

Teoria - API

Escrito por: Ana Luiza Sampaio para {Reprograma}

Me encontre no instagram: [@analu.io](https://www.instagram.com/@analu.io)

no twitter: [@analu_io](https://twitter.com/@analu_io)

ou me mande um e-mail: sampaioaanaluiza@gmail.com

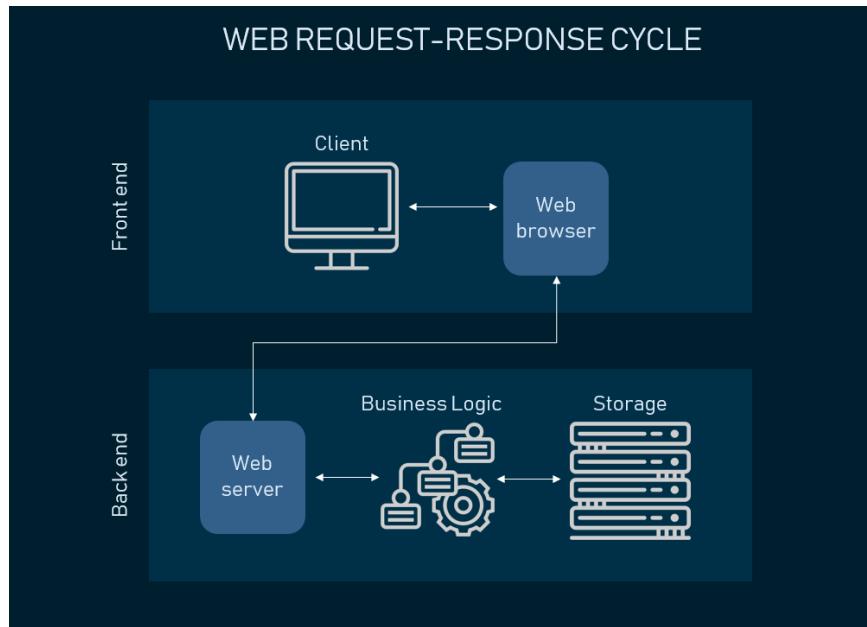


Proposta

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, ou Client, 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. Nos computadores conhecemos eles como os Browsers. 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.

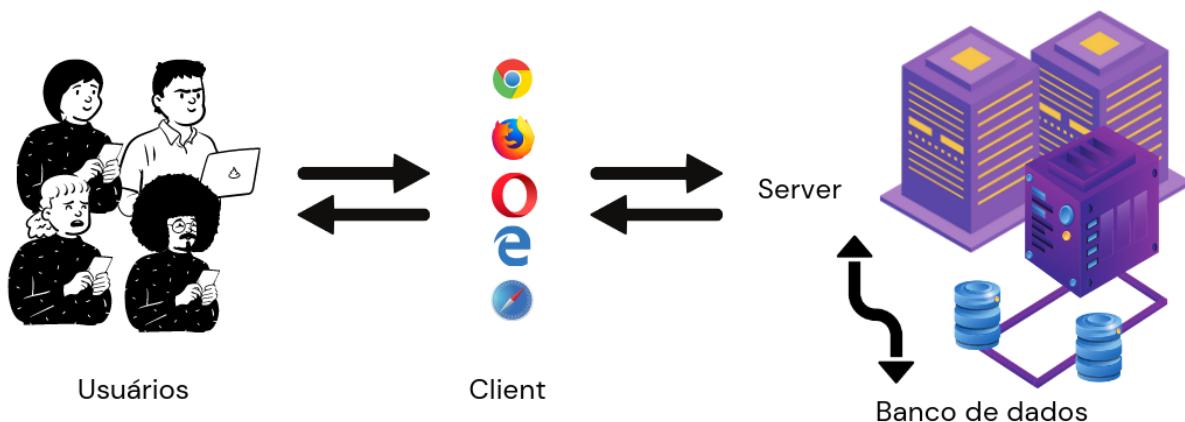
Servidor

E o Servidor, ou Server, é o responsável pelo processo, organização e gerenciamento das informações. É ele que **responde** às solicitações feitas pelo

Cliente. 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.



Importante entender que quando falamos de Client não estamos falando dos usuários, estamos falando dos softwares que são capazes de fazer requisições ao Server.
Usuárias são pessoas que utilizam e interagem com os Clients.

Comunicação

Todo site tem um **domínio**, normalmente é por ele que acessamos e conhecemos o Site.

