**Lab 1.0**

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

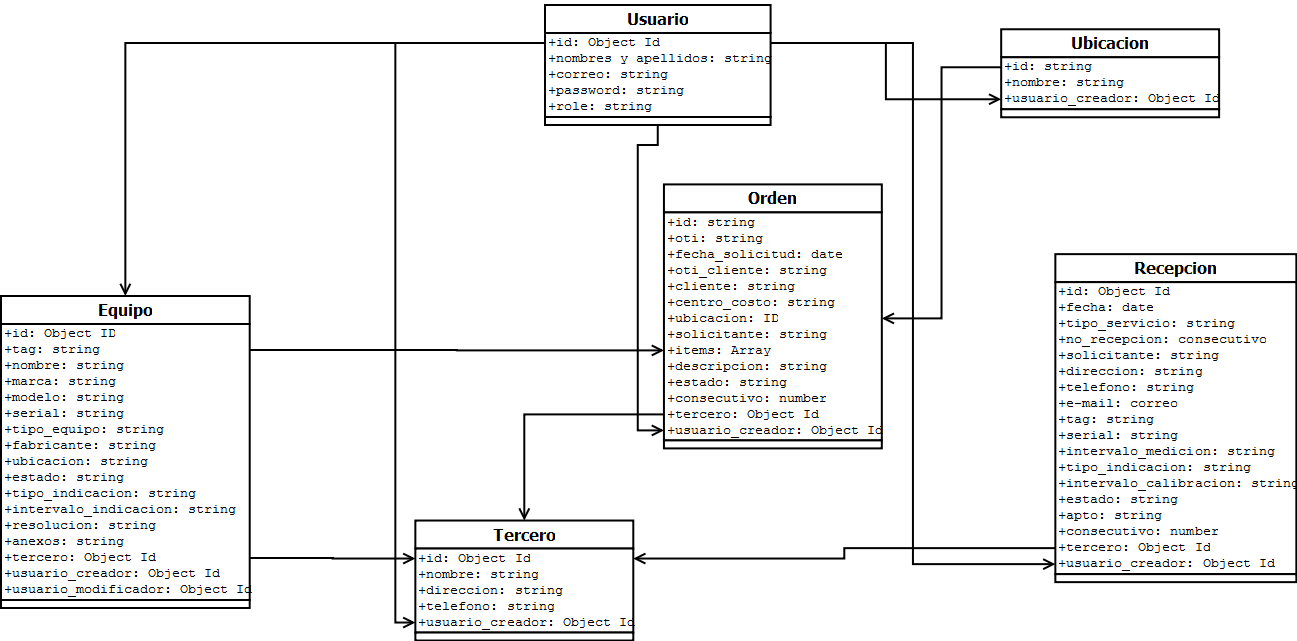
empecemos instalando las dependencias que necesitamos para nuestro backend

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);

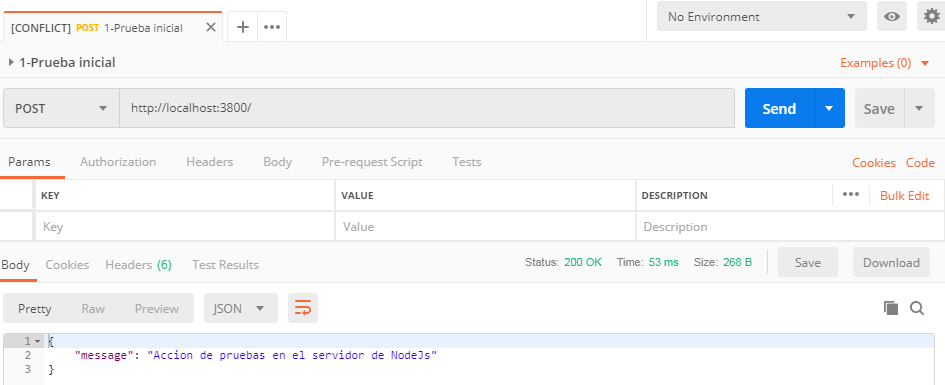
res.status(200).json({

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. **ubicacionModel.js**.

'use strict';

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

UbicacionSchema = new Schema({

nombre : {type: String, required:[true, 'el nombre de la ubicacion es requerido']},

codigo : {type: String, required:[true, 'el codigo de la ubicacion es requerido'], unique:true},

usuario\_creador: {type: Schema.ObjectId, ref:'User'}

})

module.exports = mongoose.model('Ubicacion', UbicacionSchema);

4-terceroModel.js

'use strict';

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var TerceroSchema = new Schema({

nit : {type: String, required:[true, 'el nit es requerido']},

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

direccion : {type: String},

telefono: {type: String},

usuario\_creador: {type: Schema.ObjectId, ref:'User'}

});

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},

nombre: {type:String},

marca: {type:String},

modelo: {type:String},

serial: {type:String},

tipo\_equipo: {type:String},

fabricante: {type:String},

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

estado: {type:String},

tipo\_indicacion: {type:String},

intervalo\_indicacion: {type:String},

resolucion: {type:String},

anexos: {type:String},

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

usuario\_creador: {type:Schema.ObjectId, ref:'User', required:[true, 'requiere un usuario creador']},

usuario\_modificador: {type:Schema.ObjectId, ref:'User'}

})

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

6-consecutivoModel.js

'use strict';

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var EquipoSchema = new Schema({

tag: {type:String},

nombre: {type:String},

marca: {type:String},

modelo: {type:String},

serial: {type:String},

tipo\_equipo: {type:String},

fabricante: {type:String},

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

estado: {type:String},

tipo\_indicacion: {type:String},

intervalo\_indicacion: {type:String},

resolucion: {type:String},

anexos: {type:String},

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

usuario\_creador: {type:Schema.ObjectId, ref:'User', required:[true, 'requiere un usuario creador']},

usuario\_modificador: {type:Schema.ObjectId, ref:'User'},

img: {type:String}

})

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

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

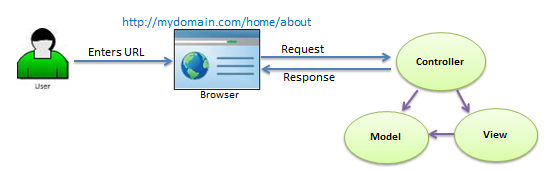
consecutivo: incrementador 1 a 1 por cada actividad

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*

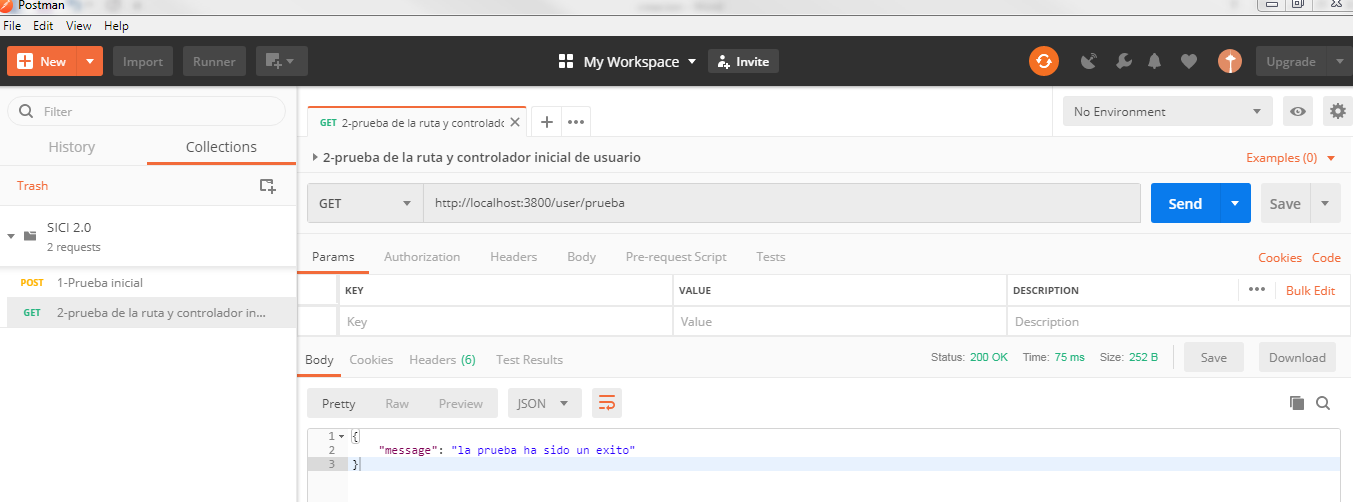
var UserRoutes = require('./routes/userRoutes');

*//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.findOne({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"})

}

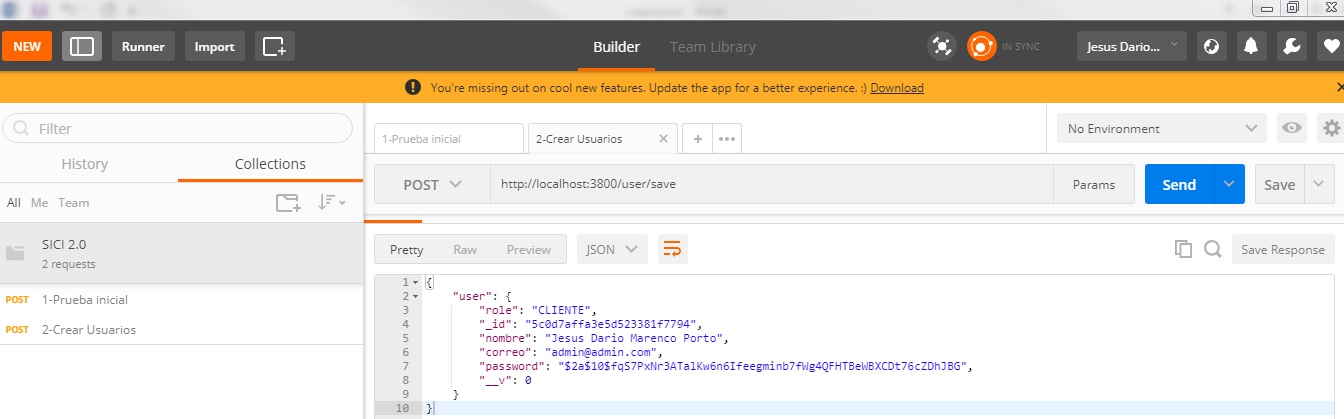
}else{

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!!"});

}

})

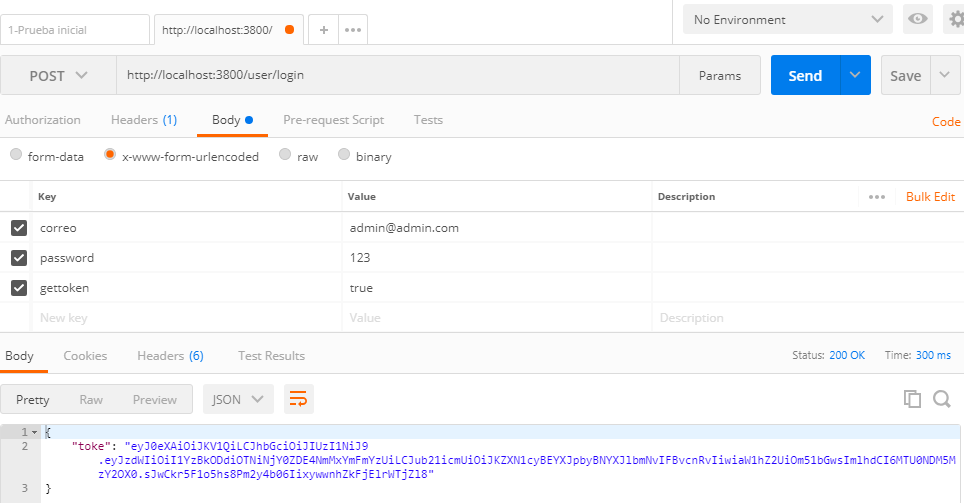
}

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();

}

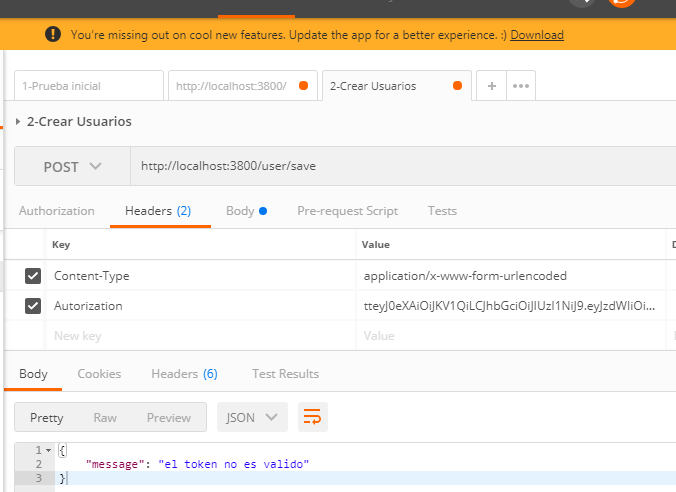
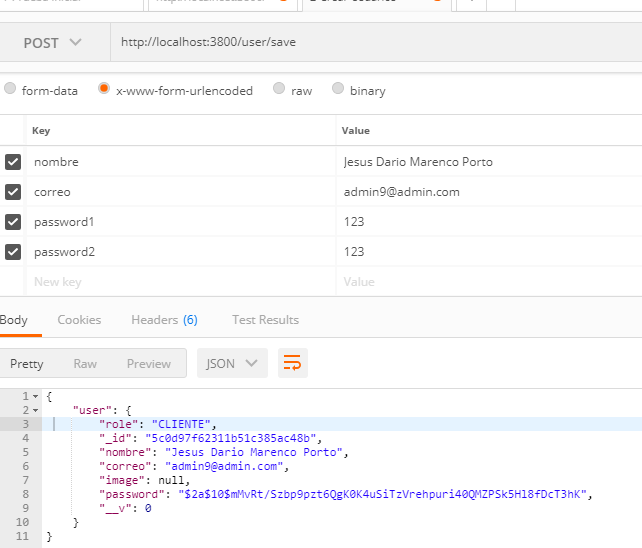
*//Estos middlewares los usaremos en las rutas de modo que deberemos cargarlos allí vamos a userRoutes.js*

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);

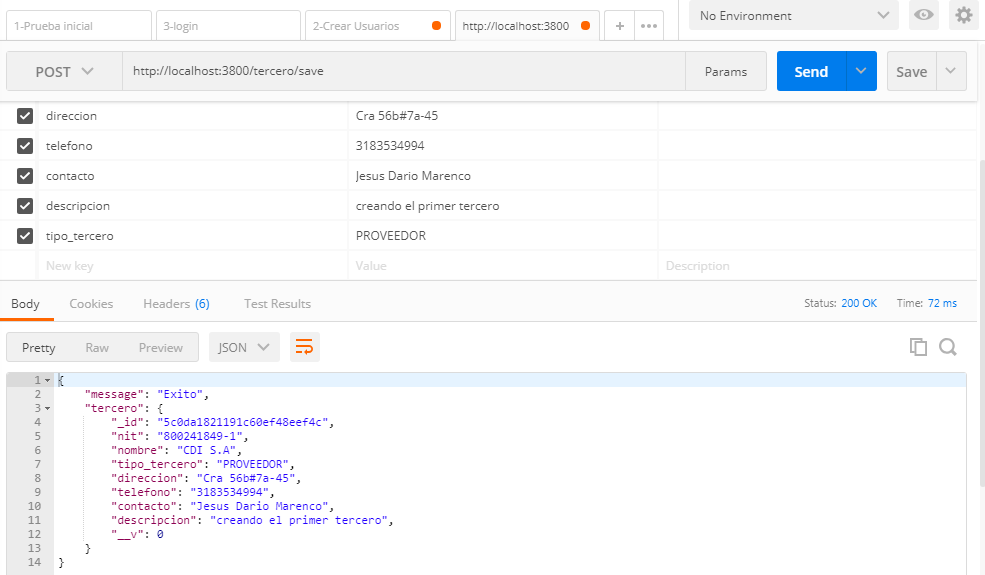
module.exports = api;

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.

