DESENVOLVIMENTO DE APIS E MICROSSERVIÇOS

Aula – Api’s REST

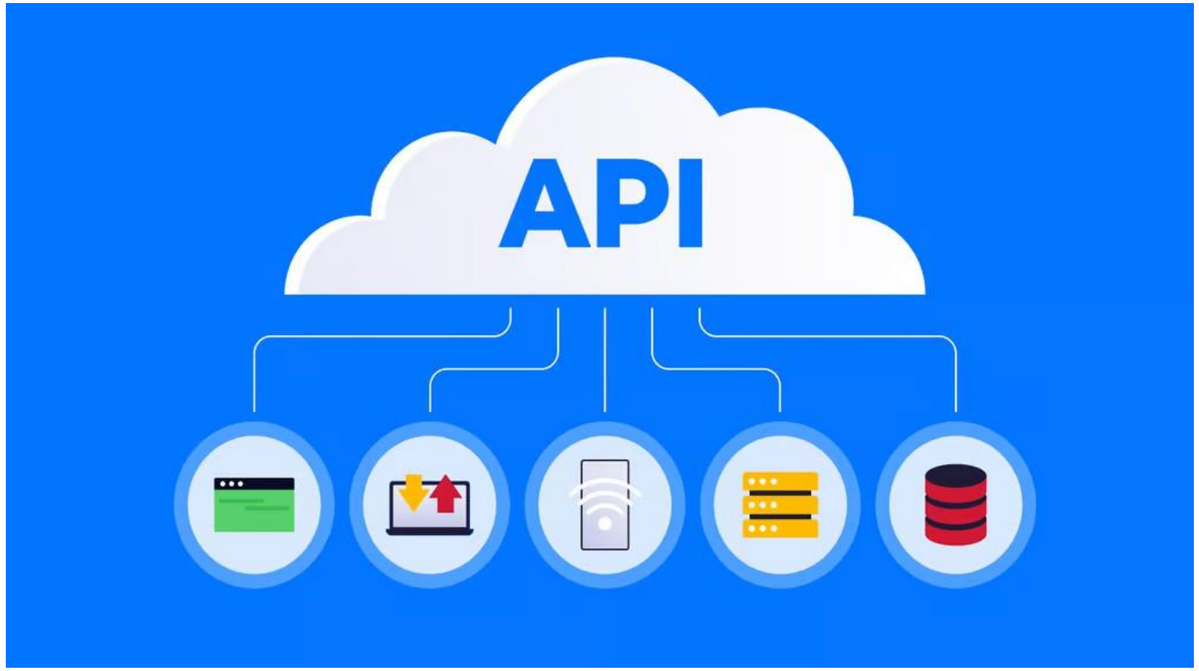
Caio Eduardo do Prado Ireno

caio.ireno@faculdadeimpacta.com.br

APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.

É um programa de software de interface que ajuda a interagir com outro software como uma interface de utilizador (UI)

Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. A aplicação para a previsão do tempo em seu telefone “fala” com esse sistema por meio de APIs e mostra atualizações meteorológicas diárias no telefone.



API

APPLICATION

Um software com uma função específica, como um programa de processamento de texto, um aplicativo de redes sociais ou um sistema de gerenciamento de banco de dados.

