Guía de ejercicios - JWT (II)

¡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: Acceso a Contenido Premium

En este ejercicio implementaremos una ruta de acceso a contenido premium en una plataforma de streaming llamada “PremiumStream”. Los usuarios deben proporcionar un token JWT válido para acceder al contenido exclusivo.

En “PremiumStream” los usuarios pueden ver películas y series en streaming.
Para acceder a las películas premium, necesitan un token JWT válido.

Implementaremos una ruta **‘/premium’** que verifica el token proporcionado en la query string. Si el token es válido y no ha expirado, el usuario podrá ver la película premium.

1. **Paso 1:** Crear un objeto con al menos 5 películas premium simuladas. Cada película debe tener un id, un título y una propiedad disponible que indique si está disponible para ver o no**.**

```
const express = require("express");
const jwt = require("jsonwebtoken");

const app = express();
const secretKey = "clave_secreta_temporal"; // Clave secreta temporal
para las pruebas

// Paso 1: Crear un objeto con al menos 5 películas premium simuladas.
const peliculasPremium = [
  { id: 1, titulo: "Tiburón", disponible: true },
  { id: 2, titulo: "Avatar", disponible: false },
  { id: 3, titulo: "Titanic", disponible: true },
  { id: 4, titulo: "Avengers", disponible: false },
  { id: 5, titulo: "Diario de una pasión", disponible: true },
];
```

2. **Paso 2**: Crear una ruta **/generar-token** que acepte un contenidoPremiumId como parámetro en la query string y genere un token JWT con ese contenidoPremiumId como payload. Utiliza la función sign de jsonwebtoken para generar el token y la secretKey proporcionada.

```
// Paso 2: Crear una ruta /generar-token que acepte contenidoPremiumId
como parámetro y genere un token JWT.
app.get("/generar-token", (req, res) => {
  const { contenidoPremiumId } = req.query;
  const pelicula = peliculasPremium.find(
    (p) => p.id === parseInt(contenidoPremiumId)
  );

  if (!pelicula || !pelicula.disponible) {
    return res.status(404).json({
      error: "404 Not Found",
      message: "Contenido premium no encontrado o no disponible.",
    });
  }

  // Generar un token JWT con el contenidoPremiumId como payload
  const token = jwt.sign({ contenidoPremiumId }, secretKey, {
    expiresIn: "1h",
  });
  res.json({ token });
});
```

3. **Paso 3**: Crear una ruta **/premium** que verifique el token proporcionado en la query string. Utiliza la función verify de jsonwebtoken para validar el token. Si el token es válido y el contenido está disponible, devuelve un mensaje indicando que el usuario puede ver la película. Si el token no es válido o el contenido no está disponible, devuelve un mensaje de error.

```
// Paso 3: Crear una ruta /premium que verifique el token y permita el
acceso al contenido premium.
app.get("/premium", (req, res) => {
  const { token } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido.",
      });
    }

    const contenidoPremiumId = decoded.contenidoPremiumId;
    const pelicula = peliculasPremium.find((p) => p.id ===
contenidoPremiumId);

    if (!pelicula || !pelicula.disponible) {
      return res.status(403).json({
        error: "403 Forbidden",
        message: "Contenido premium no disponible en este momento.",
      });
    }

    // Permitir acceso al contenido premium y enviar un mensaje con el
título de la película.
    res.send(
      `Disfruta de ${pelicula.titulo}, una película premium en
StreamHub!`
    );
  });
});
```

4. **Paso 4**: Permitir acceso al contenido premium y enviar un mensaje con el título de la película.

```
// Paso 4:
    res.send(
      `Disfruta de ${pelicula.titulo}, una película premium en
StreamHub!`
    );
```

5. **Paso 5**:Iniciar el servidor en el puerto 3000

```
app.listen(3000, () => {
  console.log('Servidor en ejecución en el puerto 3000.');
});
```

Hecho esto, ejecuta el servidor en el puerto 3000 y realiza una solicitud a **/generar-token** para obtener un token con un contenidoPremiumId válido.

Luego, realiza una solicitud a **/premium?token=TOKEN_VALIDO_AQUI** para acceder al contenido premium.

Realiza pruebas adicionales enviando solicitudes a **/generar-token** con contenidoPremiumId inválidos o a **/premium** sin proporcionar un token válido para verificar que el servidor maneja adecuadamente los casos de error.