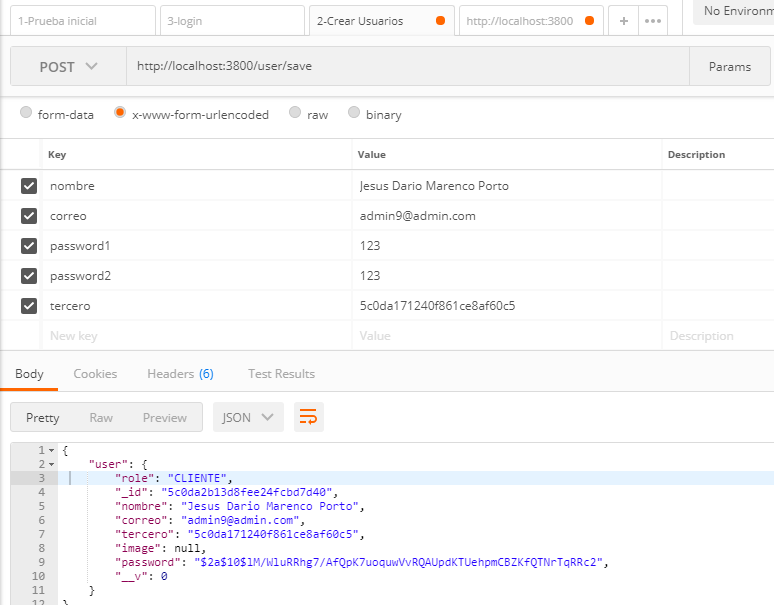
var User = new UserModel();

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});

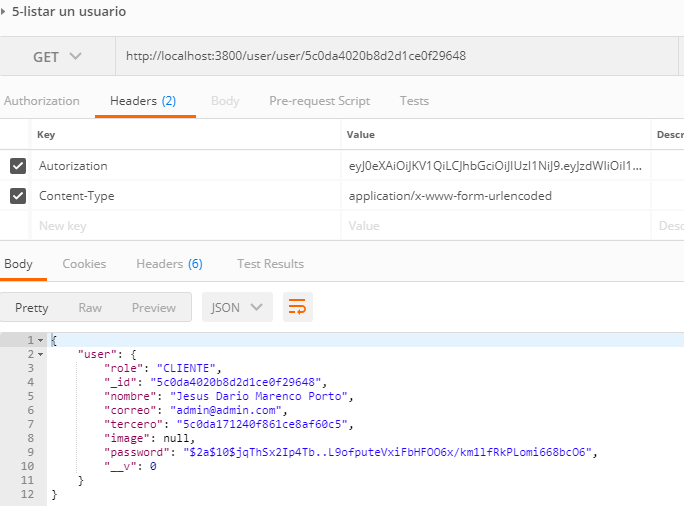
})

}

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});

})

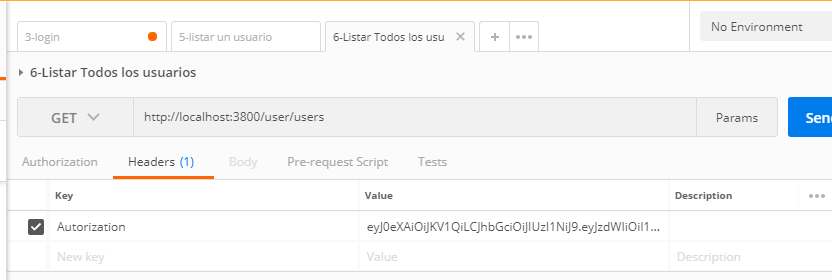
}

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});

})

})

}

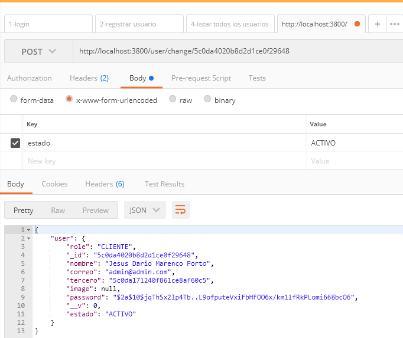
Exportamos la función

Luego creamos la ruta

*//Cambiar el Estado del usuqario*

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

probamos

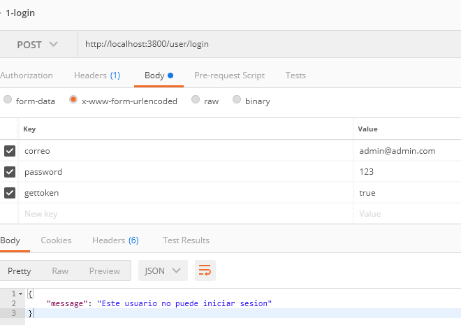


Ahora validamos la función de login para uqe los usuarios inactivos no puedan iniciar sesión

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});

});

}

});

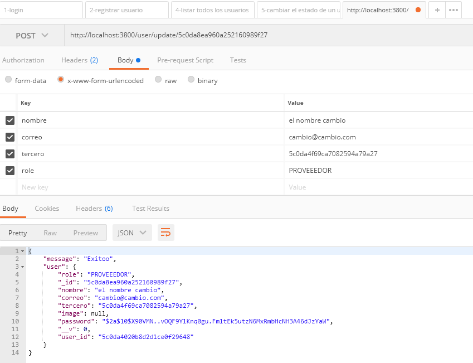
}

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

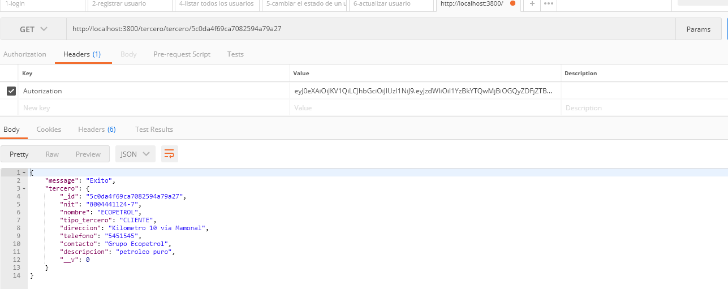
}

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

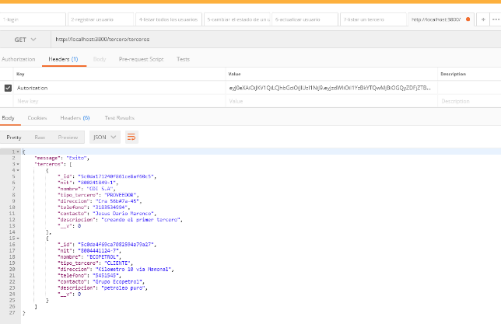
}

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,

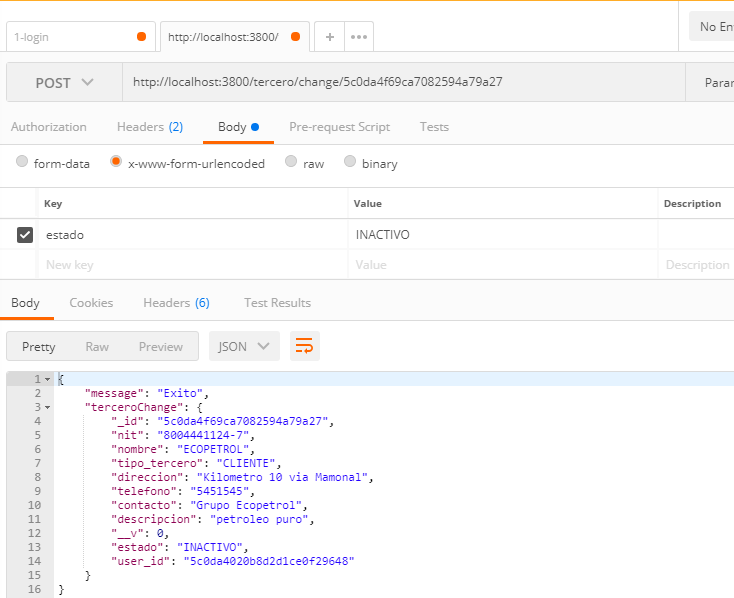
listTercero,

listarTerceros,

changeEstadoTercero

}

Probamos en postman

perfecto sigamos

**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();

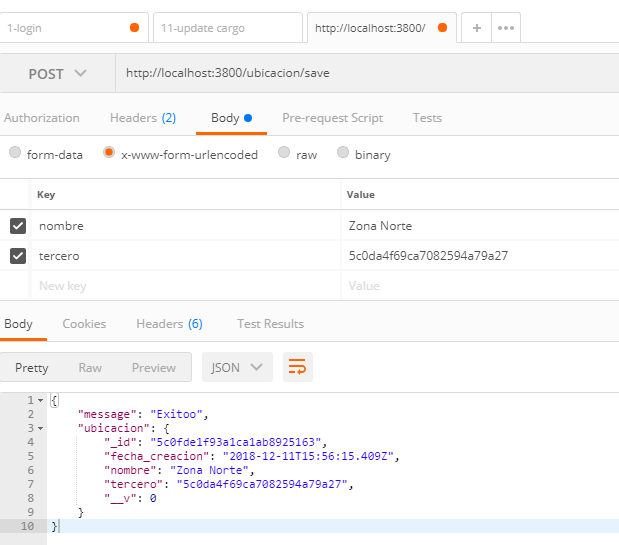
var md\_auth = require('../middlewares/autenticated');

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});

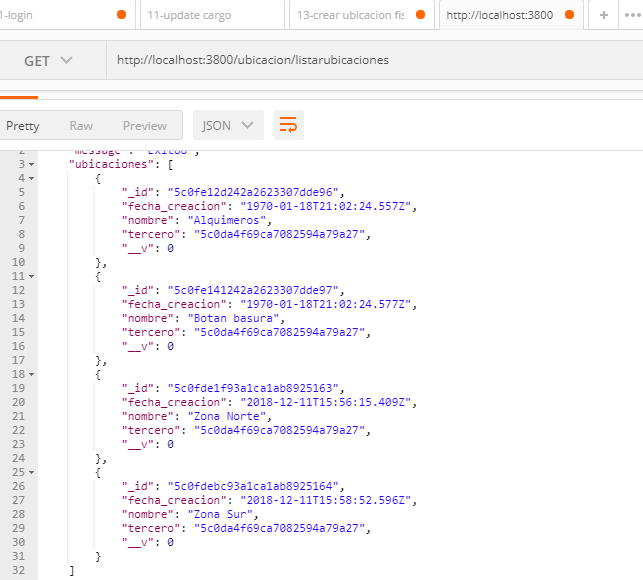
});

}

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});

});

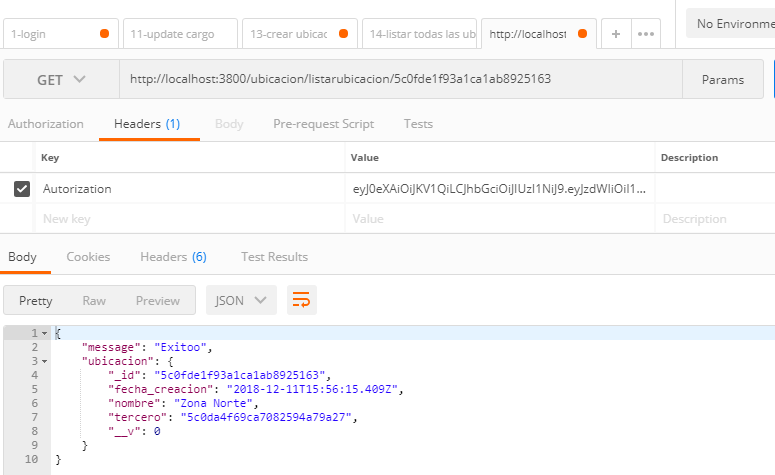
}

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');

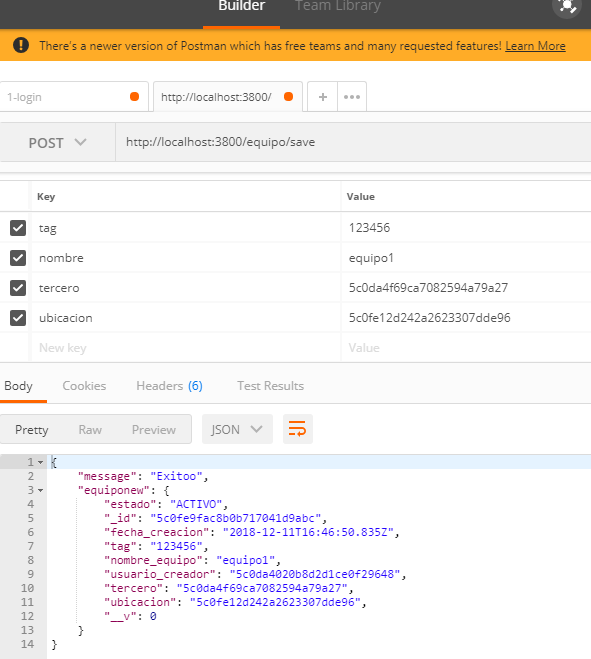
var EquipoController = require('../controllers/equipoController');

*//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});

});

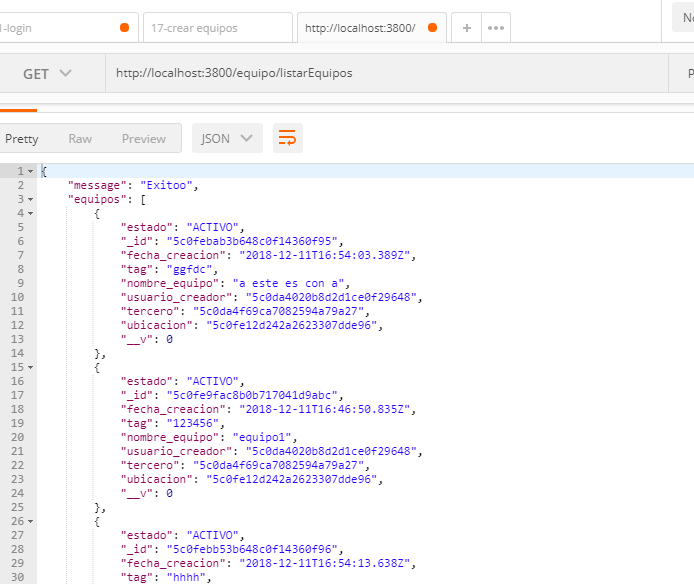
}

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});

});

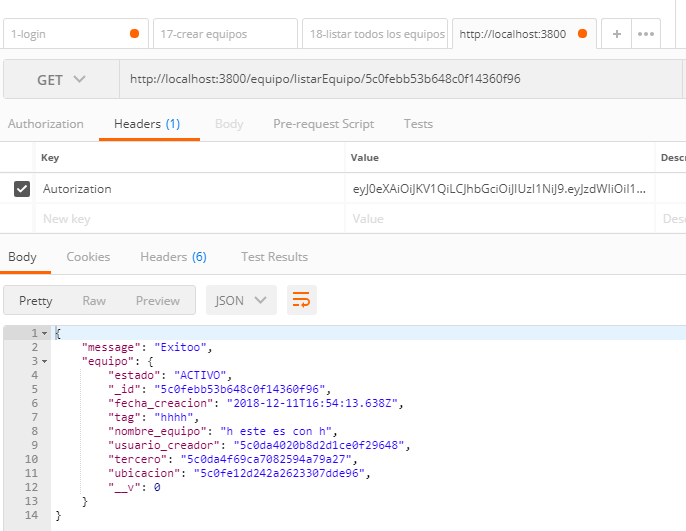
}

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});

});

});

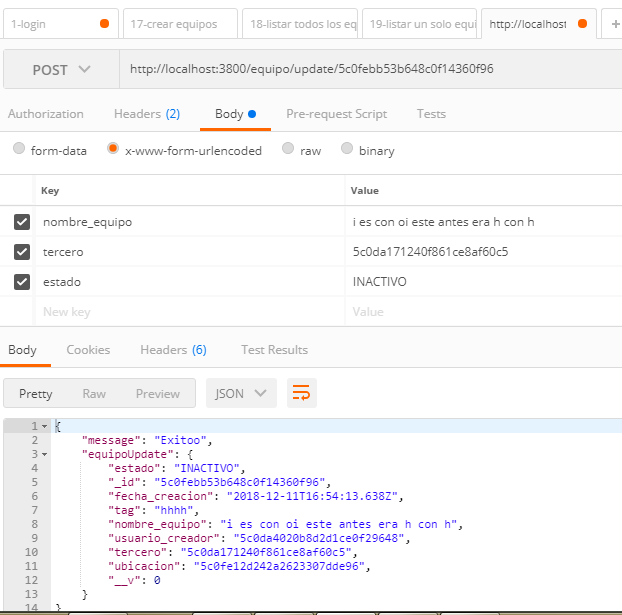
}

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 mongoose = require('mongoose');

var Schema = mongoose.Schema;

var ConsecutivoSchema = new Schema({

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

consecutivo: {type:Number, default:0, required:[true, 'El consecutivo empieza en cero y va sumando 1 a 1']},

tercero :{type:Schema.ObjectId, ref:'Tercero', unique:[true, "Este tercero ya tiene un consecutivo"], required:[true, 'El tercero es el cliente o proveedor a quien le pertenece el consecutivo']},

usuario\_creador :{type:Schema.ObjectId, ref:'Usuario' }

})

