Claims en JWT

Transcribed on July 10, 2025 at 10:57 AM by Minutes AI

Speaker 1 (00:05)

Bienvenidos a esta nueva sesión.

En esta sesión vamos a proponer un ejercicio para trabajar con los JSON Web Token y su validación utilizando librerías.

Para ello tenemos que conocer bien los claims y la importancia que estos tienen y por qué son fundamentales en la seguridad y en la autenticación de aplicaciones web y API.

En primer lugar, los claims son piezas de información que pueden incluirse en un JSON Web Token.

¿Pero, por qué son tan importantes?

Para empezar, los claims desempeñan un papel crucial en la representación de la información del usuario.

Permiten transmitir datos como el nombre de usuario, los roles, los permisos y otros atributos relevantes de manera segura y estructurada.

Esto es esencial para que las aplicaciones puedan acceder a la información necesaria para personalizar la experiencia del usuario y autorizar el acceso a los recursos protegidos.

Como segundo punto, los claims son la base de la autorización en las aplicaciones web y en las API.

Permiten a los servidores validar y restringir el acceso a los recursos basados en los roles y en los permisos del estado.

Imagina por un momento un sistema donde los claim no estén presentes.

Sería extremadamente difícil determinar quién tiene acceso a qué recursos, lo que abriría la puerta a posibles vulnerabilidades de seguridad.

En tercer lugar, los claims contribuyen a la seguridad general del sistema.

Al estar incluidos en el cuerpo del token y estar protegidos por una firma digital, garantizan que la información transmitida sea confiable y además, que ésta no haya sido manipulada durante la transmisión.

Esto proporciona una capa adicional de seguridad, asegurando que únicamente aquellos usuarios autorizados pueden acceder a los recursos protegidos.

Ahora vamos a profundizar entre los diferentes tipos de claims y cómo se utilizan en el contexto de los tokens JWT.

Los tipos de claims nos permiten transmitir información de manera segura y eficiente, adaptándonos a las necesidades específicas de cada aplicación.

Comencemos hablando de los claims registrados.

Estos son aquellos que están definidos en la especificación de Codewc y proporcionen información estándar y necesaria para la autenticación y autorización.

Los ejemplos de estos claims el emisor, el sujeto audiencia, la fecha de emisión, la fecha de expiración y la fecha de inicio de validez y también en algunas ocasiones, el identificador único del token.

Ahora después vamos a mencionarlos con detalle, pero a nivel general, estos clients proporcionan una base sólida para la implementación de la seguridad en nuestras aplicaciones.

Pasemos ahora a los claims públicos.

A diferencia de los registrados, estos no están definidos en la especificación de JWT y son específicos para cada aplicación.

Ejemplos de clink público podrían ser el nombre de usuario, su dirección de correo electrónico, distintos roles específicos de la aplicación en cuestión o cualquier otro dato que sea necesario para el funcionamiento de nuestra aplicación, pero que no esté estandarizado para que no existan colisiones.

Estos claims deben ser listados en el registro y de Jsonware Token.

Aquí podemos ver claims que utilizan otras entidades, como por ejemplo la Fundación OpenID.

Por último tenemos los claims privados.

Estos son personalizados y específicos para una aplicación o servicio en particular.

No están definidos en la especificación de JSO Web Token ni son de uso común en otras aplicaciones.

Estos claims se utilizan para información específica que no necesariamente necesita ser compartida con otros, si esto nos permite mantener cierta información de forma privada y segura dentro del token JWT.

En resumen, los diferentes tipos de claims en JWT nos proporcionan una gran flexibilidad y personalización en la transmisión de información dentro de nuestros token.

Nos permiten encontrar un equilibrio entre la estandarización necesaria para la interoperabilidad entre sistemas y la adaptabilidad a las necesidades específicas de nuestra aplicación.

Para una gestión básica como son los controles de roles o tiempo, utilizar los claims registrados es suficiente.

Vamos a ver estos claims en detalle.

Comenzamos con el claim del emisor, que está representado por issdisware.

Este claim indica la entidad que emitió el token jwt, por ejemplo, vendría a ser la URL de la aplicación, por ejemplo mi aplicación.

Este claim es esencial para que los sistemas detectores puedan validar la procedencia del token y asegurarse que proviene de una fuente confiable.

Otro claim registrado es el sujeto, que está representado por sub de Shapier.

Este cliente identifica el sujeto del token, que generalmente es el usuario al que se refiere el token.

Esto podría ser un identificador interno único del usuario, como un ID que tengamos en nuestra base de datos, por ejemplo 1234.

