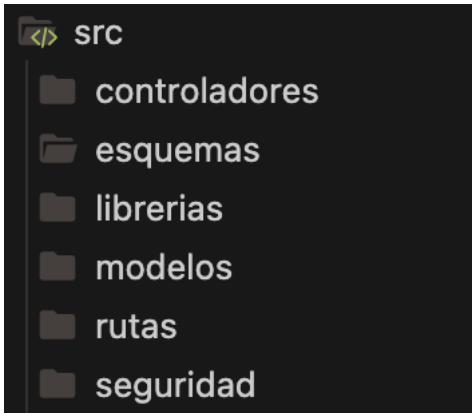
Configurando MongoDB Atlas – mongoose – express – morgan – bycryptjs - jsonwebtoken.

En este documento vamos a configurar una pequeña aplicación que nos permite crear un pequeño desarrollo de Autenticación en el que vamos a realizar funciones CRUD sobre una base de datos NoSQL. La conexión sobre la base de datos no relacional y en la nube MongoDb Atlas, la vamos a hacer desde código de javascript.

Trabajamos con un frontend, nuestro cliente realizado con HTML, javascript, Bootstrap y Electrón, y nuestro backend que sería nuestro servidor con el módulo express de la librería npm.

Crearemos un Token entre nuestro frontend y nuestro backend con jwt, que nos permite subir el nivel de seguridad en la comunicación entre los dos y utilizando una cookie para poder trabajar con el.

Para trabajar sobre todos estos conceptos, vamos a cargar un nuevo workspace que vamos a denominar “mongodb” y vamos a generar el siguiente árbol de carpetas para ir ordenando nuestros ficheros .js de forma clara y adecuada como buena práctica dentro de la profesión de programador.



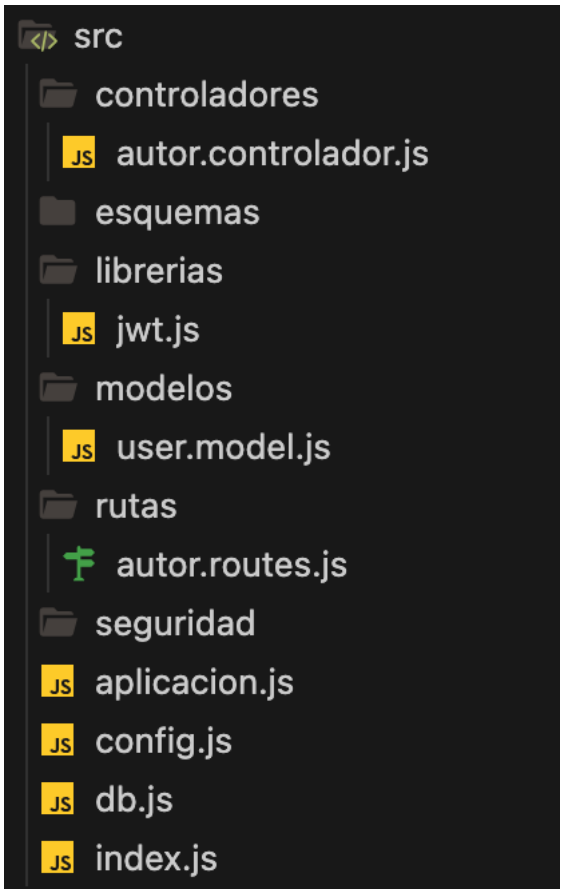
Estas carpetas las vamos a utilizar para:

- **rutas:** Vamos a crear todas las rutas que el frontend puede solicitar. Esto es interesante porque gracias a esto podemos jugar con información privada o pública.
- **controladores:** funciones que se van a ejecutar cuando se visita a una url determinada desde el frotend y que está especificada dentro del fichero .js generado en “rutas”.
- **esquemas:** Esquemas para validar los datos antes de que lleguen al backend.

IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891

- **librerias:** Escribir funciones que vamos a reutilizar desde diferentes puntos de nuestro desarrollo.
- **modelos:** En esta carpeta vamos a guardar nuestros .js en los que vamos a definir los modelos de datos que vamos a utilizar en nuestros json.
- **seguridad:** Especificamos que rutas están protegidos por los usuarios autenticados. Información pública o información privada.

