

# Trilha 04

*Ferramentas de Testes*

## Instruções para a melhor prática de Estudo

1. **Leia atentamente todo o conteúdo:** Antes de iniciar qualquer atividade, faça uma leitura detalhada do material fornecido na trilha, compreendendo os conceitos e os exemplos apresentados.
2. **Não se limite ao material da trilha:** Utilize o material da trilha como base, mas busque outros materiais de apoio, como livros, artigos acadêmicos, vídeos, e blogs especializados. Isso enriquecerá o entendimento sobre o tema.
3. **Explore a literatura:** Consulte livros e publicações reconhecidas na área, buscando expandir seu conhecimento além do que foi apresentado. A literatura acadêmica oferece uma base sólida para a compreensão de temas complexos.
4. **Realize todas as atividades propostas:** Conclua cada uma das atividades práticas e teóricas, garantindo que você esteja aplicando o conhecimento adquirido de maneira ativa.
5. **Evite o uso de Inteligência Artificial para resolução de atividades:** Utilize suas próprias habilidades e conhecimentos para resolver os exercícios. O aprendizado vem do esforço e da prática.
6. **Participe de debates:** Discuta os conteúdos estudados com professores, colegas e profissionais da área. O debate enriquece o entendimento e permite a troca de diferentes pontos de vista.
7. **Pratique regularmente:** Não deixe as atividades para a última hora. Pratique diariamente e revise o conteúdo com frequência para consolidar o aprendizado.
8. **Peça feedback:** Solicite o retorno dos professores sobre suas atividades e participe de discussões sobre os erros e acertos, utilizando o feedback para aprimorar suas habilidades.

Essas instruções são fundamentais para garantir um aprendizado profundo e eficaz ao longo das trilhas.

## 1. Postman

O Postman é uma ferramenta amplamente utilizada para desenvolvimento, teste e automação de APIs. Ele permite que desenvolvedores e testadores enviem requisições HTTP para serviços web, inspecionem respostas e validem o funcionamento das APIs de forma eficiente e intuitiva.

**Exemplo:** Vamos criar um teste simples utilizando o Postman para validar um endpoint de uma API fictícia:

### Enviando uma requisição GET

Imagine que temos uma API que retorna informações de usuários a partir do endpoint:

GET

▼

https://jsonplaceholder.typicode.com/users/1

### Resultado obtido

{ } JSON ▼

▶ Preview

🔗 Visualize ▼

```

1  {
2    "id": 1,
3    "name": "Leanne Graham",
4    "username": "Bret",
5    "email": "Sincere@april.biz",
6    "address": {
7      "street": "Kulas Light",
8      "suite": "Apt. 556",
9      "city": "Gwenborough",
10     "zipcode": "92998-3874",
11     "geo": {
12       "lat": "-37.3159",
13       "lng": "81.1496"
14     }
15   },

```

### Criando um teste automatizado no Postman

No Postman, após enviar a requisição acima, podemos adicionar um teste automatizado na aba Tests, utilizando JavaScript:

Headers (7) Body Scripts ● Settings

```
pm.test("Verificar se o status da resposta é 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Verificar se o retorno contém o campo 'name'", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("name");
});
```

## Executando os testes

Ao executar a requisição, o Postman verificará automaticamente se o status da resposta é 200 e se o retorno contém a propriedade name. Caso algum dos testes falhe, o Postman indicará o erro no painel de testes.

## 2. Insomnia

O Insomnia é uma ferramenta popular para desenvolvimento, teste e automação de APIs. Ele permite que desenvolvedores e testadores enviem requisições HTTP, inspecionem respostas e validem o funcionamento das APIs de forma prática e eficiente.

