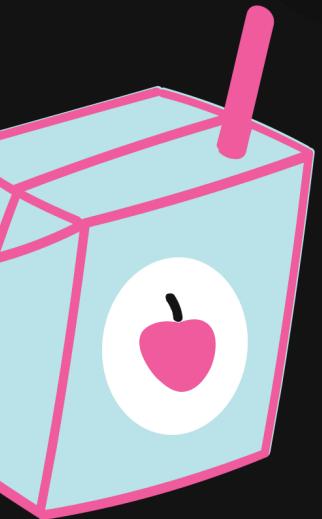




LET'S PLAY?

# API'S

**START**



COOKS

# Conceito de API

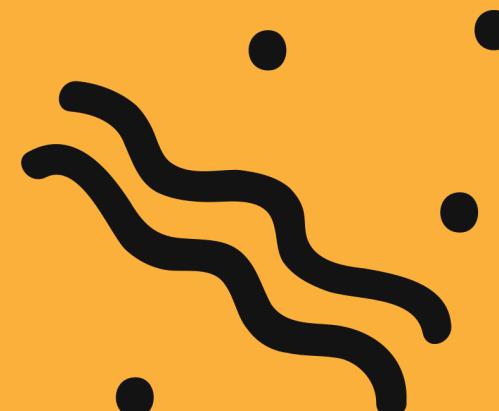


1

API (Application Programming interface) é um conjunto de protocolos, rotinas e ferramentas voltadas à construção de softwares e aplicativos, pautadas em pontos de acesso(endpoints) que então permitem que um software realize ações específicas em outro software, possibilitando a comunicação e interação entre diferentes sistemas, plataformas e até mesmo linguagens de programação.

2

Basicamente API seria um programa que funcionaria como uma ponte para a comunicação entre diferentes aplicações, assim então desenvolvedores podem acessar recursos e dados de determinado sistema. Em sistemas modernos o uso das APIs são indispensáveis para a criação de diferentes tecnologias e interação com o usuário



# Tipos de APIs

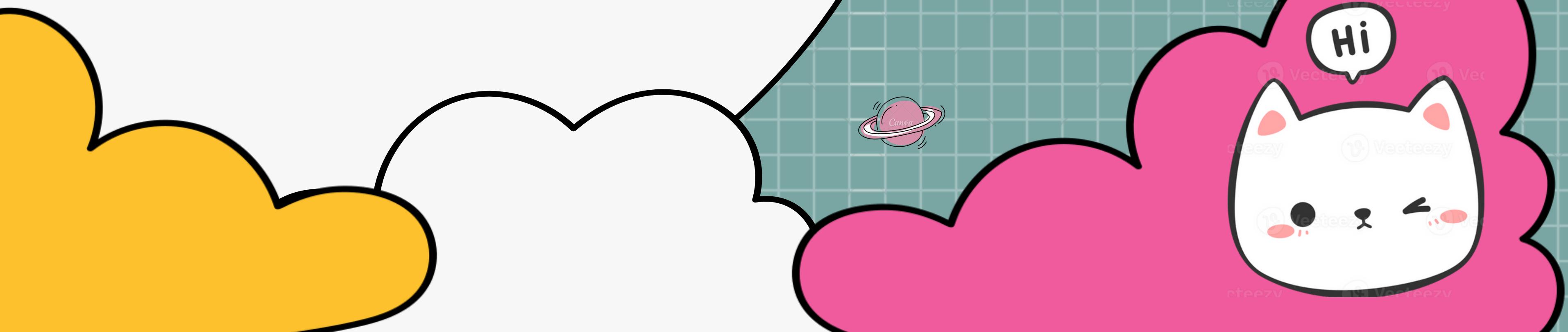
## Soap

O SOAP é capaz de manipular solicitações por meio de todos os protocolos da camada de aplicação do modelo OSI, incluindo HTTP, SMTP (para email), TCP e outros, o que lhe confere uma vantagem em relação a outros modelos arquiteturais.

As solicitações e respostas do SOAP seguem o formato XML e não são armazenadas em cache pelo navegador. Além disso, o SOAP suporta arquiteturas tanto stateful (onde o contexto da solicitação atual depende das anteriores) quanto stateless (onde cada solicitação é tratada de forma independente)

# Rest

O REST (Representational State Transfer) é um conjunto de princípios de design que visa padronizar o formato das solicitações HTTP. Ao contrário de outros modelos, o REST oferece mais liberdade no desenvolvimento e permite a utilização de diferentes formatos de solicitação, incluindo JSON, HTML, XML e plain text. A flexibilidade do REST na implementação de suas diretrizes resulta em APIs mais leves, rápidas e escaláveis, tornando o desenvolvimento mais ágil.



# GRAPHQL

3

O GraphQL é uma linguagem de consulta e ambiente de execução voltado para servidores, projetado para fornecer dados precisos para as interfaces de programação de aplicações (APIs). Com o GraphQL, as consultas sempre retornam resultados previsíveis, tornando as aplicações mais rápidas e estáveis porque controlam os dados que obtêm. Além disso, o GraphQL fornece uma descrição completa dos dados em sua API, permitindo a evolução das APIs ao longo do tempo e o uso de ferramentas poderosas. Como alternativa à arquitetura REST, o GraphQL permite que os desenvolvedores façam consultas que extraem dados de várias fontes em uma única chamada de API. O GraphQL também pode ser utilizado no ambiente GraphiQL, um IDE popular.

