# Material de apoio

Site: <u>Geração Tech</u> Impresso por: JOÃO VITOR DE MELO FREITAS

Curso: Formação em Desenvolvedor Web - Online Data: segunda-feira, 12 ago. 2024, 22:13

Livro: Material de apoio

# Índice

### 1. Implementando JWT em um Projeto Backend com Express

1.1. Vídeo i

### 2. JWT em um Projeto Backend

2.1. Vídeo 2

# 1. Implementando JWT em um Projeto Backend com Express

### Configuração Inicial

#### 1. Instalar Dependências

- o Express: Framework para criar o servidor.
- o Jsonwebtoken: Biblioteca para gerar e verificar tokens JWT.
- o Dotenv: Para gerenciar variáveis de ambiente.
- o Sequelize: ORM para interagir com o banco de dados.
- Bcrypt: Para criptografar senhas.

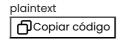
#### bash



npm install express jsonwebtoken dotenv sequelize bcrypt

#### 2. Configurar Variáveis de Ambiente

o Crie um arquivo .env na raiz do projeto com o seguinte conteúdo:



APP\_KEY=sua\_chave\_secreta\_aleatoria PORT=3000

### 3. Estrutura do Projeto

o Crie a seguinte estrutura de diretórios e arquivos:



/src /controllers authController.js /models User.js /routes userRoutes.js /config database.js server.js .env

# Configuração do Banco de Dados

#### 4. Configurar Sequelize

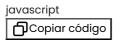
No arquivo database.js:

```
javascript
Copiar código
```

```
const { Sequelize } = require('sequelize'); require('dotenv').config(); const sequelize = new Sequelize('blog',
'root', 'password', { host: 'localhost', dialect: 'mysql' }); module.exports = sequelize;
```

#### 5. Modelo de Usuário

No arquivo User.js:

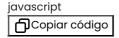


```
const { DataTypes } = require('sequelize'); const sequelize = require('../config/database'); const User =
sequelize.define('User', { username: { type: DataTypes.STRING, allowNull: false, unique: true }, password: {
type: DataTypes.STRING, allowNull: false }, email: { type: DataTypes.STRING, allowNull: false, unique: true } });
module.exports = User;
```

### Implementação do Controle de Autenticação

#### 6. Controle de Autenticação

No arquivo authController.js:



const jwt = require('jsonwebtoken'); const bcrypt = require('bcrypt'); const User = require('../models/User');
require('dotenv').config(); const generateToken = (user) => { return jwt.sign({ id: user.id, username:
 user.username }, process.env.APP\_KEY, { expiresIn: '1h' }); }; exports.register = async (req, res) => { const {
 username, password, email } = req.body; const hashedPassword = await bcrypt.hash(password, 10); try { const user
 = await User.create({ username, password: hashedPassword, email }); res.json({ user, token: generateToken(user)
 }); } catch (error) { res.status(400).json({ error: error.message }); } }; exports.login = async (req, res) => {
 const { username, password } = req.body; try { const user = await User.findOne({ where: { username } }); if
 (!user || !await bcrypt.compare(password, user.password)) { return res.status(401).json({ error: 'Invalid
 credentials' }); } res.json({ token: generateToken(user) }); } catch (error) { res.status(500).json({ error:
 error.message }); } }; exports.verifyToken = (req, res, next) => { const token = req.headers['authorization']; if
 (!token) { return res.status(403).json({ error: 'No token provided' }); } jwt.verify(token, process.env.APP\_KEY,
 (err, decoded) => { if (err) { return res.status(401).json({ error: 'Failed to authenticate token' }); }
 req.userId = decoded.id; next(); }); };

### Configuração das Rotas

#### 7. Rotas de Usuário

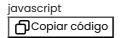
No arquivo userRoutes.js:

```
javascript
Copiar código
```

const express = require('express'); const authController = require('../controllers/authController'); const router
= express.Router(); router.post('/register', authController.register); router.post('/login',
authController.login); router.get('/me', authController.verifyToken, (req, res) => { res.json({ message:
 'Authenticated', userId: req.userId }); }); module.exports = router;

#### 8. Servidor Principal

No arquivo server.js:



const express = require('express'); const sequelize = require('./src/config/database'); const userRoutes =
require('./src/routes/userRoutes'); require('dotenv').config(); const app = express(); const PORT =
process.env.PORT || 3000; app.use(express.json()); app.use('/users', userRoutes); sequelize.sync().then(() => {
 app.listen(PORT, () => { console.log(`Server is running on port \${PORT}`); }); }).catch((error) => {
 console.error('Unable to connect to the database:', error); });

# Testando a Aplicação

#### 1. Inicie o Servidor

• No terminal:

bash Copiar código

npm start

#### 2. Utilize o Insomnia ou Postman para Testar

### Conclusão

Implementamos a geração e verificação de JWT em um projeto backend com Express, protegendo nossas rotas com tokens JWT. Agora, só usuários autenticados podem acessar rotas protegidas.

Na próxima aula, continuaremos a melhorar nossa aplicação e explorar mais funcionalidades do JWT.

Pratiquem os conceitos e até a próxima!

■ Value: Bearer YOUR\_TOKEN\_HERE

# 1.1. Vídeo 1



# 2. JWT em um Projeto Backend

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

Na aula passada, configuramos a rota de login, ajustamos parâmetros para corresponder ao banco de dados e retornamos dados de usuários.

### Ajustes na Função de Login

Primeiro, faremos um pequeno ajuste na função de login para garantir que o tratamento de dados ocorra corretamente:

### Criação de Tokens JWT

Vamos gerar um token JWT quando o usuário fizer login corretamente:

### Validação de Tokens JWT

Para validar os tokens e proteger as rotas, vamos criar um middleware:

### **Ajuste nas Rotas**

iavascript

Agora, aplicaremos esse middleware nas rotas que devem ser protegidas:

```
const express = require('express'); const authController = require('../controllers/authController'); const verifyToken =
require('../middlewares/verifyToken'); const router = express.Router(); router.post('/register', authController.register);
router.post('/login', authController.login); router.get('/me', verifyToken, (req, res) => { res.json({ message:
    'Autenticado', userId: req.userId }); }); module.exports = router;
```

# Integração no Servidor

No arquivo principal server. js, garantimos que as rotas sejam aplicadas corretamente:

```
console.log(`Servidor rodando na porta PORT'); }); }).catch((error) => { console.error('Não foi possível conectar ao banco de dados:', error); });
```

### Testando a Aplicação

Vamos utilizar o Insomnia ou Postman para testar:

| Jacobian in the content of the con

### Conclusão

Implementamos a geração e verificação de JWT em nosso projeto backend com Express, protegendo nossas rotas com tokens JWT. Isso garante que apenas usuários autenticados possam acessar determinadas rotas.

### Dicas de Prática

• Assista ao vídeo completo antes de praticar.

■ Value: Bearer SEU\_TOKEN\_AQUI

- Revise o código e entenda cada parte antes de replicar.
- Teste várias vezes para garantir que tudo está funcionando como esperado.

Espero que tenham gostado da aula.

# 2.1. Vídeo 2

