



AULA 01

API E INTRODUÇÃO AO FLASK

O QUE VEREMOS HOJE

01 API

02 HTTP

03 MÉTODOS HTTP

04 REQUEST E RESPONSE

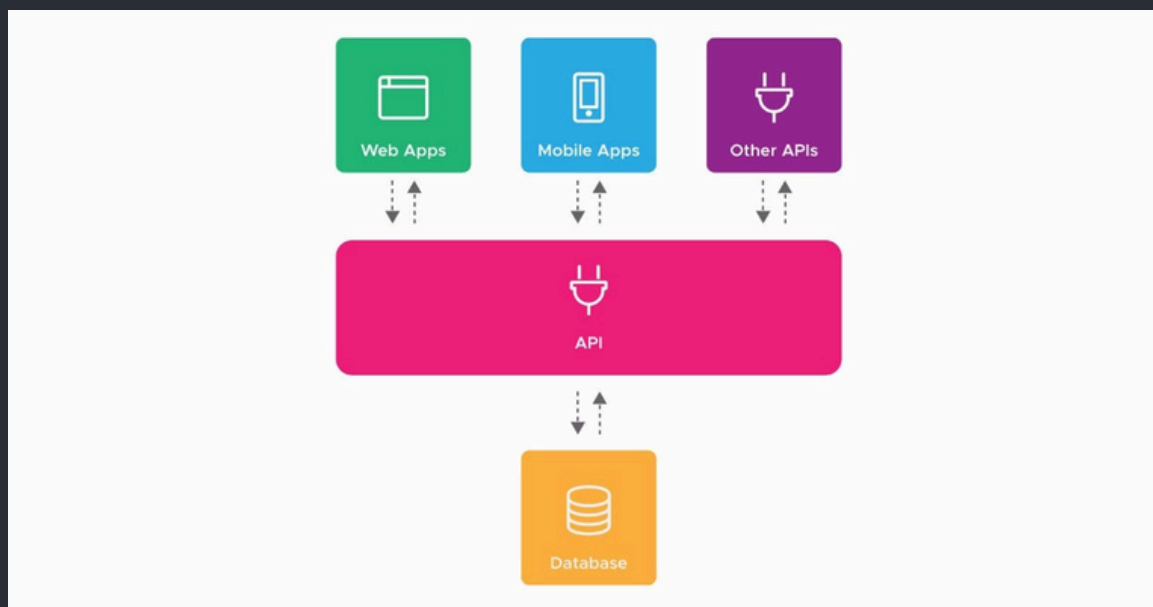
05 JSON

06 INTRODUÇÃO AO FLASK

API - APPLICATION PROGRAMMING INTERFACE

É um intermediário de software que permite que sistemas se comuniquem entre si. Elas definem regras e formatos específicos para que as informações sejam trocadas de forma organizada e eficiente.

Através de APIs, um sistema pode acessar recursos de outro sistema, como dados, funcionalidades ou serviços.



TIPOS DE API

APIs RESTful (Representational State Transfer)

São as mais comuns, utilizando o protocolo HTTP para comunicação. Elas são conhecidas por sua simplicidade e facilidade de uso, sendo amplamente adotadas em diversos cenários.

APIs GraphQL

Criada pelo Meta, é uma das mais modernas e permite ao cliente solicitar dados específicos, reduzindo a sobrecarga de informações.

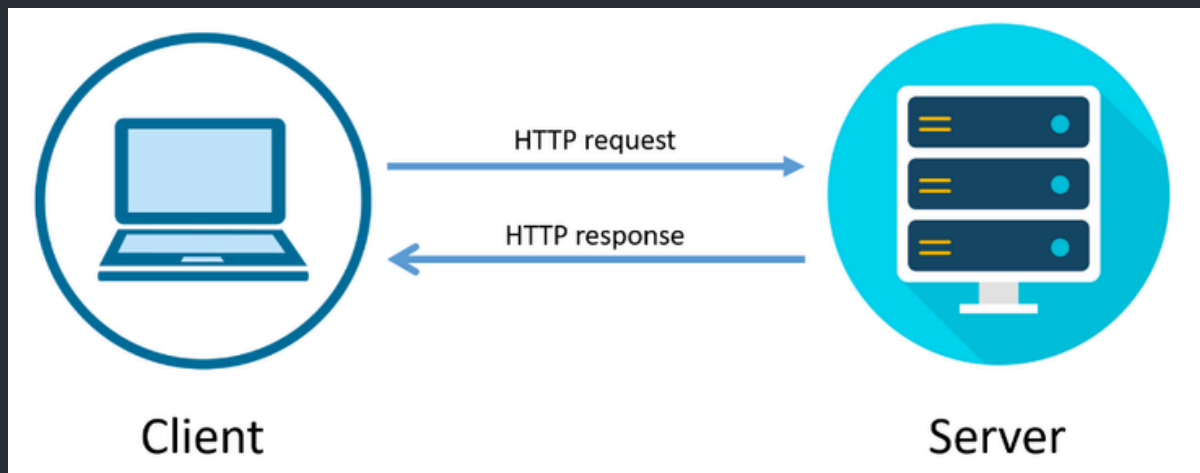
APIs SOAP (Simple Object Access Protocol)