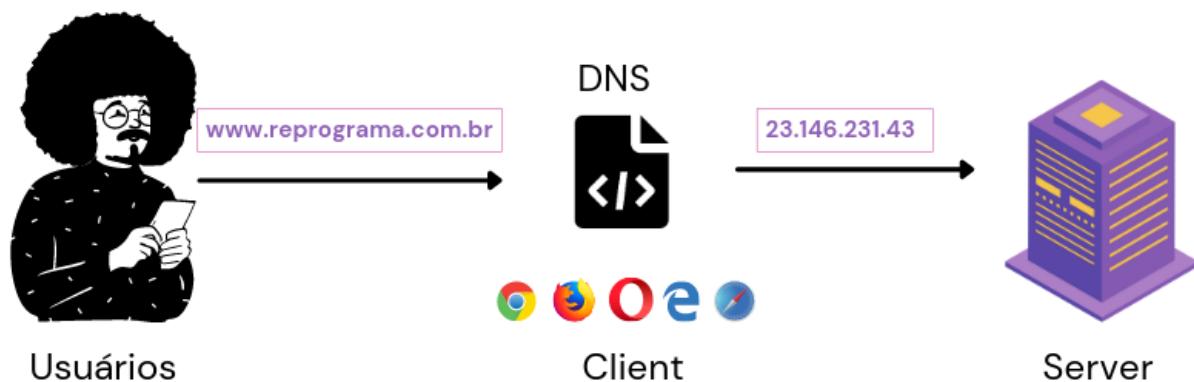
Os domínios foram criados para que nós humanos tivéssemos uma forma amigável para localizar documentos



na web, porém, no Server esse site não está registrado pelo nome de domínio, e sim pelo **endereço de IP**.

Internet Protocol Address, ou Endereço de Protocolo de Internet, é como o endereço da sua casa, ele representa a exata localização do site dentro do servidor, incluindo a porta.

Então, antes de uma requisição ser feita o domínio deve virar o IP, e pra isso, usamos o **DNS**, o Domain Name System (Sistema de Nome de Domínio) que é como um grande dicionário de domínio para IP que já vem "de fábrica" no browser. Ele tem a função de "traduzir" os domínios digitados pelas usuárias



Esse endereço completo que vemos na internet, que normalmente começa com http, é a URL.

URL - Uniform Resource Locator, ou localizador de recurso uniforme, representa um recurso específico na web, ou seja, cada uma das página, imagem, video ou arquivo web tem um endereço dentro da internet, esse endereço é a URL.

Ela normalmente é dividida em 4 partes:



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

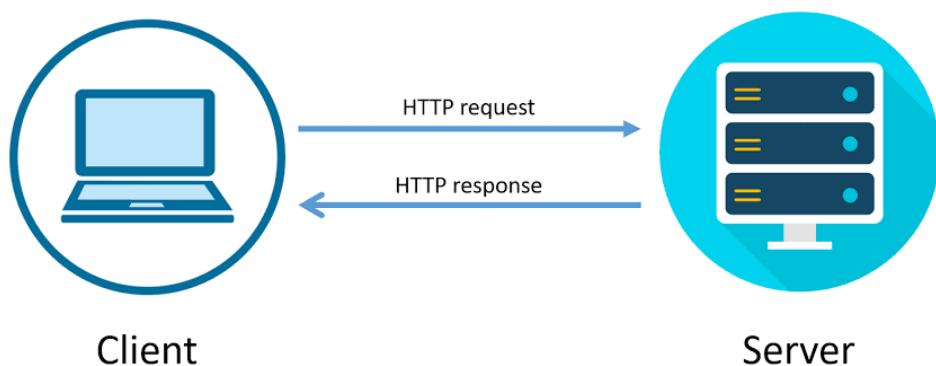
Sendo, `<servidor>:<porta>` o endereço de IP, ou endereço do servidor, que nós escrevemos somente o domínio



É esse protocolo HTTP que responsável pela comunicação.

HTTP - Hipertext Transfer 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.

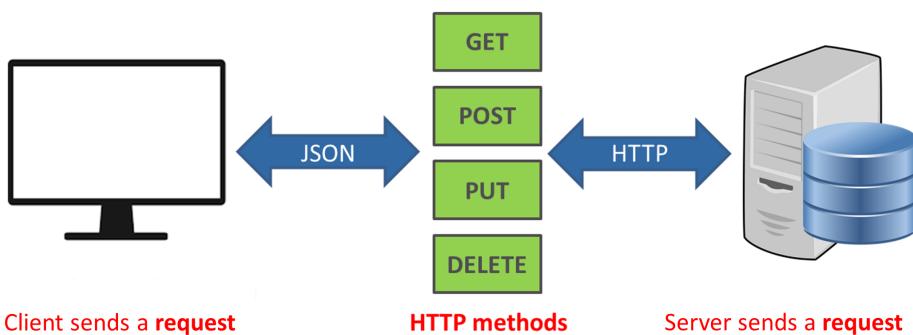


O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada. Eles são chamados de **Verbos HTTP ou Métodos HTTP**.

Os verbos HTTP mais utilizados são:

- GET
- POST
- PUT
- PATCH
- DELETE

Dessa forma o Client 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://httpstatuses.com/>

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



Hoje quase não usamos o HTTP, o mais adequado é o uso do **HTTPS**, ou Hypertext Transfer Protocol **Secure**, o grande diferencial é que o https tem diversos procedimentos e camadas de segurança como a criptografia de todos os dados enviados nas Requests e Responses. Dessa forma, dificultando que dados sejam facilmente interceptados durante a comunicação entre o Client e o Server.

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.

