

Missão Prática – Mundo 05 – Nível 05

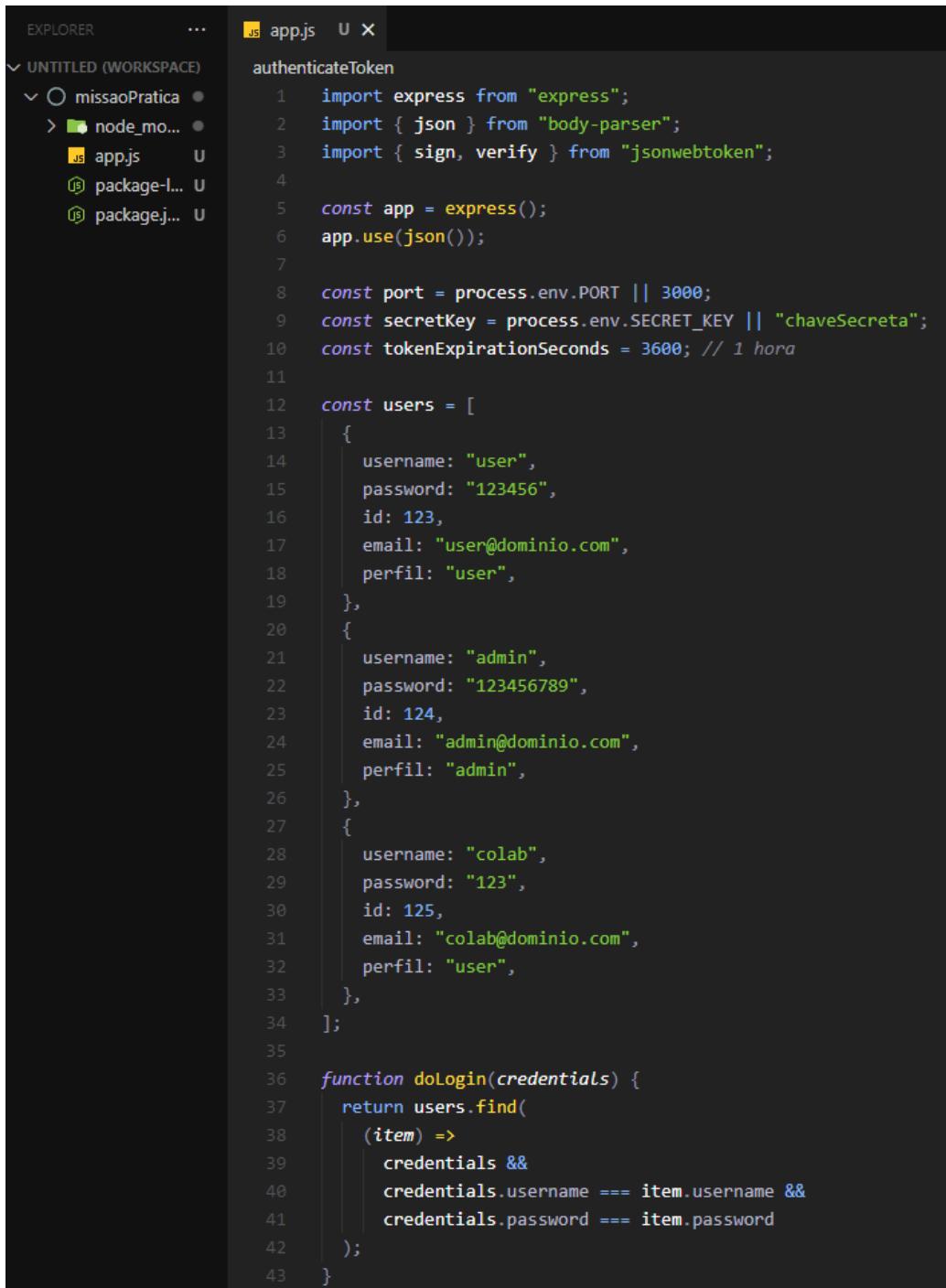
Gilvan Pereira de Oliveira – 2023.01.53256-61197

Polo Centro – São Lourenço Da Mata – PE

RPG0034 – SOFTWARE SEM SEGURANÇA NÃO SERVE! – 9001 – 2025.1

<https://github.com/GilvanPOliveira/FullStack/tree/main/Mundo05/softwareSeguranca>

Missão Prática | Software sem segurança não serve



```

EXPLORER      ...
UNTITLED (WORKSPACE)
  missaoPratica
    node_mo...
    app.js
    package-l...
    packagej...
app.js          ...
authenticateToken
1   import express from "express";
2   import { json } from "body-parser";
3   import { sign, verify } from "jsonwebtoken";
4
5   const app = express();
6   app.use(json());
7
8   const port = process.env.PORT || 3000;
9   const secretKey = process.env.SECRET_KEY || "chaveSecreta";
10  const tokenExpirationSeconds = 3600; // 1 hora
11
12  const users = [
13    {
14      username: "user",
15      password: "123456",
16      id: 123,
17      email: "user@dominio.com",
18      perfil: "user",
19    },
20    {
21      username: "admin",
22      password: "123456789",
23      id: 124,
24      email: "admin@dominio.com",
25      perfil: "admin",
26    },
27    {
28      username: "colab",
29      password: "123",
30      id: 125,
31      email: "colab@dominio.com",
32      perfil: "user",
33    },
34  ];
35
36  function doLogin(credentials) {
37    return users.find(
38      (item) =>
39        credentials &&
40        credentials.username === item.username &&
41        credentials.password === item.password
42    );
43  }