**Exemplo:** Vamos criar um teste simples utilizando o Insomnia para validar um endpoint de uma API fictícia.

### Enviando uma requisição GET

Imagine que temos uma API que retorna informações de usuários a partir do endpoint:

GET <https://jsonplaceholder.typicode.com/users/1>

Send

### Resultado obtido

```
1 {
2   "id": 1,
3   "name": "Leanne Graham",
4   "username": "Bret",
5   "email": "Sincere@april.biz",
6   "address": {
7     "street": "Kulas Light",
8     "suite": "Apt. 556",
9     "city": "Gwenborough",
10    "zipcode": "92998-3874",
11    "geo": {
12      "lat": "-37.3159",
13      "lng": "81.1496"
14    }
15  },
```

### Criando um teste automatizado no Insomnia

No Insomnia, após enviar a requisição acima, podemos validar a resposta utilizando scripts na aba **Scripts > After-response**, com JavaScript:

```
// Verificar se o status da resposta é 200
insomnia.test("Verificar se o status da resposta é 200", function () {
  insomnia.response.to.have.status(200);
});

// Verificar se a resposta contém o campo 'name'
insomnia.test("Verificar se o retorno contém o campo 'name'", function () {
  var jsonData = insomnia.response.json();
  insomnia.expect(jsonData).to.have.property("name");
});
```

### Executando os testes

Ao executar a requisição, o Insomnia verificará automaticamente se o status da resposta é 200 e se o retorno contém a propriedade name. Caso algum dos testes falhe, o Insomnia indicará o erro no painel de testes.

---

## 3. Apache JMeter

O Apache JMeter é uma ferramenta de código aberto desenvolvida pela Apache Software Foundation para a realização de testes de carga, desempenho e estresse em aplicações web, APIs, bancos de dados e outros serviços. Ele permite simular múltiplas requisições simultâneas, analisando a capacidade e o comportamento da aplicação sob diferentes níveis de carga.

**Exemplo:** Imaginemos que seja necessário realizar um teste de desempenho de uma API REST onde nossas métricas devem ser de que 5 mil usuários estejam realizando requisições, a API não pode cair ou ter instabilidades como resultado esperado.

#### Criar um Test Plan:

- Abra o JMeter e clique com o botão direito em **Test Plan > Add > Thread Group**.
- Configure o número de usuários simultâneos (exemplo: 5 usuários).

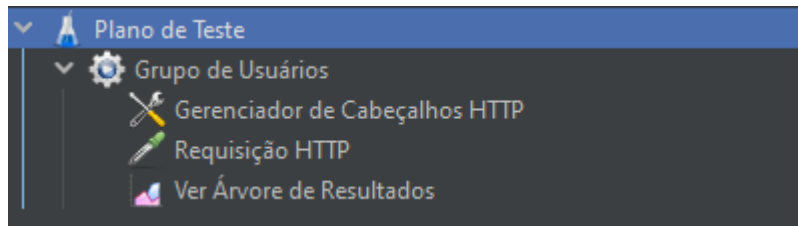
#### Adicionar um Sampler HTTP Request:

- Clique com o botão direito no **Thread Group > Add > Sampler > HTTP Request**.
- Configure o campo **"Server Name or IP"** com a URL da API (exemplo: jsonplaceholder.typicode.com).
- Escolha o método **GET**.

### Adicionar um Listener para visualizar os resultados:

- Clique com o botão direito no **Thread Group** > **Add** > **Listener** > **View Results Tree**.
- Execute o teste clicando em Run (Ctrl+R) e analise os tempos de resposta e possíveis falhas.

### Estrutura ao final do processo:



### Criando o Grupo de Usuários:

Esta é primeira fase do processo de configuração, **Test Plan** > **Add** > **Thread Group**, que consiste em definir algumas informações, como:

O Número de Usuários (Threads) representa a quantidade de usuários virtuais que o JMeter irá simular fazendo requisições para o sistema ou API que está sendo testado.

- Permite testar como a aplicação se comporta sob diferentes quantidades de usuários simultâneos.

O Tempo de Inicialização (Ramp-Up Period) define quanto tempo o JMeter levará para criar todas as threads (usuários virtuais).