# BIBLIOTECAS

Uma biblioteca é geralmente uma implementação concreta das regras de uma API. Isso significa que a biblioteca fornece a funcionalidade definida pela API, mas de uma forma mais tangível e executável. Embora não seja necessário conhecer os detalhes da implementação para usar a biblioteca, é importante que ela siga as regras da API de forma consistente.

Ao contrário da API, a biblioteca não precisa ter uma implementação estável. Ela pode mudar com o tempo e evoluir para melhor atender às necessidades dos usuários. No entanto, usar partes da biblioteca que não fazem parte da API pode trazer riscos, já que essas partes podem mudar sem aviso prévio.

Ao usar uma biblioteca, você está usando uma ferramenta que oferece funcionalidades específicas. Você escolhe quais partes da biblioteca deseja usar e como usá-las em sua aplicação.

# Web Service

O Web Service é uma interface que permite a comunicação via rede e, portanto, também é considerado uma API, o que pode gerar confusão entre os termos.

A API geralmente usa a rede HTTP como meio de comunicação, que é amplamente utilizado para esse fim. Quando se trata de enviar dados pela rede, o Web Service é a opção mais comum, pois utiliza protocolos como SOAP, REST e XML-RPC para se comunicar.

No entanto, é importante observar alguns detalhes, como o fato de que as funções de um programa são geralmente encapsuladas por uma API.

- Web service é uma aplicação enquanto a API facilita a interface direta com um aplicativo.
- Nem todas as APIs são Web services, porém, todos os Web Services são APIs.
- Web Services não executam todas as tarefas realizadas ou não de uma API.
- A API pode utilizar qualquer estilo de comunicação, porém o serviço Web só executa apenas três estilos de comunicação que são eles SOAP, REST e XML-RPC.
- A API não precisa de uma rede para seu funcionamento acontecer, enquanto o Web Server depende disto.

3

## SDK

Um SDK, ou Software Development Kit, é um conjunto de ferramentas, bibliotecas, documentação e exemplos de código que auxiliam os desenvolvedores na criação de aplicativos, softwares e sistemas. Em geral, um SDK é disponibilizado pelo fabricante de uma plataforma ou tecnologia para que outros desenvolvedores possam integrá-la em suas próprias soluções.

Com um SDK, os desenvolvedores podem ter acesso a recursos avançados, como APIs e kits de desenvolvimento para dispositivos móveis, por exemplo. Além disso, o SDK pode incluir ferramentas de teste, emuladores e outras funcionalidades que agilizam o desenvolvimento e garantem a compatibilidade com a plataforma ou tecnologia. Assim, um SDK pode ser essencial para que terceiros desenvolvam produtos e serviços para uma determinada plataforma ou tecnologia, ampliando assim suas funcionalidades e possibilidades de uso.

# HTTP e HTTPS

HTTP

Protocolo de comunicação aberto, no qual os dados são transferidos entre o navegador e o servidor em formato de texto simples, sem qualquer proteção. Assim, qualquer invasor que consiga se instalar em algum ponto dessa transferência poderá capturar seus dados sem maiores dificuldades.

HTTPS

Ao contrário de um protocolo aberto, como o HTTP, as transferências de dados do HTTPS são criptografadas. Ou seja, os dados trocados entre o navegador e o servidor são protegidos, embaralhados e decodificados somente por uma chave que só existe nas duas pontas do processo.



# **COMO TER UM HTTPS**

**É NECESSÁRIO OBTER UM CERTIFICADO DE SEGURANÇA SSL (CAMADA DE SOQUETES SEGURA) OU TLS (SEGURANÇA NA CAMADA DE TRANSPORTE).**

**O SSL NÃO É MAIS UTILIZADO. SUA ÚLTIMA VERSÃO, A 3.0, FOI DESCONTINUADA EM 2015.**

**O TLS É O SEU SUCESSOR, QUE TRAZ TECNOLOGIAS MAIS MODERNAS DE CRIPTOGRAFIA E PROTEÇÃO.**

# Principais verbos

**GET**

Requisita um determinado recurso específico. Retornam apenas dados.

**OPTIONS**

Utilizado para definir as opções para comunicação com um determinado recurso.

**POST**

Método usado para submeter dados a um determinado recurso. Esse verbo é utilizado para alterar estados de um recurso presente no servidor.

**TRACE**

Realiza um teste de loopback para verificar se uma mensagem consegue chegar a um determinado destino.

**PATCH**

Aplica modificações parciais em um determinado recurso.

**HEAD**

Similar ao método GET, entretanto não requer o corpo da resposta.

**PUT**

O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.

