

## Material de apoio

Site: [Geração Tech](#)  
Curso: Formação em Desenvolvedor Web - Online  
Livro: Material de apoio

Impresso por: JOÃO VITOR DE MELO FREITAS  
Data: segunda-feira, 12 ago. 2024, 22:12

# Índice

## **1. Implementando JWT com Expiração em uma Aplicação Express**

1.1. Vídeo Aula

## **2. Continuando a Implementação de JWT com Expiração em uma Aplicação Express**

2.1. Video aula

# 1. Implementando JWT com Expiração em uma Aplicação Express

## Configurando o Projeto

### 1. Crie uma nova pasta para o projeto:

bash

 Copiar código

```
mkdir aula-jwt2 cd aula-jwt2
```

### 2. Inicialize o projeto com npm:

bash

 Copiar código

```
npm init -y
```

### 3. Instale as dependências necessárias:

bash

 Copiar código

```
npm install express jsonwebtoken nodemon
```

### 4. Atualize o package.json para usar o nodemon:

json

 Copiar código

```
"scripts": { "start": "nodemon server.js" }
```

## Criando o Servidor Express

### 1. Crie um arquivo chamado `server.js`:

javascript

 Copiar código

```
const express = require('express'); const jwt = require('jsonwebtoken'); const app = express(); const PORT = 3000; // Chave secreta para assinar os tokens const secretKey = 'minha_chave_secreta_aleatoria'; // Dados simulados do usuário const userData = { nome: 'Márcio', email: 'marcio@gt.com.br' }; // Rota para gerar o token app.get('/gerar-token', (req, res) => { const token = jwt.sign(userData, secretKey, { expiresIn: '1h' }); res.json({ token }); }); // Rota para validar o token app.get('/validar-token', (req, res) => { const token = req.query.token; try { const decoded = jwt.verify(token, secretKey); res.json({ mensagem: 'Token válido', decoded }); } catch (err) { res.status(401).json({ mensagem: 'Token inválido ou expirado', error: err.message }); } }); app.listen(PORT, () => { console.log(`Servidor rodando na porta ${PORT}`); });
```

## Testando a Aplicação

### 1. Inicie o servidor:

bash

 Copiar código

```
npm start
```

2. Gere um token acessando a rota `/gerar-token` em seu navegador ou ferramenta de requisição HTTP como Postman ou Insomnia:

```
bash
```

```
Copiar código
```

```
http://localhost:3000/gerar-token
```

A resposta deve ser algo como:

```
json
```

```
Copiar código
```

```
{ "token": "seu_token_aqui" }
```

3. Valide o token acessando a rota `/validar-token` e passando o token gerado como query string:

```
bash
```

```
Copiar código
```

```
http://localhost:3000/validar-token?token=seu_token_aqui
```

A resposta deve ser algo como:

```
json
```

```
Copiar código
```

```
{ "mensagem": "Token válido", "decoded": { "nome": "Márcio", "email": "marcio@gt.com.br", "iat": 1626345583, "exp": 1626349183 } }
```

Se o token for alterado ou expirado, a resposta será:

```
json
```

```
Copiar código
```

```
{ "mensagem": "Token inválido ou expirado", "error": "jwt expired" }
```

## Conclusão

Nesta aula, aprendemos como implementar JWT em uma aplicação Express. Configuramos rotas para gerar e validar tokens, garantindo a segurança da nossa aplicação.

Pratiquem os conceitos aprendidos e até a próxima aula!

## 1.1. Vídeo Aula

dia 42 video 01 converted



## 2. Continuando a Implementação de JWT com Expiração em uma Aplicação Express

Olá pessoal, tudo bem? Vamos continuar nossa aula de JWT.


Na aula passada, configuramos nosso projeto com Express e implementamos algumas rotas para gerar e validar tokens.

Agora, vamos adicionar expiração ao nosso JWT e melhorar a estrutura do nosso projeto utilizando variáveis de ambiente.

### Instalando e Configurando o Dotenv

Primeiramente, vamos instalar o pacote `dotenv` para gerenciar nossas variáveis de ambiente.

bash

 Copiar código

```
npm install dotenv
```

### Estruturando o Projeto

1. Crie um arquivo `.env` na raiz do projeto para armazenar as variáveis de ambiente:

plaintext

 Copiar código

```
APP_KEY=minha_chave_secreta_aleatoria PORT=3000
```

2. Atualize seu `server.js` para carregar as variáveis de ambiente:

javascript

 Copiar código

```
require('dotenv').config(); const express = require('express'); const jwt = require('jsonwebtoken'); const app = express();
const PORT = process.env.PORT || 3000; // Chave secreta para assinar os tokens, agora vindo do .env const secretKey =
process.env.APP_KEY; // Dados simulados do usuário const userData = { nome: 'Márcio', email: 'marcio@gt.com.br' }; // Rota
para gerar o token com expiração app.get('/gerar-token', (req, res) => { const token = jwt.sign(userData, secretKey, {
expiresIn: '1h' }); res.json({ token }); }); // Rota para validar o token app.get('/validar-token', (req, res) => { const
token = req.headers['token']; if (!token) { return res.status(403).json({ mensagem: 'Token não fornecido' }); } try { const
decoded = jwt.verify(token, secretKey); res.json({ mensagem: 'Token válido', decoded }); } catch (err) {
res.status(401).json({ mensagem: 'Token inválido ou expirado', error: err.message }); } }); app.listen(PORT, () => {
console.log(`Servidor rodando na porta ${PORT}`); });
```

### Testando a Aplicação com Insomnia

1. Inicie o servidor:

bash

 Copiar código

```
npm start
```

2. Abra o Insomnia (ou outra ferramenta de sua preferência) e configure uma requisição para gerar o token:

- Método: GET
- URL: `http://localhost:3000/gerar-token`

3. Configure uma nova requisição para validar o token:

- Método: GET
- URL: `http://localhost:3000/validar-token`
- Headers:

- Key: `token`
- Value: (O token gerado pela primeira requisição)

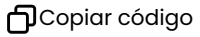
#### 4. Teste o token gerado:

- Gere um token utilizando a primeira requisição.
- Copie o token gerado.
- Valide o token utilizando a segunda requisição, passando o token no header.

## Manipulando Expiração do Token

Para demonstrar a expiração, podemos configurar o token para expirar em um tempo mais curto durante os testes:

javascript



```
app.get('/gerar-token', (req, res) => { const token = jwt.sign(userData, secretKey, { expiresIn: '1m' }); // Expira em 1 minuto res.json({ token }); });
```

Teste o token gerado e aguarde mais de um minuto para validar novamente e verificar a expiração.

## Conclusão

Implementamos a geração e validação de tokens JWT com expiração, utilizando variáveis de ambiente para gerenciar a chave secreta e a porta do servidor.

Além disso, aprendemos a utilizar o Insomnia para testar nossas requisições. Na próxima aula, continuaremos a melhorar nossa aplicação e explorar mais funcionalidades do JWT.

Pratiquem os conceitos e até a próxima!

## 2.1. Video aula

dia 42 video 02 converted

