

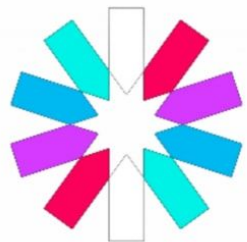
# Autenticación con JWT



Elaborado por César Guerra

[www.cesarquerra.mx](http://www.cesarquerra.mx)

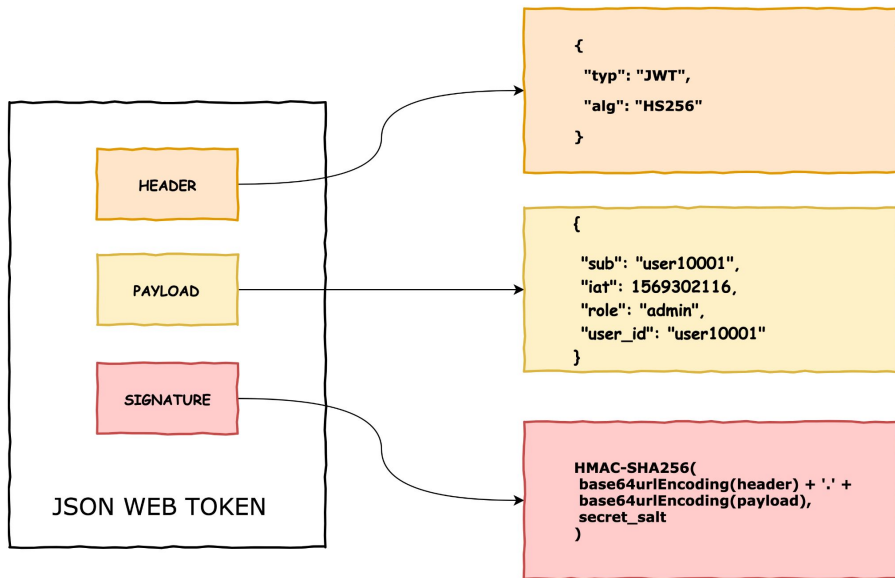
dev



# JWT

## JWT

Un **JSON Web Token** estándar abierto (RFC-7519) basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros. Contiene toda la información importante sobre una entidad, lo que implica que no hace falta consultar una base de datos ni que la sesión tenga que guardarse en el servidor (sesión sin estado).



