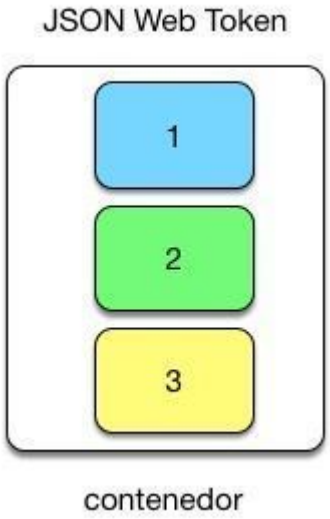
El concepto de JSON Web Token es cada día más común en el desarrollo de aplicaciones web .¿Qué es un JSON Web Token y como funciona?. Un JSON Web Token es un contenedor de información referente a la autenticación de un usuario.

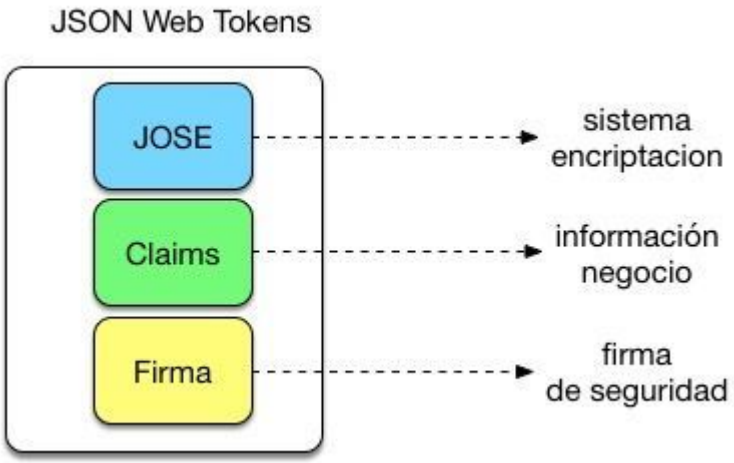


Vamos a verlo un poco más a detalle.

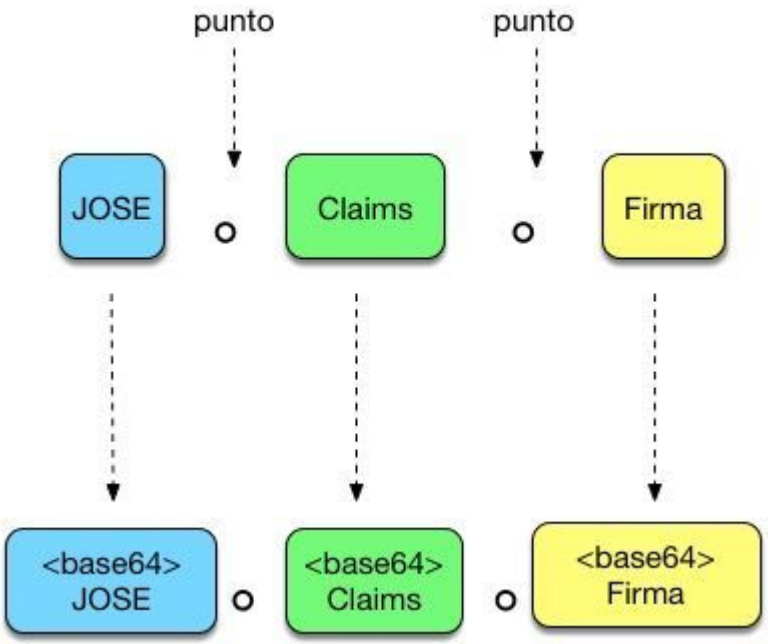
¿Cual es la estructura de este contenedor?

Un JSON Web Token esta dividido en tres partes .

1. La primera es la que se denomina JOSE o JavaScript Object Signing and Encryption y define cual es la tecnología criptográfica que se va a aplicar al token para securizar la información.
2. La segunda parte es lo que se denomina JWT PayLoad o JWT Claims y almacena la información de negocio que necesitamos en el token. Esta parte se puede estructurar de muchas formas.
3. La tercera parte es la firma JWT que se encarga de dar validez al token.



Cada una de las partes esta codificada en Base64 y separadas por un punto.



Esto está muy bien pero es difícil de entender como funciona. Vamos a intentar clarificar los conceptos usando un ejemplo de un token con sus partes.