**CONNECT**

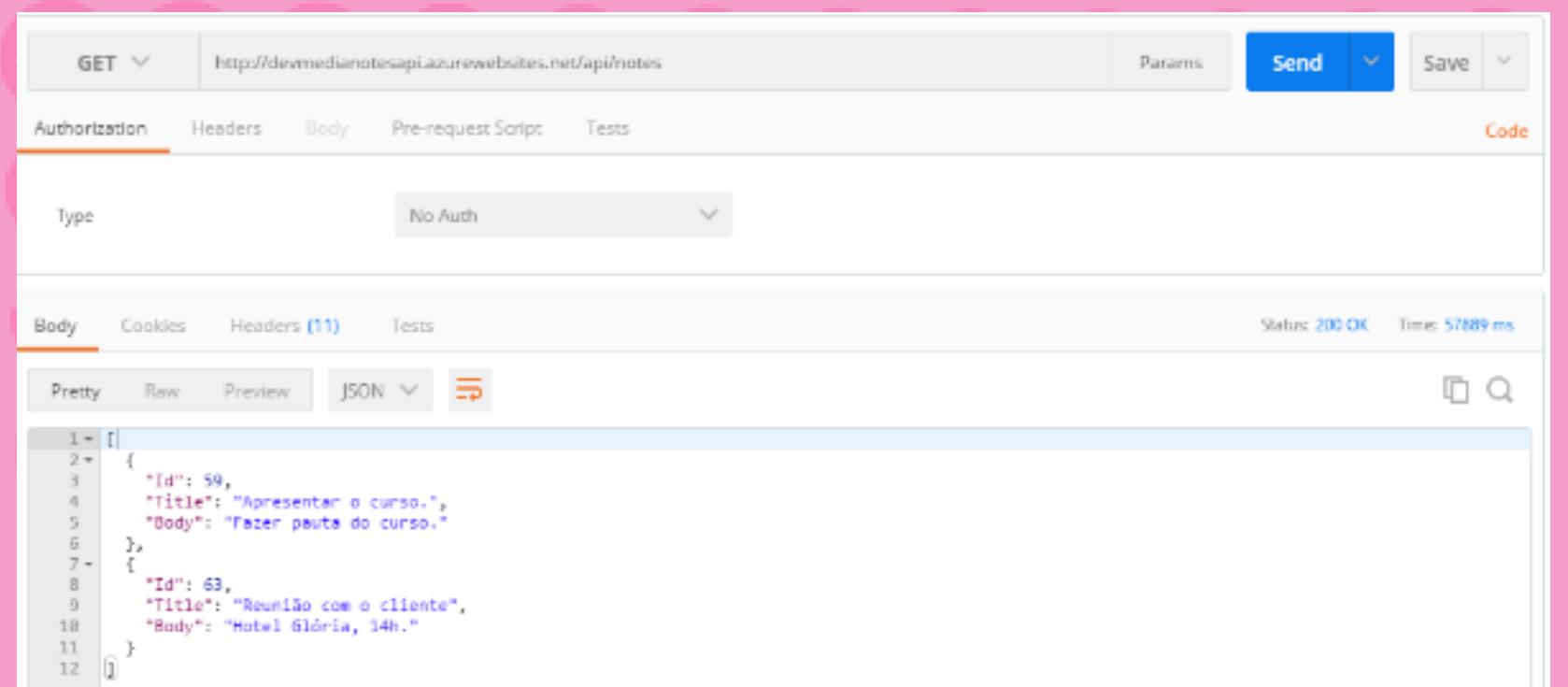
Estabelece um túnel de conexão para um servidor com base em um recurso.

**DELETE**

Apaga determinado recurso.

# Exemplos

GET



GET <https://devmedianotesapi.azurewebsites.net/api/notes> Send Save

Authorization Headers Body Pre-request Script Tests

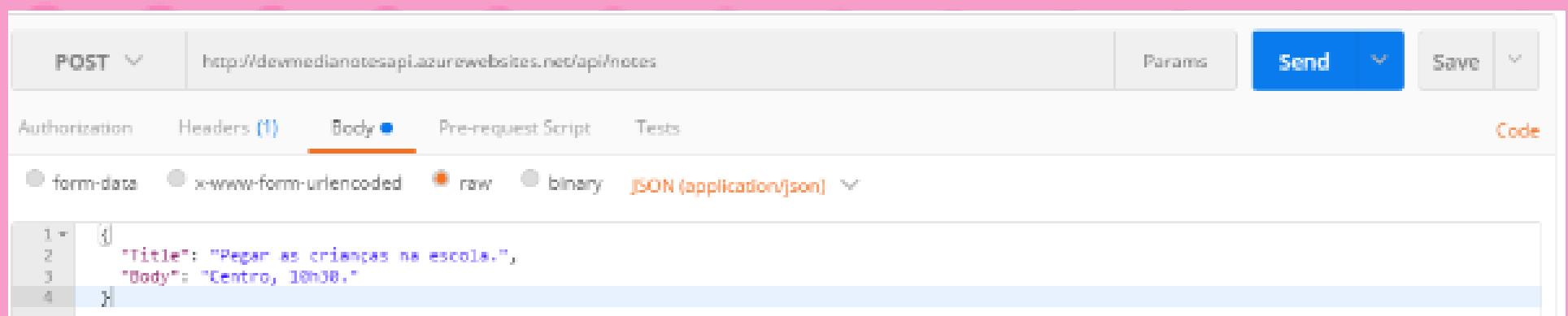
Type No Auth

Body Cookies Headers (11) Tests Status: 200 OK Time: 57889 ms

Pretty Raw Preview JSON

```
[{"Id": 59, "Title": "Apresentar o curso.", "Body": "Fazer pauta do curso."}, {"Id": 63, "Title": "Reunião com o cliente", "Body": "Hotel Glória, 14h."}]
```

