Missão Prática | Nível 5 | Mundo 5

Aluno: Rodrigo de O. Alarcon

Matrícula: 202204482321

const express = require('express');

const bodyParser = require('body-parser');

const jwt = require('jsonwebtoken');

const app = express();

app.use(bodyParser.json());

const port = process.env.PORT || 3000;

app.listen(port, () => {

  console.log(`Servcidor rodando na porta: ${port}`);

});

const jwtSecreto = 'nomeEmpresa';

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"},

];

function doLogin(credentials) {

  return users.find(item => credentials.username === item.username && credentials.password === item.password);

}

function authenticateToken(req, res, next) {

  const authHeader = req.headers['authorization'];

  const token = authHeader && authHeader.split(' ')[1];

  if (token == null) return res.sendStatus(401);

  jwt.verify(token, jwtSecreto, (err, user) => {

    if (err) return res.sendStatus(403);

    req.user = {

      id: user.usuario\_id,

      perfil: user.perfil,

      exp: user.exp

    };

    const currentTime = Math.floor(Date.now() / 1000);

    if (user.exp < currentTime) {

      return res.status(403).json({ message: 'Token expirado' });

    }

    next();

  });

}

function authorizeAdmin(req, res, next) {

  if (req.user.perfil !== 'admin') {

    return res.status(403).json({ message: 'Acesso apenas para Administradores' });

  }

  next();

}

app.post('/api/auth/login', (req, res) => {

  const credentials = req.body;

  const userData = doLogin(credentials);

  if (userData) {

    const token = jwt.sign({ usuario\_id: userData.id, perfil: userData.perfil }, jwtSecreto, { expiresIn: '1h' });

    res.json({ token });

  } else {

    res.status(401).json({ message: 'Não autorizado' });

  }

});

// Novo endpoint para recuperar os dados do usuário logado

app.get('/api/user/me', authenticateToken, (req, res) => {

  // Retornar os dados do usuário logado

  const userId = req.user.id;

  const user = users.find(u => u.id === userId);

  if (user) {

    res.status(200).json({ data: user });

  } else {

    res.status(404).json({ message: 'User not found' });

  }

});

app.get('/api/users', authenticateToken, authorizeAdmin, (req, res) => {

  res.status(200).json({ data: users });

});

app.get('/api/contracts/:empresa/:inicio', authenticateToken, authorizeAdmin, (req, res) => {

  const empresa = req.params.empresa;

  const dtInicio = req.params.inicio;

  const result = getContracts(empresa, dtInicio);

  if (result) {

    res.status(200).json({ data: result });

  } else {

    res.status(404).json({ data: 'Dados Não encontrados' });

  }

});

const mysql = require('mysql');

// Configuração da conexão com o banco de dados

const connection = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: 'password',

  database: 'nome\_do\_banco\_de\_dados'

});

// Função para buscar contratos no banco de dados

function getContracts(empresa, inicio) {

  return new Promise((resolve, reject) => {

    // Utilizando prepared statement para evitar SQL Injection

    const query = 'SELECT \* FROM contracts WHERE empresa = ? AND data\_inicio = ?';

    connection.query(query, [empresa, inicio], (error, results, fields) => {

      if (error) {

        reject(error);

      } else {

        resolve(results);

      }

    });

  });

}

class Repository {

  execute(query) {

    return [];

  }

}