JWT OAuth2

Julian F. Latorre

Índice

1. Ejercicio: Implementar autenticación JWT en una API Express

Este ejercicio guía paso a paso la implementación de un sistema de autenticación básico utilizando JSON Web Tokens (JWT) en una API Express.

1.1. Paso 1: Configuración del proyecto

Primero, configuramos nuestro proyecto y instalamos las dependencias necesarias:

```
mkdir jwt-auth-express
cd jwt-auth-express
npm init -y
npm install express jsonwebtoken bcryptjs
npm install --save-dev nodemon
```

1.2. Paso 2: Crear el archivo principal (app.js)

Creamos el archivo principal de nuestra aplicación:

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

const app = express();
app.use(express.json());

const SECRET_KEY = 'tu_clave_secreta';

// Base de datos simulada
const users = [];

// Ruta de registro
app.post('/register', async (req, res) => {
    try {
        const { username, password } = req.body;
        const hashedPassword = await bcrypt.hash(password, 10);
        users.push({ username, password: hashedPassword });
        res.status(201).json({ message: 'Usuario registrado
    exitosamente' });
    } catch (error) {
        res.status(500).json({ error: 'Error en el registro' });
    }
});

// Ruta de login
app.post('/login', async (req, res) => {
    try {
        const { username, password } = req.body;
        const user = users.find(u => u.username === username);
        if (user && await bcrypt.compare(password, user.password))
    {
```

```
            const token = jwt.sign({ username }, SECRET_KEY, {
    expiresIn: '1h' });
            res.json({ token });
        } else {
            res.status(401).json({ error: 'Credenciales inválidas'
    });
        }
    } catch (error) {
        res.status(500).json({ error: 'Error en el login' });
    }
});

// Middleware de autenticación
function authenticateToken(req, res, next) {
    const authHeader = req.headers['authorization'];
    const token = authHeader && authHeader.split(' ')[1];
    if (token == null) return res.sendStatus(401);

    jwt.verify(token, SECRET_KEY, (err, user) => {
        if (err) return res.sendStatus(403);
        req.user = user;
        next();
    });
}

// Ruta protegida
app.get('/protected', authenticateToken, (req, res) => {
    res.json({ message: 'Acceso concedido', user: req.user });
});

const PORT = 3000;
app.listen(PORT, () => console.log(`Servidor corriendo en puerto ${
    PORT}`));
```

1.3. Paso 3: Configurar script de inicio en package.json

Modificamos el archivo `package.json` para incluir scripts de inicio:

```
"scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
}
```

1.4. Paso 4: Iniciar el servidor

Iniciamos el servidor con el siguiente comando:

```
npm run dev
```

1.5. Paso 5: Probar la API

Utilizamos un cliente HTTP como Postman o cURL para probar las siguientes rutas:

1.5.1. Registrar un usuario

```
POST http://localhost:3000/register
Body: { "username": "usuario1", "password": "contraseña123" }
```

1.5.2. Iniciar sesión

```
POST http://localhost:3000/login
Body: { "username": "usuario1", "password": "contraseña123" }
```

1.5.3. Acceder a la ruta protegida

```
GET http://localhost:3000/protected
Headers: Authorization: Bearer <token_obtenido_en_login>
```