




Ejercicio Práctico

 **Título:** Autenticación y Autorización con JWT y Roles en una Aplicación Web

Objetivo del ejercicio:

Implementar un sistema de autenticación utilizando **JWT** para asegurar que solo los usuarios autenticados accedan a los recursos de la aplicación. Además, utilizar un sistema de **roles de usuario** para controlar el acceso a las funcionalidades según el tipo de usuario (Administrador, Editor y Usuario Común).

Escenario:

Estás desarrollando una aplicación web para una plataforma de gestión de proyectos. La plataforma tiene tres tipos de usuarios:

- **Administrador:** Puede ver, crear, editar y eliminar proyectos y tareas.
- **Editor:** Puede ver, crear y editar proyectos y tareas, pero no eliminarlas.
- **Usuario Común:** Puede ver solo los proyectos y tareas que le han sido asignados.

El sistema debe permitir a los usuarios autenticarse mediante **JWT**, gestionar sesiones y proporcionar acceso adecuado a los recursos según su rol.

Tu tarea:

Paso 1 – Implementar la autenticación con JWT:

1. **Crear un sistema de registro** donde los usuarios puedan crear una cuenta con un **nombre de usuario** y **contraseña**. Utiliza **bcrypt** para **hashear** las contraseñas antes de almacenarlas.
2. **Crear un sistema de inicio de sesión** donde los usuarios puedan ingresar su **nombre de usuario** y **contraseña**. Si las credenciales son correctas, el sistema

debe generar un **JWT** y enviarlo al cliente.

3. **El JWT debe incluir el rol del usuario** (Administrador, Editor, Usuario Común) y tener un tiempo de expiración limitado.

Paso 2 – Proteger las rutas con JWT:

1. **Middleware de autenticación:** Implementa un middleware que verifique si el **JWT** es válido antes de permitir el acceso a las rutas protegidas. Si el token es inválido o ha expirado, la solicitud debe ser rechazada con un mensaje de error.
2. **Proteger rutas según roles:**
 - **Administrador:** Puede acceder a todas las rutas de proyectos y tareas, incluida la eliminación de tareas.
 - **Editor:** Puede acceder a la creación y edición de proyectos y tareas, pero no puede eliminarlas.
 - **Usuario Común:** Solo puede ver los proyectos y tareas asignadas.

Paso 3 – Control de acceso basado en roles:

1. **Acceso a proyectos y tareas:** Configura el sistema para que cada usuario tenga acceso solo a los proyectos y tareas que le han sido asignados, dependiendo de su rol.
 - Los **Administradores** tienen acceso completo a todos los proyectos y tareas.
 - Los **Editores** solo pueden ver y editar los proyectos y tareas que han creado o se les han asignado.
 - Los **Usuarios Comunes** solo pueden ver los proyectos y tareas que les han sido asignados específicamente.
2. **Mostrar el rol del usuario:** Al autenticar un usuario, el sistema debe mostrar su rol y las funcionalidades disponibles para ese rol.

Paso 4 – Pruebas de acceso y seguridad:

1. **Prueba de acceso:** Asegúrate de que los **usuarios con diferentes roles** solo puedan acceder a las funcionalidades correspondientes a su rol.
2. **Verificación del JWT:** Asegúrate de que el **JWT** se valide correctamente en todas las rutas protegidas.

Paso 5 – Cerrar sesión:

1. **Cerrar sesión:** Implementa una función de cierre de sesión que invalide el **JWT** en el lado del cliente (puedes almacenar el token en **localStorage** o **sessionStorage** en el cliente).
-

✅ Resultado esperado:

- Un sistema de autenticación basado en **JWT** que permite el acceso a la aplicación solo a los usuarios autenticados.
 - Un sistema de **roles de usuario** (Administrador, Editor, Usuario Común) que controla el acceso a las funcionalidades de la plataforma.
 - **Protección de rutas** mediante el uso de JWT y un middleware de autenticación.
 - Función de **cierre de sesión** que invalida el JWT.
-

📄 Entrega sugerida:

1. **Código fuente** de la implementación de la autenticación con JWT y control de acceso por roles.
 2. **Capturas de pantalla** de las funcionalidades de autenticación, registro, inicio de sesión y pruebas de control de acceso según el rol.
 3. **Informe detallado** sobre cómo se implementaron la gestión de sesiones y el control de roles, con ejemplos de uso.
-

🧰 Herramientas recomendadas:

- **Node.js** y **Express.js** para el backend (puedes usar cualquier otro framework backend de tu preferencia).
- **JWT** (puedes usar la librería **jsonwebtoken** para crear y verificar los tokens).
- **bcrypt** para el hashing de contraseñas.

- **MongoDB** o **MySQL** para almacenar usuarios, roles, proyectos y tareas.
 - **Postman** o **Insomnia** para probar las rutas de la API.
-



Sugerencia para extender el ejercicio (opcional):

- **Autenticación Multifactor (MFA):** Implementa una capa adicional de seguridad utilizando MFA para proteger las cuentas de usuario.
 - **Token de actualización (Refresh Token):** Implementa un **refresh token** para permitir que los usuarios obtengan nuevos JWT sin necesidad de iniciar sesión nuevamente, evitando que tengan que volver a autenticarse después de que el token expire.
-