JWT o Json Web Token a diferencia de las cookies las cuales son almacenadas en el servidor es que estas se quedan junto al usuario sin darle carga a la aplicación, el token sirve para verificar la identidad del usuario y se guarda en un localstorage(dentro del almacenamiento del navegador) o en un header(con un fetch se saca el token del localstorage o sessionstorage), el token es una cadena de texto codificada única que verifica la autenticidad del usuario y funciona de esta manera

1. El usuario inicia sesión con su usuario/contraseña.
2. El servidor valida las credenciales.
3. Si son correctas, el servidor genera un JWT y se lo envía al cliente.
4. El cliente guarda ese token (ej: en localStorage o cookies).
5. En cada petición posterior, el cliente envía el JWT en el header
6. El servidor valida el token(Si es válido permite acceso si no responde con error 401 unauthorized)

Este se separa en 3 partes que se verian como HEADER.PAYLOAD.SIGNATURE

1. Header tipo de token + algoritmo usado.
2. Payload datos (ejemplo: id_usuario, rol, etc...).
3. Signature firma digital para validar que el token no fue alterado.

Por ejemplo:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
.
{
  "user": "carlos",
  "rol": "admin"
}
.
firmaDigital123
```

Ejemplo en codigo:

```
1)
const express = require('express');
const jwt = require('jsonwebtoken');
2)
const app = express();
app.use(express.json());
```

```
const SECRET_KEY = 'mi_clave_secreta';
3)
app.post('/login', (req, res) => {
  const { user, password } = req.body;
4)
  if (user === 'admin' && password === '1234') {
    const token = jwt.sign(
      { user: user, role: 'admin' },
      SECRET_KEY,
      { expiresIn: '1h' }
    );
    res.json({ token });
  } else {
    res.status(401).json({ error: 'Credenciales inválidas' });
  }
});
5)

function verificarToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.status(403).json({ error: 'Token requerido'});

  jwt.verify(token, SECRET_KEY, (err, decoded) => {
    if (err) return res.status(403).json({ error: 'Token inválido' });
    req.user = decoded;
    next();
  });
}
6)
app.get('/dashboard', verificarToken, (req, res) => {
  res.json({
    mensaje: `Bienvenido ${req.user.user}, tu rol es ${req.user.role}`
  });
});
7)
app.listen(3000, () => console.log('Servidor en
http://localhost:3000'));
```

1. Se importa xpress y jsonwebtoken

2. Se le define app a xpress(igual que createserver) y la clave secreta para firmar los tokens
3. Se define que en la ruta del login se va a generar el token (http://localhost:3000/login)
4. Se verifica que el usuario y la contraseña coincidan con una de la base de datos en el if y se crea el token con los datos del usuario que inicio sesion y la firma con el jwt.sign(el **expiresIn** es para ver cuanto tiempo dura ese token y el usuario que esté en la página si el token expira mientras el usuario está en la página depende de la api como se maneja esto pero normalmente vuelve a redirigir al usuario al inicio de sesión, también se puede no usar el expires in pero se sufre a los ataques y que esa token con la información del usuario sea robada)
5. Se usa un middleware para verificar el token y si se requiere el token o el token es invalido, primero si no hay token (!token) da error(403) y se muestra como token requerido, en jwt.verify(token, SECRET_KEY) se define err como el token y si el token efectivamente causa un error se retorna (403) se muestra como invalido

6. Aqui se muestra un ejemplo como podría ser dashboard que requiere que el usuario cuente con el token y lo verifica, si coincide entonces el usuario puede ver la pagina
7. Se define el port del servidor (http://localhost:3000/)

El jwt ayuda bastante en la seguridad ya que se hace una renovación al token casi constante o depende de cuanto dure esto pero siento que viene bien a la pagina, no es una pagina en la cual el usuario vaya a estar mucho tiempo de por sí, asi que estas limitaciones de tiempo casi no serian notadas por el usuario promedio aunque encuentro fallas con el secretario o gente que tiene que estar mas tiempo ahi, igualmente con mi poco conocimiento creo que se podría crear un token diferente para los secretarios el cual duraría mas o sin limite de tiempo de por sí pero finalmente eso está a discusión