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

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

La ruta

'use strict'

const express = require('express');

var app = express.Router();

var ConsecutivoController = require('../controllers/consecutivoController');

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

//Crear Consecutivos

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

module.exports = app;

el controlador

'use strict'

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

var mongossePaginate = require('mongoose-pagination');

//Crear Consecutivos

function crearConsecutivos(req, res){

var body = req.body;

var tercero = body.tercero;

var Consecutivo = new ConsecutivoModel();

if(body.prefijo && body.tercero){

ConsecutivoModel.findOne({prefijo:body.prefijo}).exec((err, consecutivo)=>{

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

if(consecutivo) return res.status(200).send({message: "Usted ya Tiene un Consecutivo Asignado no es Posible Asigar Mas"});

if(!consecutivo){

Consecutivo.prefijo = body.prefijo;

Consecutivo.tercero = body.tercero;

Consecutivo.usuario\_creador = body.usuario\_creador;

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 los campos requeridos"});

}

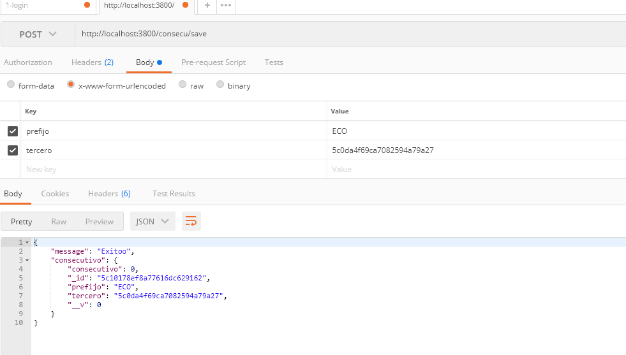
}

module.exports = {

crearConsecutivos,

}

la prueba

Exitooo

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});

});

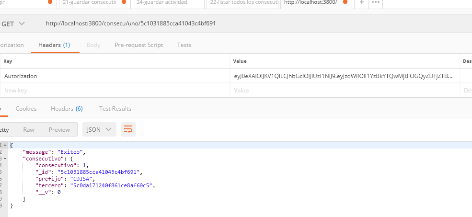
}

La ruta

*//Listar un solo consecutivo*

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

la prueba

exitooooo

**Creemos el controlador de Orden**

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

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

El modelo:

'use strict';

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var OrdenSchema = new Schema({

oti:{type:number, required},

fecha\_solicitud:{type:Date},

fecha\_requerida:{type:Date},

oti\_cliente :{type:string},

cliente: {type: Schema.ObjectId, ref:'Orden'},

centro\_costo: {type:String},

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

solicitante: {type:String},

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

usuario\_creador: {type:Schema.ObjectId, ref:'User', required:[true, 'requiere un usuario creador']},

usuario\_modificador: {type:Schema.ObjectId, ref:'User'},

items:[{

tag: {type:String},

nombre: {type:String},

marca: {type:String},

modelo: {type:String},

serial: {type:String},

tipo\_equipo: {type:String},

fabricante: {type:String},

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

estado: {type:String},

tipo\_indicacion: {type:String},

intervalo\_indicacion: {type:String},

resolucion: {type:String},

anexos: {type:String},

img: {type:String}

}]

})

la function de crear orden

function saveOrden(req, res){

var body = req.body;

var tercero = req.user.tercero;

var newOrden = new ordenModel();

*if*(body.fecha\_solicitud) newOrden.fecha\_solicitud = new Date(body.fecha\_solicitud);

*if*(body.fecha\_requerida) newOrden.fecha\_requerida = new Date(body.fecha\_requerida);

*if*(body.oti\_cliente) newOrden.oti\_cliente = body.oti\_cliente;

*if*(body.centro\_costo) newOrden.centro\_costo = body.centro\_costo;

*if*(body.ubicacion) newOrden.ubicacion = body.ubicacion;

*if*(body.solicitante) newOrden.solicitante = body.solicitante;

*if*(body.tercero) newOrden.tercero = body.tercero;

consecutivoModel.findOne({prefijo:'Oti'}).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"});

*if*(consecutivo){

consecutivo.consecutivo ++;

newOrden.oti = consecutivo.consecutivo;

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

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

*if*(err) *return* res.status(500).send({message:"error al ejecutar la accion en consecutivosn", error:err});

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

*if*(consecutivoActualizado){

newOrden.save((err, orden)=>{

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

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

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

});

};

});

;}

});

}

}

La ruta

'use strict';

var ordenController = require('../controllers/ordenController');

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

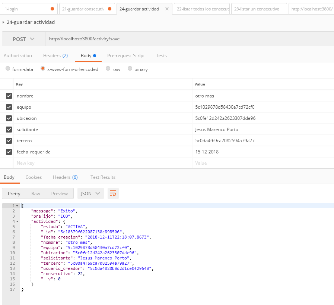
const express = require('express');

var app = express.Router();

*//save Order*

app.post('/save', md\_auth.ensureAuth, ordenController.saveOrden);

la prueba:

Exitoooo

La función de listar todas las actividades

*//Listar todas las ordenes*

function orderLists(req, res){

ordenModel.find({}).exec((err, orders)=>{

*if*(err) *return* res.status(500).send({message:"error alconsultar las ordenes", error:err});

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

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

});

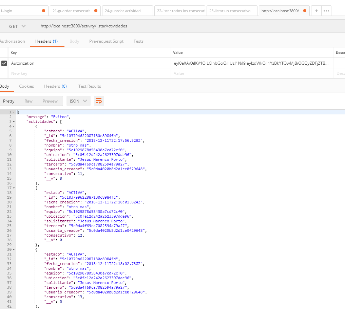
}

La ruta:

*//get Orders*

app.get('/list', md\_auth.ensureAuth, ordenController.orderLists);

la prueba

todo ok

La function de listar solo una actividad

*//List one order*

function orderListOne(req, res){

var id = req.params.id;

ordenModel.findById({\_id:id}).exec((err, order)=>{

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

*if*(!order) *return* res.status(404).send({message: "no se encontro la orden"});

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

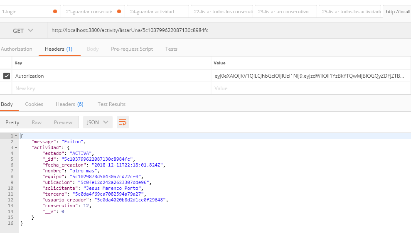
});

}

La ruta

app.get('/list/:id', md\_auth.ensureAuth, ordenController.orderListOne);

la prueba

Exitoooooo

Sigamos con el modelo, controlador, ruta y midllerare de recepción de equipos

**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

"dependencies": {

"@angular/animations": "~7.1.0",

"@angular/common": "~7.1.0",

"@angular/compiler": "~7.1.0",

"@angular/core": "~7.1.0",

"@angular/forms": "~7.1.0",

"@angular/platform-browser": "~7.1.0",

"@angular/platform-browser-dynamic": "~7.1.0",

"@angular/router": "~7.1.0",

"core-js": "^2.5.4",

"rxjs": "~6.3.3",

"tslib": "^1.9.0",

"zone.js": "~0.8.26",

"bootstrap": "4.0.0",

"jquery": "3.2.1"

},

3-ejecutamos npm update para que se actualice estas dos dependencias

4-hay que agregar Popper.js de la versión de bootstrap en el index.html

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/bootstrap/dist/js/bootstrap.min.js"

]

**Estamos listos**

1-creemos los componentes dasboard, login, navbar, sidebar con:

Ng g c components/nopmbre del componente - -spec=false

2:ahora busquemos el template llamado pages-blank.html y arrastremoslo al visual studio code.

Esto para ver las importaciones por ejemplo copiaresmos el link que imponta el icono, bootstap.css y el tema de nuestro proyecto y lo pegamos en el index.html de nuestro proyecto.

*<!-- Custom CSS -->*

<link *rel*="stylesheet" *href*="./assets/css/style.css">

*<!-- You can change the theme colors from here -->*

<link *id*="tema" *href*="./assets/css/colors/default-dark.css" *id*="theme" *rel*="stylesheet">

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

3:hay que corregir porque las imagines no las tenemos en ..assets sino en assets enyonces

4: Aun el bootstrap no esta pero lo arreglaremos copiando del material descargado el bootstrap que estara en assets/plugins/bootstrap, no copiemos todos los plugins porque son muchos solo copiemos el bootstrap entocnes deberemos crear la carpeta plugins en los assets

5: ahora sigamos con los js, que están abajo en el template, en este caso pugins de jquery que usa el template, el bootstrap, perfec-scrolbarr que es para que se vea mas bonito, el waves para que los botones tengan el efecto del material, el sidemenu, es decir todos los js.

<body

<app-root></app-root>

*<!-- All Jquery -->*

*<!-- ============================================================== -->*

<script *src*="../assets/plugins/jquery/jquery.min.js"></script>

*<!-- Bootstrap tether Core JavaScript -->*

<script *src*="../assets/plugins/bootstrap/js/popper.min.js"></script>

<script *src*="../assets/plugins/bootstrap/js/bootstrap.min.js"></script>

*<!-- slimscrollbar scrollbar JavaScript -->*

<script *src*="../assets/js/perfect-scrollbar.jquery.min.js"></script>

*<!--Wave Effects -->*

<script *src*="../assets/js/waves.js"></script>

*<!--Menu sidebar -->*

<script *src*="../assets/js/sidebarmenu.js"></script>

*<!--stickey kit -->*

<script *src*="../assets/plugins/sticky-kit-master/dist/sticky-kit.min.js"></script>

<script *src*="../assets/plugins/sparkline/jquery.sparkline.min.js"></script>

*<!--Custom JavaScript -->*

<script *src*="../assets/js/custom.js"></script>

*<!-- ============================================================== -->*

*<!-- Style switcher -->*

*<!-- ============================================================== -->*

<script *src*="../assets/plugins/styleswitcher/jQuery.style.switcher.js"></script>

</body>

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*>*

**Creemos el contenido de Breadcrumbs, Header y Sidebar:**

Sigamos usando el pages-blanck.html del template descargado y vayamos hasta el body y copiemos toda la clase y peguemosla en el index.html de nuestro proyecto.

*<*body class="fix-header card-no-border fix-sidebar"*>*

Quedaria de esta manera

*<*body class="fix-header card-no-border fix-sidebar"*>*

*<*app-root*></*app-root*>*

Usemos el prelouder, este muestra un pequeño loading que se quita cuando dispra la funcion listo de jquery, usémoslo también.

Lo clocaremos debajo del body justo antes del app-root

*<*body class="fix-header card-no-border fix-sidebar"*>* *<!-- ============================================================== -->*

*<!-- Preloader - style you can find in spinners.css -->*

*<!-- ============================================================== -->*

*<*div class="preloader"*>*

*<*div class="loader"*>*

*<*div class="loader\_\_figure"*></*div*>*

*<*p class="loader\_\_label"*>*Admin Pro*</*p*>*

*</*div*>*

*</*div*>*

*<*app-root*></*app-root*>*

Ahora empecemos, podemos observer que viene un div con id main-wrapper, desde aquí inicia el template. Y esto lo pegaremos en el app.component.html que es la vista principal de los componentes y allí borraremos todo y pegaremos ese div allí.

Debería quedar asi.

*<*div id="main-wrapper"*>*

*</*div*>*

Ahora dentro de ese div ira la etiqueta que correcponde al component del header que es <app-header> luego vamos al html del header y pegamos allí todo el header que se encuentra dentro de el main-wrapper original de el archivo pages-blank.html de la plantilla.

*<!-- ============================================================== -->*

*<!-- Topbar header - style you can find in pages.scss -->*

*<!-- ============================================================== -->*

<header *class*="topbar">

<nav *class*="navbar top-navbar navbar-expand-md navbar-light">

*<!-- ============================================================== -->*

*<!-- Logo -->*

*<!-- ============================================================== -->*

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

<a *class*="navbar-brand" *href*="index.html">

*<!-- Logo icon -->*<b>

*<!--You can put here icon as well // <i class="wi wi-sunset"></i> //-->*

*<!-- Dark Logo icon -->*

<img *src*="../assets/images/logo-icon.png" *alt*="homepage" *class*="dark-logo" />

*<!-- Light Logo icon -->*

<img *src*="../assets/images/logo-light-icon.png" *alt*="homepage" *class*="light-logo" />

</b>

*<!--End Logo icon -->*

*<!-- Logo text -->*<span>

*<!-- dark Logo text -->*

<img *src*="../assets/images/logo-text.png" *alt*="homepage" *class*="dark-logo" />

*<!-- Light Logo text -->*

<img *src*="../assets/images/logo-light-text.png" *class*="light-logo" *alt*="homepage" /></span> </a>

</div>

*<!-- ============================================================== -->*

*<!-- End Logo -->*

*<!-- ============================================================== -->*

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

*<!-- ============================================================== -->*

*<!-- toggle and nav items -->*

*<!-- ============================================================== -->*

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

*<!-- This is -->*

<li *class*="nav-item"> <a *class*="nav-link nav-toggler hidden-md-up waves-effect waves-dark" *href*="javascript:void(0)"><i *class*="ti-menu"></i></a> </li>

<li *class*="nav-item"> <a *class*="nav-link sidebartoggler hidden-sm-down waves-effect waves-dark" *href*="javascript:void(0)"><i *class*="ti-menu"></i></a> </li>

<li *class*="nav-item hidden-sm-down"></li>

</ul>

*<!-- ============================================================== -->*

*<!-- User profile and search -->*

*<!-- ============================================================== -->*

<ul *class*="navbar-nav my-lg-0">

*<!-- ============================================================== -->*

*<!-- Search -->*

*<!-- ============================================================== -->*

<li *class*="nav-item hidden-xs-down search-box"> <a *class*="nav-link hidden-sm-down waves-effect waves-dark" *href*="javascript:void(0)"><i *class*="ti-search"></i></a>

<form *class*="app-search">

<input *type*="text" *class*="form-control" *placeholder*="Search & enter"> <a *class*="srh-btn"><i *class*="ti-close"></i></a> </form>

</li>

*<!-- ============================================================== -->*

*<!-- Comment -->*

*<!-- ============================================================== -->*

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

<a *class*="nav-link dropdown-toggle waves-effect waves-dark" *href*="" *data-toggle*="dropdown" *aria-haspopup*="true" *aria-expanded*="false"> <i *class*="mdi mdi-message"></i>

<div *class*="notify"> <span *class*="heartbit"></span> <span *class*="point"></span> </div>

</a>

<div *class*="dropdown-menu dropdown-menu-right mailbox animated bounceInDown">

<ul>

<li>

<div *class*="drop-title">Notifications</div>

</li>

<li>

<div *class*="message-center">

*<!-- Message -->*

<a *href*="#">

<div *class*="btn btn-danger btn-circle"><i *class*="fa fa-link"></i></div>

<div *class*="mail-contnet">

<h5>Luanch Admin</h5> <span *class*="mail-desc">Just see the my new admin!</span> <span *class*="time">9:30 AM</span> </div>

</a>

*<!-- Message -->*

<a *href*="#">

<div *class*="btn btn-success btn-circle"><i *class*="ti-calendar"></i></div>

<div *class*="mail-contnet">

<h5>Event today</h5> <span *class*="mail-desc">Just a reminder that you have event</span> <span *class*="time">9:10 AM</span> </div>

</a>

*<!-- Message -->*

<a *href*="#">

<div *class*="btn btn-info btn-circle"><i *class*="ti-settings"></i></div>

<div *class*="mail-contnet">

<h5>Settings</h5> <span *class*="mail-desc">You can customize this template as you want</span> <span *class*="time">9:08 AM</span> </div>