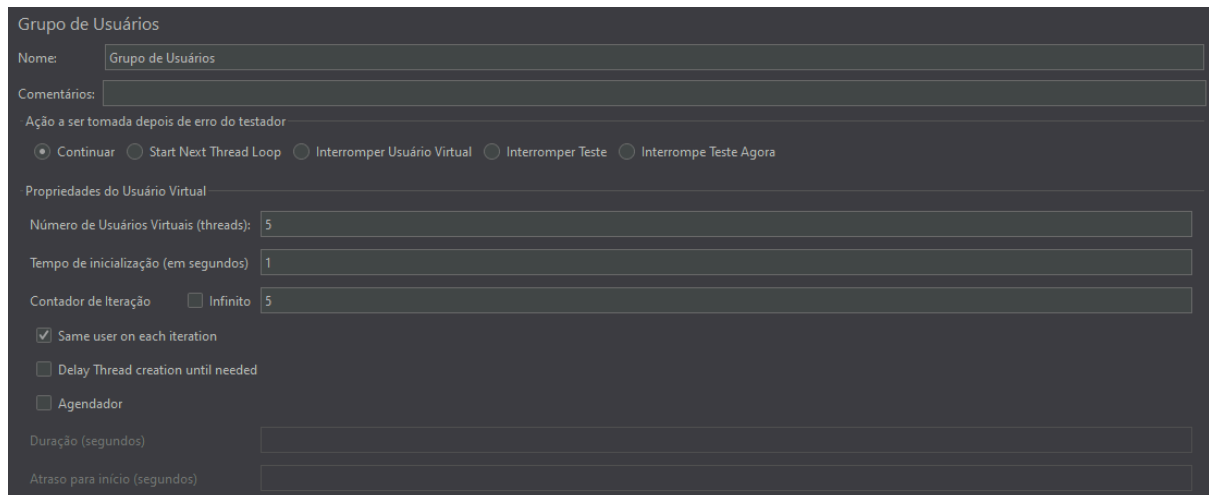
- Evita que todos os usuários iniciem ao mesmo tempo, distribuindo a carga ao longo do tempo.
- Simula um crescimento gradual do tráfego, o que reflete melhor cenários reais.
- Permite testar se o sistema consegue lidar com um aumento progressivo de usuários.

O Contador de Iteração (Loop Count) define quantas vezes cada usuário (thread) repetirá as solicitações dentro do teste.

- Simula quantas vezes um usuário faz determinada ação antes de encerrar sua execução.
- Controla o número total de requisições durante o teste.
- Permite simular cenários onde os mesmos usuários continuam interagindo com o sistema.

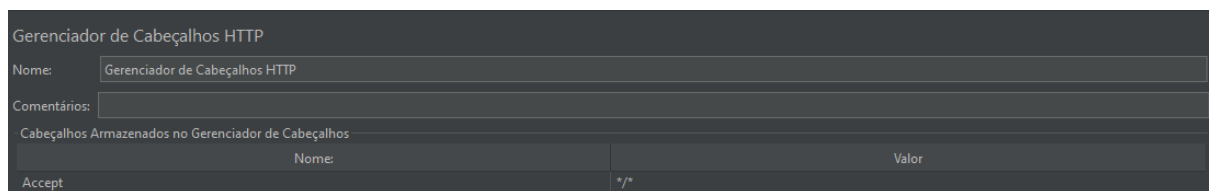
## Criando o Grupo de Usuário:

Processo padrão inicial, segue o exemplo abaixo:



## Criando o Header:

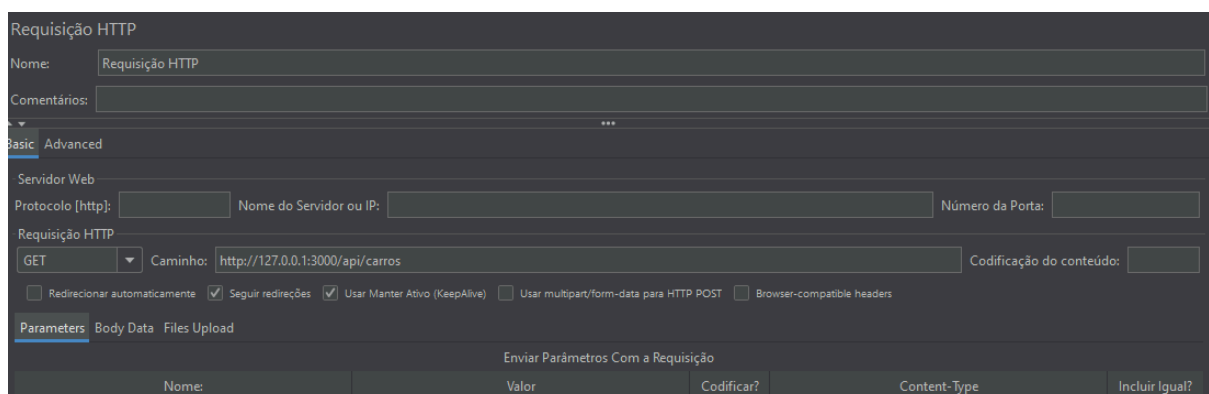
Aqui definimos o cabeçalho das requisições, serve para definir tudo o que for necessário que precisa conter para nossas requisições como por exemplo Autenticação e Autorização.