```

```

44
45 app.post("/api/autenticar/login", (req, res) => {
46   const credentials = req.body;
47   if (!credentials || !credentials.username || !credentials.password) {
48     return res.status(400).json({ message: "Credenciais inválidas" });
49   }
50
51   const userData = doLogin(credentials);
52   if (!userData) {
53     return res.status(401).json({ message: "Usuário ou senha incorretos" });
54   }
55
56   const payload = {
57     id: userData.id,
58     username: userData.username,
59     perfil: userData.perfil,
60   };
61   const token = sign(payload, secretKey, { expiresIn: tokenExpirationSeconds });
62   return res.json({ token });
63 });
64
65 function authenticateToken(req, res, next) {
66   const authHeader = req.headers["authorization"];
67   if (!authHeader)
68     return res.status(401).json({ message: "Token não fornecido" });
69
70   const parts = authHeader.split(" ");
71   if (parts.length !== 2)
72     return res.status(401).json({ message: "Token mal formatado" });
73
74   const [scheme, token] = parts;
75   if (!/^Bearer$/i.test(scheme))
76     return res.status(401).json({ message: "Token mal formatado" });
77
78   verify(token, secretKey, (err, decoded) => {
79     if (err) {
80       console.error("Erro ao verificar token:", err);
81       return res.status(401).json({ message: "Token inválido ou expirado" });
82     }
83     req.user = decoded;
84     next();
85   });
86 }
87
88 function requireAdmin(req, res, next) {
89   if (req.user && req.user.perfil === "admin") {
90     next();
91   } else {
92     res
93       .status(403)
94       .json({
95         message: "Forbidden: Acesso permitido apenas para administradores.",
96       });
97   }
98 }
99

```

```

100 app.get("/api/autenticar/usuario", authenticateToken, (req, res) => {
101   const user = users.find((u) => u.id === req.user.id);
102   if (!user) return res.status(404).json({ message: "Usuário não encontrado" });
103   res.json({ data: user });
104 });
105
106 app.get("/api/usuarios", authenticateToken, requireAdmin, (req, res) => {
107   res.json({ data: users });
108 });
109
110 app.get(
111   "/api/contratos/:empresa/:inicio",
112   authenticateToken,
113   requireAdmin,
114   (req, res) => {
115     let { empresa, inicio } = req.params;
116
117     const safeRegex = /^[a-zA-Z0-9\-\_\s\.]+$/;
118     if (!safeRegex.test(empresa) || !safeRegex.test(inicio)) {
119       return res.status(400).json({ message: "Parâmetros inválidos" });
120     }
121
122     const result = getContracts(empresa, inicio);
123     if (result) res.status(200).json({ data: result });
124     else res.status(404).json({ data: "Dados não encontrados" });
125   }
126 );
127
128 class Repository {
129   execute(query, params) {
130     return [
131       {
132         contratoId: 1,
133         empresa: params[0],
134         data_inicio: params[1],
135         detalhes: "Contrato A",
136       },
137     ];
138   }
139 }
140
141 function getContracts(empresa, inicio) {
142   const repository = new Repository();
143   const query = `SELECT * FROM contracts WHERE empresa = ? AND data_inicio = ?`;
144   const result = repository.execute(query, [empresa, inicio]);
145   return result;
146 }
147
148 app.get("/", (req, res) => {
149   res.send("API de Software Seguro - Endpoints disponíveis.");
150 });
151
152 app.listen(port, () => {
153   console.log(`Servidor rodando na porta ${port}`);
154 });

```

A implementação correta de mecanismos de segurança é essencial para garantir proteção robusta em APIs modernas, especialmente aquelas que utilizam autenticação baseada em tokens. Neste contexto, a utilização do padrão JSON Web Token (JWT) oferece vantagens como escalabilidade, desempenho e segurança aprimorada, permitindo o gerenciamento eficiente de autenticação e autorização entre cliente e servidor.

Abaixo serão apresentados os principais pontos relacionados à implementação feita a partir da autenticação JWT na API, escolhida para esta missão, destacando estratégias fundamentais adotadas para garantir integridade, controle de acessos e prevenção contra vulnerabilidades comuns, como ataques de SQL Injection:

1. Autenticação e Geração de Token JWT

- O endpoint de login (POST /api/autenticar/login) autentica o usuário com base nas credenciais fornecidas;
- Após autenticação bem-sucedida, um token JWT é gerado contendo um payload com informações essenciais (id, username e perfil) e possui validade configurada explicitamente para 1 hora;
- O token JWT é retornado no corpo da resposta ao cliente para uso nas requisições subsequentes.