Este claim es crucial para la personalización y la autorización de las solicitudes basadas en el usuario que está realizando la solicitud.

Continuamos con el claim de audiencia, que estaría representado por Aud.

Este claim indica para qué audiencia está destinado el token.

Podría ser una URL específica de nuestra aplicación o bien un array de valores que representan varias audiencias.

Por ejemplo, esto podría ser mi aplicación com o bien un array de miaplicación com y mi aplicación o otra aplicación com.

Este claim permite que los sistemas receptores verifiquen si el token está destinado a ellos o si tienen el permiso para utilizarlos.

El siguiente claim que vamos a ver sería la fecha de expiración, representado por expiration time.

Este claim indica el momento en el que expira la Validez del Token, y estaría representado en segundos desde la época Unix, es decir, haciendo uso de el Unix Timestamp, por ejemplo, esto podrá ser un valor entero que cumpla con esta Condición del Este claim nos garantiza que los tokens tengan una vida útil limitada y que no se puedan utilizar indefinidamente, lo que aumenta la Seguridad del Sistema.

Otro claim que tenemos parecido aquí sería la fecha de inicio de Validez del Token, representado por nbfdnotv.

Este claim indicaría el momento a partir del cual el token es válido, que también está representado en segundo desde la época Unix, es decir, en Unix Time.

Este clean permite que los sistemas receptores especifiquen un momento en el futuro a partir del cual el token se puede considerar válido, lo que proporciona una mayor flexibilidad en la gestión de la validez de los tokens.

Luego tenemos también la fecha de emisión, representado por IAT de eastweb App.

Este claim indica el momento en el que se inició token, que también está representado en segundos desde la época Unix.

Este claim es esencial para verificar la frescura del token y determinar si ha sido emitido recientemente o no.

Por último, añadimos el claim del identificador del propio JSON web token, que está representado como jti de JSON web tokenid.

Este claim proporciona un identificador único para token, por ejemplo, podría ser un IDE aleatorio generado por el propio servidor al emitir el token.

Este claim es útil para evitar la repetición de tokens y para realizar un seguimiento de los tokens emitidos, lo que mejora la seguridad y la gestión de las sesiones en nuestra aplicación.

Por ejemplo, se podría hacer una lista negra e ir añadiendo aquí los tokens que se quieran revocar por alguna razón, pero que a nivel de tiempo serían válidos, es decir, que no han expirado todavía.

En resumen, los claims registrados en JWT proporcionan una base sólida para la autenticación y la utilización en nuestros sistemas.

Son esenciales para verificar la procedencia, la frescura y la validez de los tokens JWT.

Conociendo ya los diferentes tipos de claims que tenemos y habiendo visto en detalle los claims registrados, es hora de poder trabajar con los JSON Web Token a través del siguiente ejercicio para la realización de esta actividad práctica vamos a trabajar con la generación de tokens utilizando una librería específica, por lo que debemos elegir un lenguaje de programación.

El objetivo de la actividad es entender cómo funcionan los JSON Web Token, cómo se generan y cómo se validan.

Además, también tenemos que conocer cómo se deben configurar los frames específicos para para controlar la expiración y la validez del token.

Para ello, el primer paso será instalar una librería que nos permita trabajar con jwt y que lo hagamos en un lenguaje de programación con el que estemos cómodos.

La recomendación en este punto es utilizar Python con la librería py jwt, ya que está bastante documentada y es relativamente sencilla de trabajar.

Para instalarlo podemos utilizar pip con el comando pininstall pyjwp.

Una vez que hemos instalado la librería, procederemos a generar un token utilizando la librería recién instalada.

Podemos realizar diferentes pruebas para asegurarnos de que el token se genera correctamente y que funciona todo como se espera.

Y ahora viene la parte más interesante.

Vamos a configurar claims específicos para controlar la inspiración y la validez del token, pero vamos a hacerlo acorde a los siguientes el token no va a ser válido hasta 5 min después de su emisión.

Por otro lado, el token no será válido después de 30 min desde su emisión.

Tendrás que revisar con qué claims podemos hacer esta configuración y además de esos claims, podemos explorar otros como la identificación del sujeto o la identificación del token, el JT que hemos comentado anteriormente.

Con esto lo que vamos a hacer va a ser entender un poco mejor el funcionamiento y la importancia dentro de la seguridad de nuestros tópicos.

Espero que resulte de interés la actividad y que logres realizar el ejercicio correctamente.

Y hasta aquí la sesión explicativa de este planteamiento de ejercicio.

Os esperamos en el siguiente vídeo.