Ejemplo de un JWT:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjEiLCJ1c2VybmFtZSI6InNlcmdpb2R4YSJ9.Qu7rv5wqk6zGjiMU8ZixwvKQGBNW9hhj55DbSP50b1g
```

Header

Payload

Signature

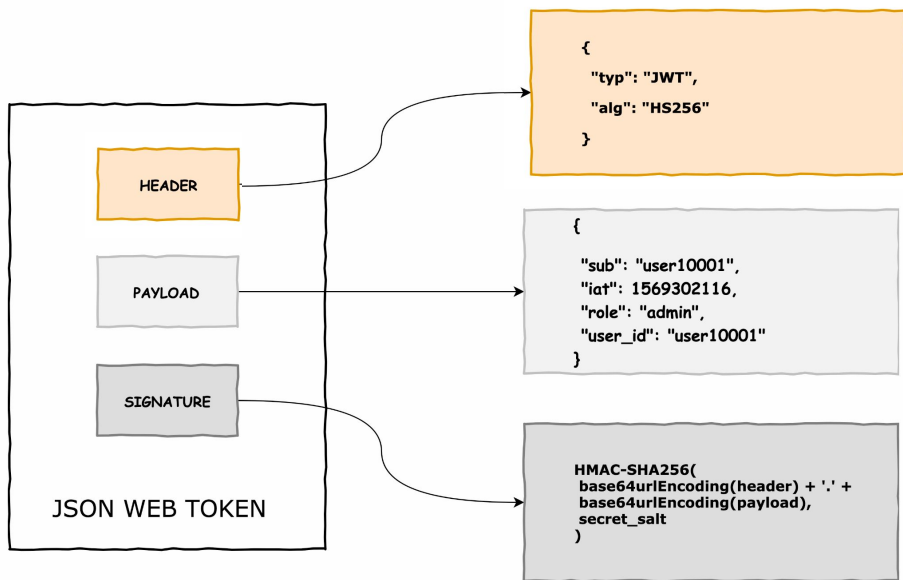
# Estructura de un JWT

Los JWT tienen una estructura definida y estándar basada en tres partes:

1. header
2. payload
3. signature

Todas las partes se concatenan con un punto quedando de la siguiente manera:

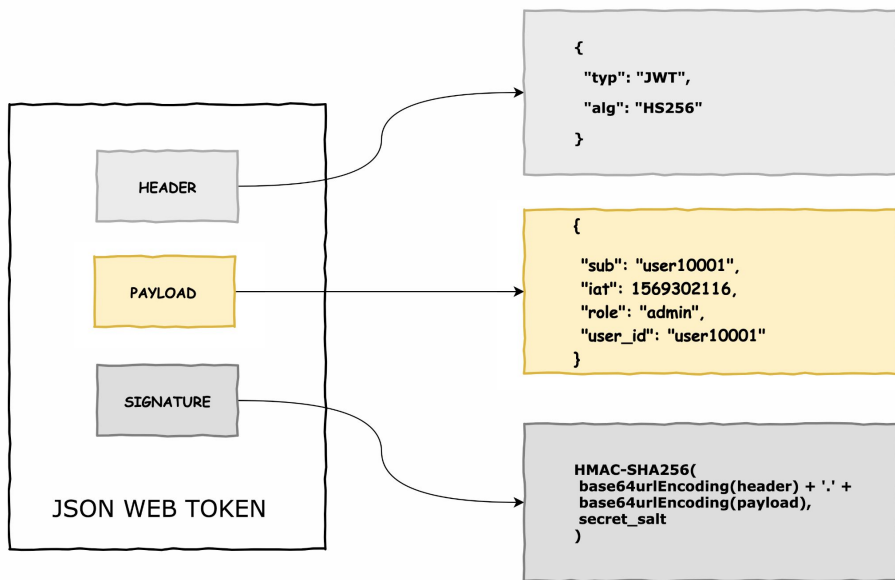
```
header.payload.signature
```



# JWT - Header

Json con Metadata del JWT el cual especifica el tipo de token y algoritmo de encriptación.

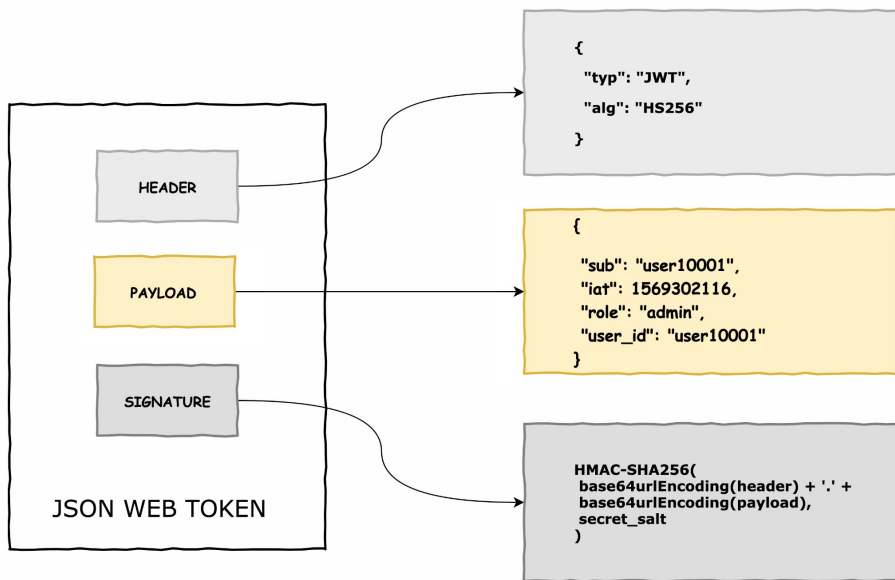
- **Tipo de token (typ):** Identifica el tipo de token. (siempre debe ser JWT)
- **Algoritmo de firmado y/o cifrado (alg):** Indica que tipo de algoritmo fue usado para firmar el token.



# JWT - Payload <sup>(1)</sup>

Es un JSON que contiene la información real que se transmitirá a la aplicación. Esta información se proporciona como pares clave/valor; las claves se denominan “claims” en JWT. Hay tres tipos diferentes de claims:

- **Claims registrados:** Son los que figuran en el IANA JSON Web Token Claim Register y cuyo propósito se establece en un estándar.
- **Claims públicos:** Son los que pueden ser definidos a voluntad por quienes usan JWTs.
- **Claims privados:** Si bien también son propiamente públicos, contienen información más concreta como datos de identificación de usuario o nombre de departamento.



## JWT - Payload (2)

Entre los claims registrados en el IANA JSON Web Token con fines específicos encontramos:

- **Creador (iss):** Identifica a quien creo el JWT
- **Razón (sub):** Identifica la razón del JWT, se puede usar para limitar su uso a ciertos casos.
- **Tiempo de expiración (exp):** Una fecha que sirva para verificar si el JWT está vencido y obligar al usuario a volver a autenticarse.
- **No antes (nbf):** Indica desde qué momento se va a empezar a aceptar un JWT.
- **Creado (iat):** Indica cuando fue creado el JWT.

Nota: Todos los claims son opcionales, por lo que no es obligatorio utilizar todos los claims registrados. Y se recomienda usar el mínimo necesario de claims.

# Explorando JWT Debugger

<https://jwt.io/>

Algorithm

## Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

## Decoded

EDIT THE PAYLOAD AND SECRET

### HEADER: ALGORITHM & TOKEN TYPE

Type of token {  
"alg": "HS256",  
"typ": "JWT"  
}

### PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

### VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
) ☐ secret base64 encoded
```