POST



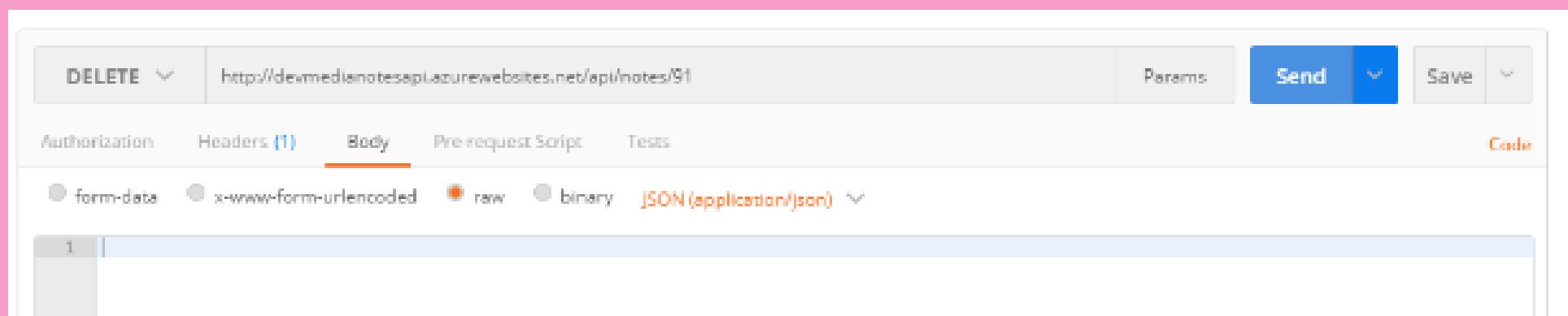
POST <https://devmedianotesapi.azurewebsites.net/api/notes> Send Save

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{"Title": "Pegar as crianças na escola.", "Body": "Centro, 10h30."}
```

DELETE



DELETE <https://devmedianotesapi.azurewebsites.net/api/notes/91> Send Save

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{ "Id": 91 }
```

# Formatos de dados:

## 1 XML

Extensible Markup Language, é uma linguagem de marcação, ou seja, um conjunto de regras utilizado para formatar documentos de maneira que os dados possam ser lidos e interpretados por diferentes sistemas.

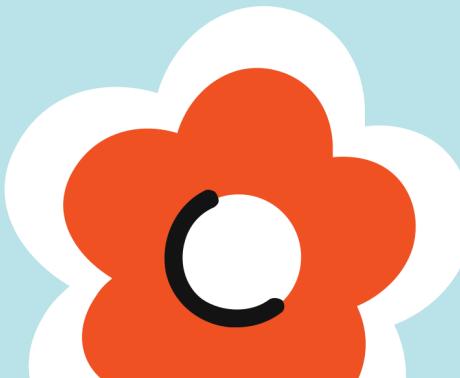
```
<?xml version="1.0">
<jogos>
  <jogo id="1">
    <titulo>Super Mario World</titulo>
    <plataforma>Super Nintendo</plataforma>
    <genero>Plataforma</genero>
    <personagens>
      <personagem>Mario</personagem>
      <personagem>Luigi</personagem>
      <personagem>Toad</personagem>
    </personagem>
  </jogo>
</jogos>
```



## 2 JSON

JavaScript Object Notation, é um formato leve de troca de dados que é fácil de ler e escrever para humanos e fácil de analisar e gerar para máquinas. É frequentemente usado para transferir dados entre um servidor e um aplicativo da web como uma alternativa ao XML.

```
Back-end > APIs > exercicios > db.json > [ ] automoveis
{
  "automoveis": [
    {
      "placa": "DUD2l22",
      "marca": "fiat",
      "modelo": "uno",
      "ano": 2006
    },
    {
      "placa": "VSF6P05",
      "marca": "Volkswagem",
      "modelo": "Fusca",
      "ano": "1976"
    },
    {
      "placa": "DUD2L11",
      "marca": "TOYOTA",
      "modelo": "corolla",
      "ano": "2018/2019"
    },
    {
      "placa": "LIN5D25",
      "marca": "PORCHE",
      "modelo": "TAYCAN",
      "ano": 2020
    }
  ]
}
```



## .FERRAMENTAS E TECNOLOGIAS

# POSTMAN

### O que é?

→ É uma ferramenta que oferece uma **interface de usuário prática** e permite a realização de solicitações HTTP sem a necessidade de escrever um monte de códigos e testar a funcionalidade de uma API.

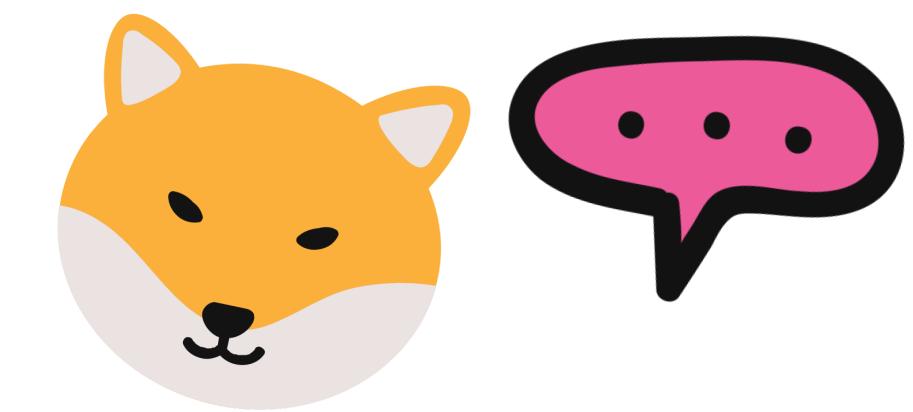
### Por que utilizar?

- Possui ambiente para a **documentação, execução de testes** de APIs e **requisições** em geral.
- Ao utilizá-lo, você constrói **solicitações** rapidamente e, ainda, pode guardá-las e analisar as respostas enviadas pela API.



