Documentação - Missão Prática 5

Italo Gabriel de Almeida Costa
Estácio

Descrição

Este projeto é uma API desenvolvida utilizando Node.js com o framework Express. O objetivo é implementar um sistema de autenticação e controle de acesso de usuários, utilizando tokens JWT, garantindo a segurança na comunicação e proteção contra vulnerabilidades comuns, como SQL Injection. A API também oferece funcionalidades para gerenciamento de usuários e contratos.

Tecnologias Utilizadas

- **Node.js**: Ambiente de execução JavaScript.
- **Express**: Framework para construção da API.
- **JWT (JSON Web Token)**: Tecnologia para autenticação e geração de tokens.
- **express-validator**: Middleware para validação dos parâmetros de entrada.
- **body-parser**: Middleware para análise de dados no corpo da requisição.

Importação das dependências

```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const { body, validationResult } = require('express-validator');
```

Configuração do servidor Express

```
const app = express();
const port = process.env.PORT || 3000;

app.use(bodyParser.json());
```

Configuração do servidor Express

```
const app = express();
const port = process.env.PORT || 3000;

app.use(bodyParser.json());
```

Mock 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 para Gerar o Token JWT

```
function generateToken(user) {
  return jwt.sign(
    { id: user.id, perfil: user.perfil },
    'secretkey',
    { expiresIn: '1h' }
  );
}
```

Endpoint de Login (POST /api/auth/login)

```
app.post('/api/auth/login', [
  body('username').not().isEmpty().withMessage('Username é obrigatório'),
  body('password').not().isEmpty().withMessage('Password é obrigatório')
], (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({ errors: errors.array() });
  }

  const { username, password } = req.body;
  let userData = users.find(user => user.username === username && user.password === password);

  if (userData) {
    const token = generateToken(userData);
    res.json({ token });
  } else {
    res.status(401).json({ message: 'Credenciais inválidas' });
  }
});
```

Endpoint de Recuperação de Dados do Usuário (GET /api/user)

```
app.get('/api/user', (req, res) => {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) {
    return res.status(401).json({ message: 'Token não fornecido' });
  }

  jwt.verify(token, 'secretkey', (err, decoded) => {
    if (err) {
      return res.status(403).json({ message: 'Falha na autenticação do token' });
```

```
    }

    const user = users.find(user => user.id === decoded.id);
    if (user) {
      res.json({ user });
    } else {
      res.status(404).json({ message: 'Usuário não encontrado' });
    }
  });
});
```

Endpoint de Listagem de Usuários (GET /api/users)

```
app.get('/api/users', (req, res) => {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) {
    return res.status(401).json({ message: 'Token não fornecido' });
  }

  jwt.verify(token, 'secretkey', (err, decoded) => {
    if (err) {
      return res.status(403).json({ message: 'Falha na autenticação do token' });
    }

    if (decoded.perfil !== 'admin') {
      return res.status(403).json({ message: 'Acesso negado. Somente administradores podem
visualizar os usuários.' });
    }

    res.json({ users });
  });
});
```

Endpoint de Recuperação de Contratos (GET /api/contracts/ /)

```
app.get('/api/contracts/:empresa/:inicio', [
  body('empresa').isString().notEmpty().withMessage('Empresa não pode estar vazio'),
  body('inicio').isDate().withMessage('Data de início inválida')
], (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({ errors: errors.array() });
  }

  const { empresa, inicio } = req.params;

  const contracts = getContracts(empresa, inicio);
  if (contracts && contracts.length > 0) {
    res.json({ contracts });
```

```
  } else {
    res.status(404).json({ message: 'Nenhum contrato encontrado' });
  }
});
```

Função Mock para Contratos

```
function getContracts(empresa, inicio) {
  return [
    { id: 1, empresa: empresa, inicio: inicio, detalhes: 'Contrato A' },
    { id: 2, empresa: empresa, inicio: inicio, detalhes: 'Contrato B' }
  ];
}
```

Inicialização do Servidor

```
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```