JWT Claims Exercise

Transcribed on July 10, 2025 at 12:25 PM by Minutes AI

Speaker 1 (00:04)

Bienvenidos a esta nueva sesión.

En esta sesión vamos a dedicarnos a corregir el ejercicio propuesto acerca de los claims en jwt.

Para la realización de este ejercicio se requería instalar en primer lugar una librería compatible con jwt que podemos encontrar en el sitio web de referencia.

Dependiendo del lenguaje de programación elegido podríamos utilizar una librería u otra.

Para el ejercicio se había propuesto utilizar la librería py jwt y el lenguaje de programación de python.

Y en primer lugar, tras instalar la librería, si hemos utilizado esta, la recomendación era hacerlo a través del comando pip con pipinstall py jwt.

Una vez que hemos instalado la librería, la idea era empezar a jugar con la librería y probar a generar un token simple y también comprobar su funcionamiento y cómo somos capaces de proceder a hacer una validación de los token que estamos generando.

Esto cualquier tipo de librería que trabaje con jwt estará preparada para hacerlo directamente pasándole simplemente unos pocos parámetros.

Después de haber jugado un poco con la librería y tener claro cómo es el manejo de la misma y vamos a empezar a trabajar con los claims registrados de tal manera que pudiéramos controlar la validez del token en función del tiempo.

Concretamente necesitábamos cumplir con dos requisitos.

Por un lado es que el token no fuera válido hasta 5 min después de que este token hubiera sido generado y además también la otra condición es que el token no fuera válido después de 30 min desde que el token fuera generado.

Esto quiere decir que el token va a ser válido durante 25 min que empezarán a contar a partir de los 5 min desde la emisión de dicho token.

Además, para poder entender un poco más esto de los claims, se recomendaba explorar algunos como podría ser el subject o la identificación del JWT.

Por supuesto que a nivel del servidor será éste el que se encargue de hacer las validaciones correspondientes relativas a quien lo está mandando o la identificación del propio token del JWTID.

Vamos entonces a pasar a un editor de código a poder realizar la práctica y ver paso a paso cómo podemos resolverla.

Antes de irnos al editor de código vamos a repasar cómo hacer la instalación de la librería y dónde encontrar toda la documentación.

En primer lugar estoy en la página jwt IO y lo primero que quiero hacer es irme directamente a esta sección que pone introducción, que de hecho también la podemos encontrar directamente en el botón principal que más tenemos destacado aquí en la página principal fijaos que desde el principio nos indican que hay una especie de libro disponible para aprender más sobre JSON Web tokens, que básicamente es una guía de referencia para aprender todo lo que está apareciendo aquí a nivel de introducción pero con mucho más nivel de detalle.

Fijaos que aquí básicamente lo que nos está indicando es una introducción a que se logra, qué nos van a servir o cuándo lo deb utilizar, cuál es la estructura.

Vemos claramente como tenemos el header, el payload y la firma y cómo tiene esa forma de que están separadas por puntos, siendo en primer lugar la cabecera, luego la carga útil y luego la firma.

A nivel de la cabecera, a nivel recordatorio de lo que lleva la cabecera se nos indica el tipo de algoritmo, en este caso en el ejemplo hs y el tipo jwt.

Entonces esto luego es modificando en base 64 y formaría la primera parte de las x, eso sería la cabecera, esta primera x para la carga útil y aquí es donde tenemos que prestar especial atención, tenemos los distintos tipos de clients, aparte de poner todo tipo de datos de información que queramos añadir.

Esto nos interesa porque se si nos venimos aquí y le pinchamos en el enlace se nos va a ir directamente al RFC de los JSON Web token donde podemos ver la información de cada uno de ellos, en este caso el issuer, el subject, la audiencia, el tiempo de expiración que no sea válido antes de cuando ha sido creado ese token y luego por último el id del propio JSON web token por si queremos identificar un token precisamente para luego por ejemplo añadirlo a un listado de lista negra por ejemplo para poder revocarlo.

Luego de hecho también se nos indica aquí que tendríamos los tipos públicos por así decirlo y los privados que son los que crearemos nosotros.