2. Envio do Token JWT via Cabeçalho HTTP

- O cliente deve enviar o token JWT no cabeçalho Authorization das requisições no formato padrão Bearer <token> (ambos nomes utilizados para seguir as melhores práticas da biblioteca utilizada);
- Um middleware (authenticateToken) verifica a existência, formato e validade do token JWT em cada requisição protegida, extraíndo e disponibilizando os dados do usuário para uso interno.

3. Validação Automática do Token

- Todas as requisições feitas a endpoints protegidos executam o middleware authenticateToken, que garante a integridade do token e verifica sua validade (inclusive a data/hora de expiração);
- Caso o token JWT esteja ausente, incorreto ou expirado, a API retorna uma mensagem clara e um código HTTP apropriado.

4. Controle de Acesso Baseado em Perfil

- O middleware requireAdmin assegura que endpoints sensíveis (/api/usuarios e /api/contratos/:empresa/:inicio) sejam acessados exclusivamente por usuários com o perfil administrativo (admin);

- Há também um endpoint dedicado (GET /api/autenticar/usuario) que permite a qualquer usuário autenticado recuperar seus próprios dados sem exigência de perfil especial, facilitando o uso da aplicação para todos os usuários logados.

5. Prevenção de SQL Injection

- Nos endpoints que recebem parâmetros de consulta (como contratos), os parâmetros (empresa e inicio) são rigorosamente validados por meio de expressões regulares seguras, permitindo exclusivamente caracteres seguros (alfanuméricos, hífens, underscores, espaços e pontos);
- As consultas ao banco de dados utilizam consultas parametrizadas (simuladas pela classe Repository e pelo método getContracts), eliminando o risco associado à concatenação direta de strings.

6. Configuração Segura via Variáveis de Ambiente

- A aplicação utiliza process.env.SECRET_KEY, com fallback seguro definido ("chaveSecreta"), assegurando maior segurança e flexibilidade entre ambientes (desenvolvimento, homologação e produção);
- A porta da aplicação também é configurada via variáveis de ambiente, garantindo adaptabilidade a diferentes contextos operacionais.

7. Organização e Melhores Práticas Gerais

- O uso de middlewares dedicados para autenticação (authenticateToken) e controle de autorização (requireAdmin) torna o código modular, de fácil manutenção e compreensão;
- A separação clara entre os endpoints e a lógica de negócios (como doLogin para autenticação e getContracts para consultas) promove uma arquitetura limpa e organizada, facilitando testes e evolução futura do código.

Após fazer o que foi proposto na missão resolvi efetuar alguns testes e modifiquei o código-fonte original para incrementar um front e testar de uma forma mais simplificada, seguem os prints dos testes:

Testar API de Software Seguro			
Usuários:			
ID	Usuário	Email	Perfil
123	user	user@dominio.com	user
124	admin	admin@dominio.com	admin
125	colab	colab@dominio.com	user

Testar API de Software Seguro			
Login			
Usuário:	admin		
Senha:		
<input type="button" value="Entrar"/>			
Endpoints Protegidos			
<input type="button" value="Obter dados do usuário"/>			
<input type="button" value="Obter todos os usuários (admin)"/>			
Consultar Contrato (admin)			
Selecionar Empresa:	<input type="button" value="-- Selecionar uma empresa --"/>		
Selecionar a Data de Início:	<input type="button" value="-- Selecionar uma data --"/>		
<input type="button" value="Consultar Contrato"/>			
<input type="button" value="Logoff"/>			

Testar API de Software Seguro			
Login			
Usuário:	admin		
Senha:		
<input type="button" value="Entrar"/>			
Endpoints Protegidos			
<input type="button" value="Obter dados do usuário"/>			
<input type="button" value="Obter todos os usuários (admin)"/>			
Consultar Contrato (admin)			
Selecionar Empresa:	<input type="button" value="-- Selecionar uma empresa --"/>		
Selecionar a Data de Início:	<input type="button" value="-- Selecionar uma data --"/>		
<input type="button" value="Consultar Contrato"/>			
<input type="button" value="Logoff"/>			

Testar API de Software Seguro			
Logout efetuado com sucesso!			
Login			
Usuário:	<input type="text"/>		
Senha:	<input type="password"/>		
<input type="button" value="Entrar"/>			
Endpoints Protegidos			
<input type="button" value="Obter dados do usuário"/>			
<input type="button" value="Obter todos os usuários (admin)"/>			
Consultar Contrato (admin)			
Selecionar Empresa:	<input type="button" value="-- Selecionar uma empresa --"/>		
Selecionar a Data de Início:	<input type="button" value="-- Selecionar uma data --"/>		
<input type="button" value="Consultar Contrato"/>			
<input type="button" value="Logoff"/>			