PROGRAMMING

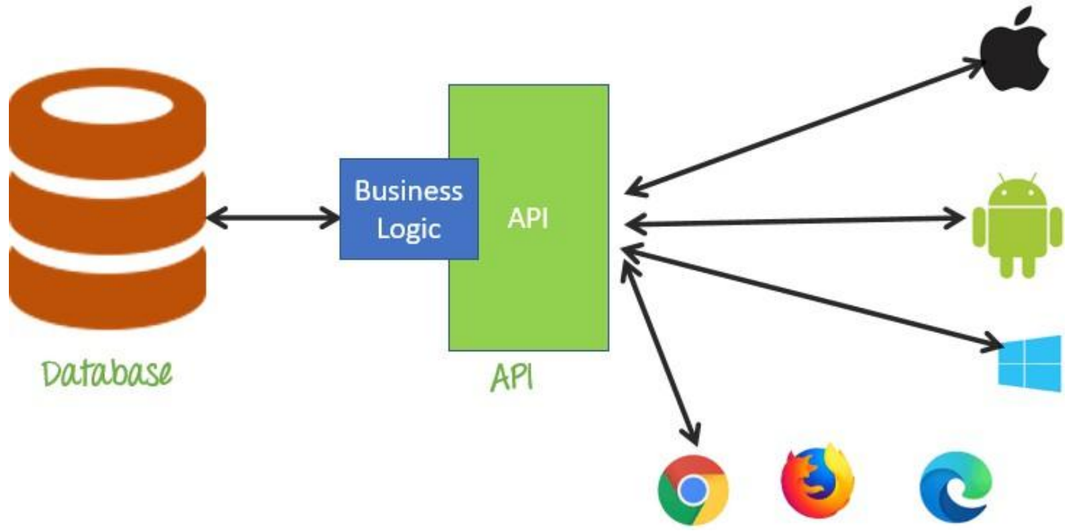
A atividade de escrever código e criar programas de computador para realizar tarefas específicas ou para interagir com outros softwares, sistemas ou dispositivos.

INTERFACE

Uma fronteira ou ponto de interação entre diferentes sistemas, componentes ou partes de um sistema. No contexto de software, uma interface define como diferentes partes do software podem se comunicar e interagir.

Quando você usa um aplicativo:

- **Ele se conecta à Internet;**
- **Envia dado para um servidor;**
- **O servidor recupera esses dados;**
- **Interpreta-os, executa as ações necessárias e envia de volta ao seu telefone;**



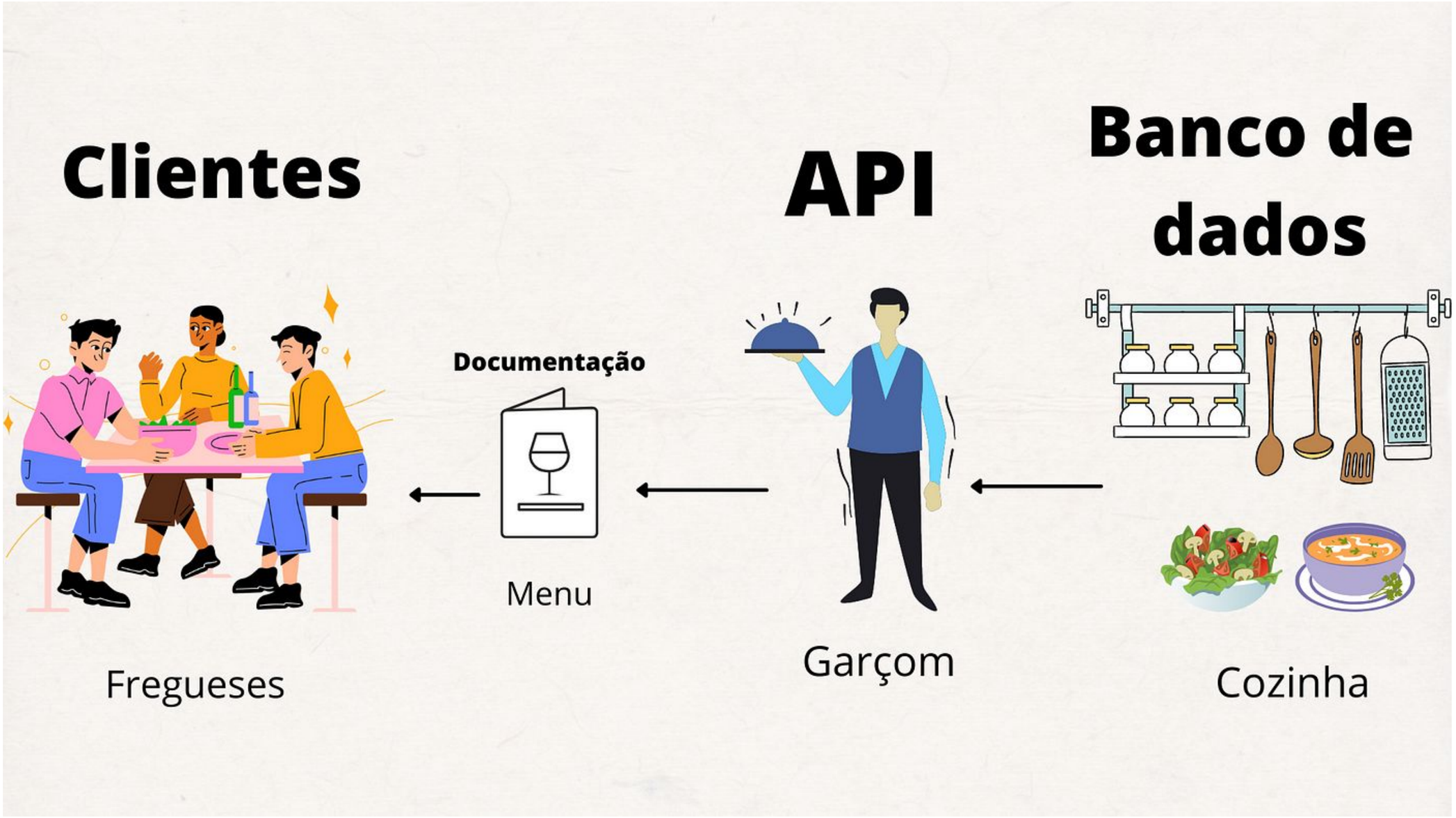
O aplicativo então interpreta esses dados e apresenta as informações que você deseja de forma legível.

