div class="col-lg-12 col-md-12 col-sx-12">

<button type="button" class="btn btn-danger tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-cog fa-4x fa-lg"></i>

<br><br> Super Configuracion

</button>

</div>

</div>

</div>

El css

/\*fondo del componente principal\*/

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 90.2% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

.container{

z-index: 2;

position: relative;

}

.espacios{

margin: 1vh;

}

.tamanoBotones{

height: 20vh !important;

margin: 5px;

font-weight: bold;

}

.efectoBotones1:hover{

background: rgba(0,0,0,0);

box-shadow: inset 0 0 0 3px #3a7999;

color : black !important;

font-weight: bold;

}

.efectoBotones2:before{

content:”;

position: absolute;

top: 0px;

left: 0px;

width: 0px;

height: 42px;

background: rgba(255,255,255,0.3);

border-radius: 5px;

transition: all 2s ease;

color: white;

}

.blanco:hover{

background: white !important;

color: black;

}

.borde:hover{

border : 3px solid #3a7999;

}

.botonletrablanca p{

color: white !important;

}

@media (max-width: 767px) {

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size:cover;

height: 100% !important;

opacity: 0.2px;

z-index: 1;

position: absolute;

}

section{

margin-top: 0px !important;

}

h1{

font-size: 2rem;

}

}

El dashboard.component.ts

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

@Component({

selector: 'app-dashboard',

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

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

})

export class DashboardComponent implements OnInit {

constructor() { }

ngOnInit() {

}

}

Creemos el componente de config cliente

El html

<div class="col-sm-12 fondo">

</div>

<div class="container">

<div class="row espacios">

<div class="col-sm-12" style="text-align:center">

<h1>Menu de opciones</h1>

</div>

<div class="col-lg-6 col-md-6 col-sx-12 ">

<button type="button" [routerLink]="['/cargos/1']" class="btn btn-info tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-handshake fa-4x fa-lg"></i>

<br><br> Cargos

</button>

</div>

<div class="col-lg-6 col-md-6 col-sx-12">

<button type="button" class="btn btn-success tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-compass fa-4x fa-lg"></i>

<br><br> Ubicacion fisica

</button>

</div>

<div class="col-lg-12 col-md-12 col-sx-12" >

<button type="button" class="btn btn-warning text-white tamanoBotones btn-block efectoBotones1 efectoBotones2">

<i class="fas fa-toolbox fa-4x fa-lg"></i>

<br><br> Equipo

</button>

</div>

</div>

</div>

El css

/\*fondo del componente principal\*/

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 90.2% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

.container{

z-index: 2;

position: relative;

}

.espacios{

margin: 1vh;

}

.tamanoBotones{

height: 20vh !important;

margin: 5px;

font-weight: bold;

}

.efectoBotones1:hover{

background: rgba(0,0,0,0);

box-shadow: inset 0 0 0 3px #3a7999;

color : black !important;

font-weight: bold;

}

.efectoBotones2:before{

content:”;

position: absolute;

top: 0px;

left: 0px;

width: 0px;

height: 42px;

background: rgba(255,255,255,0.3);

border-radius: 5px;

transition: all 2s ease;

color: white;

}

.blanco:hover{

background: white !important;

color: black;

}

.borde:hover{

border : 3px solid #3a7999;

}

.botonletrablanca p{

color: white !important;

}

@media (max-width: 767px) {

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size:cover;

height: 100% !important;

opacity: 0.2px;

z-index: 1;

position: absolute;

}

section{

margin-top: 0px !important;

}

h1{

font-size: 2rem;

}

}

El config-component.ts

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

@Component({

selector: 'app-config-cliente',

templateUrl: './config-cliente.component.html',

styleUrls: ['./config-cliente.component.css']

})

export class ConfigClienteComponent implements OnInit {

constructor() { }

ngOnInit() {

}

}

Ahora creemos el componente de cargos-cliente

El html

<div class="col-sm-12 fondo">

</div>

<div class="container">

<div class="row espacios">

<div class="col-md-4">

<div class="card">

<div class="card-body">

<h5 style="text-align:center">Creacion de Cargos</h5>

<br>

<div class="card">

<div class="card-body">

<form ngNativeValidate [formGroup]="form" #cargoFormC="ngForm" (ngSubmit)="createCargo(form, cargoFormC)">

<div class="col-md-12 col-sm-12">

<div class="form-group">

<label for="nombreUsuario" class="negrita">Nombre del Cargo</label>

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

</div>

</div>

<div class="col-md-12 col-sm-12">

<div class="form-group">

<label for="vhora" class="negrita">Valor de la Hora Hombre</label>

<input type="number" formControlName="vhora\_hombreC" class="form-control" required>

