



## Tarea 5: Autenticación

27 de Marzo

En esta tarea añadiremos la [autenticación a la aplicación](#). La haremos con tokens [jwt](#) intercambiados en [cookies](#).

Lo primero será crear una tabla de usuarios en la BD. Modificamos el archivo **schema.prisma** añadiendo:

```
model Usuario {
  correo    String @id
  nombre    String
  password  String
  rol       ROL @default(USUARIO)
}

enum ROL {
  USUARIO
  ADMINISTRADOR
}
```

y para crear las tablas:

```
> pnpm exec prisma migrate dev añadido_usuario
```

Las contraseñas se deben guardar **cifradas**.

## Dependencias para instalar

```
> pnpm i bcryptjs      # para cifrar/descifrar la contraseña
> pnpm i jsonwebtoken  # para el jwt
> pnpm i cookie-parser # middleware para poner la info de las cookies en el request
```

## Controlador

Seguiremos **Beginner's authentication, JWT and cookies**. El código estará en un nuevo archivo **routes/usuarios.mjs**, con 4 funciones:

```
import jwt from "jsonwebtoken"

...
router.get('/login', (req, res)=>{
  res.render('login.njk')
})

router.post('/login', async (req, res)=>{ // viene del formulario de login
  // comprobar las credenciales en la BD:
  ...
  // genera el token jwt, con una clave secreta en .env
  const token = jwt.sign({usuario: user.nombre, rol: user.rol}, process.env.SECRET_K

  res.locals.usuario = user.nombre // info para las plantillas en el resp

  // pone la cookie con el jwt
  res.cookie('access_token', token, {
    httpOnly: true, // Evita acceso desde JavaScript del cl
    secure: process.env.IN === 'production', // En producción aseguramos HTTPS
    maxAge: 7200000 // 2 horas en milisegundos
```

```

    }).render('index.njk') // o donde sea
  })

  router.get('/logout', (req, res)=>{
    // borrar la cookie:
    ...
    res.render('index.njk') // o donde sea
  })

  router.post('/registro', async (req, res) => { // viene del formulario de registro
    // añadir a la BD:
    ...
    res.render('index.njk') // o a una bienvenida
  })

```

## Middleware de autenticación

Además en el programa principal habrá un `middleware` para comprobar la cookie, y en su caso añadir información al request (el usuario y su rol), y también al response para modificar la cabecera de la página con la información del usuario logeado.

```

// index.mjs
import cookieParser from "cookie-parser"
import jwt from "jsonwebtoken"
...
app.use(express.urlencoded({ extended: true })) // para poner los parámetros del fc
app.use(cookieParser())

...

// middleware de
const autenticación = (req, res, next) => {
  const token = req.cookies.access_token;
  if (token) {
    const data = jwt.verify(token, process.env.SECRET_KEY);
    req.usuario      = data.usuario // en el request
    req.rol          = data.rol
  }
  next();
}

```

```
    res.locals.usuario = data.usuario    // en el response para
    res.locals.rol      = data.rol       // para que se tenga acceso en las plantilla
    console.log('En el request ', req.usuario, req.rol)
  }
  next()
}

...

app.use(authenticación)
```

## Vistas

Las IAs ya son aprovechables para el UI: [respuesta de la IA](#) para el prompt: *'una página web de login y registro hecha con la librería bulma'*.

## Referencias:

- [Authentication using cookies with JWT in ExpressJs](#)
- [Sure, here's an advanced tutorial on HTML](#)
- [JSON Web Token \(JWT\) Debugger](#)
- [Express res.locals](#)