Nome	Valor
Accept	*/*

## Criando a Requisição HTTP:

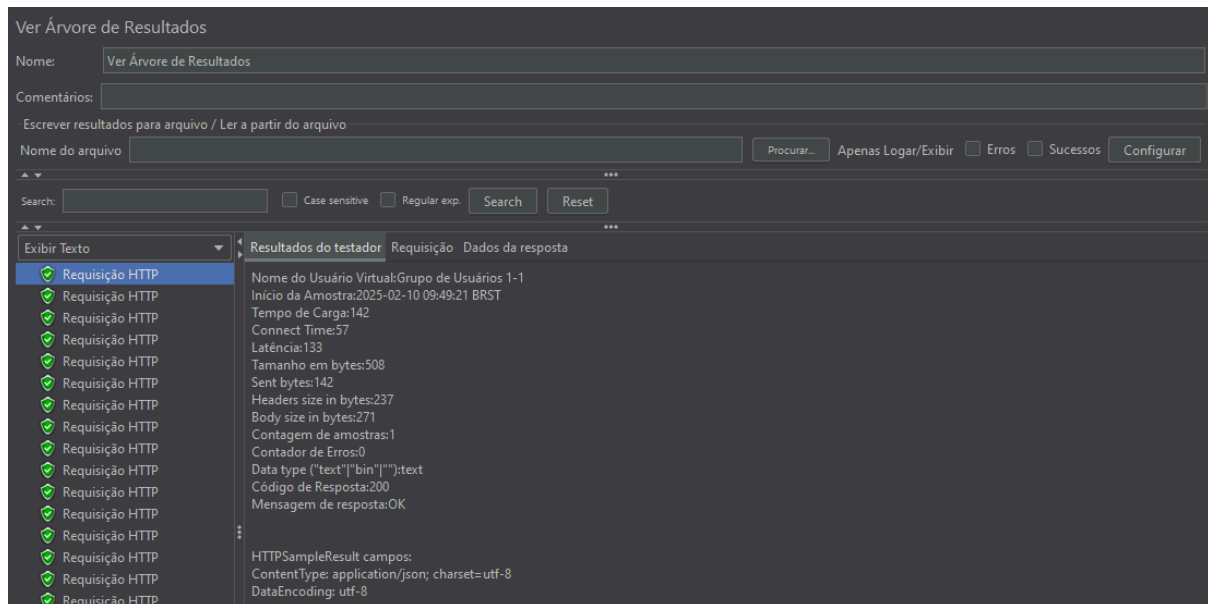
Definição dos endpoints que serão requisitados.



Nome	Valor	Codificar?	Content-Type	Incluir Igual?

## Criando a árvore de resultados:

Configuração importante para verificação das requisições que estão sendo testadas.



## 4. OWASP ZAP

O OWASP ZAP (Zed Attack Proxy) é uma ferramenta de código aberto desenvolvida pela OWASP (Open Web Application Security Project) para a realização de testes de segurança em aplicações web. Ele permite detectar vulnerabilidades comuns, como injeção de SQL, XSS (Cross-Site Scripting) e falhas de autenticação.

**Exemplo:** Imagine que estamos testando um sistema de login de um site que pode estar vulnerável a ataques de injeção SQL. Vamos utilizar o OWASP ZAP para interceptar as requisições e analisar possíveis falhas de segurança.

### Passos para criar o teste no OWASP ZAP

#### Instalar o OWASP ZAP:

- Baixe o OWASP ZAP do site oficial: [OWASP ZAP](https://owasp.org/zap2/).
- Instale e inicie a ferramenta.

#### Capturar as requisições do site

- No OWASP ZAP, vá até a aba "**Quick Start**" e clique em "**Manual Explore**".
- Insira a URL do site que deseja testar e clique em "**Start**".
- O OWASP ZAP começará a monitorar as requisições entre o navegador e o servidor.