</a>

*<!-- Message -->*

<a *href*="#">

<div *class*="btn btn-primary btn-circle"><i *class*="ti-user"></i></div>

<div *class*="mail-contnet">

<h5>Pavan kumar</h5> <span *class*="mail-desc">Just see the my admin!</span> <span *class*="time">9:02 AM</span> </div>

</a>

</div>

</li>

<li>

<a *class*="nav-link text-center" *href*="javascript:void(0);"> <strong>Check all notifications</strong> <i *class*="fa fa-angle-right"></i> </a>

</li>

</ul>

</div>

</li>

*<!-- ============================================================== -->*

*<!-- End Comment -->*

*<!-- ============================================================== -->*

*<!-- ============================================================== -->*

*<!-- Messages -->*

*<!-- ============================================================== -->*

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

<a *class*="nav-link dropdown-toggle waves-effect waves-dark" *href*="" *id*="2" *data-toggle*="dropdown" *aria-haspopup*="true" *aria-expanded*="false"> <i *class*="mdi mdi-email"></i>

<div *class*="notify"> <span *class*="heartbit"></span> <span *class*="point"></span> </div>

</a>

<div *class*="dropdown-menu mailbox dropdown-menu-right animated bounceInDown" *aria-labelledby*="2">

<ul>

<li>

<div *class*="drop-title">You have 4 new messages</div>

</li>

<li>

<div *class*="message-center">

*<!-- Message -->*

<a *href*="#">

<div *class*="user-img"> <img *src*="../assets/images/users/1.jpg" *alt*="user" *class*="img-circle"> <span *class*="profile-status online pull-right"></span> </div>

<div *class*="mail-contnet">

<h5>Pavan kumar</h5> <span *class*="mail-desc">Just see the my admin!</span> <span *class*="time">9:30 AM</span> </div>

</a>

*<!-- Message -->*

<a *href*="#">

<div *class*="user-img"> <img *src*="../assets/images/users/2.jpg" *alt*="user" *class*="img-circle"> <span *class*="profile-status busy pull-right"></span> </div>

<div *class*="mail-contnet">

<h5>Sonu Nigam</h5> <span *class*="mail-desc">I've sung a song! See you at</span> <span *class*="time">9:10 AM</span> </div>

</a>

*<!-- Message -->*

<a *href*="#">

<div *class*="user-img"> <img *src*="../assets/images/users/3.jpg" *alt*="user" *class*="img-circle"> <span *class*="profile-status away pull-right"></span> </div>

<div *class*="mail-contnet">

<h5>Arijit Sinh</h5> <span *class*="mail-desc">I am a singer!</span> <span *class*="time">9:08 AM</span> </div>

</a>

*<!-- Message -->*

<a *href*="#">

<div *class*="user-img"> <img *src*="../assets/images/users/4.jpg" *alt*="user" *class*="img-circle"> <span *class*="profile-status offline pull-right"></span> </div>

<div *class*="mail-contnet">

<h5>Pavan kumar</h5> <span *class*="mail-desc">Just see the my admin!</span> <span *class*="time">9:02 AM</span> </div>

</a>

</div>

</li>

<li>

<a *class*="nav-link text-center" *href*="javascript:void(0);"> <strong>See all e-Mails</strong> <i *class*="fa fa-angle-right"></i> </a>

</li>

</ul>

</div>

</li>

*<!-- ============================================================== -->*

*<!-- End Messages -->*

*<!-- ============================================================== -->*

*<!-- ============================================================== -->*

*<!-- mega menu -->*

*<!-- ============================================================== -->*

<li *class*="nav-item dropdown mega-dropdown"> <a *class*="nav-link dropdown-toggle waves-effect waves-dark" *href*="" *data-toggle*="dropdown" *aria-haspopup*="true" *aria-expanded*="false"><i *class*="mdi mdi-view-grid"></i></a>

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

<ul *class*="mega-dropdown-menu row">

<li *class*="col-lg-3 col-xlg-2 m-b-30">

<h4 *class*="m-b-20">CAROUSEL</h4>

*<!-- CAROUSEL -->*

<div *id*="carouselExampleControls" *class*="carousel slide" *data-ride*="carousel">

<div *class*="carousel-inner" *role*="listbox">

<div *class*="carousel-item active">

<div *class*="container"> <img *class*="d-block img-fluid" *src*="../assets/images/big/img1.jpg" *alt*="First slide"></div>

</div>

<div *class*="carousel-item">

<div *class*="container"><img *class*="d-block img-fluid" *src*="../assets/images/big/img2.jpg" *alt*="Second slide"></div>

</div>

<div *class*="carousel-item">

<div *class*="container"><img *class*="d-block img-fluid" *src*="../assets/images/big/img3.jpg" *alt*="Third slide"></div>

</div>

</div>

<a *class*="carousel-control-prev" *href*="#carouselExampleControls" *role*="button" *data-slide*="prev"> <span *class*="carousel-control-prev-icon" *aria-hidden*="true"></span> <span *class*="sr-only">Previous</span> </a>

<a *class*="carousel-control-next" *href*="#carouselExampleControls" *role*="button" *data-slide*="next"> <span *class*="carousel-control-next-icon" *aria-hidden*="true"></span> <span *class*="sr-only">Next</span> </a>

</div>

*<!-- End CAROUSEL -->*

</li>

<li *class*="col-lg-3 m-b-30">

<h4 *class*="m-b-20">ACCORDION</h4>

*<!-- Accordian -->*

<div *id*="accordion" *class*="nav-accordion" *role*="tablist" *aria-multiselectable*="true">

<div *class*="card">

<div *class*="card-header" *role*="tab" *id*="headingOne">

<h5 *class*="mb-0">

<a *data-toggle*="collapse" *data-parent*="#accordion" *href*="#collapseOne" *aria-expanded*="true" *aria-controls*="collapseOne">

Collapsible Group Item #1

</a>

</h5>

</div>

<div *id*="collapseOne" *class*="collapse show" *role*="tabpanel" *aria-labelledby*="headingOne">

<div *class*="card-body"> Anim pariatur cliche reprehenderit, enim eiusmod high. </div>

</div>

</div>

<div *class*="card">

<div *class*="card-header" *role*="tab" *id*="headingTwo">

<h5 *class*="mb-0">

<a *class*="collapsed" *data-toggle*="collapse" *data-parent*="#accordion" *href*="#collapseTwo" *aria-expanded*="false" *aria-controls*="collapseTwo">

Collapsible Group Item #2

</a>

</h5>

</div>

<div *id*="collapseTwo" *class*="collapse" *role*="tabpanel" *aria-labelledby*="headingTwo">

<div *class*="card-body"> Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. </div>

</div>

</div>

<div *class*="card">

<div *class*="card-header" *role*="tab" *id*="headingThree">

<h5 *class*="mb-0">

<a *class*="collapsed" *data-toggle*="collapse" *data-parent*="#accordion" *href*="#collapseThree" *aria-expanded*="false" *aria-controls*="collapseThree">

Collapsible Group Item #3

</a>

</h5>

</div>

<div *id*="collapseThree" *class*="collapse" *role*="tabpanel" *aria-labelledby*="headingThree">

<div *class*="card-body"> Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. </div>

</div>

</div>

</div>

</li>

<li *class*="col-lg-3 m-b-30">

<h4 *class*="m-b-20">CONTACT US</h4>

*<!-- Contact -->*

<form>

<div *class*="form-group">

<input *type*="text" *class*="form-control" *id*="exampleInputname1" *placeholder*="Enter Name"> </div>

<div *class*="form-group">

<input *type*="email" *class*="form-control" *placeholder*="Enter email"> </div>

<div *class*="form-group">

<textarea *class*="form-control" *id*="exampleTextarea" *rows*="3" *placeholder*="Message"></textarea>

</div>

<button *type*="submit" *class*="btn btn-info">Submit</button>

</form>

</li>

<li *class*="col-lg-3 col-xlg-4 m-b-30">

<h4 *class*="m-b-20">List style</h4>

*<!-- List style -->*

<ul *class*="list-style-none">

<li><a *href*="javascript:void(0)"><i *class*="fa fa-check text-success"></i> You can give link</a></li>

<li><a *href*="javascript:void(0)"><i *class*="fa fa-check text-success"></i> Give link</a></li>

<li><a *href*="javascript:void(0)"><i *class*="fa fa-check text-success"></i> Another Give link</a></li>

<li><a *href*="javascript:void(0)"><i *class*="fa fa-check text-success"></i> Forth link</a></li>

<li><a *href*="javascript:void(0)"><i *class*="fa fa-check text-success"></i> Another fifth link</a></li>

</ul>

</li>

</ul>

</div>

</li>

*<!-- ============================================================== -->*

*<!-- End mega menu -->*

*<!-- ============================================================== -->*

*<!-- ============================================================== -->*

*<!-- Language -->*

*<!-- ============================================================== -->*

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

<a *class*="nav-link dropdown-toggle waves-effect waves-dark" *href*="" *data-toggle*="dropdown" *aria-haspopup*="true" *aria-expanded*="false"> <i *class*="flag-icon flag-icon-us"></i></a>

<div *class*="dropdown-menu dropdown-menu-right animated bounceInDown"> <a *class*="dropdown-item" *href*="#"><i *class*="flag-icon flag-icon-in"></i> India</a> <a *class*="dropdown-item" *href*="#"><i *class*="flag-icon flag-icon-fr"></i> French</a> <a *class*="dropdown-item" *href*="#"><i *class*="flag-icon flag-icon-cn"></i> China</a> <a *class*="dropdown-item" *href*="#"><i *class*="flag-icon flag-icon-de"></i> Dutch</a> </div>

</li>

*<!-- ============================================================== -->*

*<!-- Profile -->*

*<!-- ============================================================== -->*

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

<a *class*="nav-link dropdown-toggle waves-effect waves-dark" *href*="" *data-toggle*="dropdown" *aria-haspopup*="true" *aria-expanded*="false"><img *src*="../assets/images/users/1.jpg" *alt*="user" *class*="profile-pic" /></a>

<div *class*="dropdown-menu dropdown-menu-right animated flipInY">

<ul *class*="dropdown-user">

<li>

<div *class*="dw-user-box">

<div *class*="u-img"><img *src*="../assets/images/users/1.jpg" *alt*="user"></div>

<div *class*="u-text">

<h4>Steave Jobs</h4>

<p *class*="text-muted">varun@gmail.com</p><a *href*="pages-profile.html" *class*="btn btn-rounded btn-danger btn-sm">View Profile</a></div>

</div>

</li>

<li *role*="separator" *class*="divider"></li>

<li><a *href*="#"><i *class*="ti-user"></i> My Profile</a></li>

<li><a *href*="#"><i *class*="ti-wallet"></i> My Balance</a></li>

<li><a *href*="#"><i *class*="ti-email"></i> Inbox</a></li>

<li *role*="separator" *class*="divider"></li>

<li><a *href*="#"><i *class*="ti-settings"></i> Account Setting</a></li>

<li *role*="separator" *class*="divider"></li>

<li><a *href*="#"><i *class*="fa fa-power-off"></i> Logout</a></li>

</ul>

</div>

</li>

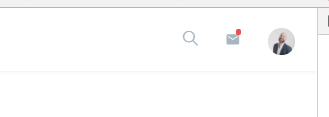
</ul>

</div>

</nav>

</header>

Quitemos algunas cosas en nuestro header es decir los iconos de notificaciones y el megamenu también comentemos lo de los lenguajes es posible que lo usemos mas adelante. Quedando de la siguiente forma.



Hacemos lo mismo para el left-sidebar en el comnponente sidebar

*<!-- ============================================================== -->*

<aside *class*="left-sidebar">

*<!-- Sidebar scroll-->*

<div *class*="scroll-sidebar">

*<!-- Sidebar navigation-->*

<nav *class*="sidebar-nav">

<ul *id*="sidebarnav">

<li *class*="user-profile">

<a *class*="has-arrow waves-effect waves-dark " *href*="#" *aria-expanded*="false"><img *src*="../assets/images/users/profile.png" *alt*="user" /><span *class*="hide-menu">Steave Jobs </span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="javascript:void()">My Profile </a></li>

<li><a *href*="javascript:void()">My Balance</a></li>

<li><a *href*="javascript:void()">Inbox</a></li>

<li><a *href*="javascript:void()">Account Setting</a></li>

<li><a *href*="javascript:void()">Logout</a></li>

</ul>

</li>

<li *class*="nav-devider"></li>

<li *class*="nav-small-cap">PERSONAL</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-gauge"></i><span *class*="hide-menu">Dashboard <span *class*="label label-rouded label-themecolor pull-right">4</span></span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="index.html">Minimal </a></li>

<li><a *href*="index2.html">Analytical</a></li>

<li><a *href*="index3.html">Demographical</a></li>

<li><a *href*="index4.html">Modern</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-bullseye"></i><span *class*="hide-menu">Apps</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="app-calendar.html">Calendar</a></li>

<li><a *href*="app-chat.html">Chat app</a></li>

<li><a *href*="app-ticket.html">Support Ticket</a></li>

<li><a *href*="app-contact.html">Contact / Employee</a></li>

<li><a *href*="app-contact2.html">Contact Grid</a></li>

<li><a *href*="app-contact-detail.html">Contact Detail</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-email"></i><span *class*="hide-menu">Inbox</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="app-email.html">Mailbox</a></li>

<li><a *href*="app-email-detail.html">Mailbox Detail</a></li>

<li><a *href*="app-compose.html">Compose Mail</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-chart-bubble"></i><span *class*="hide-menu">Ui Elements <span *class*="label label-rouded label-danger pull-right">25</span></span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="ui-cards.html">Cards</a></li>

<li><a *href*="ui-user-card.html">User Cards</a></li>

<li><a *href*="ui-buttons.html">Buttons</a></li>

<li><a *href*="ui-modals.html">Modals</a></li>

<li><a *href*="ui-tab.html">Tab</a></li>

<li><a *href*="ui-tooltip-popover.html">Tooltip &amp; Popover</a></li>

<li><a *href*="ui-tooltip-stylish.html">Tooltip stylish</a></li>

<li><a *href*="ui-sweetalert.html">Sweet Alert</a></li>

<li><a *href*="ui-notification.html">Notification</a></li>

<li><a *href*="ui-progressbar.html">Progressbar</a></li>

<li><a *href*="ui-nestable.html">Nestable</a></li>

<li><a *href*="ui-range-slider.html">Range slider</a></li>

<li><a *href*="ui-timeline.html">Timeline</a></li>

<li><a *href*="ui-typography.html">Typography</a></li>

<li><a *href*="ui-horizontal-timeline.html">Horizontal Timeline</a></li>

<li><a *href*="ui-session-timeout.html">Session Timeout</a></li>

<li><a *href*="ui-session-ideal-timeout.html">Session Ideal Timeout</a></li>

<li><a *href*="ui-bootstrap.html">Bootstrap Ui</a></li>

<li><a *href*="ui-breadcrumb.html">Breadcrumb</a></li>

<li><a *href*="ui-bootstrap-switch.html">Bootstrap Switch</a></li>

<li><a *href*="ui-list-media.html">List Media</a></li>

<li><a *href*="ui-ribbons.html">Ribbons</a></li>

<li><a *href*="ui-grid.html">Grid</a></li>

<li><a *href*="ui-carousel.html">Carousel</a></li>

<li><a *href*="ui-date-paginator.html">Date-paginator</a></li>

<li><a *href*="ui-dragable-portlet.html">Dragable Portlet</a></li>

</ul>

</li>

<li *class*="nav-small-cap">FORMS, TABLE &amp; WIDGETS</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-file"></i><span *class*="hide-menu">Forms</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="form-basic.html">Basic Forms</a></li>

<li><a *href*="form-layout.html">Form Layouts</a></li>

<li><a *href*="form-addons.html">Form Addons</a></li>

<li><a *href*="form-material.html">Form Material</a></li>

<li><a *href*="form-float-input.html">Floating Lable</a></li>

<li><a *href*="form-pickers.html">Form Pickers</a></li>

<li><a *href*="form-upload.html">File Upload</a></li>

<li><a *href*="form-mask.html">Form Mask</a></li>

<li><a *href*="form-validation.html">Form Validation</a></li>

<li><a *href*="form-dropzone.html">File Dropzone</a></li>

<li><a *href*="form-icheck.html">Icheck control</a></li>

<li><a *href*="form-img-cropper.html">Image Cropper</a></li>

<li><a *href*="form-bootstrapwysihtml5.html">HTML5 Editor</a></li>

<li><a *href*="form-typehead.html">Form Typehead</a></li>

<li><a *href*="form-wizard.html">Form Wizard</a></li>

<li><a *href*="form-xeditable.html">Xeditable Editor</a></li>

<li><a *href*="form-summernote.html">Summernote Editor</a></li>

<li><a *href*="form-tinymce.html">Tinymce Editor</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-table"></i><span *class*="hide-menu">Tables</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="table-basic.html">Basic Tables</a></li>

