

{ reprograma }

Introdução à API

Como você chega?

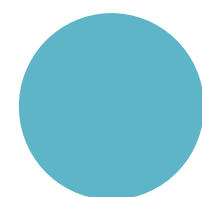
Manda no chat qual o seu **#checkin**
Sentimentos, expectativas pra aula...

{ reprograma }

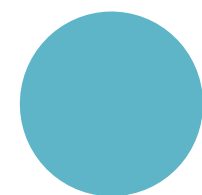
Prazer, Paula



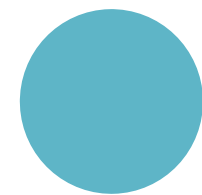
Combinados



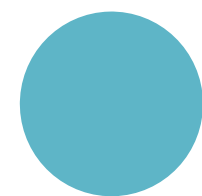
O que eu não sei, posso **descobrir**



Regra dos 3

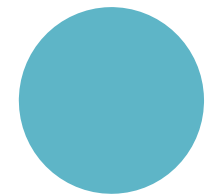


Atenção **plena!** 👁️👁️

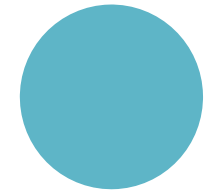


Dúvidas 🖐️

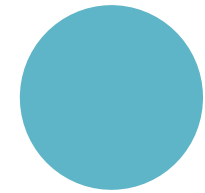
Agenda



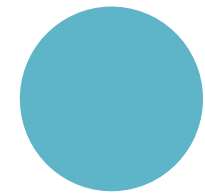
