

# Teoria - API

Escrito por: Ana Luiza Sampaio para {Reprograma}

Me encontre no instagram: [@analu.io](https://www.instagram.com/analu.io)  
no medium: [@sampaioaanaluiza](https://medium.com/@sampaioaanaluiza)  
ou me mande um e-mail: sampaioaanaluiza@gmail.com

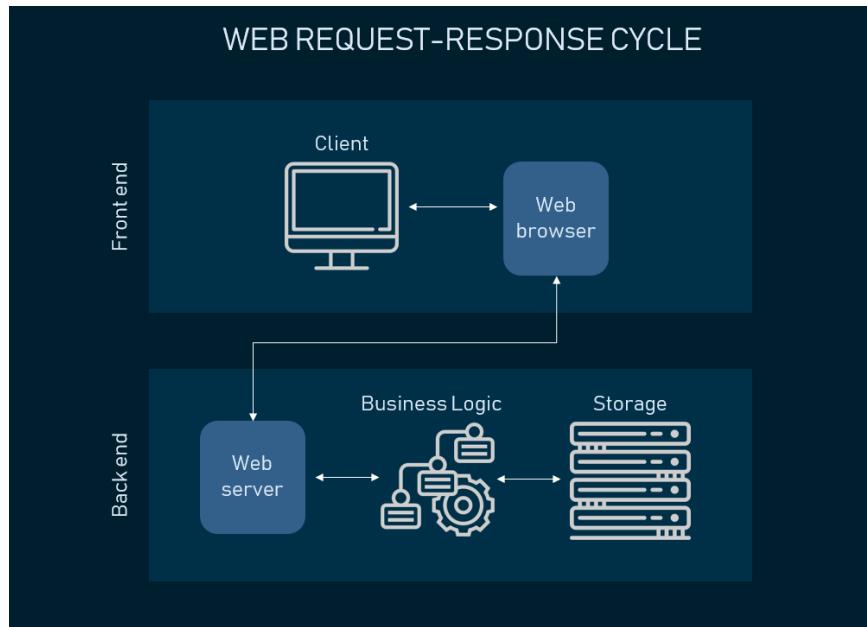


## Proposta

O propósito dessa conteúdo é definir o conceito do modelo server-client e HTTP, entender o que é uma API e uma Web API REST.

## Modelo Server/Client

A Web como conhecemos hoje tem uma arquitetura, ou seja, é construída dentro de um modelo Servidor/Cliente nele o processamento da informação é dividido em módulos ou processos distintos.



## Cliente

Entenda cliente como a interface que o usuário interage, seja por computador, celular, tablet, smart TV, ou qualquer outro dispositivo que possa se conectar com a internet. É o Cliente que **solicita** serviços e informações de um ou mais servidores.

Algumas tarefas a serem realizadas pelo Cliente:

- Manipulação de tela
- Interpretação de menus ou comandos
- Entrada e validação dos dados
- Recuperação de erro
- Manipulação de janelas
- Gerenciamento de som e vídeo (em aplicações multimídia)

Gerenciando a interação com o usuário, o cliente esconde do usuário o servidor e a rede, caso houver. Para o usuário a impressão é que a aplicação está sendo rodada completamente local.

## Servidor

E o Servidor é o responsável pelo processo, organização e gerenciamento das informações. É ele que **responde** às solicitações feitas pelo usuário. Ele é um processo reativo, disparado pela chegada de pedidos de seus clientes.

O processamento do servidor geralmente inclui:

- Acessar
- Organizar os dados compartilhados
- Fazer a comunicação com o Banco de Dados
- Atualizar dados previamente armazenados
- Gerenciamento dos recursos compartilhados.

## Comunicação

Páginas web são escritas usando HTML (Hyper Text markup language) e cada uma tem um endereço, a URL.

URL - Uniform Resource Locator(localizador de recurso uniforme). Ela representa um recurso específico na web. Recursos são coisas que eu quero interagir, como: imagens, páginas, arquivos, e videos.

As URLs tem um padrão que normalmente é composto por 4 partes:

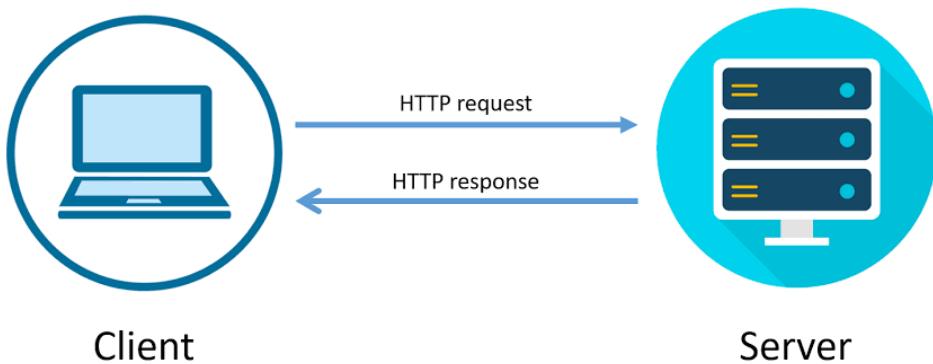
<protocolo>://<servidor>:<porta>/<recurso>



É esse protocolo HTTP que responsável pela comunicação.

