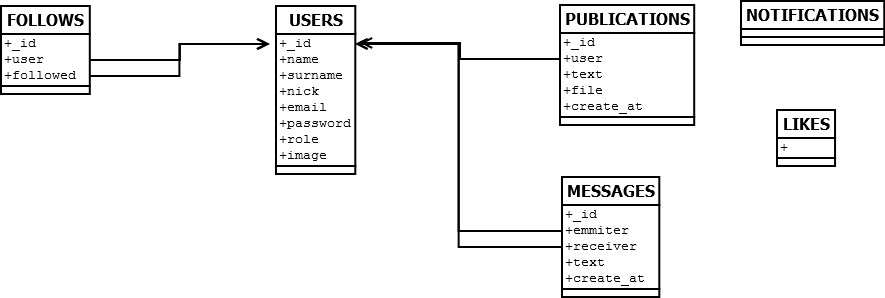
**PROYECTO MEAN 5**

**Schema de base de Datos:**



1-creamos un proyecto en nodejs con **npm init** damos nombre al proyecto y seguimos los pasos normales.

2-creamos un repositorio en git y lo enlazamos con el comando

git remote add origin <https://github.com/jesusdario10/Redsocial.git>

3-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 moment --save :** librería para trabajar las fechas, horas y todo lo que tenga que ver con tiempo.
8. **Npm install nodemon --save-dev** : librería que refresca el servidor cada que guardamos para no estar parando y arrancando el servidor.

Nuestro package.json debería tener esta forma.

{

"name": "backendred",

"version": "1.0.0",

"description": "backend de una red social para practicar los conocimientos",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "jdmp",

"license": "MIT",

"dependencies": {

"bcrypt-nodejs": "0.0.3",

"body-parser": "^1.18.3",

"connect-multiparty": "^2.2.0",

"express": "^4.16.4",

"jwt-simple": "^0.5.5",

"moment": "^2.22.2",

"mongoose": "^5.3.7"

},

"devDependencies": {

"nodemon": "^1.18.5"

}

}

**Creemos la base de datos:**

Usaremos robo3T

La base se llamara RedS

1-creemos una colleccion llamada users

2-insertemos un documento:

/\* 1 \*/

{

"\_id" : ObjectId("5bd75ca10629c34bdb9a15f9"),

"name" : "admin",

"surname" : "admin",

"nick" : "admin",

"email" : "admin@admin.com",

"password" : "123456",

"image" : null

}

**3-conectemos la api a mongodb:**

1. **Creamos el archivo index.js en la raiz**
2. **Conectar mongodb usando promesas:**

'use strict'

var mongoose = require('mongoose');

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost:27017/RedS', {useMongoClient: true})

.then(()=>{

console.log("conectado a la db RedS");

})

.catch(err=>{

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

})

1. **Configurar el comando npm start con nodemon en el package.json:**

"scripts": {

"start":"nodemon index.js",

"test": "echo \"Error: no test specified\" && exit 1"

},

**Creemos el servidor web:**

**1-creamos el fichero app.js:** aq1ui configuraremos todo lo que tiene que ver con express es decir body-parser, rutas etc y luego este fichero lo importaremos en el index.js donde culminaremos la creación final del servidor web.

1. Lo primero que haremos sera cargar express, body parser e instanciar express:

'use strict'

var express = require('express');

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

var app = express();

1. Cargaremos los middlewares del body parser que sonmetodos que se ejecutan antes de que los controladores se ejecuten. Estos middleares del body parser son necesarios para la ejecución del bodyparser, y luego hacemos que convierta lo que nos llegue en el body a un objeto json, esto ocurrirá cada que hagamos peticiones

*//cargar middlewares*

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

app.use(bodyParse.json());

1. Ahora debemos exportar la configuración: ya que lo podremos importar en otros ficheros en este caso en el index.js

module.exports = app;

1. Carguemos el app.js en el index.js

'use strict'

var mongoose = require('mongoose');

var app = require('./app')

1. Creemos también una variable port donde le indicaremos al puerto donde vamos a trabajar:

var port = 3800;

1. Usemos la variable app dentro del then para arrancar el servidor de modo que el archivo index.js quedaría de la siguiente manera.

'use strict'

var mongoose = require('mongoose');

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

var port = 3800;

*//Conexion a la base de datos*

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost:27017/RedS', {useMongoClient: true})

.then(()=>{

console.log("conectado a la db RedS");

*//Crear servidor*

app.listen(port, ()=>{

console.log("Servidor NodeJS corriendo in port 3800");

})*//-fin de la creacion del servidor*

})

.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:**

*//cargar rutas*

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

res.status(200).json({

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

})

})

Efectivamente en el navegador podemos observar como funciona

<http://localhost:3800/>

{"mensaje": "Acción de pruebas en el servidor de NodeJs"}

**El cliente ResFull**

Postman nos permitirá realizar peticiones http de cualquier tipo y manipularla a nuestro antojo para hacer pruebas ojo los middlewares siempre van primero que las rutas:

El archivo app.js quedaría de esta manera por ahora:

'use strict'

var express = require('express');

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

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'

})

})

*//cors y cabeceras*

*//rescribir rutas*

module.exports = app;

**creemos los modelos:**

1-creemos una carpeta llamada model y dentro creemos el modelo de usuario llamado userModel.js.

2-se inicia con ‘use strict’

3-cargamos el modulo de mongoose

var mongoose = require('mongoose');

4-crearemos una variable llamada Schema que nos permitirá definir nuevos Schemas.

var Schema = mongoose.Schema;

5-ahora creemos la variable UserSchema que sera la entidad que reutilizaremos cada que quedamos crear un usuario:

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var UserSchema = Schema({

name: String,

surname : String,

nick : String,

email: String,

password : String,

role : String,

image : String

})

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

**6-**creemos el modelo de publication que se llamara publicationModel.js

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var PublicationSchema = Schema({

text: String,

file: String,

create\_at:String,

user : { type: Schema.ObjectId, ref: 'User'}

})

module.exports = mongoose.model('Publication', PublicationSchema);

**7-**vamos con el modelo de followers que se llamara followModel.js

'use strict'

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var FollowSchema = Schema({

user:{type : Schema.ObjectId, ref : 'User'},//es el user que sigue

followed:{type : Schema.ObjectId, ref : 'User'}//es el user seguido

})

module.exports = mongoose.model('Follow', FollowSchema);