

# JWT: En qué consiste

Implementa la seguridad de tu API Rest con Spring Boot

# JWT

- JSON Web Token (RFC 7519)
- Es un mecanismo para propagar de forma segura la identidad (y *claims* o privilegios) entre dos partes.
- Los privilegios se codifican como objetos JSON.
- Estos objetos se usan en el cuerpo (*payload*) de un mensaje firmado digitalmente.

# Token JWT

- Se trata de una cadena de texto con 3 partes *codificadas* en Base64
- Las partes están separadas por un punto
  - eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.ikFGEvw-Du0f30vBaA742D\_wqPA5BBHXgUY6wwqab1w
- Las partes están separadas por un punto
- Podemos utilizar un *debugger* online para decodificarlo
  - <https://jwt.io/#debugger-io>

# Token JWT

- Pero... ¿si la información se ve! *¿Esto es seguro?*

The diagram illustrates the encoding process of a JWT token. It is divided into two main sections: **Encoded** and **Decoded**.

**Encoded:** Shows the final encoded JWT token as a single long string of characters, including letters, numbers, hyphens, and underscores.

**Decoded:** Shows the token's structure and payload. It is divided into three parts:

- HEADER:** Contains the algorithm used for signing (e.g., "HS256") and the token type (e.g., "JWT").
- PAYLOAD:** Contains the data being transmitted, such as user information (e.g., "username", "password", "role") and expiration time (e.g., "exp").
- Signature:** The result of applying the algorithm to the header and payload, used to verify the token's integrity.

The diagram also shows the process of decoding the token, where the encoded string is split into its three parts and then decoded back into a readable format.

# Token JWT

- Hemos dicho que tiene 3 partes
  - *Header*: indica el algoritmo y el tipo de token (HS256 y JWT)
  - *Payload*: datos del usuario y privilegios
    - Como nosotros generamos el token, podemos incluir todos los datos que estimemos convenientes.
  - *Signature*: firma para verificar que el token es válido (aquí radica el *quid* de la cuestión).

# Firma de un token JWT

- La firma se construye de tal forma que podemos verificar que el remitente es quien dice ser, y que el mensaje no ha cambiado por el camino.
- Se construye como el HMACSHA256 de
  - Codificación en base64 de *header*
  - Codificación en base 64 de *payload*
  - Un secreto (establecido por la aplicación)

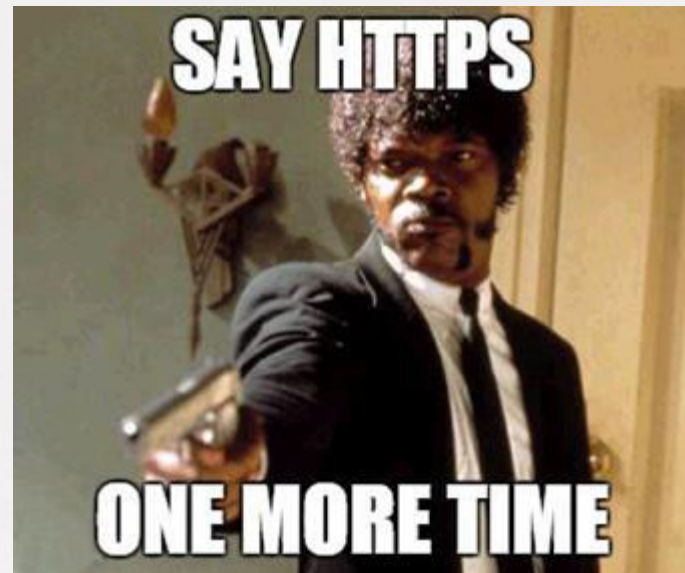


# Firma de un token JWT

- Si alguien modifica el token por el camino:  
...iOiJKV1QiLCJhbGci... a ...iOiRFH1QiLCJhbGci...
- La comprobación de la firma no será correcta
- No podemos confiar en el token recibido, y deberíamos denegarlo.
- Siempre debemos verificar la firma de un token recibido.

# Token JWT seguro

- Con todo, el *header* y *payload* no están cifrados, solo codificados en base64.
- Esto nos invita a pensar que toda comunicación que hagamos debería ser con HTTPS para encriptar el tráfico.
- *En el fondo, siempre deberíamos utilizar HTTPS y un servidor con certificado.*





# Ciclo de vida de un token JWT

