Guía de ejercicios - JWT (I)

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Actividad guiada: Dashboard restringido

La red social "Your Space" no permite que los usuarios accedan al dashboard sin ser autorizados, por lo que necesitarás agregar una ruta **/Dashboard** que reciba un token en la query string y en caso de ser válido o estar vigente, devolver una bienvenida al Dashboard, de lo contrario, devolver un JSON indicando el mensaje de error que disponemos como parámetro en el callback del método "verify". Sigue los pasos para la solución de este ejercicio:

- ***Paso 1:*** Crear una ruta **/Dashboard.**
- ***Paso 2:*** Extraer de las query Strings el parámetro token y almacenarlo en una constante.
- ***Paso 3:*** Utilizar el método "verify" para verificar el token obtenido.
- ***Paso 4:*** En caso de haber un error, devuelve el código de estado 401 y un JSON con la propiedad "message" del parámetro de error disponible en el callback del método "verify".
- ***Paso 5:*** En caso de no existir ningún error, devolver el mensaje "Bienvenido al Dashboard ${decoded.email}". Utilizamos el parámetro "decoded" para acceder a las propiedades del payload que se encontraba codificado en el token recibido.

```
// Paso 1
app.get("/Dashboard", (req, res) => {
  // Paso 2
  let { token } = req.query;
  // Paso 3
  jwt.verify(token, secretKey, (err, decoded) => {
    // Paso 4
    err
      ? res.status(401).send({
          error: "401 Unauthorized",
          message: err.message,
        })
      : // Paso 5
        res.send(`
       Bienvenido al Dashboard ${decoded.data.email}
    `);
```

```
  });
});
```

Ahora, si intentas consultar al servidor con la dirección http://localhost:3000/Dashboard recibirás lo que te muestro en la siguiente imagen:

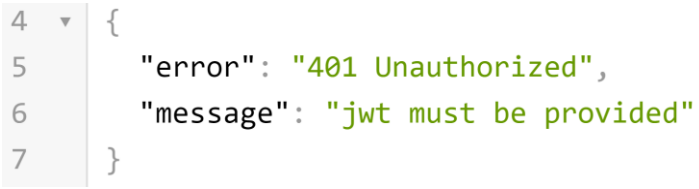


Imagen 1. Respuesta del servidor indicando que el token no está proveído.
Fuente: Desafío Latam

Como era de esperarse, no estamos escribiendo en la dirección ningún token como parámetro ¿Qué pasaría si intentamos inventar un token nosotros mismos? Consulta al servidor con la siguiente dirección para averiguarlo:

http://localhost:3000/Dashboard?token=CualquierCosa

Deberás recibir lo que te muestro en la siguiente imagen:

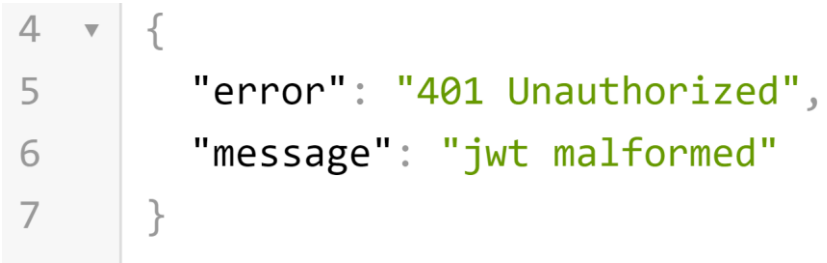
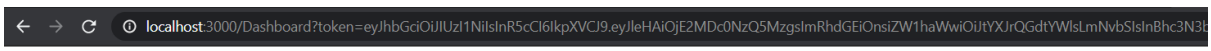


Imagen 2. Respuesta del servidor indicando que el token es inválido.
Fuente: Desafío Latam

Como puedes observar, el mensaje cambia dinámicamente en dependencia de la situación que esté sucediendo. Ahora intenta loguearte con el usuario “mark” usando la siguiente dirección:

http://localhost:3000/login?email=mark@gmail.com&password=facebook

Presiona el enlace “Ir al Dashboard”, deberás ver lo que te muestro en la siguiente imagen:



Bienvenido al Dashboard mark@gmail.com

Imagen 3. Acceso exitoso a la ruta **/Dashboard**.
Fuente: Desafío Latam

Muy bien, esto está funcionando de maravilla, pero ¿Qué sucede con el tiempo de expiración del token? ¡Averiguemoslo! espera 2 minutos y vuelve a consultar la ruta. Deberás recibir lo que te muestro en la siguiente imagen.

```
{
  "error": "401 Unauthorized",
  "message": "jwt expired"
}
```

Imagen 17. JSON de respuesta indicando que el token expiró.
Fuente: Desafío Latam

¡Excelente! Ahí lo tenemos, ya que expiró el tiempo de vida del token, el servidor no lo validó e impidió el acceso al contenido.

¡Manos a la obra! - Ejercicio propuesto 1

Basado en el ejercicio “Autorizando a Steve” de la sesión anterior, crear un servidor que devuelva en su ruta raíz un token utilizando como payload el usuario “Bill”.

¡Manos a la obra! - Ejercicio propuesto 2

Basado en el ejercicio “Decodificando el token de Steve” de la sesión anterior, crear una ruta que verifique tokens y en caso de ser inválido devuelva el parámetro “err” del callback que tiene el método verify.

¡Manos a la obra! - Ejercicio propuesto 3

Basado en el ejercicio “Persistencia del token” de la sesión anterior, generar un token que expira en 5 minutos y almacenar el token en sessionStorage.

¡Manos a la obra! - Ejercicio propuesto 4

Basado en el ejercicio guiado de esta guía “Dashboard restringido”, en caso de ser válido el token recibido, devuelve además del mensaje de bienvenida, un script que guarde el correo electrónico del usuario en LocalStorage.

Solución ejercicios propuestos

Ejercicio propuesto 1

Basado en el ejercicio “Autorizando a Steve”, crear un servidor que devuelve en su ruta raíz un token utilizando al usuario “Bill”.

```
const express = require('express')
const users = require('./data/users.js')
const app = express()
app.listen(3000, () => console.log('Servidor encendido en el puerto
3000'))
const jwt = require("jsonwebtoken")

const secretKey = 'Mi Llave Ultra Secreta'

const token = jwt.sign(users[2], secretKey)

app.get('/', (req, res) => {
  res.send(token)
})
```

Ejercicio propuesto 2

Basado en el ejercicio “Decodificando el token de Steve”, crear una ruta que verifique tokens y en caso de ser inválido devuelva el parámetro “err” del callback que tiene el método verify.

```
app.get("/token", (req, res) => {
  const { token } = req.query;
  jwt.verify(token, secretKey, (err, data) => {
    res.send( err ? err : data );
  });
});
```

Ejercicio propuesto 3

Basado en el ejercicio “Persistencia del token”, generar un token que expira en 5 minutos y almacena el token en sessionStorage.

```
app.get("/login", (req, res) => {
  const { email, password } = req.query;
  const user = users.find((u) => u.email == email && u.password ==
password);
  if (user) {
    const token = jwt.sign(
      {
        exp: Math.floor(Date.now() / 1000) + 300,
        data: user,
      },
      secretKey
    );
    res.send(`
    <a href="/Dashboard?token=${token}"> <p> Ir al Dashboard </p> </a>
     Bienvenido, ${email}.
    <script>
    sessionStorage.setItem('token', JSON.stringify("${token}"))
    </script>
    `);
  } else {
    res.send("Usuario o contraseña incorrecta");
  }
});
```

Ejercicio propuesto 4

Basado en el ejercicio "Dashboard restringido", en caso de ser válido el token recibido, devuelve además del mensaje de bienvenida, un script que guarde el correo electrónico del usuario en LocalStorage.

```
app.get("/Dashboard", (req, res) => {
  let { token } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    err
      ? res.status(401).send({
          error: "401 Unauthorized",
          message: err.message,
        })
      : res.send(`
       Bienvenido al Dashboard ${decoded.data.email}
       <script>
       localStorage.setItem('email', "${decoded.data.email}")
       </script>
     `);
  });
});
```

Preguntas de proceso

Reflexiona:

- ¿Qué he aprendido hasta ahora?
- ¿Hay algo que me está dificultando mucho?