



Sistemas Distribuídos  
Dr. Joseffe Barroso de Oliveira

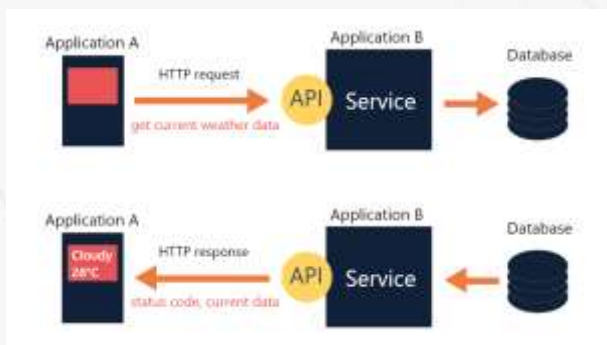


## Introdução

**APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.** Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. O aplicativo meteorológico em seu telefone “fala” com este sistema por meio de APIs e mostra atualizações meteorológicas diárias no telefone.

## O que é e como funcionam as APIs?

API significa Application Programming Interface (Interface de Programação de Aplicação). No contexto de APIs, a palavra Aplicação refere-se a qualquer software com uma função distinta. 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.



## Tipos de dados usados nas APIs - XML

Extensible Markup Language, ou simplesmente XML, é uma linguagem de marcação, ou seja, um conjunto de códigos para determinar a estrutura de dados para facilitar a troca de informação entre sistemas computacionais.

```
<?xml version="1.0"?>
- <birds>
  - <owl id="1201">
    <species>Bubo bubo</species>
    <name>Eagle Owl</name>
    <region>Eurasia</region>
  </owl>
  - <owl id="1202">
    <species>Strix occidentalis</species>
    <name>Spotted Owl</name>
    <region>North America</region>
  </owl>
</birds>
```

## Tipos de dados usados nas APIs - JSON

JSON é um acrônimo de “Javascript Object Notation” ou simplesmente “Notação de objeto JavaScript”. É um modelo para a transmissão de informações no formato de texto entre diferentes linguagens, ou seja, é um formato de serialização de dados muito utilizado em APIs.

Dentre suas diversas características, a mais atrativa sem dúvidas é a sua legibilidade, podendo facilmente ser lido por humanos, sem a necessidade de uma aplicação auxiliar.

```
{
  "First_Name": "eric",
  "Last_Name": "stuart",
  "ID_Number": 113547,
  "Age": 45,
  "Start_Date": "11/01/2000",
  "Job_Title": "programmer"
},
{
  "First_Name": "dan",
  "Last_Name": "baker",
  "ID_Number": 567821,
  "Age": 40,
  "Start_Date": "04/01/2004",
  "Job_Title": "programmer"
},
{
  "First_Name": "ella",
  "Last_Name": "wajah",
  "ID_Number": 983753,
  "Age": 26,
  "Start_Date": "05/01/2010",
  "Job_Title": "sales"
}
```



## Tipos de APIs mais usadas

### APIs SOAP

Essas APIs usam o Simple Object Access Protocol (Protocolo de Acesso a Objetos Simples). **Cliente e servidor trocam mensagens usando o formato XML.** Esta é uma API menos flexível que era mais popular no passado.

### APIs REST

REST significa Transferência Representacional de Estado. **O cliente envia solicitações ao servidor como dados.** O servidor usa essa entrada do cliente para iniciar funções internas e retorna os dados de saída ao cliente. **Cliente e servidor trocam mensagens usando o formato JSON.**

**A principal característica da API REST é a ausência de estado.** A ausência de estado significa que os servidores não salvam dados do cliente entre as solicitações. As solicitações do cliente ao servidor são semelhantes aos URLs que você digita no navegador para visitar um site. A resposta do servidor corresponde a dados simples, sem a renderização gráfica típica de uma página da Web.



## Requisições e comunicações em APIs REST

O REST precisa que um cliente faça uma requisição para o servidor para enviar ou modificar dados. Um requisição consiste em:

- Um verbo ou método HTTP, que define que tipo de operação o servidor vai realizar;
- Um header, com o cabeçalho da requisição que passa informações sobre a requisição;
- Um path (caminho ou rota) para o servidor, ou seja, uma URL;
- Informação no corpo da requisição, sendo esta informação opcional.