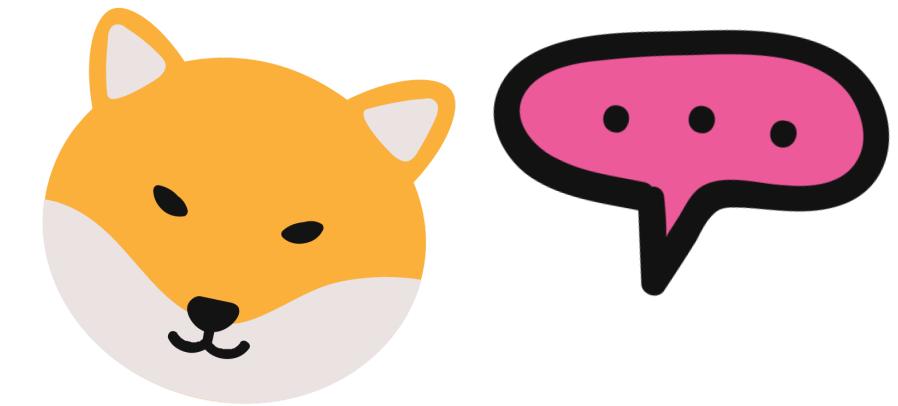
# VANTAGENS:

- **Possibilita qualquer tipo de chamada de API:** REST, SOAP ou HTTP;
- **Oferece suporte** para formatos de dados populares, como OpenAPI GraphQL e RAML;
- **Fácil acessibilidade:** Basta fazer login e pode acessar os arquivos a qualquer momento pelo aplicativo;
- **Uso de Coleções:** Permite que os usuários criem coleções para suas chamadas de API;
- **Colaboração:** Coleções e ambientes podem ser importados, exportados e compartilhados;
- **Uso de ambientes:** Ajuda a ter menos repetições de testes, pois é possível usar a mesma coleção, mas em um ambiente diferente;



# VANTAGENS:

- **Elaboração de testes:** Pontos de verificação de teste, como a verificação do status de resposta HTTP bem-sucedido, podem ser adicionados a cada chamada de API;
- **Teste de automação:** Os testes podem ser executados em várias iterações, economizando tempo para testes repetitivos;
- **Depuração:** O console do Postman ajuda a verificar quais dados foram recuperados, facilitando a depuração de testes;
- **Integração Contínua:** A partir da sua capacidade de oferecer suporte à integração contínua, são mantidas práticas de desenvolvimento.