Baseada em XML, esse tipo de API define um padrão para comunicação de mensagens entre sistemas, oferecendo mecanismos para validação de dados, controle de acesso e criptografia, entre outras ferramentas com alto nível de segurança.

O QUE É HTTP?

HTTP (Hypertext Transfer Protocol) é o protocolo de **comunicação** que permite a troca de informações entre navegadores web e servidores. É uma base fundamental para a comunicação entre APIs e aplicativos.

O HTTP define as regras de comunicação, como a estrutura das requisições e respostas, os códigos de status e os tipos de conteúdo. APIs usam o HTTP para transmitir dados e comandos entre servidores e aplicativos.



MÉTODOS HTTP

GET

É utilizado para buscar informações de um servidor. É o método mais básico e comum, e retorna dados em formato de texto, normalmente em HTML, JSON ou XML.

POST

O método POST é utilizado para enviar dados para um servidor, por exemplo, para criar um novo recurso, como um novo usuário em um site.

PUT

O método PUT é utilizado para atualizar um recurso existente em um servidor, por exemplo, para editar as informações de um usuário.

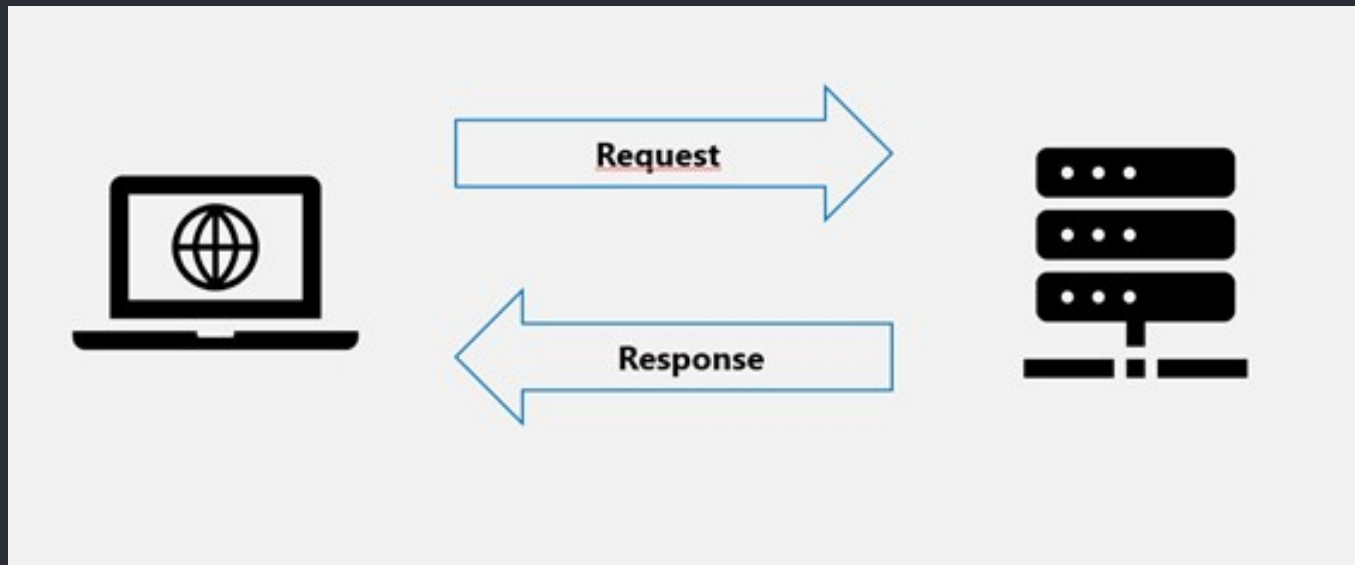
DELETE

O método DELETE é utilizado para remover um recurso de um servidor, por exemplo, para excluir um usuário de um sistema.