JSON y JOSE (I)

Empezaremos por la parte JOSE:

```
{
"alg": "HS256",
"typ": "JWT"
}
```

Aquí vemos como definimos el tipo de token y el algoritmo criptográfico que vamos a utilizar (HS256). ¿Esto que quiere decir exactamente? . En nuestro caso quiere decir que estamos usando un token JWT (el más común) y un algoritmo de HASH muy concreto denominado HMAC que genera un hash con SHA256 utilizando una clave privada. Si no hemos entendido la frase no hay que preocuparse, es normal, más adelante lo explicaremos . Veamos la segunda parte del token.

JSON Web Token y Claims

Este bloque define la información que vamos a almacenar de negocio . En nuestro caso únicamente almacenaremos el nombre del usuario:

```
{
"nombre": "Cecilio"
}
```

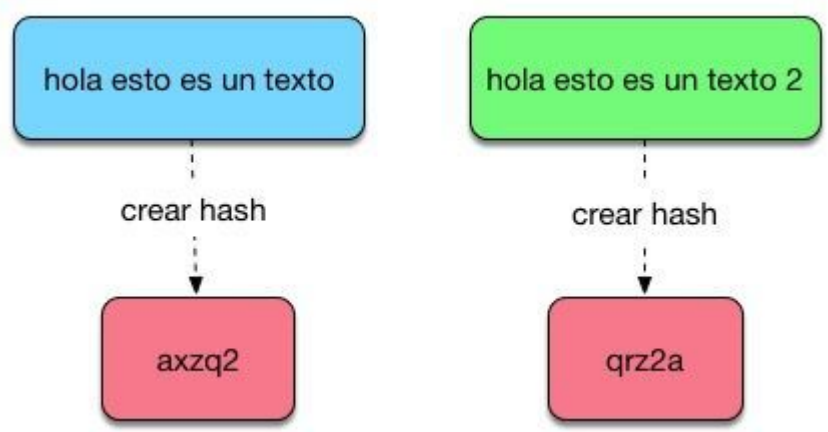
JSON y Firma

La tercera parte del token simplemente es un hash con la siguiente estructura:

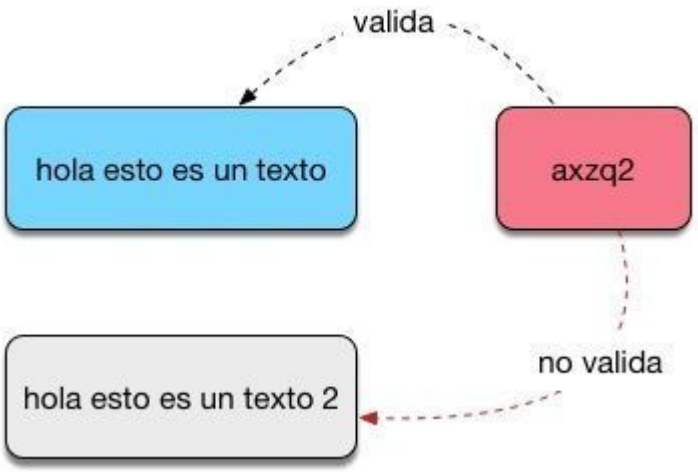
HMAC(base64UrlEncode(jose)+”.” +base64UrlEncode(claims),clave)

Criptografia y JSON Web Token

Para entender JWT necesitamos repasar algunos de conceptos criptográficos . Lo primero que tenemos que entender es el concepto de algoritmo de HASH. Un algoritmo de HASH se encarga de generar un HASH (bloque de caracteres de longitud fila) a partir de una cadena arbitraria de texto.