<li><a *href*="table-layout.html">Table Layouts</a></li>

<li><a *href*="table-data-table.html">Data Tables</a></li>

<li><a *href*="table-footable.html">Footable</a></li>

<li><a *href*="table-jsgrid.html">Js Grid Table</a></li>

<li><a *href*="table-responsive.html">Responsive Table</a></li>

<li><a *href*="table-bootstrap.html">Bootstrap Tables</a></li>

<li><a *href*="table-editable-table.html">Editable Table</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-widgets"></i><span *class*="hide-menu">Widgets</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="widget-data.html">Data Widgets</a></li>

<li><a *href*="widget-apps.html">Apps Widgets</a></li>

<li><a *href*="widget-charts.html">Charts Widgets</a></li>

</ul>

</li>

<li *class*="nav-small-cap">EXTRA COMPONENTS</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-book-multiple"></i><span *class*="hide-menu">Page Layout</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="layout-single-column.html">1 Column</a></li>

<li><a *href*="layout-fix-header.html">Fix header</a></li>

<li><a *href*="layout-fix-sidebar.html">Fix sidebar</a></li>

<li><a *href*="layout-fix-header-sidebar.html">Fixe header &amp; Sidebar</a></li>

<li><a *href*="layout-boxed.html">Boxed Layout</a></li>

<li><a *href*="layout-logo-center.html">Logo in Center</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-book-open-variant"></i><span *class*="hide-menu">Sample Pages <span *class*="label label-rouded label-success pull-right">25</span></span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="starter-kit.html">Starter Kit</a></li>

<li><a *href*="pages-blank.html">Blank page</a></li>

<li><a *href*="#" *class*="has-arrow">Authentication <span *class*="label label-rounded label-success pull-right">6</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="pages-login.html">Login 1</a></li>

<li><a *href*="pages-login-2.html">Login 2</a></li>

<li><a *href*="pages-register.html">Register</a></li>

<li><a *href*="pages-register2.html">Register 2</a></li>

<li><a *href*="pages-lockscreen.html">Lockscreen</a></li>

<li><a *href*="pages-recover-password.html">Recover password</a></li>

</ul>

</li>

<li><a *href*="pages-profile.html">Profile page</a></li>

<li><a *href*="pages-animation.html">Animation</a></li>

<li><a *href*="pages-fix-innersidebar.html">Sticky Left sidebar</a></li>

<li><a *href*="pages-fix-inner-right-sidebar.html">Sticky Right sidebar</a></li>

<li><a *href*="pages-invoice.html">Invoice</a></li>

<li><a *href*="pages-treeview.html">Treeview</a></li>

<li><a *href*="pages-utility-classes.html">Helper Classes</a></li>

<li><a *href*="pages-search-result.html">Search result</a></li>

<li><a *href*="pages-scroll.html">Scrollbar</a></li>

<li><a *href*="pages-pricing.html">Pricing</a></li>

<li><a *href*="pages-lightbox-popup.html">Lighbox popup</a></li>

<li><a *href*="pages-gallery.html">Gallery</a></li>

<li><a *href*="pages-faq.html">Faqs</a></li>

<li><a *href*="#" *class*="has-arrow">Error Pages</a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="pages-error-400.html">400</a></li>

<li><a *href*="pages-error-403.html">403</a></li>

<li><a *href*="pages-error-404.html">404</a></li>

<li><a *href*="pages-error-500.html">500</a></li>

<li><a *href*="pages-error-503.html">503</a></li>

</ul>

</li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-file-chart"></i><span *class*="hide-menu">Charts</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="chart-morris.html">Morris Chart</a></li>

<li><a *href*="chart-chartist.html">Chartis Chart</a></li>

<li><a *href*="chart-echart.html">Echarts</a></li>

<li><a *href*="chart-flot.html">Flot Chart</a></li>

<li><a *href*="chart-knob.html">Knob Chart</a></li>

<li><a *href*="chart-chart-js.html">Chartjs</a></li>

<li><a *href*="chart-sparkline.html">Sparkline Chart</a></li>

<li><a *href*="chart-extra-chart.html">Extra chart</a></li>

<li><a *href*="chart-peity.html">Peity Charts</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-brush"></i><span *class*="hide-menu">Icons</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="icon-material.html">Material Icons</a></li>

<li><a *href*="icon-fontawesome.html">Fontawesome Icons</a></li>

<li><a *href*="icon-themify.html">Themify Icons</a></li>

<li><a *href*="icon-linea.html">Linea Icons</a></li>

<li><a *href*="icon-weather.html">Weather Icons</a></li>

<li><a *href*="icon-simple-lineicon.html">Simple Lineicons</a></li>

<li><a *href*="icon-flag.html">Flag Icons</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-map-marker"></i><span *class*="hide-menu">Maps</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="map-google.html">Google Maps</a></li>

<li><a *href*="map-vector.html">Vector Maps</a></li>

</ul>

</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-arrange-send-backward"></i><span *class*="hide-menu">Multi level dd</span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="#">item 1.1</a></li>

<li><a *href*="#">item 1.2</a></li>

<li> <a *class*="has-arrow" *href*="#" *aria-expanded*="false">Menu 1.3</a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="#">item 1.3.1</a></li>

<li><a *href*="#">item 1.3.2</a></li>

<li><a *href*="#">item 1.3.3</a></li>

<li><a *href*="#">item 1.3.4</a></li>

</ul>

</li>

<li><a *href*="#">item 1.4</a></li>

</ul>

</li>

</ul>

</nav>

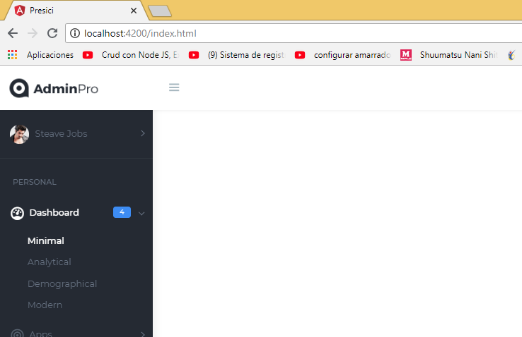
*<!-- End Sidebar navigation -->*

</div>

*<!-- End Sidebar scroll-->*

</aside>

Deberia quedar asi



Podemos observar que hay muchos elementos quedémonos con lo que nos sirve y dejemos solo lo del profile y personal el resto borremos todo. Debería quedar asi

*<!-- ============================================================== -->*

<aside *class*="left-sidebar">

*<!-- Sidebar scroll-->*

<div *class*="scroll-sidebar">

*<!-- Sidebar navigation-->*

<nav *class*="sidebar-nav">

<ul *id*="sidebarnav">

<li *class*="user-profile">

<a *class*="has-arrow waves-effect waves-dark " *href*="#" *aria-expanded*="false"><img *src*="../assets/images/users/profile.png" *alt*="user" /><span *class*="hide-menu">Steave Jobs </span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="javascript:void()">My Profile </a></li>

<li><a *href*="javascript:void()">My Balance</a></li>

<li><a *href*="javascript:void()">Inbox</a></li>

<li><a *href*="javascript:void()">Account Setting</a></li>

<li><a *href*="javascript:void()">Logout</a></li>

</ul>

</li>

<li *class*="nav-devider"></li>

<li *class*="nav-small-cap">App</li>

<li> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-gauge"></i><span *class*="hide-menu">Dashboard <span *class*="label label-rouded label-themecolor pull-right">4</span></span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="index.html">Minimal </a></li>

<li><a *href*="index2.html">Analytical</a></li>

<li><a *href*="index3.html">Demographical</a></li>

<li><a *href*="index4.html">Modern</a></li>

</ul>

</li>

</ul>

</nav>

*<!-- End Sidebar navigation -->*

</div>

*<!-- End Sidebar scroll-->*

</aside>

Ahora viene el **page-wrapper** esto es necesario para que el template funcione bien ese seria el envoltorio de lo que será el content principal y lo colocare en el app.component.html después del sidebar.

Tambein el **container-fluid** de adentro **page-wrapper** quedando de la siguiente manena inicialmente.

<div *id*="main-wrapper">

<app-header></app-header>

<app-sidebar></app-sidebar>

*<!-- ==================page-wrapper==================================== -->*

<div *class*="page-wrapper">

*<!-- ============================================================== -->*

*<!-- Container fluid -->*

*<!-- ============================================================== -->*

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

</div>

*<!-- ==End Container fluid========================================= -->*

</div>

*<!-- ==============End page-wrapper==================================== -->*

</div>

Luego también dentro vendrá el div del bread crumb que será el directorio interno lo compiamos completo y pegamos en el componente breadcrumbs en su vista y quitemos el ultimo div que tiene un botón ya que no lo usaremos. Y al app.component.html traeremos la etiqueda de configuración del componente breadcrumbs.

Breadcrumbs.component.html

*<!-- ============================================================== -->*

*<!-- Bread crumb and right sidebar toggle -->*

*<!-- ============================================================== -->*

<div *class*="row page-titles">

<div *class*="col-md-5 align-self-center">

<h3 *class*="text-themecolor">Blank Page</h3>

</div>

<div *class*="col-md-7 align-self-center">

<ol *class*="breadcrumb">

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

<a *href*="javascript:void(0)">Home</a>

</li>

<li *class*="breadcrumb-item">pages</li>

<li *class*="breadcrumb-item active">Blank Page</li>

</ol>

</div>

</div>

*<!-- ============================================================== -->*

*<!-- End Bread crumb and right sidebar toggle -->*

*<!-- ============================================================== -->*

Ahora con la etiqueta de este componente la integraremos al app.component.html dentro del container-fluid del page-wrapper

<div *id*="main-wrapper">

<app-header></app-header>

<app-sidebar></app-sidebar>

*<!-- ==================page-wrapper==================================== -->*

<div *class*="page-wrapper">

*<!-- ============================================================== -->*

*<!-- Container fluid -->*

*<!-- ============================================================== -->*

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

<app-breadcrumbs></app-breadcrumbs>

</div>

*<!-- ==End Container fluid========================================= -->*

</div>

*<!-- ==============End page-wrapper==================================== -->*

</div>

Por ultimo debajo de breadcrumbs colocaremos el div donde ira todo el contenido que es el star-page-content esto completo debería quedar asi:

<div *id*="main-wrapper">

<app-header></app-header>

<app-sidebar></app-sidebar>

*<!-- ==================page-wrapper==================================== -->*

<div *class*="page-wrapper">

*<!-- ============================================================== -->*

*<!-- Container fluid -->*

*<!-- ============================================================== -->*

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

<app-breadcrumbs></app-breadcrumbs>

*<!-- ============================================================== -->*

*<!-- Start Page Content -->*

*<!-- ============================================================== -->*

<div *class*="row">

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

<div *class*="card">

<div *class*="card-body">

This is some text within a card block.

</div>

</div>

</div>

</div>

*<!-- =====================End Page Content ====================== -->*

</div>

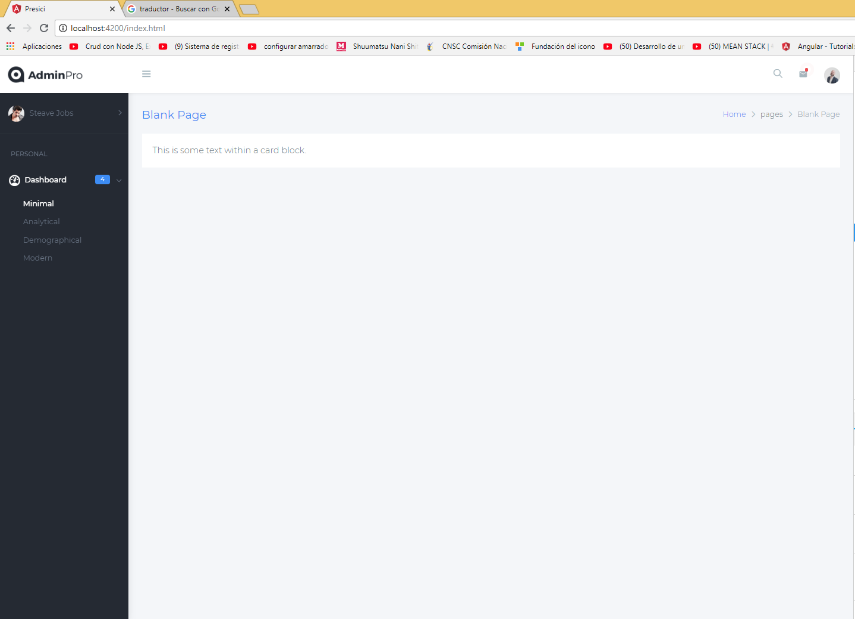
*<!-- ==End Container fluid========================================= -->*

</div>

*<!-- ==============End page-wrapper==================================== -->*

</div>

El render debería quedar asi.



**Implementando las rutas principales:**

Aquí implantaremos las rutas que cambiaran el contenido que se encuentre en medio.

Entones creemos en app el archivo app.routes.ts, este será mi archivo de rutas principales.

1: importamos el modulo routes.

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

2-Creemos una constante appRoutes de tipo Routes para definir un arreglo de rutas; Cuando termino de agregar el componente y le doy enter el me importara automáticamente el componente. Quedando la importación de componentes y la constante de la siguiente forma.

*import* { Routes } *from* "@angular/router";

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

const appRoutes : Routes = [

{path:'dashboard', component: DashboardComponent}

];

Y asi haremos con los componentes login que aun no hemos creado pero lo colocaremos, también crearemos una ruta con doble \*\* para llevar allí al componente de pagina no encontrada

*import* { Routes } *from* "@angular/router";

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

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

*import* { PagenofoundComponent } *from* './components/pagenofound/pagenofound.component';

const appRoutes : Routes = [

{path:'dashboard', component: DashboardComponent},

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

{path:'\*\*', component: PagenofoundComponent},

];

Ahora necesitamos tambien importar al lado del modulo Router el RouterModule

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

y exportemos el modulo de la siguiente manera: tambien usemos el hash asi podre enviar un objeto useHash.

export const APP\_ROUTES = RouterModule.forRoot(appRoutes, {useHash: true});

con esto basta para que funcione la aplicación

ahora para usar el app.routing.ts debere importarlo en el app.module.ts

import { APP\_ROUTES } from './app.routes';

y lo agregaremos l array de los imports.

imports: [

BrowserModule,

APP\_ROUTES

],

Ahora deberemos usar la etiqueta <router-outlet> para activar el ruteo y eso lo haremos en el app.component.html en el star Page

<div *id*="main-wrapper">

<app-header></app-header>

<app-sidebar></app-sidebar>

*<!-- ==================page-wrapper==================================== -->*

<div *class*="page-wrapper">

*<!-- ============================================================== -->*

*<!-- Container fluid -->*

*<!-- ============================================================== -->*

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

<app-breadcrumbs></app-breadcrumbs>

*<!-- ============================================================== -->*

*<!-- Start Page Content -->*

*<!-- ============================================================== -->*

<div *class*="row">

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

<div *class*="card">

<div *class*="card-body">

<router-outet></router-outet>

</div>

</div>

</div>

</div>

*<!-- =====================End Page Content ====================== -->*

</div>

*<!-- ==End Container fluid========================================= -->*

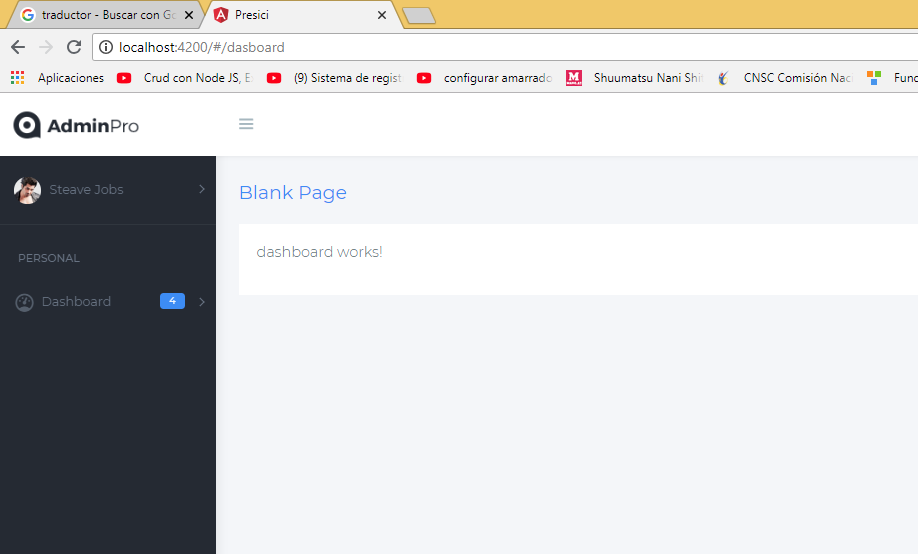
</div>

*<!-- ==============End page-wrapper==================================== -->*

</div>

Ahora si accedemos a la ruta del dashboard que ya tiene un componente creado e inicializado en el archivo de rutas podremos observar lo siguiente

<http://localhost:4200/#/dasboard>



Y si modificamos las rutas tambien cambiaran.