REQUEST

Uma request é uma solicitação enviada de um sistema para outro, geralmente através da internet. Essa solicitação contém informações sobre o que o sistema requisitante deseja obter do outro sistema.

A request geralmente inclui o método HTTP, a URL do recurso solicitado, cabeçalhos e um corpo com os dados da solicitação.



REQUEST - COMPONENTES

Método HTTP

O método indica a ação desejada, como obter dados (GET) ou enviar informações (POST).

URL

A URL identifica o recurso específico que está acessando. Ela é formada pelo endereço do servidor, o nome da API e um identificador único do recurso.

Headers

Os headers fornecem informações adicionais sobre a request, como o tipo de conteúdo que está sendo enviado.

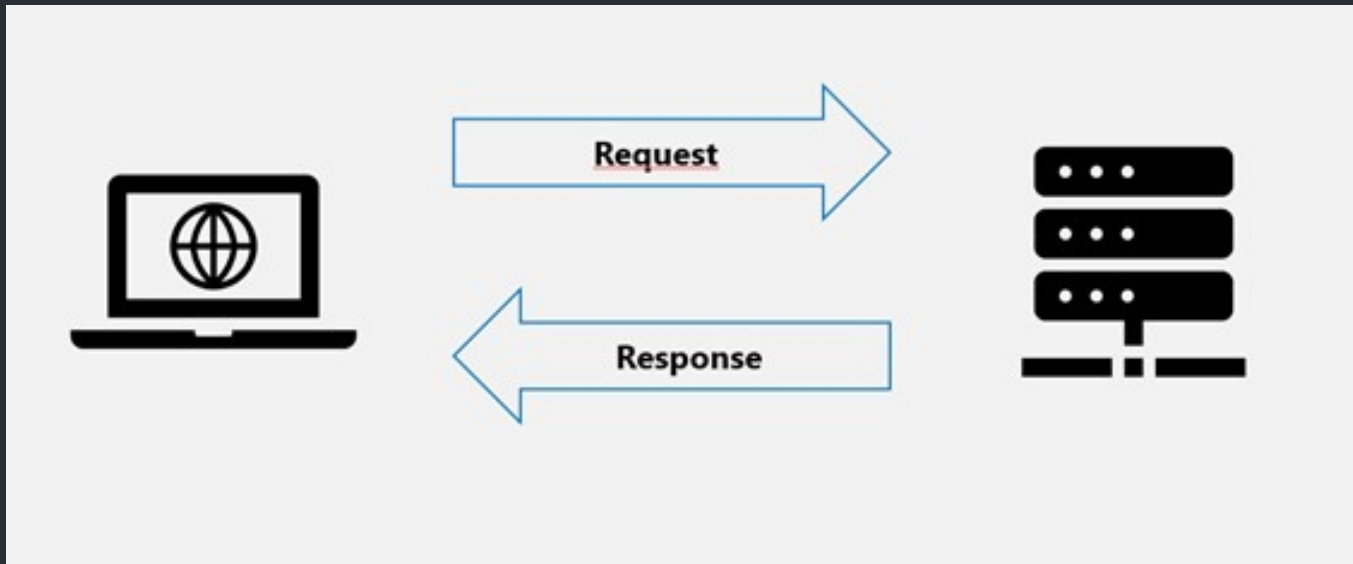
Body

O body contém os dados que está sendo enviado para o servidor. O conteúdo do body varia de acordo com o método HTTP.

RESPONSE

Response é a resposta do servidor à request. No response irá conter as informações sobre o sucesso ou falha da requisição, dados solicitados e outros detalhes.

Assim como no request, o response também possui componentes em sua estrutura.



RESPONSE - COMPONENTES

Status Code

O status code é um número que indica o sucesso ou o erro da requisição. 200 significa sucesso, 404 significa que a página não foi encontrada, e 500 indica um erro do servidor.

Headers

Os headers são metadados que fornecem informações adicionais sobre a resposta, como o tipo de conteúdo.

Body

O body da resposta contém o conteúdo real da resposta, como dados, texto ou imagens. O formato do body depende da API e pode ser JSON, XML ou outros.

JSON

JSON (JavaScript Object Notation) é um formato leve de troca de dados e usado para comunicação entre sistemas, especialmente em aplicações web.

A estrutura do JSON é similar a pares chave-valor, mas ele é sempre uma string formatada, podendo ser convertida para tipos nativos, como dicionários, em várias linguagens de programação.

