Missão Prática do Mundo 5 Nível 5

Disciplina: Software Sem Segurança Não Serve

Aluno: João Rainier de Castro Carvalho

Matrícula: 202208942661

Explicando as alterações:
Login JWT: Gera um token JWT ao logar com validade de 1 hora.
Middleware de Autenticação: authenticateToken verifica a validade do token no cabeçalho.
Middleware de Autorização: authorizeAdmin limita o acesso apenas ao perfil admin.
Proteção contra Injeção: Parâmetros empresa e início são sanitizados para evitar injeção de SQL.
Novo Endpoint de Perfil do Usuário Logado: /api/users/me retorna os dados do usuário logado.

index.html × m5nivel5.js ×

```
const express = require('express');
const bodyParser = require('body-parser');
const crypto = require('crypto');
const jwt = require('jsonwebtoken');
const app = express();
app.use(bodyParser.json());

const JWT_SECRET = 'your_jwt_secret';
const JWT_EXPIRATION = '1h';

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});

// Mock de dados de usuários
const users = [
  { "username": "user", "password": "123456", "id": 123, "email": "user@dominio.com", "perfil": "user" },
  { "username": "admin", "password": "123456789", "id": 124, "email": "admin@dominio.com", "perfil": "admin" }
  { "username": "colab", "password": "123", "id": 125, "email": "colab@dominio.com", "perfil": "user" },
];
```

```
// Função de login para autenticação e geração do token JWT
app.post('/api/auth/login', (req, res) => {
  const credentials = req.body;
  const userData = doLogin(credentials);

  if (userData) {
    const token = jwt.sign(
      { userId: userData.id, perfil: userData.perfil },
      JWT_SECRET,
      { expiresIn: JWT_EXPIRATION }
    );
    res.json({ token });
  } else {
    res.status(401).json({ message: 'Credenciais inválidas' });
  }
});

// Middleware para validar o token JWT
function authenticateToken(req, res, next) {
  const token = req.headers['authorization'];
  if (!token) return res.status(401).json({ message: 'Token não fornecido' });

  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) return res.status(403).json({ message: 'Token inválido ou expirado' });
    req.user = user;
    next();
  });
}
```

```
// Middleware para verificar se o usuário tem perfil 'admin'
function authorizeAdmin(req, res, next) {
  if (req.user.perfil !== 'admin') return res.status(403).json({ message: 'Acesso proibido' });
  next();
}

// Endpoint para recuperação dos dados de todos os usuários (restrito a admin)
app.get('/api/users', authenticateToken, authorizeAdmin, (req, res) => {
  res.status(200).json({ data: users });
});

// Endpoint para recuperação dos contratos, com validação e proteção contra injeção
app.get('/api/contracts/:empresa/:inicio', authenticateToken, authorizeAdmin, (req, res) => {
  const empresa = req.params.empresa.replace(/[^a-zA-Z0-9]/g, ''); // Sanitização de parâmetros
  const inicio = req.params.inicio.replace(/[^a-zA-Z0-9-]/g, '');   // Sanitização de data

  const result = getContracts(empresa, inicio);
  if (result) {
    res.status(200).json({ data: result });
  } else {
    res.status(404).json({ data: 'Dados não encontrados' });
  }
});

// Endpoint para recuperação dos dados do usuário logado (sem restrição a admin)
app.get('/api/users/me', authenticateToken, (req, res) => {
  const userData = users.find(user => user.id === req.user.userId);
  if (userData) {
    res.status(200).json({ data: userData });
  } else {
    res.status(404).json({ message: 'Usuário não encontrado' });
  }
});

// Função para autenticar credenciais
function doLogin(credentials) {
  return users.find(user => user.username === credentials.username && user.password === credentials.password);
}

// Classe simulada para acesso ao banco de dados
class Repository {
  execute(query) {
    return [];
  }
}

// Recuperação dos contratos com proteção contra injeção
function getContracts(empresa, inicio) {
  const repository = new Repository();
  const query = `SELECT * FROM contracts WHERE empresa = ? AND data_inicio = ?`;

  // Mecanismo seguro para executar consultas com parâmetros, como prepared statements
  const result = repository.execute(query);
  return result;
```