re S7_A2_SL6 backend

Roteiro de atividade prática

Nome: _____ Turma: _____

Título da atividade: integração com API externas e implementação de autenticação e autorização

Pontos principais do conteúdo:

- **Integração com API externas:** no desenvolvimento de sistemas *back-end*, é comum interagir com API externas para obter dados ou funcionalidades de terceiros. Isso pode incluir API de pagamento, API de redes sociais ou até mesmo API de geolocalização. Uma integração eficiente deve ser segura e gerenciar respostas de erro de maneira adequada, garantindo que o sistema funcione mesmo diante de falhas temporárias na API externa.
- **Implementação de autenticação e autorização:** autenticação é o processo de validar a identidade do usuário (como login e geração de *tokens* JWT), enquanto a autorização define o que o usuário autenticado pode acessar ou fazer no sistema. Ao integrar API externas, é comum precisar autenticar os usuários e autorizar o uso de certas funcionalidades, como chamadas a serviços externos protegidos.

Objetivos

- **Implementar a integração com uma API externa** (simulada) e garantir que o sistema possa fazer chamadas HTTP para consumir dados externos.
- **Adicionar autenticação e autorização** para proteger a API local e garantir que apenas usuários autenticados possam acessar certas funcionalidades.
- **Implementar um sistema que integre a autenticação de uma API externa** e autorize o acesso de acordo com o papel (*role*) dos usuários.

Contexto:

Você está desenvolvendo uma API de gerenciamento de projetos que precisa se integrar com uma API externa para obter informações sobre o clima da cidade onde os projetos estão sendo executados. Os usuários do sistema deverão se autenticar para acessar as funcionalidades de gerenciamento de projetos e também a integração com a API de clima. Além disso, apenas usuários com o papel "admin" poderão atualizar os dados de um projeto.

Sua tarefa é:

- **implementar a autenticação** dos usuários utilizando *tokens* JWT;
- **proteger as rotas de acesso ao gerenciamento de projetos**, permitindo que apenas usuários autenticados as acessem;
- **integrar a API de clima externa**, permitindo que a aplicação faça uma chamada a essa API e retorne as informações meteorológicas para a cidade onde o projeto está localizado;
- **implementar autorização**, garantindo que somente administradores possam atualizar as informações de um projeto.

Tarefa:

1. **Configurar a autenticação com JWT:**

- Implemente uma rota de login que valide as credenciais do usuário e retorne um ***token* JWT**.
- Adicione o papel (*role*) "admin" ou "user" no *token* para diferenciar os tipos de usuários.

2. **Proteger rotas com *middleware* de autenticação:**

- Crie um *middleware* que valide o ***token* JWT** enviado no cabeçalho Authorization antes de permitir que o usuário acesse rotas protegidas.

3. **Integrar com a API externa (clima):**

- Faça uma requisição HTTP para uma API de clima (simulada ou real, como OpenWeather API), utilizando a cidade do projeto para buscar informações. Exemplo de chamada à API de clima (HTTP GET):

GET /weather?city=São Paulo

4. **Implementar autorização para administradores:**

- Proteja a rota de atualização de projetos, permitindo que apenas usuários com o papel "admin" possam acessar essa funcionalidade.

Exemplo de rotas:

- **POST /login:** sutenticação do usuário (retorna o *token* JWT).
- **GET /projects:** retorna os projetos do usuário autenticado.
- **GET /projects/**

/weather: retorna o clima para a cidade do projeto (integração com API de clima).

- **PUT /projects/:id:** atualiza o projeto (apenas para usuários com papel "admin").

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

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

const SECRET_KEY = 'my_secret_key';

// Usuários fictícios para autenticação
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 de projetos protegida (apenas usuários autenticados)
app.get('/projects', authenticateToken, (req, res) => {
  res.json({ projects: ['Projeto A', 'Projeto B'] });
});

// Rota para obter o clima de um projeto (integração com API externa)
app.get('/projects/:id/weather', authenticateToken, (req, res) => {
  const city = 'São Paulo';  // Exemplo: cidade do projeto

axios.get(`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid
=YOUR_API_KEY`)
    .then(response => {
      res.json({ weather: response.data });
    })
    .catch(error => {
      res.status(500).json({ error: 'Erro ao obter informações do clima' });
    });
});

// Rota para atualizar projeto (apenas admins)
app.put('/projects/:id', authenticateToken, authorizeRole('admin'), (req, res) => {
  res.send('Projeto atualizado com sucesso.');
});

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

Perguntas para responder

1. Como a autenticação com JWT protege o sistema e quais são os benefícios dessa abordagem?

2. Por que é importante usar papéis (*roles*) na autorização de usuários em um sistema *back-end*?

3. Como a integração com API externas pode melhorar as funcionalidades de um sistema *back-end* e o que deve ser considerado ao integrá-las?