Finalmente el código completo quedaría de la siguiente manera:

```
const express = require("express");
const jwt = require("jsonwebtoken");

const app = express();
const secretKey = "clave_secreta_temporal"; // Clave secreta temporal
para las pruebas

// Paso 1: Crear un objeto con al menos 5 películas premium simuladas.
const peliculasPremium = [
  { id: 1, titulo: "Tiburón", disponible: true },
  { id: 2, titulo: "Avatar", disponible: false },
  { id: 3, titulo: "Titanic", disponible: true },
```

```
  { id: 4, titulo: "Avengers", disponible: false },
  { id: 5, titulo: "Diario de una pasión", disponible: true },
];

// Paso 2: Crear una ruta /generar-token que acepte contenidoPremiumId
como parámetro y genere un token JWT.
app.get("/generar-token", (req, res) => {
  const { contenidoPremiumId } = req.query;
  const pelicula = peliculasPremium.find(
    (p) => p.id === parseInt(contenidoPremiumId)
  );

  if (!pelicula || !pelicula.disponible) {
    return res.status(404).json({
      error: "404 Not Found",
      message: "Contenido premium no encontrado o no disponible.",
    });
  }

  // Generar un token JWT con el contenidoPremiumId como payload
  const token = jwt.sign({ contenidoPremiumId }, secretKey, {
    expiresIn: "1h",
  });
  res.json({ token });
});

// Paso 3: Crear una ruta /premium que verifique el token y permita el
acceso al contenido premium.
app.get("/premium", (req, res) => {
  const { token } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido.",
      });
    }

    const contenidoPremiumId = decoded.contenidoPremiumId;
    const pelicula = peliculasPremium.find((p) => p.id ===
contenidoPremiumId);

    if (!pelicula || !pelicula.disponible) {
      return res.status(403).json({
```

```
        error: "403 Forbidden",
        message: "Contenido premium no disponible en este momento.",
      });
    }

    // Paso 4: Permitir acceso al contenido premium y enviar un mensaje
con el título de la película.
    res.send(
      `Disfruta de ${pelicula.titulo}, una película premium en
StreamHub!`
    );
  });
});

// Paso 5: Iniciar el servidor en el puerto 3000.
app.listen(3000, () => {
  console.log("Servidor en ejecución en el puerto 3000.");
});
```

¡Manos a la obra! - Validación de Tokens

Crea una ruta **/validar-token** que acepte un token JWT en la query string y verifique su validez utilizando la secretKey.
Devuelve un mensaje indicando si el token es válido o no.

¡Manos a la obra! - Expiración de Tokens

Modifica la ruta **/generar-token** para que los tokens JWT expiren después de 30 minutos en lugar de 1 hora. Asegúrate de que los tokens generados tengan una fecha de expiración y verifica que un token expirado sea rechazado por la ruta **/premium**.

Solución ejercicios - Manos a la obra

Validación de Tokens

```
app.get("/validar-token", (req, res) => {
  const { token } = req.query;
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({
        error: "401 Unauthorized",
        message: "Token inválido."
      });
    }
    res.json({
      message: "Token válido. Usuario autenticado.",
      decodedData: decoded
    });
  });
});
```

Expiración de Tokens

```
app.get("/generar-token", (req, res) => {
  const { contenidoPremiumId } = req.query;
  const pelicula = peliculasPremium.find(p => p.id ===
parseInt(contenidoPremiumId));

  if (!pelicula || !pelicula.disponible) {
    return res.status(404).json({
      error: "404 Not Found",
      message: "Contenido premium no encontrado o no disponible."
    });
  }

  // Generar un token JWT con el contenidoPremiumId como payload y
expiración en 30 minutos
  const token = jwt.sign({ contenidoPremiumId }, secretKey, { expiresIn:
'30m' });
  res.json({ token });
});
```

Preguntas de proceso

Reflexiona:

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

Preguntas de cierre

- ¿Cuál es el propósito principal de utilizar tokens JWT en una aplicación web?
- Explica cómo se utilizan para la autenticación y autorización, y por qué son preferibles en comparación con otros métodos de autenticación.
- ¿Cuál es la diferencia entre autenticación y autorización en el contexto de tokens JWT?