Guía de ejercicios - JWT (III)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Actividad guiada: Acceso a Contenido Premium

En este ejercicio implementaremos un sistema de autenticación de usuarios y una ruta de acceso a libros exclusivos en la aplicación de biblioteca digital "Bookshelf". Los usuarios deben autenticarse para obtener un token JWT válido y acceder a los libros exclusivos.

1. **Paso 1:** Configurar una lista de usuarios simulados con información como nombre de usuario y contraseña.

```
const usuarios = [
  { id: 1, username: 'usuario1', password: 'clave123' },
  { id: 2, username: 'usuario2', password: 'contraseña456' }
];
```

2. **Paso 2**: Crear una ruta **/login** que acepte username y password como parámetros en la query string.
   Verificar si las credenciales son válidas comparándolas con la lista de usuarios simulados. Si las credenciales son correctas, generar un token JWT con el id del usuario como payload y una expiración de 1 hora.

```
app.get("/login", (req, res) => {
  const { username, password } = req.query;
  const usuario = usuarios.find(u => u.username === username &&
u.password === password);

  if (!usuario) {
    return res.status(401).json({
      error: "401 Unauthorized",
      message: "Credenciales incorrectas. Por favor, verifica tu nombre
de usuario y contraseña."
    });
  }

  // Generar un token JWT con el id del usuario como payload y
expiración en 1 hora
  const token = jwt.sign({ userId: usuario.id }, secretKey, { expiresIn:
'1h' });
  res.json({ token });
});
```

3. **Paso 3**: Crear una ruta **/libro-exclusivo** que verifique el token y permita el acceso al libro exclusivo. Verificar si el libro correspondiente al libroId del token existe y está disponible.
   Si es así, permitir el acceso al libro y enviar un mensaje indicando el título del libro que el usuario puede leer.

```
app.get("/libro-exclusivo", (req, res) => {
  const { token, libroId } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido."
      });
    }

    const userId = decoded.userId;
    const usuario = usuarios.find(u => u.id === userId);
    if (!usuario) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Usuario no autorizado."
      });
    }

    const libro = librosExclusivos.find(l => l.id === parseInt(libroId)
&& l.disponible);
    if (!libro) {
      return res.status(403).json({
        error: "403 Forbidden",
        message: "Libro exclusivo no disponible en este momento."
      });
    }

    // Permitir acceso al libro exclusivo y enviar un mensaje con el
título del libro.
    res.send(`Disfruta de "${libro.titulo}", un libro exclusivo en
Bookshelf!`);
  });
});
```

4. **Paso 4**: Implementar una ruta **/validar-token** que acepte un token JWT en la query string y verifique su validez utilizando la secretKey. Devuelve un mensaje indicando si el token es válido o no.

```
app.get("/validar-token", (req, res) => {
  const { token } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido."
      });
    }
    res.json({
      message: "Token válido. Usuario autenticado.",
      decodedData: decoded
    });
  });
});
```

5. **Paso 5**: Iniciar el servidor en el puerto 3000

```
app.listen(3000, () => {
  console.log('Servidor en ejecución en el puerto 3000.');
});
```

Ahora asegúrate de que la ruta **/libro-exclusivo** esté protegida y solo sea accesible para usuarios autenticados con un token válido.

Puedes agregar manejo de errores para casos donde las credenciales sean incorrectas, el token sea inválido o el libro exclusivo no esté disponible.