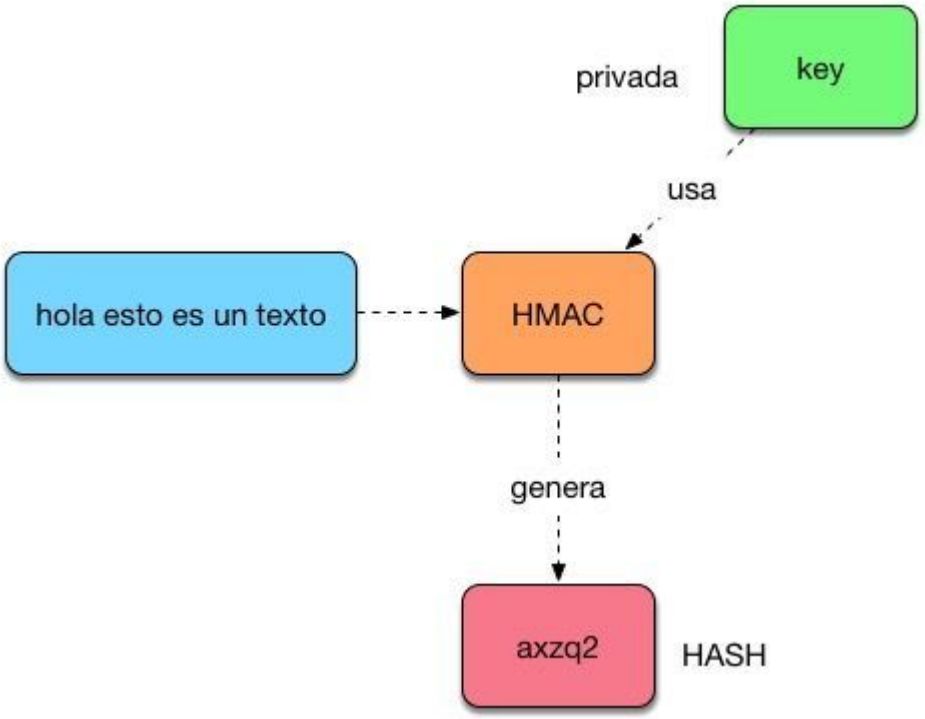


Los algoritmos de hash sirven para comprobar que en ningún momento se ha modificado el texto original ya que se aseguran de que ante dos textos distintos siempre se genera un hash diferente. Por lo tanto si alguien nos cambia el texto original y lo intenta dar por valido podemos regenerar el hash y comprobar si cumple.

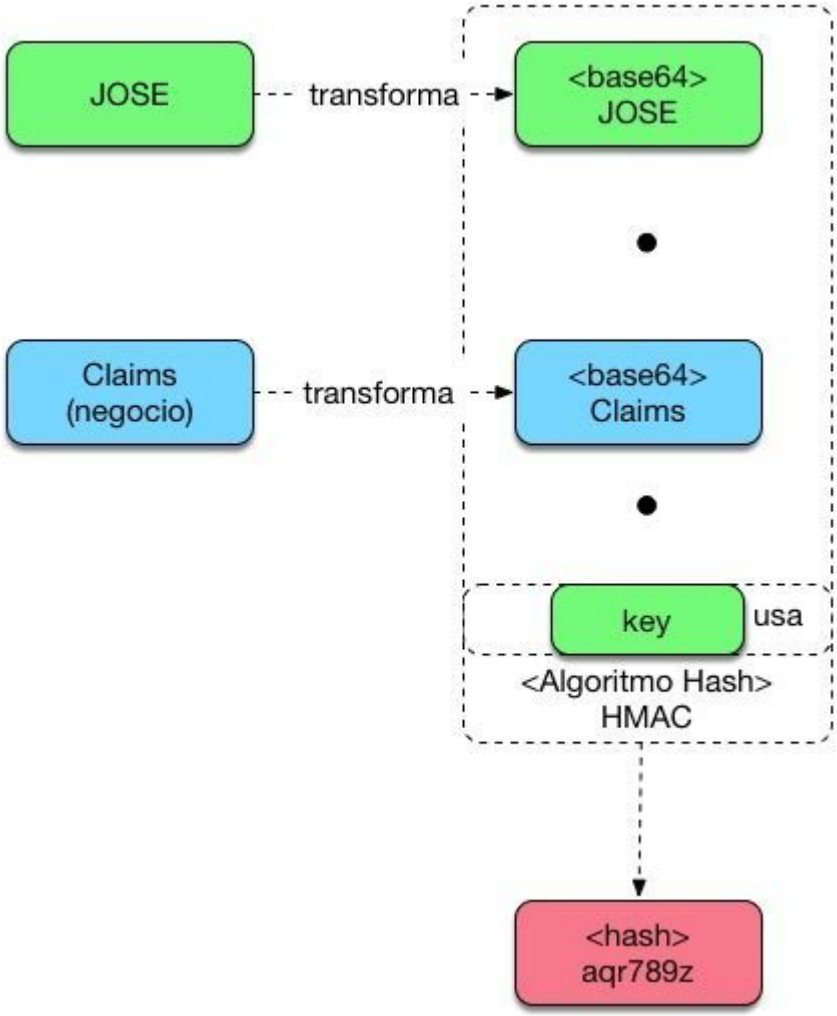


JSON Web Token y HMAC

¿Qué es lo que tiene en especial usar un algoritmo HMAC para generar el hash?. Lo que tiene de especial es que este algoritmo criptográfico se encarga de generar un hash para un texto utilizando una clave privada.

