Consigna

Presentar un servidor backend con Node.js y Express que implemente autenticación mediante JWT (JSON Web Tokens), cumpliendo con los siguientes requisitos y estructura general de trabajo.

Estructura base del proyecto

- El proyecto deberá incluir, como mínimo, los siguientes archivos y carpetas:
- `server.js` o `app.js` (archivo principal del servidor)
- `rutas/usuarios.js` (rutas de registro e inicio de sesión)
- `rutas/privadas.js` (rutas que requieren autenticación)
- `middlewares/autenticacion.js` (middleware para verificación de tokens)
- `.env.example` (archivo de configuración con variables de entorno de ejemplo)
- `.gitignore` (archivo para excluir node_modules y .env del repositorio)
- `package.json` (archivo de configuración de npm)

Objetivo

Desarrollar una API REST con Node.js que implemente un sistema de autenticación básico utilizando JWT, encriptación de contraseñas, y manejo de variables de entorno. El proyecto deberá demostrar comprensión de conceptos básicos de seguridad backend y middleware.

Requisitos técnicos

1. Configuración inicial del proyecto:

- Inicializar un proyecto Node.js
- Instalar las dependencias necesarias:
  - `express`
  - `jsonwebtoken`
  - `bcrypt` o `bcryptjs`
  - `dotenv`
- Crear un archivo `.env` con al menos:
  - `PUERTO` (puerto del servidor)
  - `CLAVE_SECRETA` (clave para firmar tokens)
  - Otras variables que consideren necesarias

2. Sistema de autenticación:

- Implementar en `rutas/usuarios.js`:
  - **POST /api/usuarios/registro**: Endpoint para crear nuevos usuarios
    - Recibir datos del usuario (al menos nombre de usuario y contraseña)
    - Validar los datos recibidos

        - Encriptar la contraseña antes de almacenarla
        - Guardar el usuario (puede ser en memoria o base de datos)
        - Devolver respuesta apropiada
    - **POST /api/usuarios/acceso**: Endpoint para iniciar sesión
        - Recibir credenciales del usuario
        - Validar las credenciales
        - Generar y devolver un JWT si son correctas
        - El token debe incluir información del usuario

3. Middleware de autorización:

- Crear `middlewares/autenticacion.js` que:
    - Verifique la presencia de un token JWT en las peticiones
    - Valide el token usando la clave secreta
    - Permita o deniegue el acceso según la validez del token
    - Agregue la información del usuario a la petición si es válido

4. Rutas protegidas:

- Implementar en `rutas/privadas.js`:
    - Al menos una ruta que requiera autenticación
        - Aplicar el middleware de autenticación
        - Devolver contenido solo a usuarios autenticados
        - Ejemplo: **GET /api/privado/perfil** o cualquier otra ruta de su elección

5. Servidor principal:

- Configurar en `server.js` o `app.js`:
    - Cargar variables de entorno
    - Configurar Express
    - Montar las rutas creadas
    - Iniciar el servidor

6. Consideraciones de seguridad:

- Las contraseñas deben estar encriptadas
- La clave secreta del JWT debe estar en variables de entorno
- Implementar manejo básico de errores

7. Funcionalidad esperada:

- El sistema debe permitir:

1. Registrar usuarios
2. Iniciar sesión
3. Acceder a rutas protegidas con un token válido
4. Denegar acceso a rutas protegidas sin token

Pruebas recomendadas

Para verificar el correcto funcionamiento, se puede usar cualquier herramienta de prueba de APIs:

1. Registrar usuario: `POST /api/usuarios/registro`
2. Iniciar sesión: `POST /api/usuarios/acceso`
3. Acceder a ruta protegida con token válido
4. Intentar acceder a ruta protegida sin token

Criterios de evaluación

- Implementación funcional de JWT
- Encriptación de contraseñas
- Uso de variables de entorno
- Estructura organizada del proyecto
- Funcionamiento del flujo de autenticación
- Documentación básica del proyecto

Formato de entrega

Deberá entregarse la URL de un repositorio público de GitHub donde se encuentre el proyecto completo. El repositorio debe incluir:

- Todos los archivos fuente del proyecto
- Un archivo README.md con instrucciones de instalación y uso
- El archivo `.gitignore` configurado correctamente (excluyendo `node_modules` y `.env`)
- Un archivo `.env.example` mostrando las variables de entorno requeridas (sin valores reales)

Importante: Verificar que el repositorio es público y puede accederse desde una sesión incógnita del navegador. No se debe incluir el archivo `.env` real.