## Requisições e comunicações em APIs REST

The screenshot displays a REST client interface with the following components:

- Request Section:**
  - Method:** GET
  - URL:** `https://backend-tarefa.herokuapp.com/tarefa` (Annotated with "API" and "Recursos (End point)")
  - Body:** Empty (Annotated with "Verbo HTTP")
  - Auth:** Empty (Annotated with "Servidor")
  - Query:** Empty
  - Header:** Empty
  - Docs:** Empty
- Response Section:**
  - Status Code:** 200 OK (Annotated with "Status code")
  - Time:** 180 ms (Annotated with "Tempo da response")
  - Size:** 379 B (Annotated with "Tamanho da response")
  - Preview:** A JSON array of 3 tasks (Annotated with "JSON da response (Preview)")

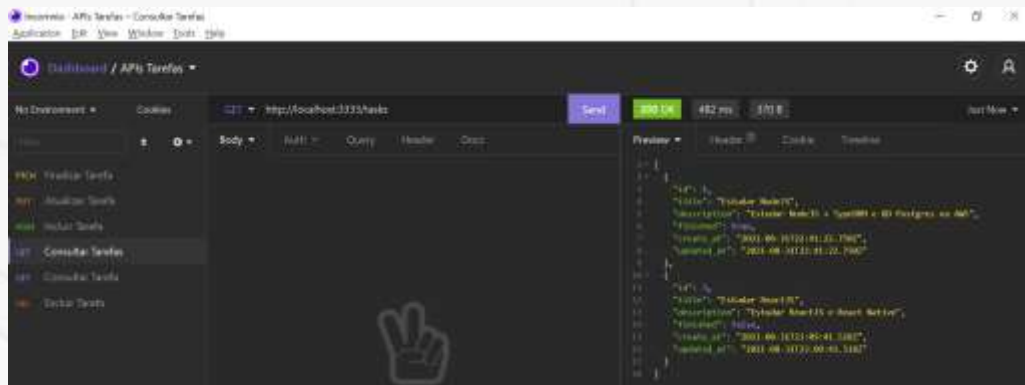
**JSON Response (Preview):**

```
[{"id": 01, "name": "Preparar o ambiente de desenv", "createdAt": "2021-08-18T16:17:55.789Z", "updatedAt": "2021-08-18T16:17:55.789Z"}, {"id": 02, "name": "Criar o Banco de Dados no Ad5", "createdAt": "2021-08-18T16:18:20.536Z", "updatedAt": "2021-08-18T16:18:20.536Z"}, {"id": 03, "name": "Iniciar o desenv no NodeJS", "createdAt": "2021-08-18T16:18:36.886Z", "updatedAt": "2021-08-18T16:18:36.886Z"}]
```



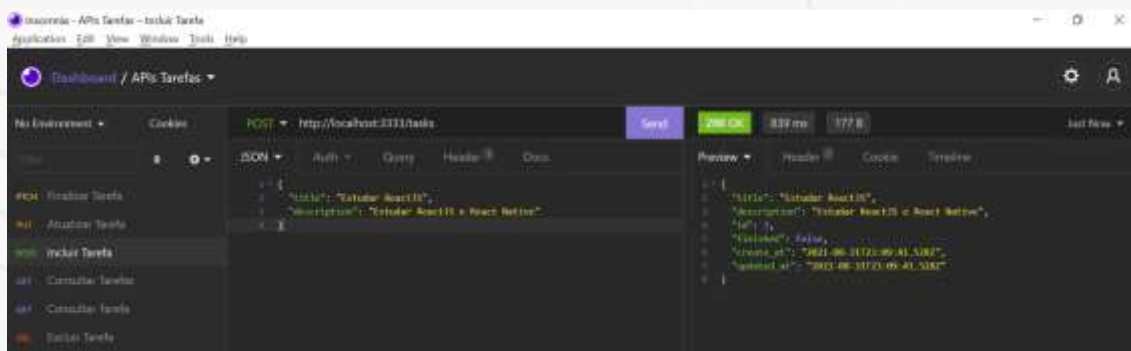
## Verbs HTTP - GET

O método GET é o método mais comum, geralmente é usado para solicitar que um servidor envie um recurso



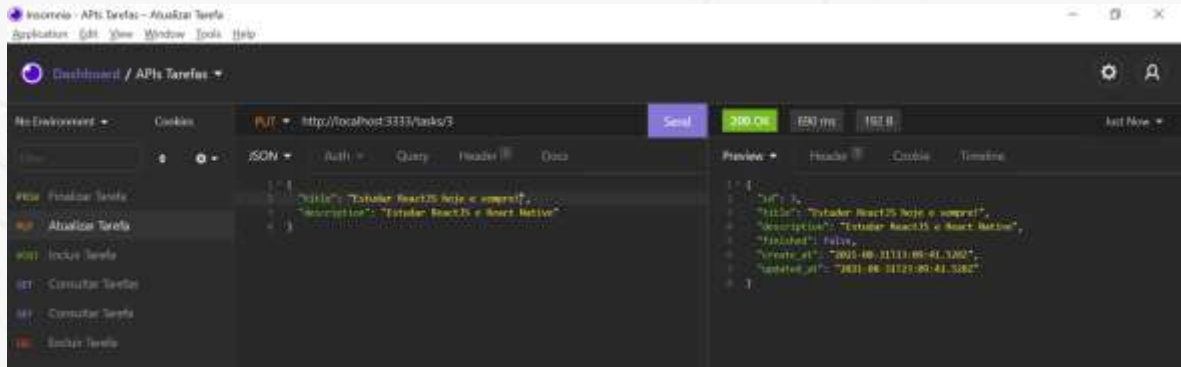
## Verbs HTTP - POST

O método POST foi projetado para enviar dados de entrada para o servidor. Na prática, é frequentemente usado para suportar formulários HTML