**Trabajemos con rutas hijas para separar el login de los demás componentes**

1-organicemos mejor la estructura del código:

Todo aquello que forme parte de la estructura del frontend es decir **header**, **breadbrumbs**, **sidebar** lo meteremos dentro de una carpeta que crearemos dentro de components llamada **frontstructure**

2-creemos un nuevo componente en una nueva carpeta llamada pages, esta carpeta será la base para la creación de las rutas hijas.

**ng g c pages/pages --flat -is --spect=false esto nos creara el .ts y .html dentro de la carpeta pero sin una subcarpeta que lo contenga, no creara tampoco el css y tampoco el spec.**

Ahora todo el contenido del app.component.html cortemoslo y peguemoslo en el pages.component.html.

En app.component.html colocare la etiqueta de routeo <router-outlet></ router-outlet>

Ahora implementaremos unas rutas hijas o secundarias para que el contenido vaya cambiando mientras nuestra ruta padre sea path‘ ’, Entonces en el app.routter.ts declararemos las rutas hijas dentro del path que acabamos de modificar. Esto lo haremos de la siguiente manera.

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

*import* { DashboardComponent } *from* './components/pages/dashboard/dashboard.component';

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

*import* { PagenofoundComponent } *from* './components/pagenofound/pagenofound.component';

*import* { PagesComponent } *from* './components/pages/pages.component';

const appRoutes : Routes = [

{path:'',

component: PagesComponent,

children:[

{path:'dashboard', component: DashboardComponent},

{path:'', redirectTo:'/dashboard', pathMatch:'full'}

]

},

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

{path:'\*\*', component: PagenofoundComponent},

];

*export* const APP\_ROUTES = RouterModule.forRoot(appRoutes, {useHash:true});

De este modo volverá a funcionar el dashboard y sus rutas hijas desde el page.component.ts, también usaremos redirectTo para cuando la ruta hija este vacia lo redireccione siempre al dashboard asi separamos todo de la vista login.

Ahora creemos la pagina de login, abramos en visual page-login2.htm del material descargado.

Copiaremos solo el contenido que esta dentro del section concomentario Main wrapper y lo pegaremos en el componente de login.

*<!-- ============================================================== -->*

*<!-- Main wrapper - style you can find in pages.scss -->*

*<!-- ============================================================== -->*

<section *id*="wrapper" *class*="login-register login-sidebar" *style*="background-image:url(../assets/images/background/login-register.jpg);">

<div *class*="login-box card">

<div *class*="card-body">

<form *class*="form-horizontal form-material" *id*="loginform" *action*="index.html">

<a *href*="javascript:void(0)" *class*="text-center db"><img *src*="../assets/images/logo-icon.png" *alt*="Home" /><br/><img *src*="../assets/images/logo-text.png" *alt*="Home" /></a>

<div *class*="form-group m-t-40">

<div *class*="col-xs-12">

<input *class*="form-control" *type*="text" *required*="" *placeholder*="Username">

</div>

</div>

<div *class*="form-group">

<div *class*="col-xs-12">

<input *class*="form-control" *type*="password" *required*="" *placeholder*="Password">

</div>

</div>

<div *class*="form-group row">

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

<div *class*="checkbox checkbox-primary pull-left p-t-0">

<input *id*="checkbox-signup" *type*="checkbox" *class*="filled-in chk-col-light-blue">

<label *for*="checkbox-signup"> Remember me </label>

</div>

<a *href*="javascript:void(0)" *id*="to-recover" *class*="text-dark pull-right"><i *class*="fa fa-lock m-r-5"></i> Forgot pwd?</a> </div>

</div>

<div *class*="form-group text-center m-t-20">

<div *class*="col-xs-12">

<button *class*="btn btn-info btn-lg btn-block text-uppercase btn-rounded" *type*="submit">Log In</button>

</div>

</div>

<div *class*="row">

<div *class*="col-xs-12 col-sm-12 col-md-12 m-t-10 text-center">

<div *class*="social"><a *href*="javascript:void(0)" *class*="btn btn-facebook" *data-toggle*="tooltip" *title*="Login with Facebook"> <i *aria-hidden*="true" *class*="fa fa-facebook"></i> </a> <a *href*="javascript:void(0)" *class*="btn btn-googleplus" *data-toggle*="tooltip" *title*="Login with Google"> <i *aria-hidden*="true" *class*="fa fa-google-plus"></i> </a> </div>

</div>

</div>

<div *class*="form-group m-b-0">

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

Don't have an account? <a *href*="pages-register2.html" *class*="text-primary m-l-5"><b>Sign Up</b></a>

</div>

</div>

</form>

<form *class*="form-horizontal" *id*="recoverform" *action*="index.html">

<div *class*="form-group ">

<div *class*="col-xs-12">

<h3>Recover Password</h3>

<p *class*="text-muted">Enter your Email and instructions will be sent to you! </p>

</div>

</div>

<div *class*="form-group ">

<div *class*="col-xs-12">

<input *class*="form-control" *type*="text" *required*="" *placeholder*="Email">

</div>

</div>

<div *class*="form-group text-center m-t-20">

<div *class*="col-xs-12">

<button *class*="btn btn-primary btn-lg btn-block text-uppercase waves-effect waves-light" *type*="submit">Reset</button>

</div>

</div>

</form>

</div>

</div>

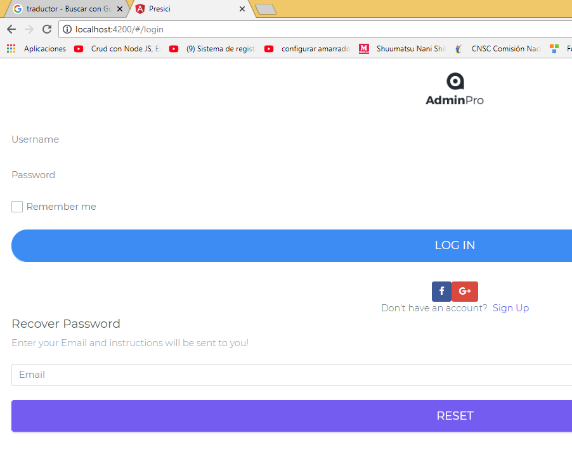
</section>

*<!-- ============================================================== -->*

*<!-- End Wrapper -->*

*<!-- ============================================================== -->*

esto es lo que aparece si escribimos en nuestra url la siguiente direccion. <http://localhost:4200/#/login>

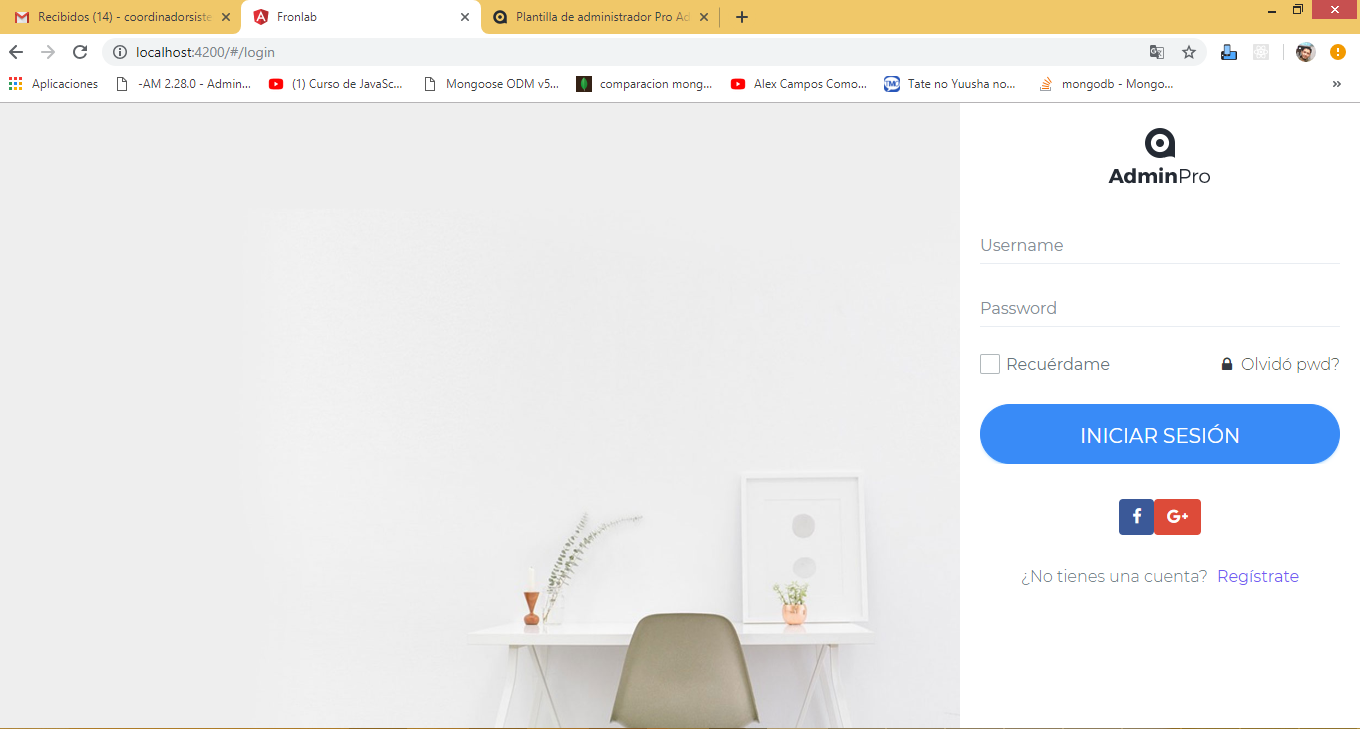


Podemos observar que no se parece en nada al template de la plantilla y es porque aun no le hemos aplicado los estilos, en el encabezado de la plantilla hay un estilo que se llama

<link href="css/pages/login-register-lock.css" rel="stylesheet">

Desde la plantilla abriremos ese css lo copiaremos y lo pegaremos en el css del component de login

Para que se arregle



Como ya tenemos rutas hijas debemos utilizar una organización mayor, lo ideal es que las rutas hijas, tengan un modulo aparte del de la raíz asi que en pages crearemos este modulo llamándolo pages.module.ts

la estructura de un modulo se compone de la importación del NgModule y en @NgModule

pages.module.ts

en declaration colocaremos los componentes hijos de page y en exports colocaremos los mismos componentes para poderlos usar fuera de su padre.

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

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

@NgModule({

declarations:[

DashboardComponent

],

exports:[

DashboardComponent

]

})

*export* class PageModule{}

Ahora importaremos el modulo creado en el app.module.ts.

imports: [

BrowserModule,

APP\_ROUTES,

PageModule

],

También deberemos quitarlos del @NgModule en el apartado del array declaratios, ya que los estamos importando en el PageModule. El app.module.ts debería quedar algo asi.

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

*import* { APP\_ROUTES } *from* './app.routes';

*import* { PageModule } *from* './components/pages/pages.module';

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

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

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

*import* { PagenofoundComponent } *from* './components/pagenofound/pagenofound.component';

@NgModule({

declarations: [

AppComponent,

LoginComponent,

PagenofoundComponent,

],

imports: [

BrowserModule,

APP\_ROUTES,

PageModule

],

providers: [],

bootstrap: [AppComponent]

})

*export* class AppModule { }

es que si yo tengo un modulo debo de importarlo en el archivo de app.modules.ts, como quite del Array **declarations** los modulos de DashboarComponent, ProgressComponent, PagesComponent, Graficas1Component me arrojara un error, aunque yo después haya importado en el app.module.ts el modulo de PagesModule.ts que es quien tienen estos componentes, pero porque se da el error en realidad, simplemente porque el componente PagesComponent requiere en su vista los componentes sidebar, breadcrums y header quienes no están importados en el modulo page.module.ts

Por lo tanto para que todo empiece a funcionar y sigamos resumiendo el app.module.ts, hagamos también lo mismo para los componentes que están en **frontstructure** es decir breadbrumbs, header y sidebar y luego importemos ese modulo en el page.module.ts y en un array de imports en el declaratios coloquemoslo, deshabilitemos los componentes antes mencionados del app.module.ts y probemos debería funcionar pero aun no del todo.

El decir creemos un modulo pararte para el **frontstructure**:

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

*import* { HeaderComponent } *from* './header/header.component';

*import* { SidebarComponent } *from* './sidebar/sidebar.component';

*import* { BreadcrumbsComponent } *from* './breadcrumbs/breadcrumbs.component';

@NgModule({

declarations:[

HeaderComponent,

SidebarComponent,

BreadcrumbsComponent

],

exports:[

HeaderComponent,

SidebarComponent,

BreadcrumbsComponent

]

})

*export* class Frontstructure{}

y lo importaremos en el pages.module.ts

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

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

*import* { Frontstructure } *from* '../frontstructure/frontstructure.module';

*import* { PagesComponent } *from* './pages.component';

@NgModule({

declarations:[

DashboardComponent,

PagesComponent

],

exports:[

DashboardComponent

],

imports:[

FrontstructureModule

]

})

*export* class PageModule{}

el app.module.ts debería quedar asi

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

*import* { APP\_ROUTES } *from* './app.routes';

*import* { PageModule } *from* './components/pages/pages.module';

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

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

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

*import* { PagenofoundComponent } *from* './components/pagenofound/pagenofound.component';

@NgModule({

declarations: [

AppComponent,

LoginComponent,

PagenofoundComponent,

],

imports: [

BrowserModule,

APP\_ROUTES,

PageModule

],

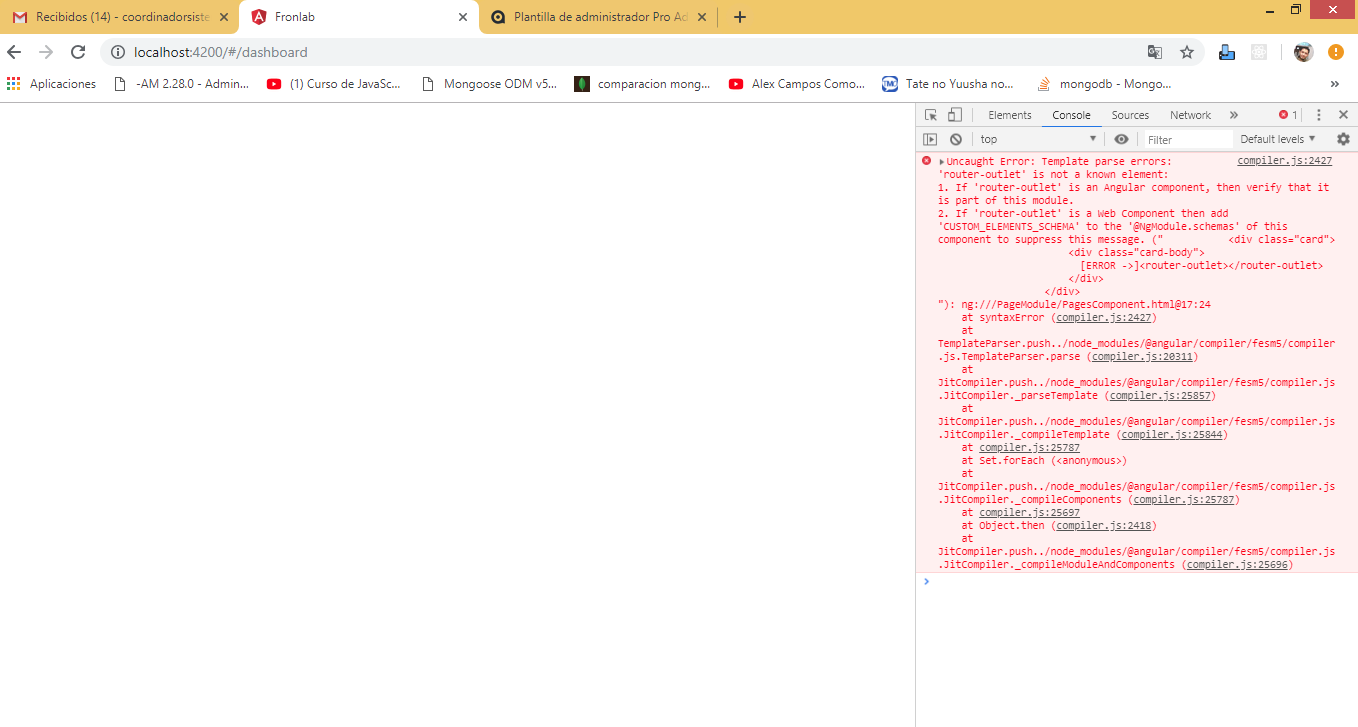
providers: [],

bootstrap: [AppComponent]

})

*export* class AppModule { }

por ultimo en el pages.component.html deberemos comentar el router-outlet por el momento al ejecutar y mirar el navegador deberemos tener este resultado hasta ahora. Observaremos que el ruteo no se ejecuta por la falta del router-outlet pero si lo ponemos nos arrojara un error. Este error ocurrio porque se desactivaron las rutas originales



Arreglemos el problema de las rutas que se desactivaron, para mejor organización de las rutas crearemos rutas por carpeta contenedora en este caso crearemos un routing para **pages**.