De hecho volviendo para atrás podríamos ver el registro y Ana de los Jason Webtoon donde se nos indica cuáles son los claims que se están utilizando, por ejemplo OpenID estaría utilizando pues el nombre, family name, given name, middle name, nickname, etc.

Todo esto que vemos aquí dependiendo tenemos distintas entidades que están y por último aquí también tendríamos la parte de la firma que pues ya vemos que dependiendo del tipo de algoritmo que estemos eligiendo le estamos indicando h que sea 256.

Tendríamos la primera parte que es lo que veríamos aquí arriba del todo, lo que se correspondería con esto literal relleno en base 64 y las z en este caso comprendería la firma que sería de pasarle el hma xia de esta primera parte y aplicarle nuestro secreto, que el secreto también podrá ser puesto luego en base 64 para utilizar el secreto como base 64 en vez de utilizar normal.

Esto sería ya la forma que tendría un token y que deberíamos validar.

Para validarlo podríamos hacer uso directamente del debugger o en este caso lo que vamos a hacer es uso de las librerías.

Vamos a filtrar por librerías para y aquí nos encontramos con cuatro librerías distintas y bueno pues cada una de ellas nos dice qué es lo que va a comprobar.

En este caso vamos a utilizar esta primera que es la más extendida y vamos a ver la documentación, pero fijaos que no tiene validación.

Si queremos aplicar el Claim sub o el del Identificador del JSON Web Token, estos dos no lo tienen.

A nivel de algoritmo de cifrado sí que son compatibles con todos, por ejemplo este otro de aquí pues no cubriría todos los algoritmos de cifrado y por ejemplo este Jotra WC Crypto pues vemos que tiene todo pero por alguna razón a nivel de popularidad dentro de la plataforma de GitHub es un recurso menos apoyado.

Por último también tendríamos el outlib que vemos que es compatible con prácticamente con todos menos con un tipo concreto de issuer.

Así que vamos nosotros a instalar este que será simplemente indicando bit install py jwt pero antes de instalarlo vamos a visitar el repo.

A nivel del repo pues aquí tenemos un poco la documentación más rápida pero podemos ver la documentación con todo tipo de detalle.

Para instalarlo es tan sencillo como hacer y pins pyjwt, pero si revisamos las opciones de instalación dependiendo el tipo de algoritmo que estemos utilizando, por ejemplo si queremos utilizar RSA o ECDSA pues necesitaremos instalar las dependencias de las librerías criptográficas y por tanto se nos indica que lo deberíamos instalar de esta manera.

Nosotros vamos a hacer instalación con Pitinstall para jwt y luego vamos a proceder a venirnos a la sección de ejemplos para probar la parte más sencilla.

A nivel de código estaríamos importando la librería, vamos a definir aquí la clave o el secreto que vamos a querer utilizar y luego para poder utilizar directamente jwt lo que tenemos que hacer es llamar un método llamado encode que está dentro de esta lista jwt, le vamos a pasar aquí lo que sería la carga útil, por eso vemos aquí son payload y la clave y también el tipo de algoritmo, en este caso h.

Si esto lo imprimimos nos va a devolver directamente el token.

¿Qué pasa?

Que si utilizamos por ejemplo el decode con el texto que hemos conseguido con este token y le tenemos que indicar el algoritmo, pues vamos a ver que nos lo decodifique.

Aquí es interesante forzar el cambio de algún tipo de parámetro aquí arriba para que podamos trabajar y ver los errores que aparecen.

Así que vamos a empezar primero viendo esta parte de la librería y luego pasamos a utilizar los claims de expiración para que no sean válidos antes de un tiempo.

Pasamos aquí al Visual Studio Code, ya tengo preparado un fichero y luego que hago es hacer el bit install by jw.

Vamos a instalarlo, ya lo tendríamos, vamos por ir más rápido, pues directamente esta parte que la podría copiar y vamos a adaptarlo aquí un poco.

Vamos a poner primero, vamos a hacerlo bien hasta aquí y también nos quedaría el Decode, que lo vamos a poner así.

Entonces vamos a limpiar la p final y vamos a poner Python Example JW.

Bueno, como vemos este ejemplo no hemos tenido ningún tipo de problema.

Ahora lo que quiero hacer es lo siguiente, ya hemos visto que este es nuestro token.

¿Qué pasa si yo ahora digo, oye, pues quiero encode?

¿Le vamos a poner uno falso, vamos a forzar el fallo y es qué pasa si aquí modifico cualquier tipo de valor?

Aquí, bueno, meto un seis, vamos a poner, bueno, lo vamos a poner mejor aquí en la carga útil, suponiendo que hemos modificado algún tipo de dato de aquí, que esto sería tan sencillo como cambiar algo aquí, por ejemplo así, y pasar esto a base 64, que es lo que estaría haciendo por debajo.

Pero bueno, vamos a simular que estamos cambiando algún tipo de dato aquí.

¿Claro, qué va a pasar?

Que al final esta firma ahora mismo nos va a fallar porque no se corresponde con la misma firma que esto, pero bueno, vamos a verlo.

Vamos a convertir esto a String y le digo que quiero hacer el Decode del Facebook y lo lanzamos.

Bien, lo que obtenemos básicamente es una excepción.

Vemos aquí que nos está indicando que la firma, la verificación de la firma ha fallado, así que lo que tendría que hacer para que esto estuviera haciendo bien sería poner aquí un drive cat.

Si yo pusiera esto y aquí ponemos un excep, ya lo vamos a controlar, luego lo ideal sería controlar qué tipo de excepción es.

En este caso estamos viendo que es esta excepción, así que en el caso de que sea esta excepción que se nos imprima precisamente para que no nos quede el código feo y se nos salga sin verlo, bueno, podemos poner el mismo mensaje, pero en este caso vemos que si hubiera algo por aquí debajo el código no se saldría, no se bloquearía el código, no nos forzaría Alex.

Entonces lo siguiente que tenemos que hacer, ya sabiendo que tenemos esto, si vemos la documentación, aquí tenemos una sección de ejemplos en los cuales se trabaja con los claims registrados y aquí, bueno, nos hacen un poco con los que trabajan aquí con el tiempo de expiración con que no sea válido antes de, el issuer, la audiencia y cuando ha sido lanzado.

¿Entonces si nos fijamos aquí decimos oye, cuánto queremos que expire?

¿Por ejemplo, está poniendo un valor concreto o bien está haciendo uso de la librería datetime para poder indicarle, bueno en este caso estamos poniendo el tiempo de ahora, qué pasa?

Nos va a dar un error porque aquí nos dice que al igual que antes, aquí estábamos consiguiendo un error, una excepción de firma inválida, luego podemos conseguir otro tipo de excepciones, en este caso una excepción de que ha expirado o si bajamos un poco más, por ejemplo que no sea válido antes de, pues ocurrirá lo mismo, si tenemos algún tipo de error pues va a decir que ha expirado o que éste todavía no es válido, que es con lo que nosotros vamos a trabajar.

Así que lo que vamos a hacer va a ser aquí importar una librería nativa de python que es datetime, aquí la tenemos, lo que vamos a hacer va a ser definir ahora unas variables específicas para la expiración y la validez del token, así que le voy a decir, le vamos a poner datetime, datetime utcnow, vamos a llamar a esta función, esta función lo que nos está devolviendo es el tiempo que tenemos actualmente.

Por otro lado vamos a poner la expiración, en este caso vamos a sumarle un delta, vamos a utilizar este valor que es el valor que va a coger actual cuando se ejecute el código y vamos a sumarle por aquí un time delta, en este caso para la fecha de expiración con 30 min y esto mismo lo vamos a poner para not before time, vamos a poner lo mismo de antes, la de time timedelta, pero lo que queremos ahora son 5 min, es decir, que después de 5 min sea válido.

Vamos a definir la carga útil aquí, para ello voy a crear un diccionario, voy a poner claims y quiero aquí definir, lo vamos a definir el claim iat, que era como se llamaba cuando ha sido puesto, este me está fallando, aquí sería, vamos a poner también la fecha de expiración que la tendríamos aquí y vamos a poner también not valid y fortnite, aquí lo tendríamos si quisiéramos añadir más claims, que luego tendríamos también que seguir validando, pero podríamos añadir por ejemplo el issuer, por ejemplo si tuviésemos aquí algún tipo de aplicación, por ejemplo tendríamos que ponerlo así, nuestro sitio web o cualquier cosa, si quisiésemos también indicar por ejemplo un usuario concreto, pues podríamos poner el sup y al sup asignarle algún tipo de valor, imaginemos 123-456-7890 suponiendo que este dato lo sacaríamos de una base de datos y que se corresponda con el usuario, es decir, cuando un usuario se autentica se le está generando el token con esta información que va a ser para él e incluso también podríamos asignar un identificador único.

Hay que tener en cuenta que esta librería, por lo que hemos estado viendo, no nos va a validar directamente este id, es una librería que no es válida con esto.

Entonces esta parte lo tendríamos que hacer nosotros a mano, pero que sepáis que lo podríamos añadir algo de este estilo, por ejemplo, como ven aquí tenemos o lo generamos con un random, por ejemplo aquí tenemos o lo generamos con un random o cualquier cosa.

Ahora estamos definiendo, aquí tenemos la parte de la clave y lo que vamos a hacer va a ser generar el token para que esto sea mejor vamos a poner token igual y vamos a poner el jwt code, a ver si me lo va a pillar bien.

Aquí vamos a poner los claims, vamos a ponerle también la clave y por último pues el algoritmo con el que ya hemos estado trabajando.

Entonces si esto, si ahora no nos hemos equivocado en nada, podremos hacer aquí la impresión de el token.

Vamos a poner por ejemplo aquí, vamos a poner format string token, jwt generado y vamos a poner el token.

Pues vamos entonces en primer lugar, esto lo vamos a comentar, esto no lo queremos y vamos en primer lugar a imprimir eso.

Aquí lo vemos, oye, token generado, esto lo tenemos aquí, fijaos que esto si nos venimos aquí y hacemos la verificación de la ficha, nos podíamos venir aquí atrás, vámonos al debug un poquito más atrás y aquí en el debugger vamos a poder pegar nuestro token.

Nos está diciendo que la firma es inválida, tenemos que ponerle aquí el secreto con el que nosotros estamos firmando, nos dice ahora sí la firma es válida.

¿Qué pasa?

Que hasta aquí está bien, es decir, esto lo único que nos está haciendo es comprobar si la firma es válida o no.

Pero sin embargo, si nosotros ahora cogemos y tratamos de.

Vamos a ponerle aquí un try, vamos a poner decode token y esto es igual a lo que teníamos abajo, j decode token con la clave icon, el algoritmo, vamos a intentar hacer esto, vamos a poner también una excepción genérica para que no la imprima y en el caso de que esto funcione bien, que también nos imprima.

Token wt guardado, aquí está imprimiendo, vamos a probar esto, lo guardamos.

Aquí hay algo en lo que me habré equivocado, algoritmo me pide los va con una s, el decode vamos a probar de nuevo que me había colado eso y vemos que efectivamente me está diciendo oye, es que este token no es válido, tal y como se ha generado este token todavía no es válido, es decir, tenemos que esperar 5 min para poder encontrar esta que el token sea válido, porque no va a ser válido.

Claro, si lo hubiésemos hecho, si ahora yo por ejemplo, a esto le restamos, vamos a hacer una cosa que el ISO edad le vamos a decir que tenemos un time delta de -60 min, vamos a ver si esto nos lo coge, vale, bueno, claro, nos dice que el token todavía no es válido, es decir, claro, me está comprobando la propia librería, me está comprobando el tiempo actual y me está diciendo que el tiempo actual de cuando se ha lanzado tenemos un esweb bastante atrás.

La otra opción que podríamos hacer para que veamos que ya ha expirado el token es lo que estaba pasando, si recordáis en la parte de la documentación de la librería tenemos una sección donde a la vez que se crea se está terminando este toque.

Si nosotros asignásemos por ejemplo esto de aquí, la parte de expiración, este le vamos a comentar, nos lo podemos quitar, no hace falta que pongamos todo y en la fecha de expiración le vamos a poner el mismo, vamos a limpiar y vamos a lanzarlo, que nos dice que la firma ha expirado, obviamente que ya no es válido porque en el momento en el que lo hemos generado y lo hemos querido imprimir, pues a nivel de segundos o de microsegundos ya ha cambiado y por tanto ya ha estirado.

Así que bueno, con esta pequeña demostración vemos cómo es el uso de los claims jwt, lo sencillo que es utilizar una librería y en este caso hemos cogido el ejemplo de python para poder realizar el ejercicio, pero que perfectamente con todas las librerías disponibles podríamos utilizar cualquier otro lenguaje de programación.

Lo interesante de aquí es ver cómo funcionan los claims, cómo funciona la validación y cómo perfectamente a la hora de intentar decodificar el token, si no nos cuadra, si hay algo incorrecto en la firma o si a nivel de tiempo ha expirado, esto nos lo va a hacer y por tanto luego será cuestión nuestra que implementemos una lógica que permita únicamente

Si imprimimos por el tipo aquí sabemos que es del tipo de que está expirada.

Si hacemos lo que teníamos antes, pues vamos a ver otro tipo de de extensión aquí Vale, creo que lo teníamos así esto una f y le ponemos el not before time.

Vamos a lanzarle aquí nos dice que otra excepción que es Immature signature R.

Como vemos, con esto queda ya demostrado el uso de la librería de Pi jwt y cómo podemos trabajar con estos tokens y validarlos a nivel de tiempo y a nivel de roles.

Tras realizar esta actividad práctica de generación de tokens JWt con claims registrados, hemos llegado a varias conclusiones importantes que resumen nuestro aprendizaje y experiencia.

En primer lugar, la configuración cuidadosa de los claims en los tokens jwt.

Esto es esencial para controlar su comportamiento y la seguridad de estos.

Hemos aprendido que la correcta configuración de los claims relacionados con la expiración y la validez o que no sean válidos antes de cierto tiempo.

Esto es crucial para garantizar su uso seguro y evitar posibles accesos no autorizados en tiempos no autorizados.

Utilizar librerías especializadas como Pi Jwt nos ha permitido trabajar de manera eficiente y segura en la manipulación de estos tokens.

A través de esta experiencia hemos comprendido la importancia de utilizar herramientas adecuadas para tareas específicas, lo que contribuye a un desarrollo más robusto y confiable de nuestras aplicaciones.

Con la realización de esta actividad, hemos adquirido una comprensión sólida sobre cómo controlar la vida útil de un token mediante la configuración de los claims xp y nbf.

Esto nos permite establecer restricciones temporales precisas para el uso del token, lo que mejora significativamente la seguridad de nuestras aplicaciones y la protección de los datos sensibles.

Además de los los claims relacionados con la aspiración y la validez del token, hemos explorado otros claims útiles como es la identificación del sujeto o la identificación del token.

Estos claims adicionales nos brindan la flexibilidad para agregar información adicional al token y mejorar aún más su seguridad y su utilidad en diferentes escenarios de aplicación.

Hemos aprendido la importancia práctica de los conocimientos adquiridos en esta actividad en el desarrollo web y APIs.

Por ejemplo, la capacidad para generar y manejar tokens JWT de manera segura es fundamental para garantizar la autenticación y la autorización adecuada en nuestro sistema, lo que contribuye a la protección de nuestros usuarios y de nuestros datos.

Por último, y como resumen final, esta actividad nos ha proporcionado una comprensión más profunda y práctica de los tokens JWT y su configuración, así como su importancia en la seguridad de nuestras aplicaciones.

Estamos listos para aplicar estos conocimientos en nuestros proyectos futuros y así proteger la seguridad de nuestros datos.

Y con esto llegamos al final de esta sesión de Resolución del Ejercicio.

Los esperamos en el siguiente vídeo.