- Os dados do seu telefone nunca são totalmente expostos ao servidor.
- O servidor nunca fica totalmente exposto ao seu telefone

Cada um se comunica com pequenos pacotes de dados compartilhando apenas o que é absolutamente necessário.

Podemos pensar no conceito acima semelhante a pedir comida para viagem em seu restaurante favorito.

Você, cliente, diz ao garçom o que gostaria de comer e ele lhe dirá o que precisa em troca e, no final, você recebe sua refeição!


Clientes
API
Banco de dados
Documentação
Menu
Fregueses
Garçom
Cozinha

Qual é a vantagem de criar um API?

- **Reduz a quantidade de código repetitivo necessário para criar a sua aplicação**

- **Pode-se criar um ambiente de utilizador mais seguro, uma vez que apenas pessoas específicas podem acessar aos dados à sua aplicação.**

- **As APIs possibilitam a automatização de diversas operações;**

- **Operam com os comandos HTTP GET, PUT, POST e DELETE; suportando também as quatro operações básicas – create, read, update, and delete (ou CRUD);**

Como as APIs funcionam?

A arquitetura da API geralmente é explicada em termos de cliente e servidor. A aplicação que envia a solicitação é chamada de cliente e a aplicação que envia a resposta é chamada de servidor. Então, no exemplo do clima, o banco de dados meteorológico do instituto é o servidor e o aplicativo móvel é o cliente.

REST

Representational State Transfer

REST é um conjunto de princípios de arquitetura que atende às necessidades de aplicações mobile e serviços web. Como se trata de um grupo de diretrizes, são os desenvolvedores que precisam implementar essas recomendações.

Imagine um aplicativo de previsão do tempo. Ele pode usar uma API REST para obter dados atualizados sobre o clima de uma cidade específica. Através da API REST, o aplicativo pode enviar uma solicitação para a URL da API contendo o nome da cidade. A API processa a solicitação e retorna os dados do clima no formato desejado, como JSON. Isso permite que o aplicativo exiba informações precisas sobre o clima para o usuário.

REST

As APIs REST seguem um conjunto de princípios que definem a sua arquitetura e comportamento. Esses princípios tornam as APIs REST flexíveis e escaláveis.

Recursos: As APIs REST são baseadas em recursos, que são entidades que podem ser acessadas e manipuladas através das operações da API.

Métodos HTTP: As operações sobre os recursos são mapeadas para os métodos HTTP, como GET, POST, PUT e DELETE. Cada método tem um propósito específico, como obter, criar, atualizar ou excluir um recurso.