</div>

</div>

<div style="text-align:center">

<input type="submit" class="btn btn-outline-success" value="Agregar">

</div>

</form>

</div>

</div>

</div>

</div>

</div>

<div class="col-md-7 col-sm-12">

<div class="card">

<div class="card-body">

<h5 style="text-align:center">Buscar Cargos</h5>

<br>

<div class="card">

<div class="card-body">

<label for="">Buscar</label>

<input #input (keyup)="buscarCargo(input.value)" type="text" class="form-control" name="" id="">

<div class="row">

<div class="col-md-12">

<table class="table table-hover table-bordered table-responsive">

<thead class="table-info">

<tr>

<th>#</th>

<th>Nombre Cargo</th>

<th>Valor HH</th>

<th>Estado</th>

<th>Editar</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let cargo of cargos; let i = index">

<td>{{i}}</td>

<td>{{cargo.nombre}}</td>

<td>{{cargo.vhora\_hombre | currency:'USD':true:'1.0-3'}}</td>

<td>{{cargo.estado}}</td>

<td style="text-align:center">

<i class="fas fa-edit pointer" [routerLink]="['/cargosedit', cargo.\_id]"></i>

</td>

</tr>

</tbody>

</table>

<!--PAGINACION -->

<div class="antsig" style="text-align:center">

<button type="button" \*ngIf="page >= 2" [routerLink]="['/cargos', pre\_page]" class="btn btn-outline-info">Anterior</button>

<button type="button" \*ngIf="pages != page" [routerLink]="['/cargos', next\_page]" class="btn btn-outline-info ">Siguiente</button>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

El css

/\*fondo del componente principal\*/

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 118% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

.container{

z-index: 2;

position: relative;

}

.espacios{

padding: 2vh;

}

button{

margin: 3vh;

}

.espacios div{

padding-top: 3vh;

}

.pointer{

cursor: pointer;

}

El cargoClienteComponenent.ts

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

import { UserService } from 'src/app/services/user.service';

import { CargoModel } from 'src/app/models/cargoModel';

import { CargoService } from 'src/app/services/cargo.service';

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

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

declare var swal:any;

@Component({

selector: 'app-cargos-cliente',

templateUrl: './cargos-cliente.component.html',

styleUrls: ['./cargos-cliente.component.css']

})

export class CargosClienteComponent implements OnInit {

public identity : any;

public cargos : CargoModel[]=[];

public newCargo : CargoModel;

public form : FormGroup;

public formSubmit: boolean;

page: number;

next\_page: number;

pre\_page: number;

total: any;

pages: any;

constructor(

private \_userServices : UserService,

private \_cargoServices : CargoService,

private fb: FormBuilder,

private \_router : Router,

private \_route:ActivatedRoute,

) {

this.page = 1;

}

ngOnInit() {

this.identity = this.\_userServices.getIdentity();

//inicializando el formulario

this.form = this.fb.group({

nombreC : [ "", Validators.required ],

vhora\_hombreC: [ 0, Validators.required ]

});

//listarCargos

this.actualPage();

this.listarCargoCliente(1);

}

//Crear Cargos

createCargo(form, createCargo){

const formModel = this.form.value;

let saveCargo: CargoModel ={

nombre : formModel.nombreC as string,

vhora\_hombre :formModel.vhora\_hombreC as number,

usuario\_creador : this.identity.\_id,

tercero : this.identity.tercero,

};

this.\_cargoServices.createCargo(saveCargo)

.subscribe((datos:any)=>{

console.log(datos);

form.reset();

swal("Exito", "Cargo Creado", "success");

this.listarCargoCliente(this.page);

});

}

actualPage(){

//asi capturo paramtros de la url

this.\_route.params.subscribe(params=>{

let page = +params['page'];//con el + delante lo convierto en un entero

this.page = page;

if(!params['page']){

page = 1;

}

if(!page){

page = 1

}else{

this.next\_page = page + 1;

this.pre\_page = page - 1;

if(this.pre\_page <= 0){

this.pre\_page = 1;

}

}

if(page >= this.pages){

page = this.pages

}

//ejecutar traer todos los usuarios

this.listarCargoCliente(page)

})

}

//Listar Cargos de los clientes

listarCargoCliente(page){

this.\_cargoServices.listarCargosClientes(page)

.subscribe((datos:any)=>{

console.log(datos);

this.cargos = datos.cargos;

this.total = datos.total;

this.pages = datos.pages;

})