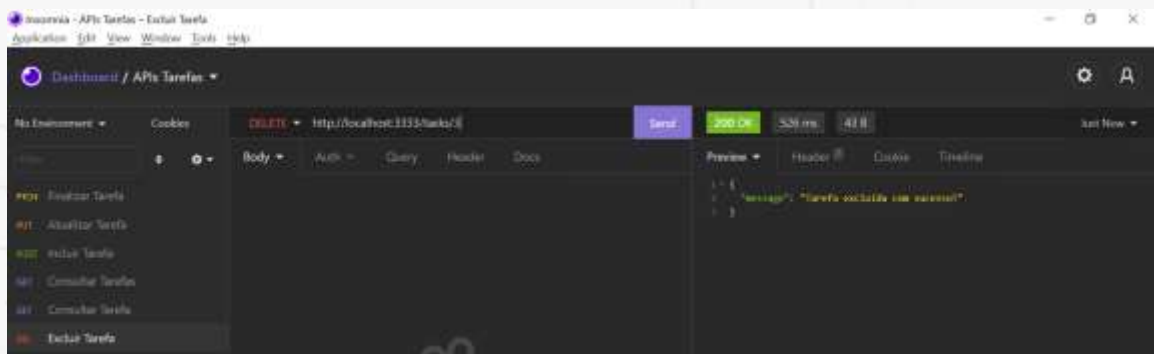
## Verbos HTTP - PUT

O método PUT edita e atualiza documentos em um servidor



## Verbos HTTP - DELETE

O método DELETE que como o próprio nome já diz, deleta certo dado ou coleção do servidor



## Códigos de respostas HTTP

Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição. São eles:

- 200 (OK), requisição atendida com sucesso;
- 201 (CREATED), objeto ou recurso criado com sucesso;
- 204 (NO CONTENT), objeto ou recurso deletado com sucesso;
- 400 (BAD REQUEST), ocorreu algum erro na requisição (podem existir inumeras causas);
- 404 (NOT FOUND), rota ou coleção não encontrada;
- 500 (INTERNAL SERVER ERROR), ocorreu algum erro no servidor.



## Exemplo de APIs Públicas

**ViaCep**

<https://viacep.com.br/>

**VIA CEP**  
Consulte CEPs de todo o Brasil

Procurando um **webservice** gratuito e de alto desempenho para consultar Códigos de Endereçamento Postal (CEP) do Brasil? Utilize nosso serviço, melhore a qualidade de suas aplicações web! e colabore para manter esta base de dados atualizada.

**Acessando o webservice de CEP**

Para acessar o webservice, um CEP no formato de **00** dígitos deve ser fornecido, por exemplo: **"01001000"**. Após o CEP, deve ser fornecido o tipo de retorno desejado, que deve ser **"json"**, **"xml"**, **"pipes"** ou **"query"**.

Exemplo de pesquisa por CEP:  
[viacep.com.br/vs/01001000/json/](https://viacep.com.br/vs/01001000/json/)





## Exemplo de APIs Públicas

Exemplo de API SOAP: <https://viacep.com.br/ws/01001000/xml/>

This XML file does not appear to have any style information associated with it. The document tree is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<cep>01001-000</cep>
<logradouro>Praça da Sé</logradouro>
<complemento>Lado ímpar</complemento>
<bairro>Sé</bairro>
<localidade>São Paulo</localidade>
<uf>SP</uf>
<ibge>3550308</ibge>
<gia>1004</gia>
<ddd>11</ddd>
<siafi>7107</siafi>
</cep>
```

Exemplo de API REST: <https://viacep.com.br/ws/01001000/json/>

```
{
  "cep": "01001-000",
  "logradouro": "Praça da Sé",
  "complemento": "lado ímpar",
  "bairro": "Sé",
  "localidade": "São Paulo",
  "uf": "SP",
  "ibge": "3550308",
  "gia": "1004",
  "ddd": "11",
  "siafi": "7107"
}
```



## Exemplo de APIs Públicas

### GitHub

<https://api.github.com/users/<<seu login>>>

### Repositório de APIs Públicas

<https://github.com/public-apis/public-apis>

### APIs do Star Wars

<https://swapi.dev/>