Stateless (Sem Estado): Cada solicitação para a API deve conter todas as informações necessárias. O servidor não mantém estado entre as solicitações, facilitando a escalabilidade e a distribuição.

Representações: Os recursos podem ser representados em diferentes formatos, como JSON, XML, HTML, etc. O cliente escolhe o formato de representação que melhor atende às suas necessidades.

REST

Uma solicitação REST consiste em diferentes partes que especificam o que o cliente deseja fazer com um recurso.

Endpoint: É a URL que aponta para um recurso específico. Ela identifica onde a ação será realizada.

Método HTTP: Determina qual ação será executada sobre o recurso, como GET, POST, PUT ou DELETE.

Cabeçalhos: Contêm informações adicionais sobre a solicitação, como o formato de representação desejado (por exemplo, JSON) e informações de autenticação.

Corpo: Opcionalmente, contém os dados enviados para a criação ou atualização de um recurso. Geralmente usado em solicitações POST e PUT.

```python
import requests

BASE_URL = "http://example.com/api"  # Substitua isso pela sua URL base

def createUser(user):
    url = f"{BASE_URL}/users"
    headers = {"Content-Type": "application/json"}
    response = requests.post(url, headers=headers, json=user)

    if response.status_code == 201:
        data = response.json()
        print(f"Usuário {data['name']} criado com sucesso")
        return data
    else:
        print(f"Erro ao criar usuário: {response.text}")
        return None
```

Endpoint

Cabeçalhos

Método HTTP

Corpo

REST

As respostas de uma API REST contêm informações sobre o resultado da solicitação e o estado atual do recurso.

Código de Status HTTP: Indica se a solicitação foi bem-sucedida ou se houve algum erro. Exemplos comuns incluem 200 OK, 201 Created e 404 Not Found.

Cabeçalhos: Fornecem informações adicionais sobre a resposta, como o formato da representação, tamanho do conteúdo, etc.

Corpo: Contém os dados do recurso solicitado ou mensagens de erro em formato JSON ou outro formato definido.

REST

As APIs REST oferecem diversos benefícios que tornam a comunicação entre aplicações mais eficiente e flexível.

Interoperabilidade: As APIs REST podem ser consumidas por diferentes tipos de clientes (por exemplo, aplicativos móveis, navegadores) independentemente das tecnologias usadas.

Escalabilidade: O modelo stateless permite que os servidores sejam escalados horizontalmente para lidar com cargas maiores.

Cacheabilidade: As respostas podem ser armazenadas em cache para melhorar o desempenho e reduzir a carga no servidor.

Simplicidade: O uso de métodos HTTP padronizados e a estrutura da API facilitam o desenvolvimento e a manutenção.