1:En la carpeta **pages** creare un nuevo archivo llamado pages.routes.ts, este archivo es casi igual al archivo de rutas padre.

2: en pages.routes.ts cortaremos todas las rutas que le conciernen del app.routing.ts es decir de los componentes que le corresponden:

el pages.routes.ts quedara de la siguiente manera, como son rutas hijas en la exportación al usar el RouterModule en ves de forRoot usaremos forChild.

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

*import* { PagesComponent } *from* './pages.component';

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

const pagesRoutes : Routes = [

{

path: '',

component : PagesComponent,

children:[

{path:'dashboard', component:DashboardComponent}

]

}

]

*export* const PAGES\_ROUTES = RouterModule.forChild(pagesRoutes)

ahora debemos importar el PAGES\_ROUTES en el modulo de pages es decir en pages.module.ts ya que harán parte del pages.module.ts y las rutas deben ser añadidas al array de imports.

Importamos

*import* { PAGES\_ROUTES } *from* './pages.routes';

añadir a los imports

imports:[

Frontstructure,

PAGES\_ROUTES

]

Ahora en el app.router.ts solo dejare lo que le corresponde por el momento

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

*import* { DashboardComponent } *from* './components/pages/dashboard/dashboard.component';

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

*import* { PagenofoundComponent } *from* './components/pagenofound/pagenofound.component';

*import* { PagesComponent } *from* './components/pages/pages.component';

const appRoutes : Routes = [

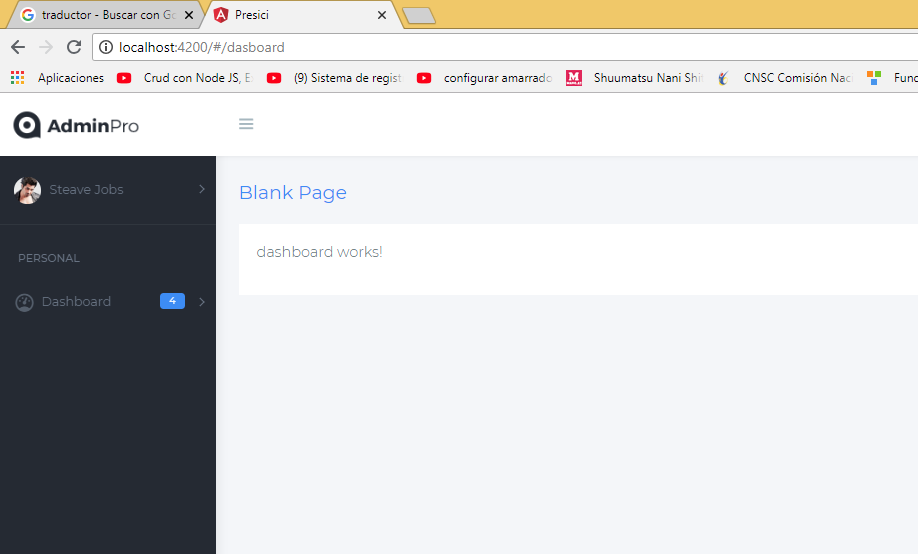
{path:'login', component: LoginComponent},

{path:'\*\*', component: PagenofoundComponent},

];

*export* const APP\_ROUTES = RouterModule.forRoot(appRoutes, {useHash:true});

Ahora volveré a activar el router-outlet. Del pages.component.html. y todo estará funcionando nuevamente



**Ahora modularicemos los servicios que usaremos mas adelante para poder usarlos en cualquier modulo y componente que quiera.**

**1-** creemos un modulo de servicios con el siguiente comando

**ng g m services/service - -spec=false - -flat**



**2-** Dentro creare un servicio llamado frontstructure para mis components de header, breacrumbs, sidebar.

Empecemos con el sidebar para administrar este menú.

Utilizando el siguiente comando podemos crear nuestro servicio en su propia carpeta dentro de servicios

Ng g s services/sidebar/sidebar - -spec=false

Ahora en la carpera services creare un archive llamado services.index.ts donde convergeran todos los servicios creados.

Importaremos servicio de sidebar

*import* { SidebarService } *from* "./sidebar/sidebar.service";

ahora cambiaremos el import por export para que se pueda exportar

*export* { SidebarService } *from* "./sidebar/sidebar.service";

ahora lo agregaremos al service.module.ts que administrara todo, los servicios los agregaremos a los prividers, quedaría asi.

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

*import* { CommonModule } *from* '@angular/common';

*import* { SidebarService } *from* './sidebar/sidebar.service';

@NgModule({

declarations: [],

imports: [

CommonModule

],

providers:[

SidebarService

]

})

*export* class ServicesModule { }

y también lo importaremos al app.module.ts para poder usarlo en modulos externos, como es un modulo va en los imports

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

*import* { APP\_ROUTES } *from* './app.routes';

*import* { PageModule } *from* './components/pages/pages.module';

*import* { ServicesModule } *from* './services/services.module';

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

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

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

*import* { PagenofoundComponent } *from* './components/pagenofound/pagenofound.component';

@NgModule({

declarations: [

AppComponent,

LoginComponent,

PagenofoundComponent,

],

imports: [

BrowserModule,

APP\_ROUTES,

PageModule,

ServicesModule

],

providers: [],

bootstrap: [AppComponent]

})

*export* class AppModule { }

**Creemos servicios para administrar el sidebar, vayamos al sidebarservices.ts**

Por el momento vayamos al servicio sidebar.service.ts y creemos un objeto menú de tipo any que va a ser igual a un arreglo de objetos, con las siguientes propiedades.

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

@Injectable({

providedIn: 'root'

})

*export* class SidebarService {

menu : any = [

{

titulo: "Equipos",

icono: "mdi mdi-gauge",

submenu:[

{ titulo:"Patrones", url:"/dashboard" },

{ titulo:"Equipos", url:"/reception" },

{ titulo:"Grafica1", url:"/graficas1" },

{ titulo:"Grafica2", url:"/graficas2" }

]

}

];

constructor() { }

}

Ahora vamos al componente del sidebar para implementar el objeto menú. Es decir abramos el archivo sidebar.component.ts. en este archivo deberemos importar el servicio sidebar.service.ts e inicializarla en el constructor.

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

*import* { SidebarService } *from* 'src/app/services/services.index';

@Component({

selector: 'app-sidebar',

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

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

})

*export* class SidebarComponent implements OnInit {

constructor( public \_sidebar:SidebarService) { }

ngOnInit() {

}

}

Ahora para poder hacer ruteo necesitare importar en el frontstructure.module.ts el modulo de Router que se llama RouterModule. y agreguémoslo a los imports del @NgModule

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

*import* { HeaderComponent } *from* './header/header.component';

*import* { SidebarComponent } *from* './sidebar/sidebar.component';

*import* { BreadcrumbsComponent } *from* './breadcrumbs/breadcrumbs.component';

*import* { RouterModule } *from* '@angular/router';

@NgModule({

imports:[

RouterModule

],

declarations:[

HeaderComponent,

SidebarComponent,

BreadcrumbsComponent

],

exports:[

HeaderComponent,

SidebarComponent,

BreadcrumbsComponent

]

})

*export* class Frontstructure{}

Ahora en la vista del sidebar.component.html usaremos el router lo usaremos despues.

El routerLink envía la ruta y el routerLinkActive activa la opción.

Ahora si consentremonos en el menú, utilicemos la directiva \*ngFor para tomar el objeto del menú que me trae el servicio.

Sidebar.component.html

*<!-- ============================================================== -->*

<aside *class*="left-sidebar">

*<!-- Sidebar scroll-->*

<div *class*="scroll-sidebar">

*<!-- Sidebar navigation-->*

<nav *class*="sidebar-nav">

<ul *id*="sidebarnav">

<li *class*="user-profile">

<a *class*="has-arrow waves-effect waves-dark " *href*="#" *aria-expanded*="false"><img *src*="../assets/images/users/profile.png" *alt*="user" /><span *class*="hide-menu">Steave Jobs </span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="javascript:void()">My Profile </a></li>

<li><a *href*="javascript:void()">My Balance</a></li>

<li><a *href*="javascript:void()">Inbox</a></li>

<li><a *href*="javascript:void()">Account Setting</a></li>

<li><a *href*="javascript:void()">Logout</a></li>

</ul>

</li>

<li *class*="nav-devider"></li>

<li *class*="nav-small-cap">App</li>

<li *\*ngFor*="let menu of \_sidebar.menu"> <a *class*="has-arrow waves-effect waves-dark" *href*="#" *aria-expanded*="false"><i *class*="mdi mdi-gauge"></i><span *class*="hide-menu">Dashboard <span *class*="label label-rouded label-themecolor pull-right">4</span></span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="index.html">Minimal </a></li>

<li><a *href*="index2.html">Analytical</a></li>

<li><a *href*="index3.html">Demographical</a></li>

<li><a *href*="index4.html">Modern</a></li>

</ul>

</li>

</ul>

</nav>

*<!-- End Sidebar navigation -->*

</div>

*<!-- End Sidebar scroll-->*

</aside>

Esto nos dara un error sino importamos el CommonModule en el frontstructure.module.ts, dicho modulo nos activa las directivas angulares.

import { CommonModule } from '@angular/common';

ahora tambien agreguemos a los imports

@NgModule({

imports:[

RouterModule,

CommonModule

],

Ahora si de regreso al html es decor el sidebar.component.html

Quitemos el href del menú principal para que no redireccione, cambiemos su icono usando la implementación de las clases de angular. Y el titulo también cambiémoslo utilizando la interpolación de las {{}}, el 4 estatico que tiene cambiémoslo por el numero de sub elementos del objeto menú {{menú.submenu.length}}

Ahora el hay que traer las opciones del submenú y ese también lo haremos desde un ngForm.

Los RouterLinks como son de manera dinámica y como puede que lleven algún parámetro serán declarados de la siguiente forma [routerLink]=”[submenú.url]”

Sidebar.component.html

*<!-- ============================================================== -->*

<aside *class*="left-sidebar">

*<!-- Sidebar scroll-->*

<div *class*="scroll-sidebar">

*<!-- Sidebar navigation-->*

<nav *class*="sidebar-nav">

<ul *id*="sidebarnav">

<li *class*="user-profile">

<a *class*="has-arrow waves-effect waves-dark " *href*="#" *aria-expanded*="false"><img *src*="../assets/images/users/profile.png" *alt*="user" /><span *class*="hide-menu">Steave Jobs </span></a>

<ul *aria-expanded*="false" *class*="collapse">

<li><a *href*="javascript:void()">My Profile </a></li>

<li><a *href*="javascript:void()">My Balance</a></li>

<li><a *href*="javascript:void()">Inbox</a></li>

<li><a *href*="javascript:void()">Account Setting</a></li>

<li><a *href*="javascript:void()">Logout</a></li>

</ul>

</li>

<li *class*="nav-devider"></li>

<li *class*="nav-small-cap">App</li>

<li *\*ngFor*="let menu of \_sidebar.menu">

<a *class*="has-arrow waves-effect waves-dark" *aria-expanded*="false">

<i *class*="mdi mdi-gauge"></i>

<span *class*="hide-menu">{{menu.titulo}}

<span *class*="label label-rouded label-themecolor pull-right">{{menu.submenu.length}}</span>

</span>

</a>

<ul *aria-expanded*="false" *class*="collapse">

<li *\*ngFor*="let submenu of menu.submenu"><a *[routerLink]*="[submenu.url]">{{submenu.titulo}}</a></li>

</ul>

</li>

</ul>

</nav>

*<!-- End Sidebar navigation -->*

</div>

*<!-- End Sidebar scroll-->*

</aside>

Ahora creemos los demas components que usaremos por ahora para completer el menú y sus respectivas rutas funcionales

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

@Injectable({

providedIn: 'root'

})

*export* class SidebarService {

menu : any = [

{

titulo: "Device",

icono: "mdi mdi-calculator",

submenu:[

{ titulo:"Dasboard", url:"/dashboard" },

{ titulo:"Equipos", url:"/equipos" },

{ titulo:"Patrones", url:"/patrones" }

]

},

{

titulo: "Operation",

icono: "mdi mdi-owl",

submenu:[

{ titulo:"Leyend", url:"/leyendOperation" },

{ titulo:"Reception", url:"/reception" },

{ titulo:"Order", url:"/order" },

{ titulo:"Delivery", url:"/delivery" },

{ titulo:"Certificate", url:"/certificate" },

]

},

];

constructor() { }

}

**Implementando el Breadcrumbs con observables y promesas**

Lo primero que hay que hacer es ir a pages.routes.ts y agregar un atributo data a todas las rutas con su respectivi significado, eto lo va a usar el breadcrumbs para poder ser dinamico.

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

*import* { PagesComponent } *from* './pages.component';

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

*import* { EquipoComponent } *from* './equipo/equipo.component';

*import* { PatronComponent } *from* './patron/patron.component';

*import* { LeyendOperationComponent } *from* './leyend-operation/leyend-operation.component';

*import* { ReceptionComponent } *from* './reception/reception.component';

*import* { OrderComponent } *from* './order/order.component';

*import* { DeliveryComponent } *from* './delivery/delivery.component';

*import* { CertificateComponent } *from* './certificate/certificate.component';

const pagesRoutes : Routes = [

{

path: '',

component : PagesComponent,

children:[

{path:'dashboard', component:DashboardComponent, data :{title:'Dashboard'}},

{path:'equipos', component:EquipoComponent, data :{title:'Machine'}},

{path:'patrones', component:PatronComponent, data :{title:'Patter'}},

{path:'leyendOperation', component:LeyendOperationComponent, data :{title:'Leyend'}},

{path:'reception', component:ReceptionComponent, data :{title:'Reception'}},

{path:'order', component:OrderComponent, data :{title:'Order'}},

{path:'delivery', component:DeliveryComponent, data :{title:'Delivery'}},

{path:'certificate', component:CertificateComponent, data :{title:'Certificate'}},

]

}

]

*export* const PAGES\_ROUTES = RouterModule.forChild(pagesRoutes)

como podemos obtrener ese title dsesde el pages.routes.ts desde el breadcrumbs.component.ts?