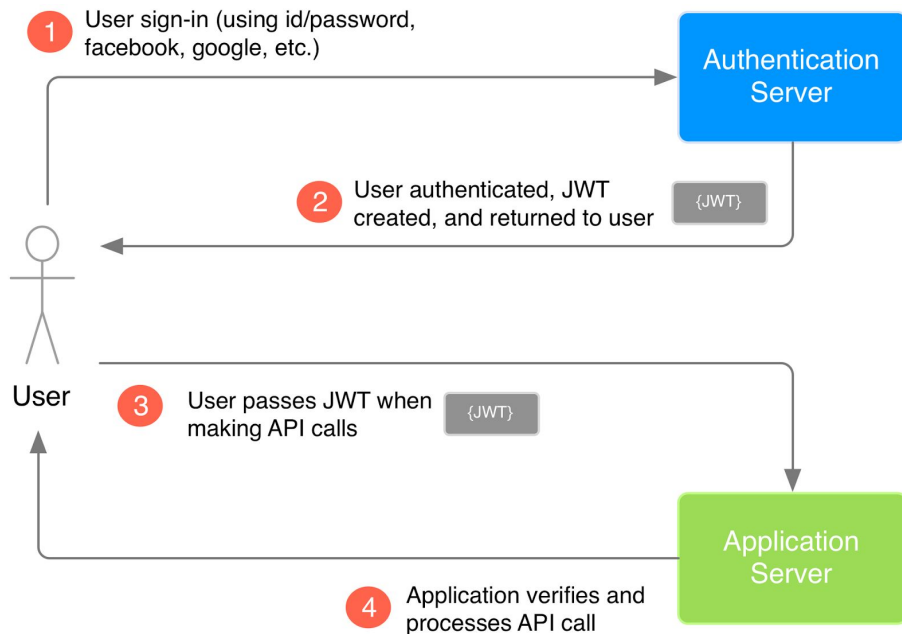
Herramienta auxiliar: <https://www.base64encode.org/>

# Esquema de Authentication con JWT

En la autenticación, cuando el usuario inicia sesión correctamente con sus credenciales, se devolverá un token web JSON.

Dado que los tokens son credenciales, se debe tener mucho cuidado para evitar problemas de seguridad. En general, no debe conservar los tokens más tiempo del necesario.

Tampoco debe almacenar datos de sesión confidenciales en el almacenamiento del navegador debido a la falta de seguridad.