## HTTP - Hipertext Tranfer Protocol

Protocolo de Transferência de Hipertexto é um protocolo usado dentro do modelo Client/Server é baseado em **pedidos (requests) e respostas (responses)**. Ele é a forma em que o Cliente e o Servidor se comunicam.

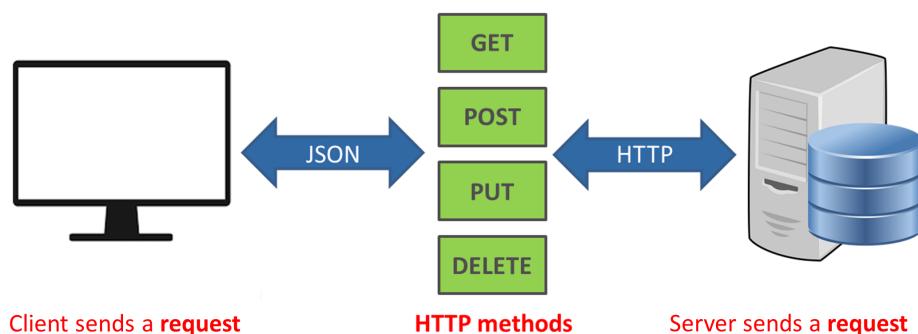


O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada. Eles são chamados de **Verbos HTTP**.

Os verbos HTTP mais utilizados são:

- GET
- POST
- PUT
- PATCH
- DELETE

Dessa forma o Cliente manda um request solicitando um dos verbos e o servidor deve estar preparado para receber e responde-lo.



## CRUD

CRUD é a composição da primeira letra de 4 operações básicas de um banco de dados, e são o que a maioria das aplicação faz:

-  **C:** Create (criar) - criar um novo registro
-  **R:** Read (ler) - exibir as informações de um registro
-  **U:** Update (atualizar) - atualizar os dados do registro
-  **D:** Delete (apagar) - apagar um registro

É importante o entendimento desses conceitos pois com os verbos HTTP podemos executar o CRUD.

## Operações CRUD com HTTP

Verbos HTTP	Operação CRUD
GET	Ler
POST	Criar
PUT	Substituir
PATCH	Modificar
DELETE	Excluir

## HTTP Status Codes

Quando o Client faz uma requisição o Servidor responde com um código de status numérico também padronizado.

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

- Respostas de informação (100-199)
- Respostas de sucesso (200-299)
- Redirecionamentos (300-399)
- Erros do cliente (400-499)
- Erros do servidor (500-599)

Você pode ver todo os codigos e seus usos em:

## HTTP Status Codes

httpstatuses.com is an easy to reference database of HTTP Status Codes with their definitions and helpful code references all in one place. Visit an individual status code via httpstatuses.com/code or browse the list below.

[http https://httpstatuses.com/](https://httpstatuses.com/)

É a desenvolvedora Back end que coloca na construção do servidor quais serão as situações referentes a cada resposta.

---

# API

Application Programming Interface, em português **Interface de Programação de Aplicativos**, é um conceito um pouco abstrato, né? Uma Interface que nos possibilita Programar Aplicativos... Muitas vezes quando as pessoas desenvolvedoras falam em API elas estão se referindo a **Web APIs**, que é um pouco diferente. Quero que você entenda o que de fato é uma Interface de Programação de Aplicativos, para isso precisamos definir o que é uma **interface**.

## Interface - o I de API

Preciso que você pense em um rádio.



Nós sabemos que ao aperta ou rodar um determinado botão podemos mudar de musica e até diminuir ou aumentar o volume certo? Porém nós não

sabemos ao **como** isso está acontecendo.

Como ao rodar um botão do rádio pra um lado a musica aumenta e quando viramos para o lado contrario ela diminui?

Não sabemos! E nem precisamo saber. O importante é que podemos usar mesmo não sabendo como de fato isso funciona internamente.

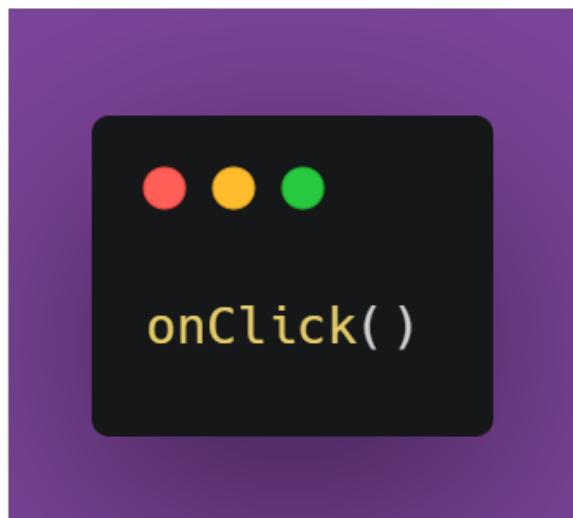


A informação de como o rádio funciona foi **abstraída** para que possamos usar o aparelho, ou seja, a informação de como o rádio funciona foi isolada para que possamos usar somente a função final dos botões, mesmo que ambas estejam conectadas.

