🛠️ Ejercicio Práctico

📌 Título: *Implementación de Autenticación y Autorización en una Aplicación Web con JWT y RBAC*

🎯 Objetivo del ejercicio:

Aplicar conceptos de **autenticación** y **autorización** en una aplicación web utilizando **JWT (JSON Web Tokens)** para la autenticación y **RBAC (Role-Based Access Control)** para la autorización. Los estudiantes crearán un sistema de acceso controlado donde diferentes tipos de usuarios tienen distintos permisos.

📝 Escenario:

Imagina que estás desarrollando una **aplicación web** para un sistema de gestión de empleados. El sistema tiene tres tipos de usuarios:

- **Empleado**: Puede ver su información personal.
- **Gerente**: Puede ver la información personal de los empleados a su cargo.
- **Administrador**: Tiene acceso completo a todos los registros del sistema.

Para proteger el acceso a las diferentes secciones del sistema, utilizarás **JWT** para la autenticación de los usuarios y **RBAC** para gestionar los permisos de acceso en función del rol de cada usuario.

🔍 Tu tarea:

Paso 1 – Crear un sistema de autenticación con JWT:

1. **Implementar la autenticación JWT** para que los usuarios se registren e inicien sesión en la aplicación. Utiliza **JWT** para generar un token de acceso cuando un

usuario se autentique con su nombre de usuario y contraseña.

2. **Proteger las rutas del backend** utilizando el token JWT para que solo los usuarios autenticados puedan acceder a los recursos.
    - Implementa un middleware que verifique el token en cada solicitud a las rutas protegidas.

Paso 2 – Implementar RBAC para la autorización:

1. **Definir roles** para los usuarios: **Empleado**, **Gerente**, y **Administrador**. Cada rol tendrá diferentes permisos en la aplicación.
2. **Asignar roles** a los usuarios durante el registro o al realizar la autenticación.
3. **Configurar rutas protegidas** con acceso basado en roles:
    - **Empleados**: Pueden acceder solo a su propia información.
    - **Gerentes**: Pueden acceder a la información de los empleados a su cargo.
    - **Administradores**: Pueden acceder a toda la información en el sistema.
4. **Gestionar acceso a los recursos** basándote en los roles de los usuarios. Por ejemplo:
    - Si un **Empleado** intenta acceder a los datos de otro empleado, el sistema debe denegar el acceso.
    - Si un **Gerente** intenta acceder a los datos de otro **Gerente**, el acceso debe ser denegado.

Paso 3 – Verificación y Pruebas:

1. **Probar el sistema**:
    - Asegúrate de que el sistema genere un **token JWT** al iniciar sesión correctamente.
    - Verifica que el token es necesario para acceder a las rutas protegidas.
    - Realiza pruebas para asegurarte de que los permisos de acceso se aplican correctamente según el rol del usuario.
2. **Pruebas de seguridad**:

- Verifica que un **Empleado** no pueda acceder a la información de otros empleados.
- Verifica que un **Gerente** solo pueda acceder a la información de los empleados a su cargo.
- Verifica que un **Administrador** tenga acceso completo a todos los registros.

✅ Resultado esperado:

- Un sistema funcional que permita a los usuarios autenticarse mediante **JWT**.
- Implementación de **RBAC** que limite el acceso de los usuarios según su rol (Empleado, Gerente, Administrador).
- Seguridad implementada para proteger las rutas y recursos según el **rol del usuario**.

📄 Entrega sugerida:

1. **Código fuente** de la implementación de la autenticación JWT y autorización RBAC.
2. **Capturas de pantalla** que muestren el registro, inicio de sesión y las pruebas de acceso (con tokens JWT).
3. **Informe detallado** sobre cómo se implementaron las rutas protegidas y cómo los roles controlan el acceso.

🧰 Herramientas recomendadas:

- **Node.js** y **Express.js** para el backend (puedes usar otros frameworks si lo prefieres).
- **JWT** para autenticación (puedes usar librerías como **jsonwebtoken**).
- **MongoDB** o **MySQL** para almacenar usuarios y roles.
- **Postman** o **Insomnia** para probar las rutas de la API.

📝 Sugerencia para extender el ejercicio (opcional):

- Implementar **Autenticación Multifactor (MFA)** junto con JWT para aumentar la seguridad de la aplicación.

- Agregar **sesiones de usuario** y **expiración de tokens JWT** para mejorar la seguridad.