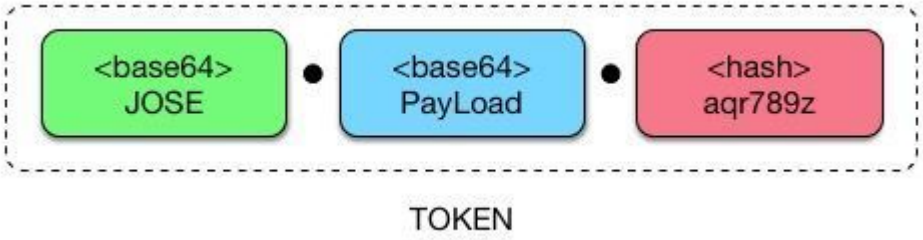


Esto implica una diferencia importante ya que los HASH solo los pueden generar aquellas personas que conozcan la clave privada. Por lo tanto no solo sabemos ahora que el contenido no ha sido modificado sino también podemos saber quien es su creador. Vamos a profundizar un poco mas con este diagrama:



El diagrama explica como se genera el HASH

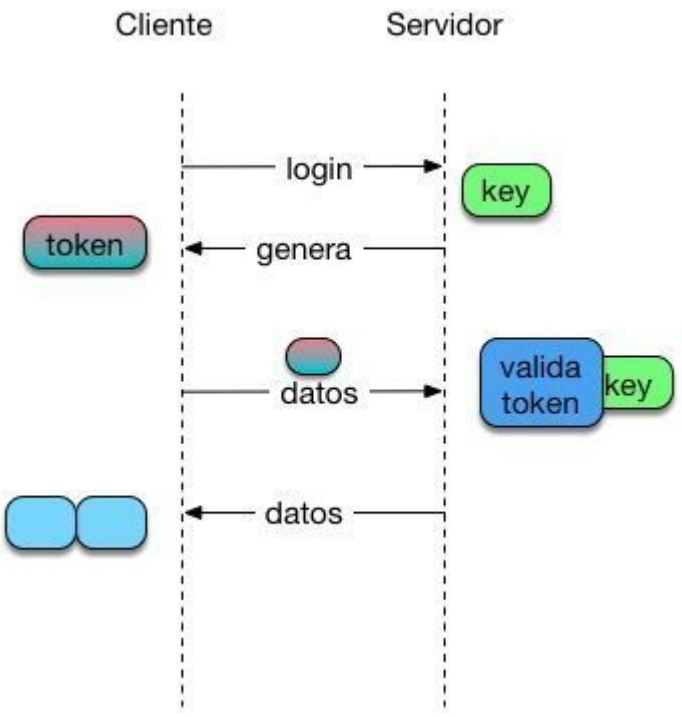
1. En primer lugar generamos las estructuras de base64 tanto de la parte JOSE como de los Claims
2. En segundo lugar usaremos el algoritmo HMAC con su clave privada para generar un HASH basado en las estructuras de Base64.
   Realizadas ambas operaciones el último paso es sumarlo todo y generar un token:



Este token será el que enviemos al cliente y le permitirá autenticarse más adelante.

JWT y Servidor

La forma de procesar los tokens JWT esta ligada al servidor pero básicamente es algo del estilo:



1. El cliente se logea en el servidor y envía usuario y clave
2. El sistema le valida y genera un token usando el algoritmo HMAC y la clave privada

3. El cliente recibe el token
4. El cliente solicita unos datos y pasa como identificador el token
5. El servidor decodifica los bloques de base64 y usa su clave privada para comprobar el HASH .Si todo es correcto permite el acceso y envía la información solicitada al cliente.
   Estos son los conceptos fundamentales detrás de JWT. A continuación se muestra un ejemplo de Token:

   eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJub21icmUiOiJjZWNpbGlvIn0.MCfaorSC7Wdc8rSW7BJi9xuJb7G8RQzasfzsm_y-COI

   Podemos ver los puntos que separan cada parte del token y si queremos podemos decodificar los dos primeros bloques.En el siguiente artículo veremos como construir una implementación sencilla de JWT.

   Otros artículos relacionados

1. REST JSON y Java
2. JAX-RS Client y JSON
3. Java Security y anotaciones JAAS