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

Parte – 4

En esta parte de nuestro proyecto vamos a ver como trabajar con los token´s que hemos generado en la parte-3. Para hacer esto, vamos a trabajar con rutas protegidas y con funciones intermedias de verificación. Para ello vamos a utilizar la carpeta de nuestro proyecto "seguridad" y vamos a generar una nueva ruta, en la que para poder acceder a la función asociada, tendrá que pasar por una función intermedia de verificación del token. Esta ruta nueva la vamos a generar en un documento .js nuevo de la carpeta "rutas".

Vamos a crear una tarea nueva ruta que se va a denominar "**perfilUsuario**" en la que se nos va a mostrar el perfil del usuario que se acaba de registrar o que se ha logeado. Esto lo vamos a hacer a través de la verificación de la cookie "token", como como hemos visto en el documento anterior, guarda el token codificado.

Dentro de la carpeta de "**seguridad**", vamos a crear un documento denominado "**verificacionRuta.js**"

Lo primero que vamos a hacer es coger el valor de la cookie para comprobar que el usuario está logeado correctamente. Para leer una cookie debemos de instalar el módulo "**cookie-parse**".

npm i cookie-parser

Una vez cargado el módulo y al igual que hicimos con el módulo para poder leer el .json, en nuestro fichero "**aplicacion.js**", las siguientes líneas:

Import nuestraCookie from 'cookie-parser';

aplicacion.use(nuestraCookie());

El fichero queda como se muestra en la imagen:

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 express from 'express'; //importamos la libreria expres.
import morgan from 'morgan'; //Nos sirve para ver las peticiones que se le hacen al servidor.
import ruta from './rutas/autor.routes.js';
//import rutaOperaciones from './rutas/operaciones.routes.js';
import cookieParser from "cookie-parser";
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.
aplicacion.use(cookieParser());
aplicacion.use("/api", ruta);

//aplicacion.use(rutaOperaciones());

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.
```

Una vez configurado nuestro proyecto para poder trabajar con los datos recogidos desde una cookie, configuramos nuestro fichero de rutas para añadir la nueva con el punto intermedio de seguridad. El fichero "**autor.routes.js**", quedaría de la siguiente manera:

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

const ruta = new Router();

ruta.post('/register', register);
ruta.post('/login', login);
ruta.post('/logout', logout);
ruta.get('/miPerfil', autorizacionRequerida, miperfil);

export default ruta;
```

Lo primero que hacemos es importar la función "**autorizacionRequerida**" desde el fichero "**verificaciónRuta.js**". Esta función la vemos un poco más abajo.

Lo segundo es añadir la ruta nueva, en este caso **"/miPerfil**" y, a diferencia del resto, le pasamos tres parámetros:

- El string con la ruta nueva.
- El punto medio de seguridad que estamos creando, en este caso, **autorizacionRequerida**.
- Si se pasa el punto intermedio de seguridad, la función **miperfil** del fichero "**autor.controlador.js**".

Fijaros que para añadir la función miperfil, debemos de añadirlo en el import:

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

La función autorizacionRequerida del fichero "**verificacionRuta.js**" dentro de la carpeta "**seguridad**", quedaría de la siguiente forma:

```
import jwt from 'jsonwebtoken';
import {TOKEN_SECRET_KEY} from '../config.js';


export const autorizacionRequerida = (req, res, next) => {  //Esta función es
intermedia, por eso tiene tres parámetros de entrada.
    const {token} = req.cookies;
    if (!token) return res.status(401).json({message: "No autorizado"});

    jwt.verify(token, TOKEN_SECRET_KEY, (err, user) => {
        if (err) return res.status(401).json({message : "Token invalido"});
        req.user = user;
        next();
    })
}
```

Lo primero que vemos es que importamos el "**jsonwebtoke**", que vamos a utilizar más tarde para verificar el token e importamos la variable "**TOKEN_SECRET_KEY**", que lo único que hace es guardar el string que he utilizado como clave principal.

```
export const TOKEN_SECRET_KEY = "secret123";
```

Luego, nos debería llamar la atención que, en este caso, la función flecha **autorizacionRequerida**, recibe tres parámetros:

- req -> Datos con los que trabaja el backend (servidor).
- res-> Datos con los que trabaja el frontend (cliente).
- next -> Si todo es correcto, ejecutamos la siguiente función definida en el fichero autor.routes.js.

En esta función, lo primero que hacemos es recoger el valor de la cookie "token" y la guardo en una variable token.

```
const {token} = req.cookies;
```

Si la variable "token" está vacía devolvemos un error, puesto que si el usuario está logeado, debe de tener un valor. Recordar que en la cookie token, se guarda el token que se genera sobre el id del usuario.
Si se provoca el error, se envía un código de estado y un mensaje de error.

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

```
if (!token) return res.status(401).json({message: "No autorizado"});
```

Si el token existe, entonces lo que hacemos es validar ese token que cogemos de las cookies con otro que generamos de nuevas:

```
jwt.verify(token, TOKEN_SECRET_KEY, (err, user) => {
        if (err) return res.status(401).json({message : "Token invalido"});
        req.user = user;
        next();
    })
```

Esto lo hacemos con jwt.verify(). Tres parámetros de entrada, el token que tenemos guardado, la password que me sirve de semilla para volver a generar ese token. Si hay algún error en la comparación, se activa “err” y si todo está bien, nos quedamos con toda la información del usuario.

Una vez se supera esta función intermedia, ejecutamos la función miperfil en la función “**autor.controlador.js**”.

```
export const miperfil = async (req, res) => {
    const usuarioEncontrado = await usuario.findById(req.user.id);

    if (!usuarioEncontrado) return res.status(400).json({message: "Usuario
desconocido"});

    return res.json({
        id : usuarioEncontrado._id,
        username : usuarioEncontrado.username,
        email : usuarioEncontrado.email,
        CreateAt : usuarioEncontrado.createdAt,
        UpdateAt : usuarioEncontrado.updatedAt,
    });
}
```

Es importante que prestéis atención a como siempre utilizamos los parámetros de entrada de “req” y “res”, esto nos permite tener acceso a esos valores (dentro de la tecnología web), de estos parámetros.

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