A interface é uma forma de usarmos uma ferramenta sem precisar saber necessariamente o como.

O mesmo acontece num aplicativo de ouvir música no celular. Nós sabemos que ao clickar num botão podemos parar ou começar uma música, porém não sabemos como isso de fato funciona.

Acredito que a desenvolvedora que fez esse aplicativo, quando estava criando essa função de parar e começar musicas, deve ter provavelmente usado uma função pronta da linguagem, como por exemplo a função onClick( ) do JavaScript.



A desenvolvedora não precisou criar do zero todas as linhas dessa função e não sabe de fato o que está acontecendo nela, ela já veio pronta na linguagem.

Provavelmente, muitas outras pessoas desenvolvedoras tinham que criar essa função de click que dever ser trabalhosa, por isso, as pessoas que desenvolveram a linguagem resolveram colocar essa função lá, sem que as desenvolvedoras precisassem cria-la todas as vezes.

Foi criada um função pronta, **uma interface**, que facilita a programação de aplicativos.

Da mesma forma, acredito que a desenvolvedora não precisou programar a função que interage com o sistema operacional do celular (iOS ou Android) para ativar as saídas de sons dos aparelhos. A pessoa desenvolvedora provavelmente não sabe como de fato a musica começa tocar no aparelho, o próprio sistema operacional teve criar uma interface amigável para

desenvolvedora de apps, ou seja, existe uma função já pronta para que a desenvolvedora só interaja com ela.

Assim como no rádio, a API é uma interface que possibilita que usemos uma certa função, dado ou ferramenta sem precisar "reinventar a roda" na programação.

Ela não necessariamente está num link na Web, ela pode ser uma lib, um framework, uma função já pronta em uma linguagem específica, etc.

## Web API e API RESTful

Agora que sabemos o que é uma API fica mais fácil definir uma Web API.

Web API é uma interface que é disponibilizada de forma remota, pela web, que possibilita a programação aplicativos e softwares.

Uma empresa de software lança sua API para o público de modo que outros criadores de software possam desenvolver produtos usando esse serviço ou dados.

Alguns exemplos de Web APIs:

- API do Google Maps: onde não precisamos saber criar mapas naveáveis, só precisamos nos conectar a API que a própria google disponibiliza e ter os mapas criados pela Google dentro do nosso site.
- API do Correios: não precisamos criar um banco de dados com todos os CEPs do Brasil, o próprio correio disponibilizou uma API para que eu possamos utilizar e pesquisar os CEPs que desejamos e essa informação fica armazenada somente com eles.

E as APIs RESTfull são aquelas que são capazes de fazer o REST. Que nada mais é uma API que usa os protocolos HTTP.

## Bibliografia e referências

- **Artigo:** OLUWATOSIN, Haroon Shakirat. [Client-server model](#). IOSR J Comput Eng (IOSR-JCE), v. 16, n. 1, p. 67, 2014.

- **Site: UFRGS - Conteúdo da matéria de Redes**  
([http://www.penta.ufrgs.br/redes296/cliente\\_ser/cliente.htm](http://www.penta.ufrgs.br/redes296/cliente_ser/cliente.htm))
- **PDF: UFPE - Conteúdo da matéria de Desenvolvimento Web**  
([https://cin.ufpe.br/~erp/DesenvWeb/aulas/http\\_servlet/http.pdf](https://cin.ufpe.br/~erp/DesenvWeb/aulas/http_servlet/http.pdf))
- **PDF: UFRN - Conteúdo da matéria de Arquitetura**  
([https://cin.ufpe.br/~erp/DesenvWeb/aulas/http\\_servlet/http.pdf](https://cin.ufpe.br/~erp/DesenvWeb/aulas/http_servlet/http.pdf))
- **PDF: Universidade de Stanford - Conteúdo da matéria CS142**  
([https://cin.ufpe.br/~erp/DesenvWeb/aulas/http\\_servlet/http.pdf](https://cin.ufpe.br/~erp/DesenvWeb/aulas/http_servlet/http.pdf))
- **Artigo: "HTTP: Desmistificando o protocolo da Web"**  
(<https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1>)
- **Site: MDN web docs - Métodos de requisição HTTP**  
(<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>)
- **Blog: Angelo Públis - crud**  
(<https://angelopublio.com.br/blog/crud>)
- **Site: MDN web docs - Códigos de status de respostas HTTP**  
(<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>)
- **Video: freeCodeCamp - APIs for Beginners - How to use an API**  
(<https://www.youtube.com/watch?v=GZvSYJDk-us&t>)