OBS: No JSON só é possível usar aspas duplas.



```
1  {  
2      "nome": "José Maria",  
3      "email": "jose@gmail.com",  
4      "idade": 30,  
5      "cidade": "São Paulo",  
6      "estado": "SP",  
7      "status": "Ativo"  
8  }
```

POSTMAN

O Postman é uma ferramenta gráfica usada para testar APIs.

Essa ferramenta permite:

- Enviar requisições HTTP como GET, POST, PUT, DELETE
- Visualizar as respostas da API (texto, JSON, XML)
- Configurar cabeçalhos, corpo da requisição e parâmetros
- Testar sua API sem precisar criar uma interface gráfica



ATIVIDADE PRÁTICA

Utilizando a API Dummy JSON e faça as seguintes requisições:

- uma requisição para buscar a lista de produtos disponíveis.
- uma requisição buscar um único produto.

Verifique a documentação para fazer as requisições.

FLASK

Flask é um framework web minimalista e leve, escrito em Python, ideal para a criação de APIs REST e aplicações web simples.

Ele é conhecido por sua simplicidade, flexibilidade e por dar ao desenvolvedor bastante liberdade para estruturar o projeto da forma que preferir.

O Flask não vem com muitas coisas prontas (como autenticação, ORM, etc.), mas isso também o torna mais fácil de aprender e customizar, sendo uma ótima escolha para projetos pequenos ou para aprender os fundamentos de APIs.



AMBIENTE VIRTUAL

Um ambiente virtual é uma ferramenta que permite a criação de ambientes isolados para cada projeto. Cada projeto pode ter suas próprias dependências, evitando conflitos e garantindo que as bibliotecas necessárias estejam disponíveis.

No Python isso é feito por meio da **venv** que é um um diretório separado que contém uma cópia independente do interpretador Python e um conjunto de pacotes



COMO INICIAR UM AMBIENTE VIRTUAL

Para criar um ambiente virtual (venv) é só digitar o comando no terminal:

```
python -m venv venv
```

Esse comando deve criar uma pasta chamada venv no diretório.

Para ativar o ambiente virtual temos que usar o seguinte comando:

- Windows: **venv/Scripts/activate**
- macOS/Linux: **source venv/bin/activate**

FLASK - CONFIGURAÇÃO

Com a venv ativada, é possível instalar o Flask

Para instalar, rode o comando no terminal: **pip install flask**

Para criar o arquivo requirements.txt usamos o comando: **pip freeze > requirements.txt**

Este arquivo lista todas as dependências do projeto.


```
• (venv) aline@aline-Aspire-A515-54:~/WorkArea/Infinity/aula_01$ pip freeze > requirements.txt
○ (venv) aline@aline-Aspire-A515-54:~/WorkArea/Infinity/aula_01$
```

FLASK - CRIANDO UM APP

Com o Flask instalado, podemos criar um arquivo Python para iniciar a aplicação.

Para começar a usar o Flask, é necessário importar a biblioteca e instanciar o objeto Flask com o módulo principal (**`__name__`**).

O decorador **`@app.route()`** define a rota que será acessada no navegador, e a função abaixo dele indica a resposta retornada ao usuário.



```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def hello():
7      return "Olá mundo"
```

FLASK - EXECUTANDO O APP

Para rodar o servidor Flask, use o terminal com o ambiente virtual ativado.

```
flask --app main run
```

O terminal mostrará um link como:

```
Running on http://127.0.0.1:5000/
```

Esse endereço pode ser acessado no navegador ou Postman para testar sua aplicação.

Também é possível rodar o servidor Flask em modo de desenvolvimento, com recarregamento automático ativado usando o comando:

```
flask --app main --debug run
```

ATIVIDADE PRÁTICA

Adicione um novo endpoint GET no arquivo main.py que retorne uma lista de tarefas.

Cada tarefa deve conter:

- id (número)
- título (texto)
- status (ex: 'concluída' ou 'pendente')
- categoria (ex: 'trabalho', 'estudo', 'pessoal')

Crie a lista de tarefas como uma lista de dicionários no seu código.

Teste no Postman e verifique se a lista aparece corretamente no formato de resposta.

ATIVIDADE PRÁTICA

Crie um novo endpoint GET que retorne apenas as tarefas concluídas.

Você pode escolher a melhor forma de representar isso:

- status: True ou False (valores booleanos)
- ou status: 'concluída' / 'pendente' (valores em texto)

No novo endpoint, filtre a lista original e retorne apenas as tarefas com status de concluída.

Teste no Postman e verifique se o retorno está correto.