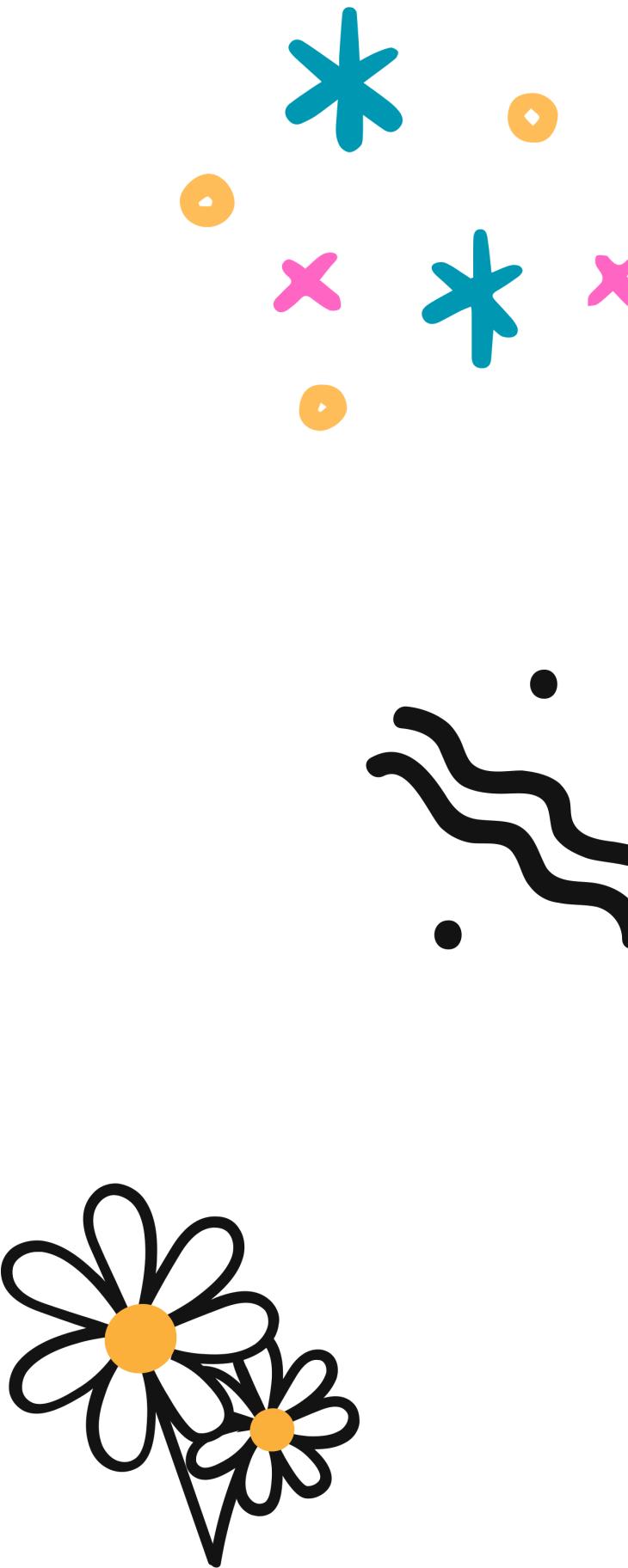


## .FERRAMENTAS E TECNOLOGIAS - POSTMAN

**Com o Postman, os testes de funcionalidade são muito mais otimizados.** Tudo o que precisa ser feito é:

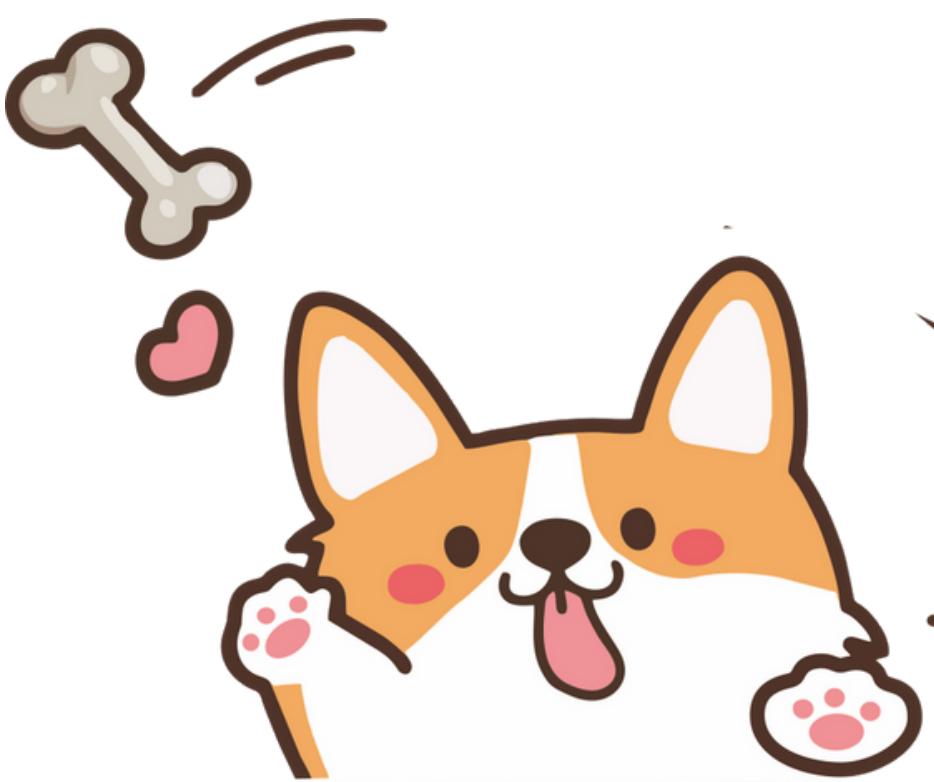
- **Obter a conexão** da rota à barra de endereços;
- **Selecionar o método de resposta** GET na caixa suspensa à esquerda;
- **Digitar a chave da API** na seção "Headers";
- **Especificar o formato da resposta**, que poderia ser em JSON, por exemplo, e;
- **Clicar em enviar.**

Em seguida, será obtido os **dados de resposta em JSON** acompanhado do código de status 200 (solicitação bem-sucedida).



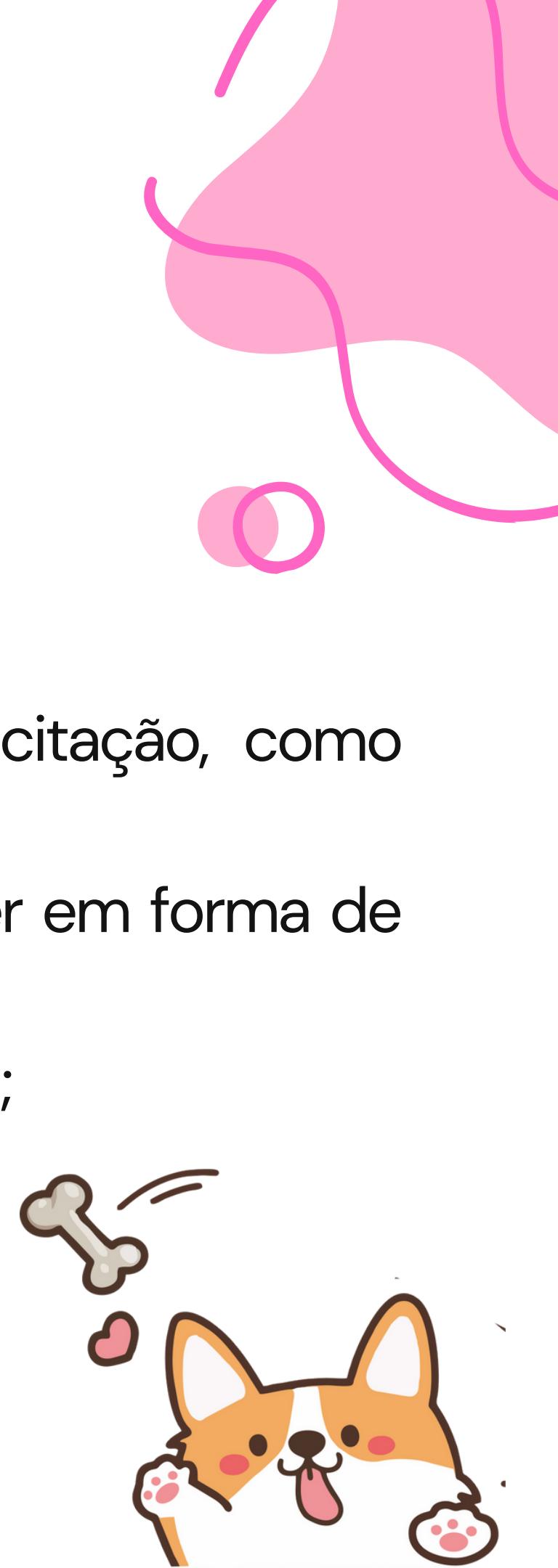
# RECURSOS DA INTERFACE:

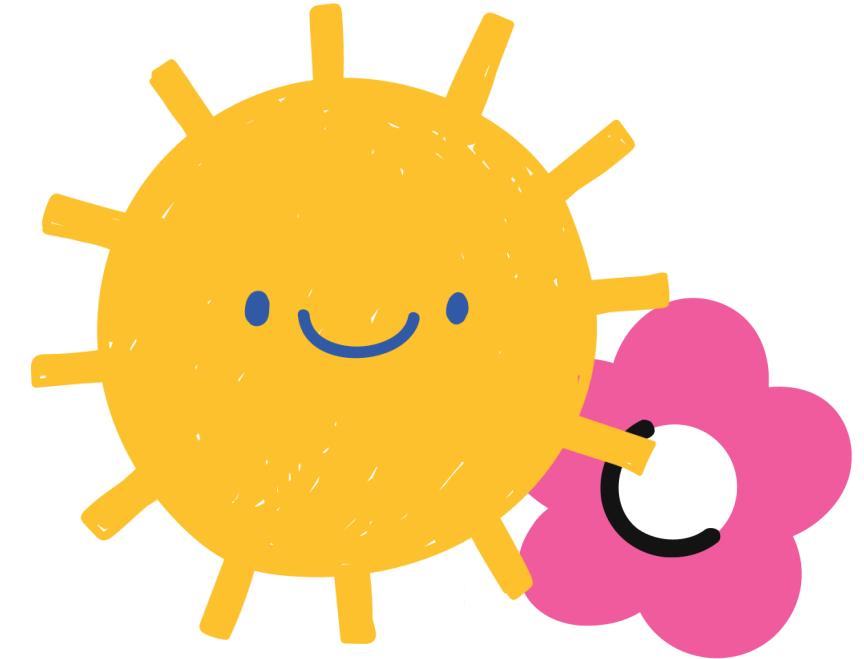
- **New:** Cria uma nova solicitação, coleção ou ambiente;
- **Import:** Importa uma coleção ou ambiente.;
- **Runner:** Pode ser utilizado para executar os testes de automação;
- **Open New:** Abre uma nova guia, Postman Window ou Runner Window;
- **My Workspace:** Criar um novo espaço de trabalho;
- **Invite:** Permite convidar membros da equipe;
- **History:** Exibe as solicitações enviadas anteriormente;
- **Collections:** Permite criar coleções;
- **Request tab:** Exibe o título de uma solicitação;
- **HTTP Request:** Exibe uma lista suspensa de solicitações diferentes;



# RECURSOS DA INTERFACE:

- **Request URL:** Identifica o link para o qual a API se comunicará;
- **Save:** Salva novas alterações;
- **Params:** Onde se escreve os parâmetros necessários para uma solicitação, como valores-chave;
- **Authorization:** Para acessar APIs, é necessária uma autorização (pode ser em forma de nome de usuário);
- **Headers:** Define cabeçalhos, como o tipo de conteúdo JSON, por exemplo;
- **Body:** Possibilita personalizar detalhes em uma solicitação;
- **Pre-request Script:** Scripts que serão executados antes da solicitação;
- **Tests:** Scripts executados durante a solicitação.

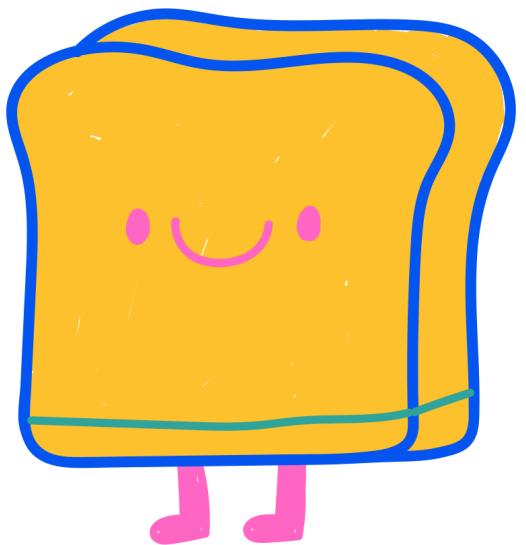




# EXEMPLO - GET

Na área de trabalho do aplicativo, faça o seguinte:

- 1 – Defina sua solicitação HTTP como GET.
- 2 – No campo URL da solicitação, coloque o link de entrada.
- 3 – Clique em Enviar.
- 4 – Você verá na aba “Status” a mensagem “200 OK”.



GET

<https://api.enotasgw.com.br/v1/empresas/:empresaid?=:MTIhMjE5YmMtNWFIMy00ZmU4LTljNWMtNTk5NjI>

Send

Save

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Code

 none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body

Cookies

Headers (4)

Test Results (1/1)

Status: 200 OK

Time: 1050 ms

Size: 1.6 KB

Save Response

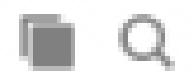
Pretty

Raw

Preview

Visualize

JSON



```
1  {
2      "id": "21996fc4-1edc-470e-b02b-f4e29d330500",
3      "status": "Habilitada",
4      "prazo": 0,
5      "dadosObrigatoriosPreenchidos": true,
6      "cnpj": "14422279000106",
7      "inscricaoMunicipal": "4301000010",
8      "inscricaoEstadual": null,
9      "razaoSocial": "SOLUCOES INTELIGENTES EM DESENVOLVIMENTO DE SISTEMA LTDA - ME",
10     "nomeFantasia": "VIRTUAL GROUP (Empresa Teste)",
11     "optanteSimplesNacional": true,
12     "dataCriacao": "2019-08-22T14:13:29Z",
13     "dataUltimaAlteracao": "2020-04-15T20:08:15Z",
14     "email": "teste@enotas.com.br",
15     "telefoneComercial": "31123456789",
16     "endereco": [
17         {
18             "pais": "Brasil",
19             "uf": "MG",
20             "codigoIbgeUf": 31,
```

