JWT

Antes de esta tecnología, la autenticación en aplicaciones web se realizaban con sesiones cookies, donde el servidor almacena información de cada usuario conectado, generando problemas de escalabilidad, especialmente en aplicaciones distribuidas o con microservicios. Este sistema evoluciona a lo que conocemos hoy como JWT (JSON Web Token).

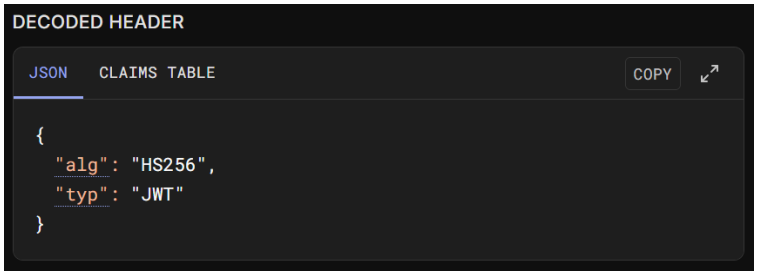
JSON Web Token es un estándar abierto basado en JSON que sirve para crear token de acceso, estos token nos permiten transmitir información entre dos sistemas como un cliente navegador y un servidor, y la información se transmite a través de un objeto JSON, por último estos JSON son firmados digitalmente con una clave secreta y por esa razón se pueden verificar en el servidor. Este sistema es perfecto para intercambiar información de manera segura, compacta y auto-contenida entre las dos partes.

¿Cómo funciona?

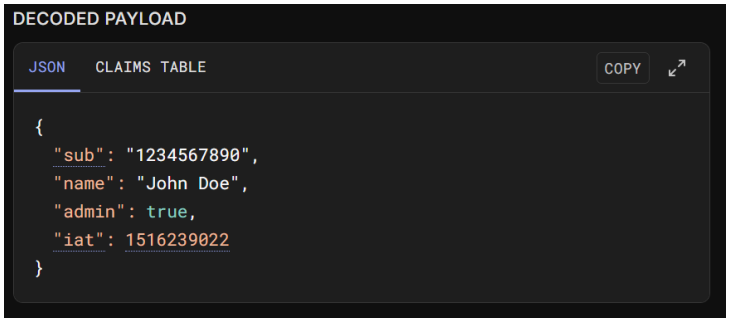
Un JWT se compone de tres partes fundamentales de un código separadas en puntos (.) :

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmF
tZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImlhdCI6MTUxNjIzOTAyMn0.KMUFsID
TnFmyG3nMiGM6H9FNFUROf3wh7SmqJp-QV30
```

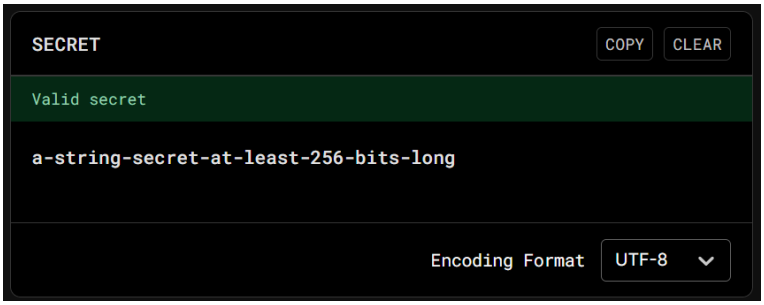
1. **Header**: Contiene el tipo de token y el algoritmo de firma.



2. **Payload**: Incluye los datos (claims), como el *id_usuario,* roles o permisos.



3. **Signature**: Se genera combinando el **header**, el **payload** y una **clave secreta,** para verificar que el token no fue alterado.



Flujo de uso:

El usuario envía sus credenciales (Correo contraseña).	El servidor valida y genera un JWT con los datos del usuario.	El cliente guarda el token en el LocalStorage o en una cookie segura.	En cada petición, el cliente envía el token en el encabezado	El servidor valida la firma y concede acceso según los permisos del usuario.

Principalmente, se utiliza para garantizar la identidad del usuario y definir permisos según roles (ej: administrador, profesor, estudiante), útil si la idea principal es realizar intercambio seguro y confiable de información entre cliente y servidor sin la necesidad de mantener sesiones de memoria.

En nuestro proyecto de Med Leave Manager, JWT se podrá aplicar para:

- Autenticación de usuarios, cuando un estudiante o profesor inicia sesión el servidor generará un JWT con la información básica (id_usuario, rol).
- Autorización o restricción a accesos como lo serían subir licencias, ver licencias asociadas al usuario o gestionar las licencias e incluso los mismos usuarios.

- Al no depender de sesiones guardadas en servidor, JWT permite que el sistema sea compatible con una arquitectura de microservicios en el futuro.

¿Cómo empezamos?

1. Instalar la librería: ***npm install jsonwebtoken***

2. importar y generar un token:

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

// Datos del usuario
const payload = { id: 1, rol: 'estudiante' };

// Generar token
const token = jwt.sign(payload, 'mi_secreto', { expiresIn: '1h' });

console.log('Token generado:', token);
```

3. Verificar un token en una ruta:

```
app.get('/protegida', (req, res) => {
  const token = req.headers['authorization']?.split(' ')[1];

  if (!token) return res.status(401).send('Acceso denegado');

  try {
    const verificado = jwt.verify(token, 'mi_secreto');
    res.send(`Bienvenido, tu rol es: ${verificado.rol}`);
  } catch (err) {
    res.status(403).send('Token inválido o expirado');
  }
});
```

Con esto ya tienes lo mínimo para generar tokens cuando un usuario inicia sesión y verificarlos en rutas protegidas.