Dentro de las diferentes carpetas de proyecto, he creado los siguientes ficheros .js y es aconsejable que los alumnos utilicen los mismos para que no se pierdan en la explicación.



El contenido de estos ficheros se irá comentando a lo largo de este documento.

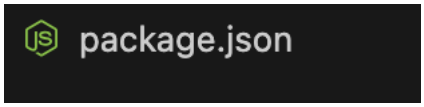
Vamos a configurar nuestro backend y nuestra conexión con el servidor “MongoDB Atlas”.

Lo primero que vamos a hacer es cargar nuestra librería npm y generar nuestro fichero .json y poder configurarlo como hacíamos en el caso de “electrón”.

Para generarlo vamos a lanzar desde el terminal de nuestro visual Code la siguiente instrucción:

IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891

npm init -y



Vamos a cargar el primer módulo, el de “nodemon” que nos va a permitir que el servidor se nos vaya actualizando con los diferentes cambios que vamos realizando desde visual code. Es similar a la extensión de “Live Server”.

Para cargar este módulo, lo podemos hacer de dos maneras:

npm install nodemon -D

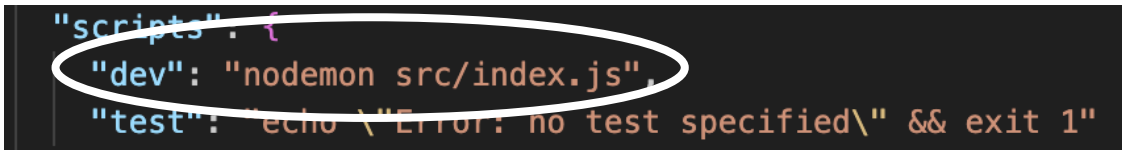
npm i nodemon -D

Dentro del fichero “package.json”, vamos a encontrar la siguiente información:

```
{
  "name": "mongodb",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  ▷ Depurar
  "scripts": {
    "dev": "nodemon src/index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^7.6.0",
    "morgan": "^1.10.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.1"
  }
}
```


IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891

Al igual que hicimos con “electrón”, vamos a agregar la siguiente línea asociada al parámetro “dev” para arrancar nuestro backend y que se esté actualizando siempre que realicemos algún cambio.



Para que no nos de ningún problema el exportar diferentes variables, debemos introducir otra línea en el .json

El fichero “index.html” se encuentra dentro de nuestro árbol de carpetas y ficheros y es desde el que vamos a lanzar todos los .js para que funcione nuestro pequeño proyecto.

El siguiente paso es configurar nuestro pequeño servidor. Para ello vamos a cargar el módulo de “express”.

npm install express

ó

npm i express

configurar nuestro pequeño servidor es simple, en principio. Si añadimos las siguientes líneas dentro del fichero “aplicacion.js”

```
import express from 'express';
const aplicacion = new express(); //instanciamos.
aplicacion.listen(3000); //Le decimos a nuestro pequeño servidor que se quede escuchando en el puerto 3000.
console.log("Servidor abierto en puerto 3000"); //Ponemos un mensaje de comprobación.
```

Con estas cuatro líneas, ponemos en funcionamiento este servidor. Lo podemos ver en funcionamiento si ejecutamos desde el terminal de visual Code la orden

node src/aplicacion.js

IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891

y abrimos nuestro navegador en el link:

localhost:3000

veremos que se nos conecta con el servidor, pero nos aparece un mensaje en el que se nos informa de que no podemos coger nada.

Para controlar las conexiones con nuestro servidor, vamos a instalar el siguiente módulo:

npm i morgan

Para gestionar nuestra conexión a mongodb, tenemos que instalar el siguiente módulo:

npm i mongoose

Una vez instalados estos módulos vamos a terminar de configurar nuestro fichero "aplicacion.js" en relación con "index.js", siempre teniendo en cuenta que, tal cual hemos configurado nuestro .json, queremos arrancar nuestra aplicación desde el comando

npm run dev

El fichero "aplicacion.js", quedaría de la siguiente forma:

```
import express from 'express'; //importamos la libreria expres.
import morgan from 'morgan'; //Nos sirve para ver las peticiones que se le
hacen al servidor.
const aplicacion = new express(); //instanciamos.

aplicacion.use(morgan("dev")); //Esta entrada, lo único que nos hace es
mostrarnos un mensaje corto por consola.
aplicacion.use(express.json()); //Para que nuestro pequeño servidor pueda
trabajar con json.

export default aplicacion; //esto nos va a servir para exportar las variables
a otro .js

//aplicacion.listen(3000); //Le decimos a nuestro pequeño servidor que se
quede escuchando en el puerto 3000.
//console.log("Servidor abierto en puerto 3000"); //Ponemos un mensaje de
comprobación.
```