## .FERRAMENTAS E TECNOLOGIAS

# SWAGGER



### O que é?

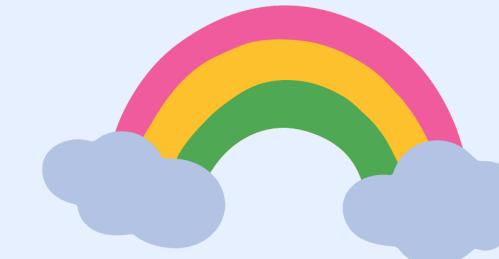
→ É um framework composto por diversas ferramentas que, independente da linguagem, **auxilia a descrição, consumo e visualização de serviços de uma API REST.**

### Por que utilizar?

→ Interface amigável e diversas opções de **personalização**;

→ Permite a **modelagem e desenvolvimento de documentação** da API e a **geração de códigos** do Cliente e do Servidor, com suporte a várias linguagens de programação.

# VANTAGENS:

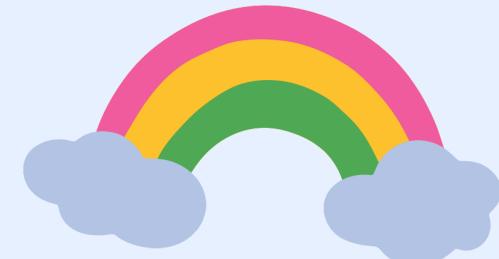


- **Especificação da API:** Consiste em **determinar os modelos de dados** que serão entendidos pela API e as funcionalidades presentes na mesma. Para cada funcionalidade, é preciso especificar o seu nome, os parâmetros que devem ser passados no momento de sua invocação e os valores que irão ser retornados aos usuários da API. Entre estas ferramentas, podemos citar o **OpenAPI Specification**;
- **Facilitação da implementação da especificação da API:** Com a ferramenta **Swagger Codegen** é possível **montar o código inicial automaticamente** nas principais linguagens de programação;
- **Testes de API:** Ajudam a **garantir o funcionamento, o desempenho e a confiabilidade** da sua aplicação. O Swagger oferece ferramentas para teste manuais, automatizados e de desempenho;



# VANTAGENS:

- **Auxílio na utilização da API:** O Swagger dispõe de ferramenta para deixar a visualização da API mais intuitiva e interativa;
- O Swagger permite **criar a documentação da API** de 3 formas:
  1. **Automaticamente:** Simultaneamente ao desenvolvimento da API é gerada a documentação;
  2. **Manualmente:** Permite ao desenvolvedor escrever livremente as especificações da API e as publicar posteriormente em seu próprio servidor;
  3. **Codegen:** Converte todas as anotações contidas no código fonte das APIs REST em documentação.



# RECURSOS DA INTERFACE:



- **Swagger Editor:** Permite criar manualmente a **documentação da API**. Possui um conjunto de templates de documentos que servem como base.
- **Swagger UI:** Permite fornecer um **ambiente visual** do que criamos com o Swagger Editor, basicamente. O módulo de UI possibilita que os usuários da API interajam intuitivamente com ela usando uma **sandbox**.
- **Swagger Codegen:** Gera automaticamente o **“esqueleto”** da API em diferentes linguagens. Com algumas linhas de comando você cria todo o código inicial da sua API na linguagem que desejar.
- **Swagger Inspector e ReadyAPI:** Auxilia nos **testes manuais** da sua API. O Swagger Inspector não precisa ser instalado, todos seus testes são feitos na nuvem.

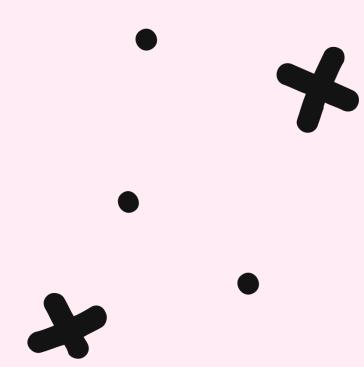
# POSTMAN VS SWAGGER

## POSTMAN

- Usado principalmente para **testar e depurar APIs**;
- Não possui os recursos de **design e documentação** da API
- Permite **fácil colaboração e script**;
- Permite que os desenvolvedores escrevam **scripts de teste** para suas APIs;

## SWAGGER

- Usado principalmente para **projetar, documentar e gerenciar APIs**;
- Oferece uma estrutura para **projetar e documentar APIs RESTful e documentação interativa**;
- **Não possui suporte integrado para automação de teste**, mas pode ser integrado a outras ferramentas;



# POSTMAN VS SWAGGER

## CONCLUSÃO



Cada um tem seus próprios pontos fortes e fracos, dependendo dos requisitos individuais do projeto. Embora o Postman ofereça uma interface fácil de usar com muitos recursos úteis prontos para uso, o uso do Swagger oferece mais opções de personalização devido à sua dependência de código em vez de opções clicáveis.

# OBRIGADA POR ASSISTIR!

## REFERÊNCIAS

<https://www.hostinger.com.br>

<https://www.devmedia.com.br>

<https://diegomariano.com>

<https://enotas.com.br>

<https://medium.com>

<http://www2.decom.ufop.br>

<https://www.youtube.com/watch?v=3LHSyha0xNO>