## Realizar um Ataque de Fuzzing para Testar SQL Injection

- Vá até a aba Sites e encontre a página de login do site.
- Localize a requisição POST do formulário de login e clique com o botão direito.
- Selecione **"Attack"** → **"Fuzz"**.
- No campo de **"Username"** ou **"Password"**, clique em **"Add Payloads"**.
- Escolha **"File Fuzzers"** e selecione a opção **"SQL Injection"**.
- Clique em **"Start Fuzzing"** e aguarde o processo.

## Analisar os Resultados

- Após o ataque, verifique os códigos de resposta HTTP e mensagens de erro.
- Se houver uma resposta inesperada, como um erro de SQL ou um retorno suspeito, o site pode estar vulnerável.

## Exemplo de resposta suspeita:

```
SQL Syntax Error: "SELECT * FROM users WHERE username = 'admin' OR 1=1 --'"
```

## Outro Exemplo Prático

Vamos realizar um teste simples de segurança em um site utilizando o OWASP ZAP.



## Configurar um teste de segurança:

- No OWASP ZAP, insira a URL da aplicação a ser testada no campo **"URL to Attack"**.
- Clique em **"Attack"** para iniciar a varredura, abaixo o exemplo:

URL to attack:

Use traditional spider: ☒

Use ajax spider:  with

Progresso: Ataque completo - veja a aba de alertas para ver os detalhes dos problemas encontrados

## Analisar os resultados:

- Após o escaneamento, visualize as vulnerabilidades detectadas na aba **"Alerts"**.
- Detalhes sobre cada vulnerabilidade e sugestões de correção estarão disponíveis, abaixo o exemplo:

- ▼ Alertas (9)
  - > Configuração Incorreta Entre Domínios (25)
  - > Content Security Policy (CSP) Header Not Set (4)
  - > Missing Anti-clickjacking Header (2)
  - > Strict-Transport-Security Header Not Set (25)
  - > X-Content-Type-Options Header Missing (23)
  - > Divulgação de Informações - Comentários Suspeitos (5)
  - > Modern Web Application (4)
  - > Re-examine Cache-control Directives (4)
  - > Retrieved from Cache (31)

## 5. Swagger/OpenAPI

O Swagger é um conjunto de ferramentas que facilitam a criação, documentação, teste e manutenção de APIs. O OpenAPI é a especificação padrão para descrever APIs RESTful de forma estruturada e legível por humanos e máquinas. Juntos, eles ajudam no desenvolvimento e consumo de APIs de forma eficiente.

**Exemplo:** Acesse <https://editor.swagger.io/> para usar a versão online ou instale localmente.

Após realizar o primeiro acesso você irá se deparar com uma estrutura toda pronta que vem de exemplo, basta seguir a documentação e os padrões de escrita para a criação da documentação e testes da API.

### Exemplo Prático

Vamos criar uma documentação de API simples utilizando o Swagger/OpenAPI.

- Acesse <https://editor.swagger.io/> para usar a versão online.

### Criar um arquivo YAML com a especificação da API:

Aqui fica todo nosso script de configuração para nossa API, para escrita existe alguns padrões estabelecidos, conforme exemplo abaixo:

```

openapi: 3.0.0
info:
  title: Exemplo de API
  description: Uma API simples para demonstração do Swagger/OpenAPI
  version: 1.0.0
servers:
  - url: https://exemploapi.com/users/1
paths:
  /usuario:
    get:
      summary: Obtém um usuário
      responses:
        '200':
          description: Usuário retornado com sucesso
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                    example: 1
                  name:
                    type: string
                  username:
                    type: string
                  email:
                    type: string

```

### Resultado da estrutura:

Na própria ferramenta conforme vamos escrevendo os testes ao lado possui uma parte onde o script é renderizado em uma forma de documentação, segue o exemplo abaixo:

## Exemplo de API 1.0.0 OAS 3.0

Uma API simples para demonstração do Swagger/OpenAPI

Servers

https://exemploapi.com/users/1

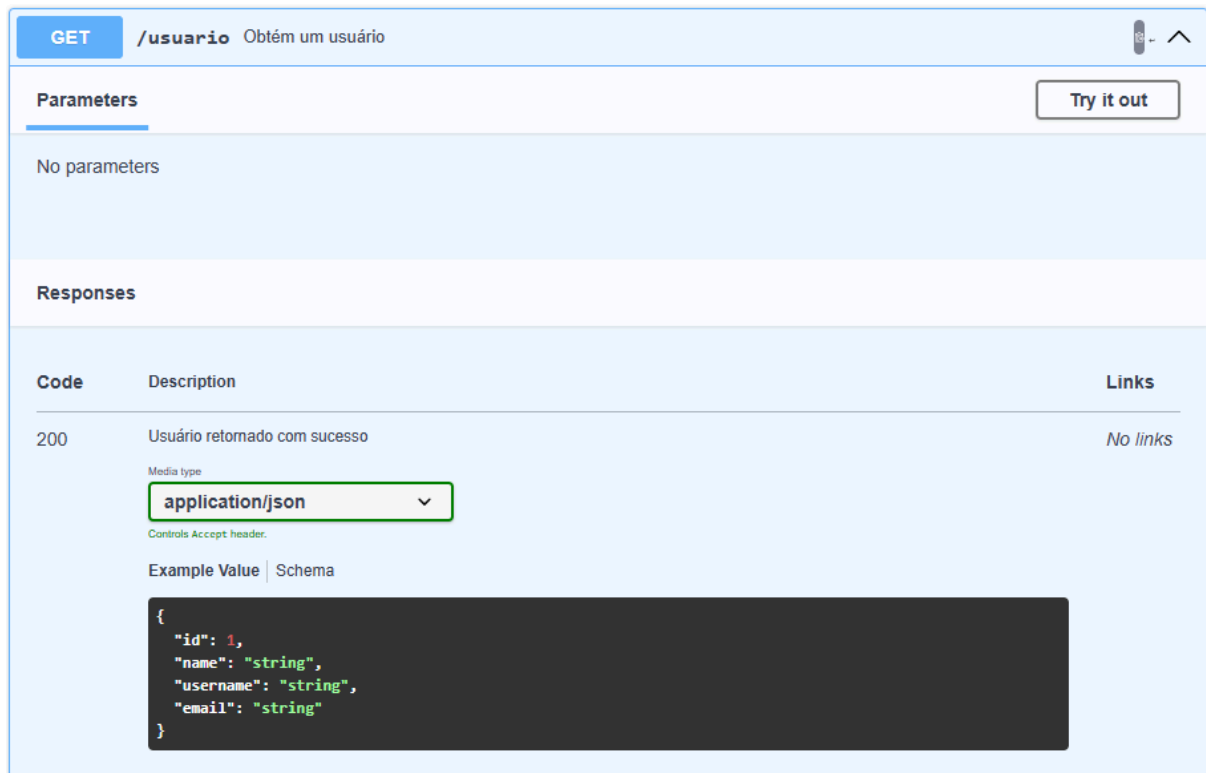
default

GET

/usuario Obtém um usuário

## Exemplo de como ficaria na interface:

Aqui vemos nosso primeiro endpoint documentado, também é possível realizar os testes diretamente na sua interface:



**GET** /usuario Obtém um usuário

**Parameters** Try it out

No parameters

**Responses**

Code	Description	Links
200	Usuário retornado com sucesso	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": 1,
  "name": "string",
  "username": "string",
  "email": "string"
}
```

## Lista de Exercícios de Fixação

### Questões Dissertativas

1. O Postman é amplamente utilizado para testar APIs e automatizar requisições HTTP. Explique como ele pode ser utilizado para testes automatizados de API e quais são as principais vantagens de usá-lo no processo de desenvolvimento de software.
2. O Insomnia é uma ferramenta intuitiva para testar APIs. Compare o uso do Insomnia com o Postman, destacando semelhanças e diferenças entre as duas ferramentas. Em qual situação você escolheria uma em vez da outra?
3. O Apache JMeter é uma ferramenta essencial para testes de carga e estresse em aplicações web. Explique como ele pode ser utilizado para testar o desempenho de uma API e quais métricas são analisadas durante um teste de carga.
4. O OWASP ZAP é utilizado para testes de segurança em aplicações web. Explique como essa ferramenta pode ser usada para identificar vulnerabilidades em um sistema e cite três exemplos de vulnerabilidades que ela pode detectar.
5. O Swagger/OpenAPI facilita a documentação e o consumo de APIs. Explique a importância de documentar uma API utilizando Swagger/OpenAPI e como isso melhora a colaboração entre desenvolvedores e testadores.