}

//Buscar Cargo desde el campo input dinamicamente

buscarCargo(termino:string){

if(termino.length<=0){

this.listarCargoCliente(this.page)

return;

}

this.\_cargoServices.buscarCargosInputDinamico(termino)

.subscribe((datos:any)=>{

this.cargos = datos.cargos;

})

}

}

Ahora el conponente para editar los cargos del cliente

Cargo-client-componen

**El html**

<div class="col-sm-12 fondo">

</div>

<div class="container">

<div class="row">

<div class="col-md-6 offset-md-3 ">

<div class="card">

<h5 style="text-align:center; padding: 1vh;">Editar Cargos</h5>

<div class="card-body">

<div class="row">

<div class="col-md-12">

<form ngNativeValidate [formGroup]="form" #cargoFormE="ngForm" (ngSubmit)="editCargo(form, cargoFormE)">

<div class="col-md-12 col-sm-12">

<div class="form-group">

<label for="nombreCargo" class="negrita">Nombre del Cargo</label>

<input type="text" formControlName="nombreC" [(ngModel)]="nombreCargo" class="form-control" required>

</div>

</div>

<div class="col-md-12 col-sm-12">

<div class="form-group">

<label for="vhora" class="negrita">Valor de la Hora Hombre</label>

<input type="number" formControlName="vhora\_hombreC" [(ngModel)]="valorHHombre" class="form-control" required>

</div>

</div>

<div class="form-group">

<div class="form-group padselect">

<label for="selectCargo">Estado</label>

<select class="form-control" name="estadoC" formControlName="estadoC" [(ngModel)]="estado">

<option [value]="estado" selected>{{estado}}</option>

<option>ACTIVO</option>

<option>INACTIVO</option>

</select>

</div>

</div>

<div style="text-align:center">

<input type="submit" class="btn btn-outline-success" value="Editar">

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

Css

/\*fondo del componente principal\*/

.fondo{

background-image: url(../../../assets/img/logo/bgc.jpg);

background-position: left;

border: 1px solid black !important;

background-size: cover;

height: 90% !important;

opacity: 0.2;

z-index: 1;

position: absolute;

}

.container{

z-index: 2;

position: relative;

padding: 3vh;

}

.padselect{

padding-left: 14px;

padding-right: 14px;

}

Ts

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

import { UserService } from 'src/app/services/user.service';

import { CargoModel, CargoEdit } from 'src/app/models/cargoModel';

import { CargoService } from 'src/app/services/cargo.service';

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

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

declare var swal:any;

@Component({

selector: 'app-cargo-cliente-edit',

templateUrl: './cargo-cliente-edit.component.html',

styleUrls: ['./cargo-cliente-edit.component.css']

})

export class CargoClienteEditComponent implements OnInit {

public identity : any;

public cargos : CargoModel[]=[];

public ediCargo : CargoModel;

public nombreCargo : string;

public valorHHombre : number;

public estado :string;

public form : FormGroup;

public formSubmit: boolean;

constructor(

private \_userServices : UserService,

private \_cargoServices : CargoService,

private fb: FormBuilder,

private \_router : Router,

private \_route:ActivatedRoute,

) {

}

ngOnInit() {

this.identity = this.\_userServices.getIdentity();

//inicializando el formulario

this.form = this.fb.group({

nombreC : [ "", Validators.required ],

vhora\_hombreC: [ 0, Validators.required ],

estadoC :["", Validators.required]

});

//buscando el cargo para editar

this.buscarCargo();

}

//Editar Cargos

editCargo(form, cargoFormE){

const formModel = this.form.value;

let editCargo: CargoModel ={

nombre : this.nombreCargo,

vhora\_hombre :this.valorHHombre,

estado : this.estado

};

let urlActual = window.location.href;

let extraer = urlActual.split('/');

let id = extraer[4];

this.\_cargoServices.editarCargoCliente(editCargo, id)

.subscribe((datos:any)=>{

console.log(datos);

form.reset();

swal("Exito", "Cargo Editado", "success");

this.\_router.navigate(['cargos/1'])

});

}

//Buscar un solo cargo para guardarlo en una variable y editarlo despues

buscarCargo(){

let urlActual = window.location.href;

let extraer = urlActual.split('/');

let id = extraer[4];

this.\_cargoServices.buscarUnCargoCliente(id)

.subscribe((datos:any)=>{

console.log(datos);

this.ediCargo = datos.cargo;

console.log(this.ediCargo);

this.nombreCargo = datos.cargo.nombre;

this.valorHHombre = datos.cargo.vhora\_hombre;

this.estado = datos.cargo.estado;

})

}

}