Finalmente el código completo quedaría de la siguiente manera:

```
const express = require('express');
const jwt = require('jsonwebtoken');

const app = express();
const secretKey = 'clave_secreta_temporal'; // Clave secreta temporal
para las pruebas

const usuarios = [
  { id: 1, username: 'usuario1', password: 'clave123' },
  { id: 2, username: 'usuario2', password: 'contraseña456' }
];

const librosExclusivos = [
  { id: 1, titulo: 'Libro Exclusivo 1', disponible: true },
  { id: 2, titulo: 'Libro Exclusivo 2', disponible: false },
  { id: 3, titulo: 'Libro Exclusivo 3', disponible: true },
  { id: 4, titulo: 'Libro Exclusivo 4', disponible: true },
  { id: 5, titulo: 'Libro Exclusivo 5', disponible: false }
];

app.get("/login", (req, res) => {
  const { username, password } = req.query;
  const usuario = usuarios.find(u => u.username === username &&
u.password === password);

  if (!usuario) {
    return res.status(401).json({
      error: "401 Unauthorized",
      message: "Credenciales incorrectas. Por favor, verifica tu nombre
de usuario y contraseña."
    });
  }

  // Generar un token JWT con el id del usuario como payload y
expiración en 1 hora
  const token = jwt.sign({ userId: usuario.id }, secretKey, { expiresIn:
'1h' });
  res.json({ token });
});

app.get("/libro-exclusivo", (req, res) => {
```

```
  const { token, libroId } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido."
      });
    }

    const userId = decoded.userId;
    const usuario = usuarios.find(u => u.id === userId);
    if (!usuario) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Usuario no autorizado."
      });
    }

    const libro = librosExclusivos.find(l => l.id === parseInt(libroId)
&& l.disponible);
    if (!libro) {
      return res.status(403).json({
        error: "403 Forbidden",
        message: "Libro exclusivo no disponible en este momento."
      });
    }

    // Permitir acceso al libro exclusivo y enviar un mensaje con el
título del libro.
    res.send(`Disfruta de "${libro.titulo}", un libro exclusivo en
Bookshelf!`);
  });
});

app.get("/validar-token", (req, res) => {
  const { token } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido."
      });
    }
    res.json({
```

```
      message: "Token válido. Usuario autenticado.",
      decodedData: decoded
    });
  });
});

app.listen(3000, () => {
  console.log('Servidor en ejecución en el puerto 3000.');
});
```

¡Manos a la obra! - Registro de usuarios

Implementa una nueva ruta **/registro** que permita a los usuarios registrarse en la aplicación. Los usuarios deben proporcionar un nombre de usuario y una contraseña. Al registrarse, se les asignará un ID único y se almacenarán en una lista de usuarios simulados.

¡Manos a la obra! - Listado de libros exclusivos

Implementa una nueva ruta **/libros-exclusivos** que devuelva una lista de libros exclusivos disponibles para los usuarios. Esta lista debe incluir sólo los libros que están marcados como "disponibles" en la lista librosExclusivos.

Solución ejercicios - Manos a la obra

Registro de usuarios

```
// Paso 1: Crear una lista de usuarios registrados simulados.
const usuariosRegistrados = [];

// Paso 2: Crear una ruta /registro que acepte username y password como
parámetros en la query string.
app.get("/registro", (req, res) => {
  const { username, password } = req.query;
  const nuevoUsuario = { id: usuariosRegistrados.length + 1, username,
password };
  usuariosRegistrados.push(nuevoUsuario);
  res.json({ message: "Usuario registrado exitosamente.", userId:
nuevoUsuario.id });
});
```

Listado de libros exclusivos

```
// Paso 1: Crear una ruta /libros-exclusivos que devuelva la lista de
libros exclusivos disponibles.
app.get("/libros-exclusivos", (req, res) => {
  const librosDisponibles = librosExclusivos.filter(libro =>
libro.disponible);
  res.json(librosDisponibles);
});
```

Preguntas de proceso

Reflexiona:

- ¿Qué he aprendido hasta ahora?
- ¿Hay algo que me está dificultando mucho?

Preguntas de cierre

- ¿Por qué es importante mantener la secretKey en secreto al usar tokens JWT?
- ¿Cuál es la importancia de la expiración de los tokens JWT en un sistema de autenticación y cómo puede ayudar a mejorar la seguridad de una aplicación web?
- ¿Qué diferencias clave existen entre la autenticación y la autorización en el contexto de las aplicaciones web?