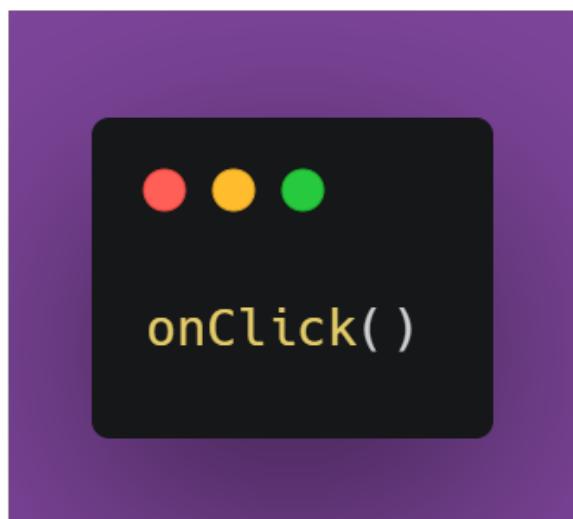


A informação de como o rádio funciona foi **abstraída** para que possamos usar o aparelho, ou seja, a informação de como o rádio funciona foi isolada para que possamos usar somente a função final dos botões, mesmo que ambas estejam conectadas.

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.

Tipos de Web APIs

APIs podem ser divididas principalmente pelo seu tipo de acesso, elas podem ser:

- **Públicas:** São aquelas que são disponibilizadas gratuitamente para desenvolvedoras e usuários com restrição mínima. Podem precisar de

cadastro, o uso de API Key ou ser completamente abertas.
Elas estão relacionadas com uso externo de dados ou serviços.

- **Privadas:** São oposto das APIs públicas. Somente pessoas autorizadas terão acesso. Elas normalmente estão ligadas à serviços sigilosos, dados sensíveis, transações de empresas privadas, comunicação e ferramentas interna da empresa ou organizações.

REST e RESTfull

As APIs RESTfull são aquelas que são capazes de fazer o REST. Que por enquanto podemos entender como uma API que usa os protocolos HTTP dentro do sistema Server/Client. Porém podemos nos aprofundar um pouco mais nessa arquitetura:

Rest, que é a abreviatura de Representational State Transfer, é um conjunto de princípios e definições utilizadas para que as requisições HTTP atendam as diretrizes definidas na arquitetura.

Dessa forma, uma API RestFull é aquela que atende todas as definições do Rest.

As Definições são:

1. **Interface Uniforme:** Uso correto dos verbos HTTP: GET, POST, PUT, DELETE, entre os demais verbos.
2. **Cliente-Servidor:** Separação de responsabilidades entre Front-end e Backend-end.
3. **Sem Estado:** Cada requisição que o Client faz para o Server, deverá conter todas as informações necessárias para o servidor entender e responder (Response) a requisição (Request).
ex.: A sessão do usuário deverá ser enviada em todas as requisições, para saber se aquele usuário está autenticado e apto a usar os serviços, e o servidor não pode lembrar que o cliente foi autenticado na requisição anterior.
4. **Cacheável:** As respostas do servidor devem determinar se aquela informação pode entrar ou não em um cache. Assim o cliente pode confiar se aquela resposta pode ser usada novamente em uma requisição equivalente.
5. **Camadas em Sistema:** permite que uma arquitetura seja composta de camadas hierárquicas, restringindo o comportamento do componente de

forma que cada componente não possa "ver" além da camada imediata com a qual está interagindo.

JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Para nós é mais tranquilo de compreender de ler e escrever. Para máquinas, é fácil de interpretar e gerar. Está baseado em um subconjunto da linguagem de programação JavaScript. JSON é em formato de texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados.

JSON é baseado em duas estruturas:

- Uma coleção de pares de nome / valor. Em várias linguagens, isso é realizado como um objeto, registro, estrutura, dicionário, tabela de hash, lista de chaves ou matriz associativa.
- Uma lista ordenada de valores. Na maioria das linguagens, isso é realizado como um array, vetor, lista ou sequência.

Estas são estruturas de dados universais. Virtualmente todas as linguagens de programação modernas as suportam, de uma forma ou de outra. É aceitável que um formato de troca de dados que seja independente de linguagem de programação se baseie nestas estruturas.

Em JSON, os dados são apresentados desta forma:

Um objeto é um conjunto desordenado de pares nome/valor. Um objeto começa com {chave de abertura e termina com }chave de fechamento. Cada nome é seguido por :dois pontos e os pares nome/valor são seguidos por ,vírgula.

```
"propriedade": "valor"
```

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

- **PDF: UFPE - Conteúdo da matéria de Desenvolvimento Web**
(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)
- **PDF: Universidade de Stanford - Conteúdo da matéria CS142**
(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>)
- **Site: What Is the Difference Between HTTP and HTTPS?**
(<https://www.keycdn.com/blog/difference-between-http-and-https>)
- **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>)
- **Repositório: Aula Api Rest e Restfull da Rocketseat**
(<https://github.com/rocketseat-content/youtube-api-rest-restful>)
- **Documentação: JSON**
(<https://www.json.org/json-pt>)