1- Importamos el modulo de Router

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

2-inicializar el servicio en el constructor, el router tiene algo que se llama events, el router events es un observable desde aquí podemos acceder a una propiedad que tiene la información que necesito

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

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

*import* { filter, map } *from* 'rxjs/operators';

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

@Component({

selector: 'app-breadcrumbs',

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

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

})

*export* class BreadcrumbsComponent implements OnInit {

public title:string;

constructor( private \_router:Router,

private \_title: Title) {

*this*.getDataRoute()

.subscribe(event=>{

*//console.log(event);*

*this*.title=event.title;

*this*.\_title.setTitle(*this*.title);

});

}

ngOnInit() {

}

getDataRoute(){

*return* *this*.\_router.events.pipe(

filter(evento=> evento instanceof ActivationEnd),

filter( (evento: ActivationEnd) => evento.snapshot.firstChild === null),

map((evento: ActivationEnd) => evento.snapshot.data)

)

}

}

**Uso de scripts de Archivos importados en el index.html en typescript:**

Hagamos que desde la vista login al dar en el botón ingresar nos lleve al dashboard. Debemos ir al login.component.html.

En el form borraremos el id y el action y declaremos un evento submit que ejecute un metodo en el componente llamado ingresar.

*<*form class="form-horizontal form-material" (ngSubmit)="ingresar()"*>*

Ahora creemos ese metodo en el login.component.ts, este metodo hara uso del routerlink para que me lleve al dahsboard.

Debemos importar el servicio Router e inicializarlo en el constructor.

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

constructor(public router: Router) { }

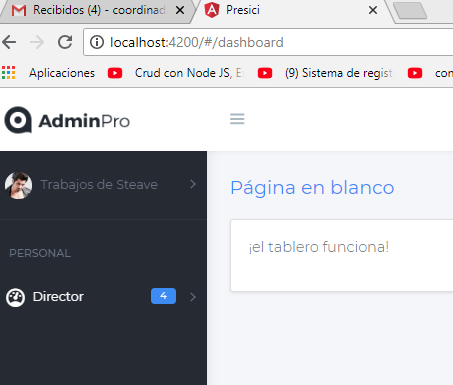
y ahora utilicemos el metodo navigate en el metodo login para que se de la redirección al dashboard:

login(){

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

}

Aquí esto nos redireccionara al dashboard pero existe un problema que las opciones del menú no funcionan, ya no se abren.



Y esto se debe a que el template esta utilizando un plugin para el menú, este plugin comprime ítems y las opciones y demás. Pero este plugin ya no funciona.

Y no funciona porque cuando el plugin es llamado en el index.html el plugin se llama

*<!--Menu sidebar -->*

*<*script src="assets/js/sidebarmenu.js"*></*script*>*

Enconces cuando pasaaos del login al dashboard el menú en el login no existe por consecuencia no se dispara la inicialización y cuando llegamso al dashboard los menus no funcionan. Avbramos este archivo pero como esta minimizado va a ser difícil de leer, asi que como tenemos el normal enlacemoslo a ese es decir de .min pasemos solo a .js como se muestra a continuación.

abramos ese archivo custon.js, ojo este solo es un segmento para ilustrar, en este caso mostremos la inicialziacion del sidebar

var set = function() {

var width = (window.innerWidth > 0) ? window.innerWidth : this.screen.width;

var topOffset = 0;

if (width < 1170) {

$("body").addClass("mini-sidebar");

$('.navbar-brand span').hide();

$(".sidebartoggler i").addClass("ti-menu");

Como podemos llamar a todos los plugins estos de jquery fuera de angular desde algún componente de angular que me la active?

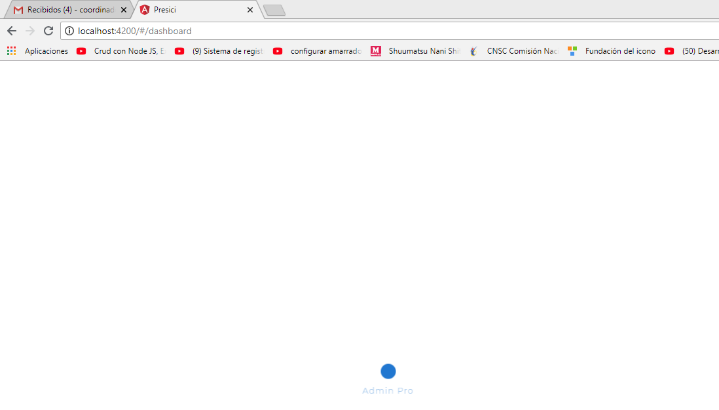
Lo primero es que en el plugin custon.js declararemos una función al inicio llamada **init\_plugins()** que me enciere toda la función principal de jquery es decir que encapsule todo el código del archivo. Para ser mas claro la función **init\_plugins()** tendrá dentro todo el código del archivo custon.js algo como esto.

**function init\_plugins(){**

**todo el código del archivo**

**}**

Cuando hagamos esto y guardemos nos daremos cuenta que se quedara cargando siempre es decir aparecerá el icono de cargue y no se quitara.



Como resolvemos esto para que cargue todo y por ende se quite el prelouder y carguen los componentes para los menus?

Para eso iremos al login.component.ts y llmaremos a la función que creamos en el ngOnInit

ngOnInit() {

init\_plugins();

}

Como yo se que existe lo que hare sera declararla al antes de la clase del componente de la siguiente manera:

declare function init\_plugins();

si yo hago esto cuando le de al botón ingresar del login me va a redirigir al dashboar, pero si yo recargo en el dashboard nuevamente se quedara cargando. Entonces resumanos

hasta ahora este es el componente login:

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

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

*import* { NgForm } *from* '@angular/forms';

declare function init\_plugins();

@Component({

selector: 'app-login',

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

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

})

*export* class LoginComponent implements OnInit {

constructor(public \_router : Router) { }

ngOnInit() {

init\_plugins();

}

login(){

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

}

}

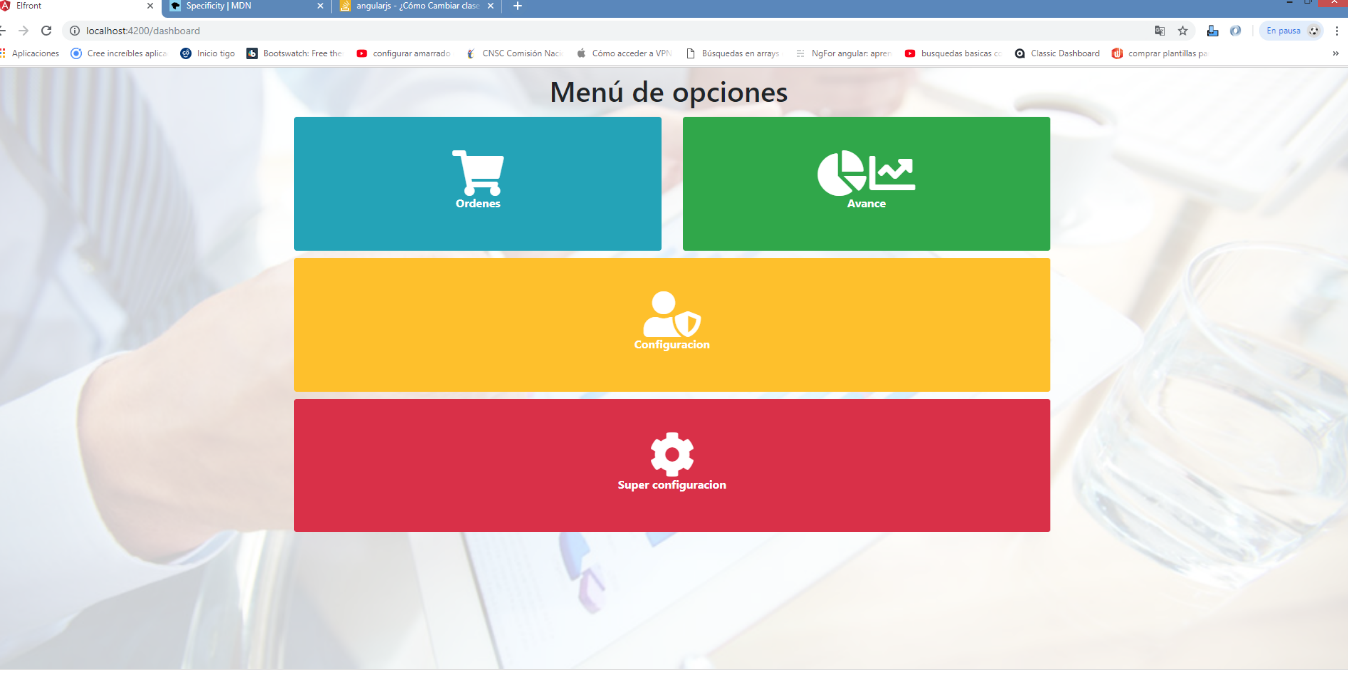
Para resolver el problema actual deberemos ir componente que embuelve todos las componentes y paginas? Pages.component.ts

Deberemos ir alla y hacer lo mismo que hicimos con el loginComponent.ts.

Con esto resolvemos el problema. Los plugins se cargaran tanto en el login como al inicializar el pages.component.ts. De esta manera podremos llamar cualquier archivo que se encuentre fuera de angular en un archivo de javascript, esto funciona mucho con carruseles, tooltips y otras cosas, que están externas que están hechas en jquey y que no cargan directamente en angular o typescrip.

**Creacion de los modelos en el frontend**

aasa



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 password?:string,

public usuario\_creador? : string,

public usuario\_modificador? : string,

public image? : string,

public role?: string,

public estado? : string,

public tercero? : string,

public \_id? : string

){}

}

otro

*export* class UbicacionModel{

constructor(

public nombre? : string,

public codigo? : string,

public usuario\_creador?:string,

public usuario\_modificador?:string,

public fecha\_creacion?:string,

public \_id? : string

){}

}

otro

*export* class TereceroModel{

constructor(

nit?: string,

nombre?:string,

direccion?:string,

telefono?:string,

contacto?:string,

usuario\_creador?:string,

\_id?:string

){}

}

otro

*export* class RecepcionModel{

constructor(

public fecha?:string,

public tipo\_servicio?:string,

public no\_recepcion?:string,

public solicitante?:string,

public tercero?:string,

public tag?:string,

public serial?:string,

public intervalo\_medicion?:string,

public tipo\_medicion?:string,

public tipo\_indicacion?:string,

public intervalo\_calibracion?:string,

public apto?: string,

public estado?:string,

public archivo?:string

){}

}

otro

*export* class OrdenModel{

constructor(

public oti? : number,

public fecha\_solicitud? : string,

public fecha\_requerida? : string,

public fecha\_ejecucion? : string,

public oti\_cliente? : string,

public centro\_costo? : string,

public ubicacion? : string,

public solicitante? : string,

public tercero? : string,

public usuario\_creador? : string,

public usuario\_modificador?: string,

public items? : Item,

public estado?: string,

public \_id?: string

){}

}

*export* class Item{

constructor(

public tag? : string,

public nombre? : string,

public marca? : string,

public modelo? : string,

public serial? : string,

public tipo\_equipo?: string,

public fabricante? : string,

public ubicacion? : string,

public tipo\_indicacion?: string,

public intervalo\_indicacion?: string,

public resolucion?: string,

public anexos?: string,

public img?: string,

){}

}

otro

*export* class EquipoModel{

constructor(

public tag? : string,

public nombre? : string,

public marca? : string,

public modelo? : string,

public serial? : string,

public tipo\_equipo?: string,

public fabricante? : string,

public ubicacion? : string,

public estado?: string,

public tipo\_indicacion?: string,

public intervalo\_indicacion?: string,

public resolucion?: string,

public anexos?: string,

public tercero?: string,

public usuario\_creador?:string,

public usuario\_modificador?: string,

public img?: string,

public \_id?: string

){}

}

otro

*export* class ConsecutivoModel{

constructor(

public prefijo?: string,

public consecutivo?: string,

public tercero?: string,

public usuario\_creador?: string,

public \_id?: string

){}

}

**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

) { }

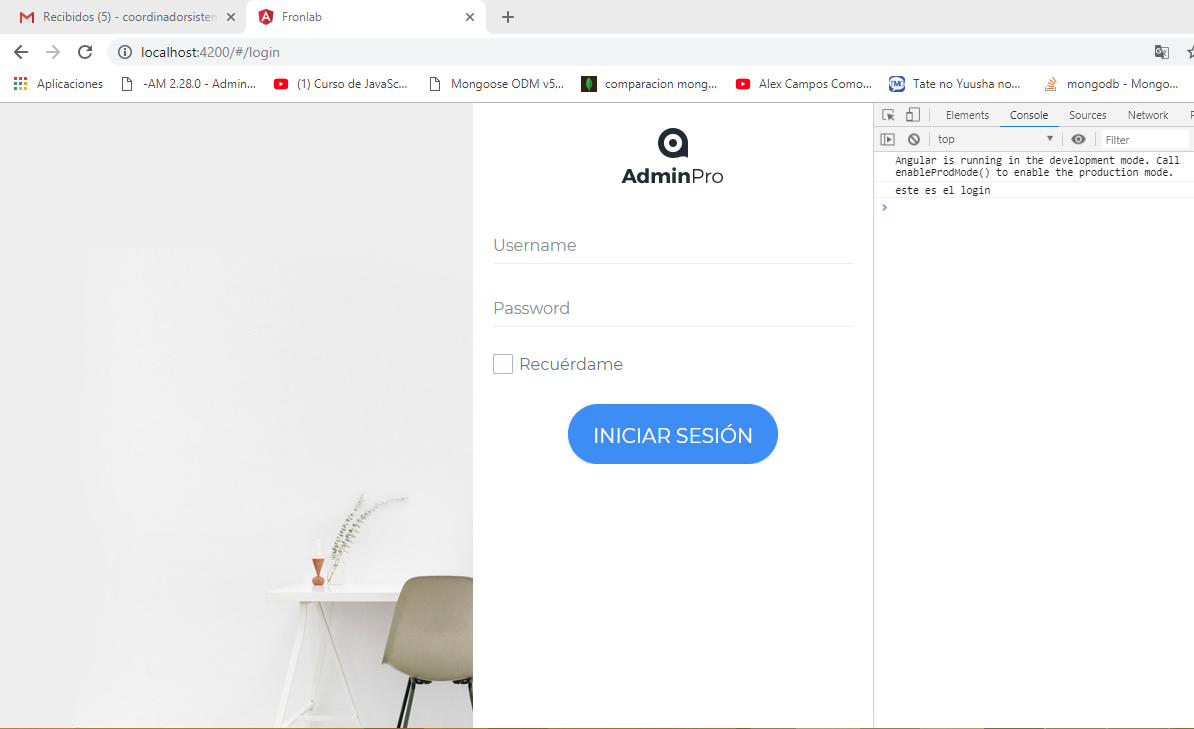
ngOnInit() {

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

*import* { NgForm } *from* '@angular/forms';

*import* { UserService } *from* 'src/app/services/services.index';

*import* { UserModel } *from* 'src/app/models/userModel';

declare function init\_plugins();

@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(

public \_router : Router,

public \_userServices :UserService

) {

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

}

ngOnInit() {

init\_plugins();

}

*//login del usuario*

login(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"

}

)

}

}

*<!-- ============================================================== -->*

*<!-- Main wrapper - style you can find in pages.scss -->*

*<!-- ============================================================== -->*

<section *id*="wrapper" *class*="login-register login-sidebar" *style*="background-image:url(../assets/images/background/login-register.jpg);">

<div *class*="login-box card">

<div *class*="card-body">

<form *#loginForm*="ngForm" *class*="form-horizontal form-material" *(ngSubmit)*="login(loginForm)">

<a *class*="text-center db"><img *src*="../assets/images/logo-icon.png" *alt*="Home" /><br/><img *src*="../assets/images/logo-text.png" *alt*="Home" /></a>

<div *class*="form-group m-t-40">

<div *class*="col-xs-12">

<input *class*="form-control" *placeholder*="Username" *type*="email"*name*="correo" *#correo*="ngModel" *[(ngModel)]*="user.correo" *required* *pattern*="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$">

</div>

</div>

<div *class*="form-group">

<div *class*="col-xs-12">

<input *class*="form-control" *placeholder*="Password" *type*="password" *name*="password" *#password*="ngModel" *[(ngModel)]*="user.password" *required*>

</div>

</div>

<div *class*="form-group text-center m-t-20">

<div *class*="col-xs-12">

<button *class*="btn btn-info btn-lg text-uppercase btn-rounded" *type*="submit" >Log In</button>

</div>

</div>

</form>

</div>

</div>

</section>

*<!-- ============================================================== -->*

*<!-- End Wrapper -->*

*<!-- ============================================================== -->*

Ahora regresamos 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 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**(es 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 **pages.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']);

}

}

Construyamos el componente del dashboard: