¿Qué es JSON Web Token?

Neilton Seguins
28/04/2023

Introducción

¿Alguna vez has estado en un evento en el que necesitabas presentar un **documento de identificación** para demostrar que realmente fuiste la persona que compró el boleto? El acto de solicitar un documento de **identificación** es una forma de autenticación para que usted reciba la **autorización** para ingresar. En la web, este proceso funciona de manera similar. Para realizar solicitudes de algunos servicios o acceder a determinadas páginas, deberá identificarse de alguna forma y esta identificación deberá ser segura y única.

¿Qué es token?

Actualmente, escuchamos mucho la palabra token relacionada con NFT (acrónimo de “non-fusible tokens”), metaverso, criptomonedas, etc. Sin embargo, fuera de este medio, un token es una **firma digital, una clave.**

Cuando abre una cuenta bancaria, debes definir una contraseña y tus datos personales. Estos datos se convierten en una firma digital que lo identificarás de **manera única** para ese banco y cada vez que acceda sa tu banco e ingreses tu contraseña y datos personales, el banco comprenderá y confirmará que tú es el usuario que inició la sesión. Similar a ingresar el evento cuando presentamos nuestro documento de identidad.

Existen varios algoritmos y estándares que transforman tu información en un token, es decir, una clave de autenticación única, que tiene sentido para el servicio o la aplicación a la que intentas acceder en ese momento. Uno de estos estándares es JWT, que es seguro porque permite la autenticación entre las dos partes a través de un **token firmado**.

¿Qué es JWT?

Un JWT es un estándar para la autenticación y el intercambio de información definido por RFC7519. Es posible almacenar objetos JSON de forma segura y compacta. Este token es un código Base64 y se puede firmar con un par de claves secretas o privadas/públicas.

Los tokens firmados pueden verificar la integridad de la información que contienen, a diferencia de los tokens cifrados que ocultan esta información. Si un JWT está firmado por un par de claves pública/privada, la empresa certifica que la parte que tiene la clave privada está realmente firmada.

¿Cuándo y dónde puedo usar JWT?

Se puede utilizar, por ejemplo, en un escenario de **autorización**. Una vez que el usuario haya iniciado sesión, puede ver cada solicitud y verificar que incluye el JWT, lo que le permite acceder a rutas, servicios y otros recursos.

Otro escenario para el uso de JWTs es el **intercambio de información** porque, una vez firmado, es posible estar seguro de que los remitentes son quienes dicen ser. Además, podemos identificar si el contenido de la empresa ha cambiado o no debido a la composición de un JWT.

Cómo surgió JWT?

Forma parte de una familia de especificaciones: la familia JOSE.

JOSE significa JSON Object Signing and Encryption, en inglés **JSON Object Signing and Encryption**. JWT es parte de esta familia de especificaciones y representa el token. A continuación, puedes ver otras especificaciones de esta familia:

- JWT (JSON Web Tokens): representa el propio token;
- JWS (JSON Web Signature): representa la firma del token;
- JWE (JSON Web Encryption): representa la firma para el cifrado de tokens;
- JWK (JSON Web Keys): representa las claves para la firma;
- JWA (JSON Web Algorithms): representa los algoritmos para firmar el token.

Ahora que sabe qué son, para qué sirven y cuándo usar un JWT, comprendamos más profundamente cómo funciona y cuáles son los componentes de un JWT. ¡Ven conmigo!

Componentes básicos de un JSON Web Token

Un JWT tiene una estructura básica compuesta por encabezado, carga útil y firma. Estas tres partes están separadas por puntos (.). De esta forma quedaría algo así como: `header.payload.signature`. ¡Comprendamos mejor cada una de estas partes!

Header

Headers son los encabezados del token donde pasamos básicamente dos piezas de información: el `alg` que informa qué algoritmo se usa para crear la firma y el `typ` que indica el tipo de token.

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

Payload

El payload es el componente donde podemos encontrar datos de autenticación como contraseñas y correos electrónicos, por ejemplo:

```
{
    "email" : "nombre@alura.com.br"
```

```
    "password" : "HuEKW489!j445*"
}
```

Signature

La firma del token (*firma*) está compuesta por la codificación del encabezado y el payload más una clave secreta y es generada por el algoritmo especificado en el encabezado.

```
HS256SHA256(
        base64UrlEncode(header) + "." + base64UrlEncode(payload), secret_key)
```

El resultado son tres cadenas separadas por puntos que se pueden usar fácilmente en entornos HTML y protocolos HTTP.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.
SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Ahora que entendemos cómo es por "dentro" un JWT, ¡creemos nuestro propio JSON Web Token!

Creando un JWT token

Para comenzar, creemos una carpeta llamada `jwt` en el directorio que quieras. Crees un archivo JavaScript con el nombre que elija, estoy usando un archivo llamado index.js. Instale la jwt lib de su elección. Hay varias bibliotecas que ayudan en la generación de JWTs Usaré jsonwebtoken, que es uno de los más populares, pero siéntete libre de explorar otras opciones.

El primer paso es importar la lib a nuestro archivo:

```
const jwt = require ('jsonwebtoken');
```

Ahora creamos nuestra clave secreta. La idea es que solo tú conozcas tu clave secreta y que sea difícil para dificultar que se produzcan ataques maliciosos. El mío se veía así:

```
const secretKey = 'skljaksdj99834983274531sldkjf';
```

Hecho esto, vamos a crear nuestro token usando el método de sign. Este método acepta como parámetros el payload, la clave secreta y la cabecera, en ese orden.

```
const nuestroToken = jwt.sign(
  {
    email: 'nome@alura.com.br',
    password: 'HuEKW489!j445*',
  },
  secretKey,
  {
    expiresIn: '1y',
    subject: '1',
  }
);
```

Para este JWT, estoy ingresando un correo electrónico y una contraseña en la carga útil; mi clave secreta; y en el encabezado estoy informando un asunto, que en la biblioteca de este ejemplo funciona como una identificación. Además, estoy diciendo que nuestro token caduca en 1 año. De forma predeterminada, el algoritmo de codificación es HS256.

Para ver el resultado en nuestra terminal, utilicé la biblioteca **Nodemon** que puedes instalar y ver cómo funciona accediendo a este enlace. nodemon es una herramienta que nos ayuda a desarrollar aplicaciones basadas en Node.JS al reiniciar automáticamente la aplicación cuando se detectan cambios en los archivos del directorio.

Podemos ver nuestro token generado al pasar la variable ourToken en un archivo console.log:

```
console.log(nuestroToken);
```

La salida debe ser:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Im5vbWVAYWx1cmEuY29tLmJyIiwic
```

Verificando nuestro JWT

Para verificar nuestro token, podemos usar un método de la biblioteca jsonwebtoken llamado `decode`, pasando el token generado.

```
const tokenGenerado = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Im5vbW
```

```
console.log(jwt.decode(tokenGerado));
```

La salida de este código es:

```
{
  email: 'nome@alura.com.br',
  password: 'HuEKW489!j445*',
  iat: 1651683517,
  exp: 1683241117,
  sub: '1'
}
```

Donde los parámetros `iat`, `exp` y `sub` son, respectivamente, las fechas de creación y caducidad, en formato UTC, en la que se creó el token y en la que caducará, y el asunto que le pasamos en nuestro código con valor 1. Otra alternativa para comprobar nuestro token es accediendo al enlace: https://jwt.io/. En este caso, solo necesitamos pasar el token generado y visualizaremos la información decodificada.

Ahora, es posible que se pregunte: "Ahora que sé qué es un token web JSON y cómo funciona, ¿cómo puedo usarlo en mis aplicaciones front-end?"

¡Vamos a averiguar!

Autenticación con Tokens

Imagina que eres desarrollador y estás creando el front-end de una aplicación para un banco. En la página de inicio de sesión, obtén datos de usuario y envía estos datos a una API utilizando fetch o axios, por ejemplo.

```
fetch(`${baseUrl}/auth/login`, {
method: ‘POST’
      headers: {
       ‘Content Type’: ‘Application/json’,
      },
body: usuario,
})
```

```
.then((respuesta) => {
...alguna cosa
})
.catch((error) => {
...alguna cosa
});
```

El servidor tomará estos datos y por lógica devolverá un token que identificará a ese usuario. Ahora, cada vez que este usuario inicie sesión en la plataforma, pasará por la **autenticación** y si todo está correcto con los datos, estará **autorizado** a acceder a ciertas áreas de la aplicación, como ver el saldo. Por lo general, esta **codificación y tokenización** es manejada por el backend, pero deberá asegurarse de que este usuario que ha iniciado sesión pueda continuar accediendo a otras áreas de la aplicación.

También puede guardar el token en el almacenamiento de sesión de su navegador o en el almacenamiento local, para asegurarte de que, mientras el token no caduque, el usuario permanezca conectado a la aplicación. Además, es importante que, al iniciar sesión, el usuario sea redirigido a una página de Inicio, donde podrá ver otras funcionalidades de la aplicación.

Cuándo este usuario intente acceder a la página que muestra su saldo, por ejemplo, puede realizar una solicitud utilizando axios o fetch pasando en los encabezados un campo de "Autorización" con el token generado. Esto hará que el servidor verifique si el usuario tiene o no permiso para acceder a esa página específica.

```
fetch(`${baseUrl}/saldo`, {
      headers: {
       'Authorization': Token,
      },
})
.then((respuesta) => {
...alguna cosa
})
.catch((error) => {
...alguna cosa
});
```

Cuando el usuario cierra la sesión de nuestra aplicación, puede redirigirlo a otra página, y cuando el token caduca, redirige al usuario a la página de inicio de sesión nuevamente.

Conclusión

Genial, ¿verdad?

En este artículo entendiste qué son los JSON Web Tokens, para qué sirven, cuáles son sus componentes y cómo usarlos en tus aplicaciones. También ha visto cómo usar tokens en una aplicación front-end para autenticar a los usuarios.

Neilton Seguins

Soy desarrollador front-end e instructor de React. Soy graduado como Licenciado en Ciencias y Tecnología y en Ingeniería Mecánica. Tengo experiencia en desarrollo usando JavaScript/TypeScript, React js, Next js y Node.js. Me encanta la música, leer libros y manga y ver series.

Traducido para **Alura Latam** por **Luis Puig**.

ARTÍCULOS DE TECNOLOGÍA

un solo pago de US$ 65.90

- 305 cursos
- Acceso a TODOS los cursos por 6 meses
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Luri, la inteligencia artificial de Alura
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

un solo pago de US$ 99.90

- 305 cursos
- Acceso a TODOS los cursos por 1 año
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Luri, la inteligencia artificial de Alura
- Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

Acceso a todos los cursos

Estudia las 24 horas, dónde y cuándo quieras

Nuevos cursos cada semana