## ¿Cómo trabaja?

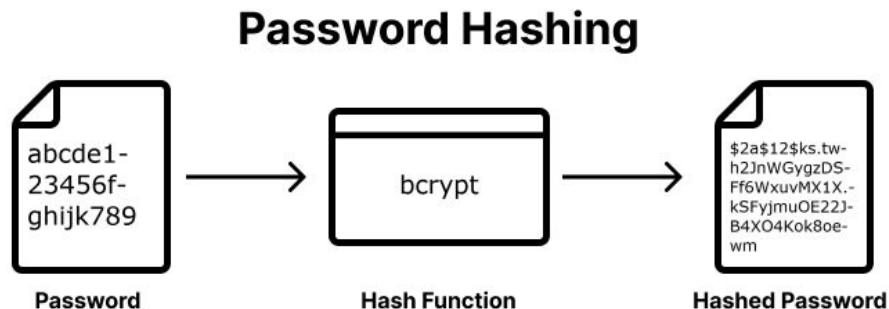
Cada vez que el usuario desee acceder a una ruta o recurso protegido, el agente de usuario debe enviar el JWT, generalmente en el encabezado de Autorización utilizando el esquema “**Bearer**” que es una petición http. El contenido del encabezado debe ser similar al siguiente:

```
Authorization: Bearer <token>
```

# Hash Passwords

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



## ¿Qué es HASH?

Una función criptográfica hash- usualmente conocida como “**hash**”- es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija.

Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud.



# HASH y las contraseñas

El almacenamiento seguro de contraseñas es un punto conflictivo para la seguridad del sistema. **En lugar de almacenar las contraseñas directamente, se recomienda utilizar hashes criptográficos.** Esto permite la validación de contraseñas sin revelar la contraseña real, lo que dificulta que los atacantes obtengan acceso incluso si obtienen los datos almacenados.


Dado que el hash es una operación unidireccional, los atacantes tendrían que recurrir a ataques de fuerza bruta para descifrar las contraseñas, lo que requiere mucho tiempo y recursos.

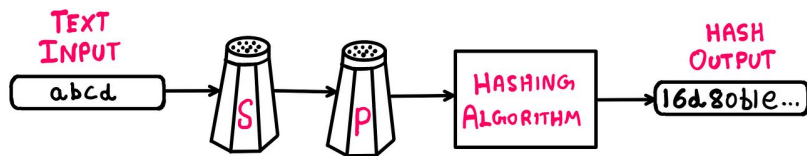


## Salt (Sal)

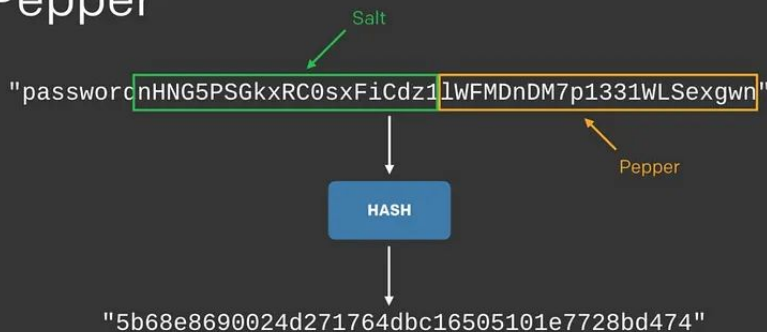
La sal es un conjunto de datos aleatorios que se añade a la contraseña antes de aplicar la función de hashing.

**El propósito de la sal es asegurar que, incluso si dos usuarios tienen la misma contraseña, los resultados del hash sean diferentes.** Esto se debe a que cada usuario tendría una sal única que se añade a su contraseña antes de aplicar el hash. La sal ayuda a proteger contra ataques de diccionario y ataques de tabla arcoiris, donde los atacantes precalcular los hashes de contraseñas comunes.

				
Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	1vn49sa	z32i6t0



## Pepper



## Pimienta (Pepper)

### Clave Secreta / Secreto

La pimienta es similar a la sal en que también es un dato añadido a la contraseña antes del hashing. Sin embargo, **a diferencia de la sal, la pimienta no se almacena junto con el hash en la base de datos.** Por lo general, se guarda como un secreto en el código de la aplicación o en una configuración segura.

**La pimienta es la misma para todas las contraseñas** y su objetivo es añadir una capa adicional de seguridad, ya que un atacante que tenga acceso a la base de datos aún necesitaría conocer la pimienta para generar hashes válidos.





# ¡GRACIAS!



César Guerra



[www.cesarguerra.mx](http://www.cesarguerra.mx)