Hank Erubiel Cárdenas Zamora

2-1

Seguridad Basada en tokens JWT

Configuración Inicial y Dependencias:

```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const clientes = require('./clientes');
const puerto = 3000;
```

- Se utiliza Express como framework web
- bodyParser para procesar JSON en las peticiones
- jwt para manejar la autenticación con tokens
- Se importa un módulo clientes que maneja la base de datos
- El servidor corre en el puerto 3000

Sistema de Autenticación:

```
app.post('/login', (req, res) => {
    const { usuario, password } = req.body;
    if (usuario == 'admin' && password == '123') {
        const token = jwt.sign({ usuario }, secretKey, { expiresIn: '1h' }); // utilizar JWT
        res.send(token)
    } else {
        res.status(404);
    }
})
```

- Endpoint /login para autenticación
- Verifica credenciales (usuario: 'admin', password: '123')
- Genera un token JWT que expira en 1 hora
- El token se usa para autenticar las siguientes peticiones

Middleware de Verificación de Token:

```
function verificarToken(req, res, next) { // middleware
    const header = req.header('Authorization') || '';
    const token = header.split(' ')[1];
    if (!token) {
        res.status(401).json({mensaje: 'Token no proporcionado'});
    } else {
        try {
            const payload = jwt.verify(token, secretKey);
            next();
        } catch {
            res.status(401).json({mensaje: 'Token incorrecto'});
        }
    }
}
```

- Middleware que verifica la autenticación
- Extrae el token del header 'Authorization'
- Verifica la validez del token
- Si el token es válido, permite continuar con la petición
- Si no es válido, devuelve error 401

Endpoints CRUD para Clientes:

```
app.post('/clientes',verificarToken, async (req, res) => { // create
    const { nombre, correo, telefono, direccion } = req.body;
    const data = await clientes.create({
        nombre, correo, telefono, direccion
    });
    res.send(data);
});

app.get('/clientes',verificarToken, async (req, res) => { // read
    const data = await clientes.findAll();
    res.send(data);
});

app.put('/clientes/:id',verificarToken, async (req, res) => { // update
    const { nombre, correo, telefono, direccion } = req.body;
    const { id } = req.params;
    const data = clientes.update({
        nombre, correo, telefono, direccion
    }, {
        where: {
            id
        }
    })
    res.send(data);
});

app.delete('/clientes/:id',verificarToken, async (req, res) => { // delete
    const { id } = req.params;
    const data = await clientes.destroy({
```

```
        where: {
            id
        }
    })
    res.send(data);
});
```

- Implementa las operaciones CRUD (Create, Read, Update, Delete) para clientes
- Todos los endpoints están protegidos con el middleware verificarToken
- Cada operación es asíncrona y utiliza el módulo clientes para interactuar con la base de datos
- Los endpoints son:
- POST /clientes: Crear nuevo cliente
- GET /clientes: Obtener todos los clientes
- PUT /clientes/:id: Actualizar cliente específico
- DELETE /clientes/:id: Eliminar cliente específico

Aspectos importantes de seguridad:

1. El token JWT expira en 1 hora
2. Todas las operaciones CRUD requieren autenticación
3. Se manejan errores de autenticación apropiadamente
4. Se utiliza una clave secreta para firmar los tokens