**Criar uma API em Node.js gerar JWT**

Vamos lá! Para usar JWT em Node.js para gerar códigos de autenticação, você precisa:

1. **Instalar a biblioteca jsonwebtoken:**
2. npm install jsonwebtoken

content\_copyUse code [with caution](https://support.google.com/legal/answer/13505487).Bash

1. **Criar um endpoint para gerar o token JWT:**
2. const express = require('express');
3. const jwt = require('jsonwebtoken');
4. const app = express();
5. const port = 3000;
6. const secretKey = 'sua\_chave\_secreta'; // Substitua por uma chave secreta forte
7. app.post('/authenticate', (req, res) => {
8. const { username, password } = req.body;
9. // Aqui você deve validar as credenciais do usuário, por exemplo:
10. // - Consultar um banco de dados
11. // - Comparar a senha com o hash armazenado
12. // - Verificar se o usuário está ativo
13. if (/\* Credenciais válidas \*/) {
14. const payload = {
15. userId: 1, // Substitua por um ID de usuário válido
16. username: username,
17. iat: Math.floor(Date.now() / 1000) // Tempo de emissão do token
18. };
19. const token = jwt.sign(payload, secretKey, { expiresIn: '1h' }); // Expire em 1 hora
20. res.json({ token });
21. } else {
22. res.status(401).json({ error: 'Credenciais inválidas' });
23. }
24. });
25. app.listen(port, () => {
26. console.log(`Servidor ouvindo na porta ${port}`);
27. });

content\_copyUse code [with caution](https://support.google.com/legal/answer/13505487).JavaScript

**Explicação:**

* **secretKey:** Define uma chave secreta forte para assinar o token JWT. Mantenha essa chave em segredo.
* **app.post('/authenticate'):** Cria um endpoint POST para a rota /authenticate.
* **req.body:** Obtém as credenciais (username e password) enviadas pelo usuário no corpo da requisição.
* **Validação de credenciais:** Substitua o comentário /\* Credenciais válidas \*/ com sua lógica de validação de credenciais.
* **payload:** Define os dados que serão incluídos no token JWT.
* **jwt.sign(payload, secretKey, { expiresIn: '1h' }):** Gera o token JWT com o payload, a chave secreta e um tempo de expiração de 1 hora.
* **Resposta:** Se a autenticação for bem-sucedida, retorna o token JWT para o cliente. Caso contrário, retorna um erro 401 (não autorizado).

**3. Usar o token JWT em outras requisições:**

* **Criar um middleware para verificar o token JWT:**
* const express = require('express');
* const jwt = require('jsonwebtoken');
* const app = express();
* const port = 3000;
* const secretKey = 'sua\_chave\_secreta'; // Substitua pela chave secreta
* // Middleware para verificar o token JWT
* const authenticateToken = (req, res, next) => {
* const authHeader = req.headers['authorization'];
* const token = authHeader && authHeader.split(' ')[1];
* if (token == null) return res.status(401).json({ error: 'Token não encontrado' });
* jwt.verify(token, secretKey, (err, user) => {
* if (err) return res.status(403).json({ error: 'Token inválido' });
* req.user = user;
* next(); // Passa para a próxima rota se o token for válido
* });
* };
* // Aplica o middleware à rota protegida
* app.get('/protected', authenticateToken, (req, res) => {
* // Ação a ser realizada se o usuário estiver autenticado
* res.json({ message: `Olá ${req.user.username}, você está autenticado!` });
* });
* app.listen(port, () => {
* console.log(`Servidor ouvindo na porta ${port}`);
* });

content\_copyUse code [with caution](https://support.google.com/legal/answer/13505487).JavaScript

**Explicação:**

* **authenticateToken:** Um middleware que extrai o token JWT do cabeçalho da requisição, verifica sua validade e, se válido, define o usuário decodificado em req.user.
* **app.get('/protected', authenticateToken, ...):** Aplica o middleware authenticateToken à rota /protected, garantindo que o usuário esteja autenticado antes de acessar essa rota.

**4. Importante:**

* Utilize uma chave secreta forte e segura para proteger o token JWT.
* Guarde a chave secreta em um local seguro e não a compartilhe.
* Configure o tempo de expiração do token de acordo com suas necessidades.

Com essas etapas, você terá uma API de autenticação com JWT em Node.js que gera tokens para autorização e valida tokens em outras requisições.