REST vs OUTROS

REST é Arquitetura de software

REST == HTTP + Princípios

REST não é HTTP

REST !== SOAP

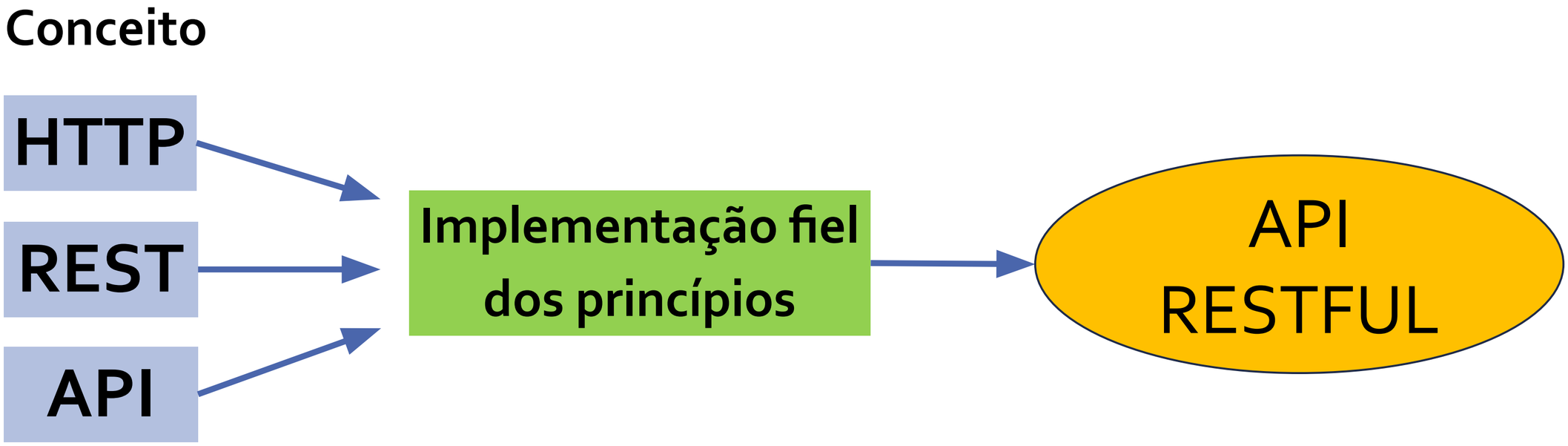
SOAP é um protocolo de comunicação, enquanto REST é um estilo arquitetural

SOAP temos contrato de serviço (wsdl) expondo as chamadas do servidor e cliente é **fortemente** acoplado ao servidor.

REST !== GraphQL

GraphQL é uma linguagem de consulta para a API. Permite o cliente solicitar apenas os dados. Um único endereço.

O que é uma API Restful?



Etapas para Criar uma API REST

1.Definir Recursos:

Identifique os recursos que sua API irá gerencia.

Recursos são as entidades que os clientes poderão acessar e manipular através da API.

Recurso: Usuário
Recurso: Produtos
Recurso: Casas

Etapas para Criar uma API REST

2. Definir Endpoints

Uma vez que você definiu seus recursos, é hora de determinar os endpoints correspondentes. Endpoints são URLs específicas que os clientes usarão para interagir com esses recursos.

Endpoint: /users
Endpoint:/products
Endpoint: /casas

Etapas para Criar uma API REST

3. Métodos HTTP

Atribuir os métodos HTTP corretos aos endpoints é fundamental para definir as operações possíveis em cada recurso.

Método GET: Obter informações sobre um recurso.
Método POST: Criar um novo recurso.
Método PUT: Atualizar um recurso existente.
Método DELETE: Excluir um recurso.

Etapas para Criar uma API REST

4. Criar Estrutura de Dados

A estrutura de dados usada para representar os recursos é crucial. Geralmente, o formato JSON é escolhido devido à sua flexibilidade e legibilidade.

```json
{
  "id": 1,
  "username": "joedoe",
  "email": "joe@example.com"
}
```

Etapas para Criar uma API REST

5. Configurar o Servidor

Utilize uma Linguagem de programação com uma biblioteca ou frameworks para criar o servidor que irá receber e processar as solicitações HTTP dos clientes.

No projeto, vamos usar o Flask que é um framework Python usado para criar aplicativos web, incluindo servidores web

Etapas para Criar uma API REST

6. Implementar Operações

Escreva o código para cada operação definida nos endpoints. Isso envolve criar funções que realizam as ações necessárias, como criar, ler, atualizar ou excluir recursos.

```python
import requests

BASE_URL = "http://example.com/api"  # Substitua isso pela sua URL base
ENDPOINT = "/users"  # Substitua isso pelo endpoint específico

def createUser(user):
    url = BASE_URL + ENDPOINT
    headers = {"Content-Type": "application/json"}
    response = requests.post(url, headers=headers, json=user)

    if response.status_code == 201:
        data = response.json()
        print(f"Usuário {data['name']} criado com sucesso")
        return data
    else:
        print(f"Erro ao criar usuário: {response.text}")
        return None
```

Etapas para Criar uma API REST

7. Tratamento de Erros

Implemente o tratamento de erros para lidar com solicitações inválidas ou falhas do servidor. Isso garante que a API forneça respostas claras em caso de problemas.

API pronta para uso