at_S7_A1_SL7_backend

Roteiro de atividade prática

Nome: _____ Turma: _____

Título da atividade: implementação de autenticação e autorização no *back-end*

Explicação dos conceitos

- **Autenticação:** é o processo de verificar a identidade de um usuário. Em uma aplicação, a autenticação geralmente ocorre quando um usuário faz login fornecendo credenciais como nome de usuário e senha. Se as credenciais forem válidas, o usuário recebe um *token* de autenticação ou sessão.
- **Autorização:** o processo de verificar se um usuário autenticado tem permissão para acessar determinados recursos ou realizar certas ações. Ela ocorre após a autenticação e é utilizada para garantir que o usuário tenha permissões adequadas para acessar API ou realizar operações específicas.

Objetivos

- **Implementar um sistema básico de autenticação** utilizando *tokens* JWT (JSON Web Tokens), permitindo que os usuários façam login em um sistema *back-end*.
- **Implementar autorização baseada em papéis** (*roles*), garantindo que usuários autenticados com diferentes permissões (como "admin" ou "user") possam acessar recursos diferentes.
- **Demonstrar como proteger rotas** em um servidor *back-end*, exigindo autenticação e autorizando o acesso com base nos papéis dos usuários.

Contexto:

Você está desenvolvendo uma API para um sistema de gerenciamento de tarefas. No sistema, os usuários devem se autenticar para acessar suas tarefas e realizar operações como adicionar, atualizar e excluir tarefas. Além disso, alguns usuários terão permissões administrativas, que lhes permitirão gerenciar todos os usuários e suas tarefas.

Sua tarefa é implementar um sistema de **autenticação e autorização** que atenda aos seguintes requisitos:

- Os usuários devem poder se autenticar utilizando credenciais (nome de usuário e senha) e receber um ***token* JWT**.
- O sistema deve proteger as rotas de gerenciamento de tarefas, exigindo que o usuário esteja autenticado.
- Algumas rotas devem ser acessíveis apenas por usuários com o papel de **"admin"**.

Tarefa:

Passos para implementação:

1. **Criar um sistema de autenticação com JWT:**
    - Implemente uma rota de login que aceite o nome de usuário e senha, valide essas credenciais e gere um **token JWT** para o usuário autenticado. Exemplo de resposta da rota de login:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

2. **Proteger rotas com *middleware* de autenticação:**
    - Crie um *middleware* que valide o ***token* JWT** enviado no cabeçalho da requisição (authorization: Bearer <token>) antes de permitir o acesso às rotas protegidas.

3. **Implementar a autorização baseada em papéis (*roles*):**

    - Adicione um atributo de **papel** (role) ao token JWT gerado, como "admin" ou "user".
    - Crie um *middleware* adicional que verifique se o usuário autenticado tem permissão (com base no papel) para acessar determinadas rotas.

Exemplo de rotas: POST/login: rota para autenticação de usuários (gera e retorna o *token* JWT).

- **GET /tasks:** retorna as tarefas do usuário autenticado (acesso permitido para todos os usuários autenticados).
- **POST /admin/users:** rota disponível apenas para administradores, permitindo a criação de novos usuários (acesso restrito ao papel "admin").

Código de exemplo (Node.js/Express com JWT):

```
const jwt = require('jsonwebtoken');
const express = require('express');
const app = express();
app.use(express.json());

const SECRET_KEY = 'my_secret_key';

// Dados fictícios para exemplificar
const users = [
  { id: 1, username: 'user1', password: 'password1', role: 'user' },
  { id: 2, username: 'admin', password: 'adminpass', role: 'admin' }
];

// Função para gerar token JWT
function generateToken(user) {
  return jwt.sign({ id: user.id, role: user.role }, SECRET_KEY, { expiresIn: '1h' });
}
```

```
// Middleware de autenticação
function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) return res.sendStatus(401);

  jwt.verify(token, SECRET_KEY, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}

// Middleware de autorização baseado em papéis
function authorizeRole(role) {
  return (req, res, next) => {
    if (req.user.role !== role) {
      return res.sendStatus(403);
    }
    next();
  };
}

// Rota de login
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username && u.password ===
password);

  if (user) {
    const token = generateToken(user);
    res.json({ token });
  } else {
    res.sendStatus(401);
```

```
  }
});

// Rota protegida (acesso para qualquer usuário autenticado)
app.get('/tasks', authenticateToken, (req, res) => {
  res.json({ tasks: ['Tarefa 1', 'Tarefa 2'] });
});

// Rota apenas para admins
app.post('/admin/users', authenticateToken, authorizeRole('admin'), (req, res)
=> {
  res.send('Usuário criado com sucesso.');
});

app.listen(3000, () => {
  console.log('Servidor rodando na porta 3000');
});
```

Perguntas para responder

1. Qual é a diferença entre autenticação e autorização em uma aplicação *back-end*?

2. Por que o uso de *tokens* JWT é popular em sistemas de autenticação?

3. Como você pode garantir que apenas usuários com o papel de "admin" tenham acesso a rotas específicas no sistema?