Y nuestro fichero index.js (recordar que es el que hemos configurado en el .json con el atributo dev), quedaría de la siguiente forma:

IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891

```
import aplicacion from "./aplicacion.js";


aplicacion.listen(4000); //Le decimos a nuestro pequeño servidor que se quede escuchando en el puerto 3000.
console.log("Servidor abierto en puerto 4000"); //Ponemos un mensaje de comprobación.
```

Si os fijáis, podemos exportar variables de un fichero .js a otro. Para ello vamos a utilizar la clausula “export”. Fijaros que en el fichero “aplicacion.js” aparece la línea:

```
export default aplicacion; //esto nos va a servir para exportar las variables a otro .js
```

Aparece la cláusula “export” acompañada por “default”. Este “default” interviene en como vamos a importar esa variable en otro fichero .js del proyecto. En nuestra aplicación lo vamos a “importar” como se muestra en la imagen:

```
import aplicacion from "./aplicacion.js";
```

fijaros que en este caso no ponemos el nombre de la variable “aplicacion” entre llaves, porque al exportarle le hemos pasdo el default. Otra cosa importante es la información que le pasamos al from, le pasamos la dirección relativa del .js en el que exportamos la variable. En nuestro caso y siguiendo el árbol de carpetas y archivos indicado en la parte superior de este documento, sería **“./aplicacion.js”**.

Configurando diferentes rutas:

En este punto, podemos empezar a configurar las diferentes rutas a las que podemos acceder desde nuestro frotend en el backend. Esto nos permitirá más adelante a tener información privada y/o pública.

En nuestro fichero “autor.routes.js” vamos a añadir el siguiente código:

```
import {Router} from "express";
import {login, register, logout} from "../controladores/autor.controlador.js";

const ruta = Router();

ruta.post('/register', register);
ruta.post('/login', login);
ruta.post('/logout', logout);

export default ruta;
```


IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891

Importamos “Router” desde el módulo “express”. Como podéis observar importamos también las funciones flecha “login”, “register”, “logout” de nuestro fichero autor.controlador.js.

La idea es que desde “**autor.routes.js**” definimos las diferentes routas que queremos configurar y les asociamos las funciones que las vamos a definir en nuestro fichero “**autor.controlador.js**”.

Desde el fichero **“autor.routes.js”** creamos una variable “ruta” y la instanciamos con Router().

A partir de aquí, a través de “ruta.post()” en la que mandamos la información del frontend al backend. Vemos que los parámetros que le pasamos a “ruta.post()” son por un lado el nombre de la ruta a configurar y por otro lado, la función que queremos que se ejecute cuando entramos en esa ruta.

El contenido del fichero “autor.controlador.js” es el que se muestra a continuación:

```
export const register = async (req, res) => res.send("registrando");
export const login = async (req, res) => res.send("logeado");
export const logout = async (req, res) => res.send("saliendo");
```

En este contenido vemos que exportamos las variables flecha y trabajamos de una forma asíncrona que es una forma de trabajar que habéis empezado a trabajar con Pedro. Tenemos dos parámetros:

- req: parámetro para trabajar con los datos de entrada.
- res: parámetro para trabajar con los datos de salida. En el ejemplo lo utilizamos para enviar el mensaje de registrado, logeado o saliendo. Más adelante vamos a configurar estas funciones flecha para gestionar todo lo referente a nuestra base de datos sobre mongoDB.

IES VLLABLANCA
C/Villablanca, 79 - 28032 Madrid
Teléfono 917 752 214 – Fax 917 766 577
e-mail ies.villablanca.madrid@educa.madrid.org
C.C.: 28039891