Modelo Server/Client



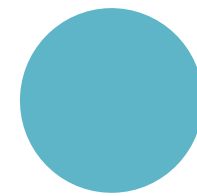
URL, Dominio, IP e DNS



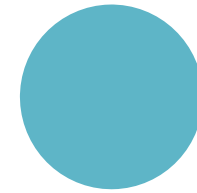
Protocolo HTTP



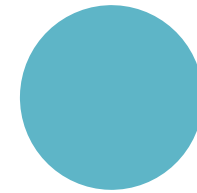
Request e Response



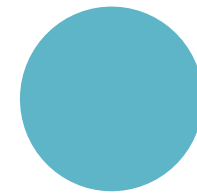
API e API Rest



JSON



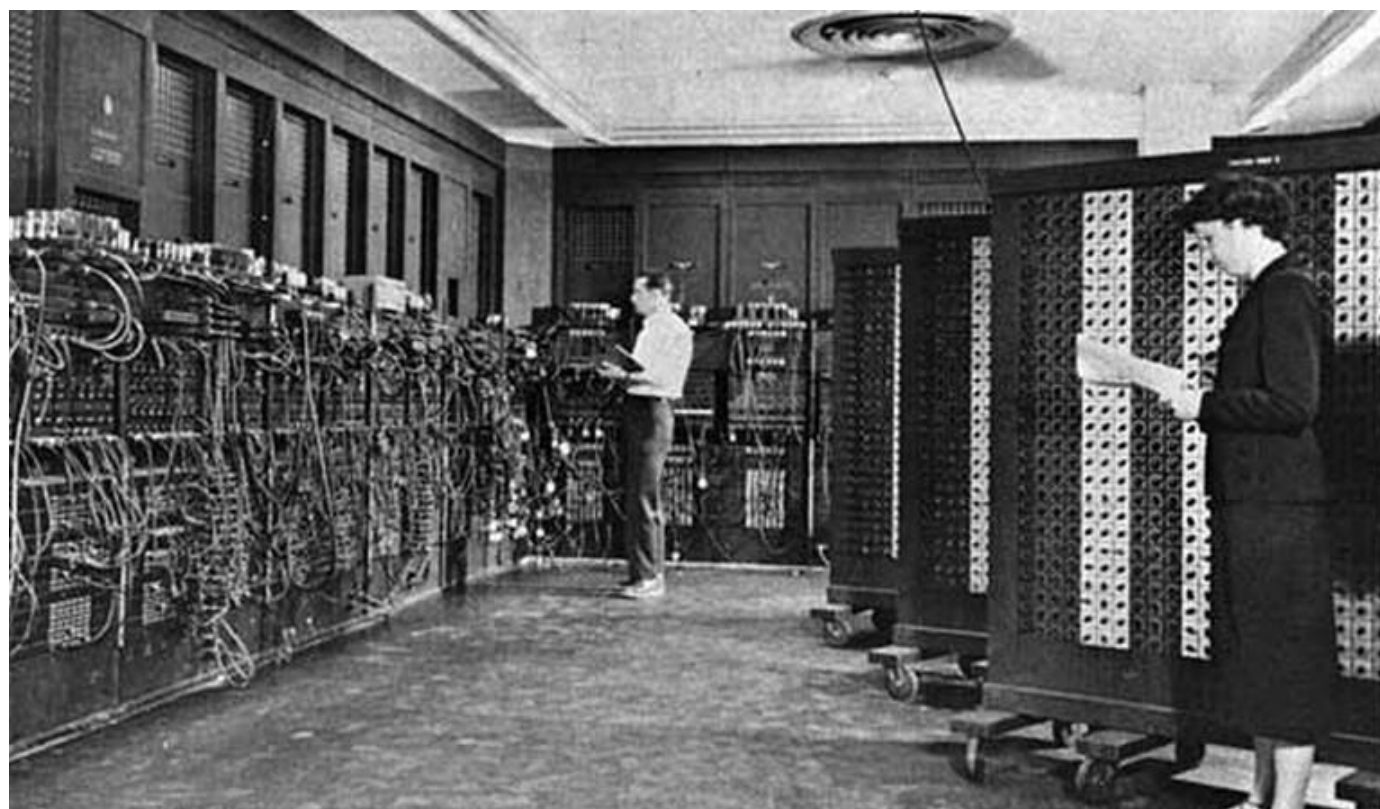
Consumindo APIs



Para o lar

Como o computador entende o mundo?

{ reprograma }



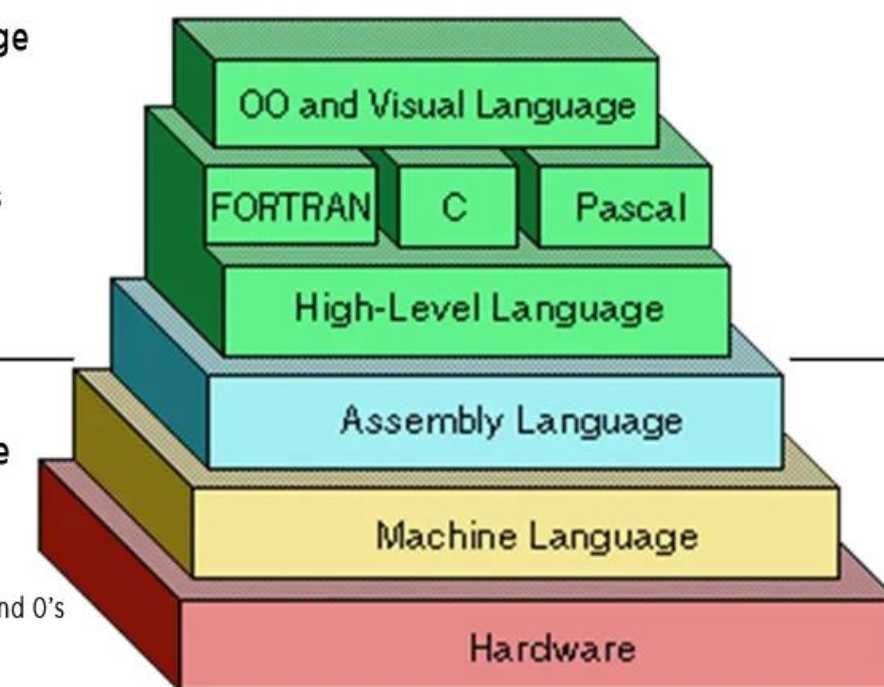
Machine Code
10011101000110100000
01100011010001110110
10000010111101101110
11110110001011011000
10000010011100011011
10010011000111000000

High Level Language

- Easy for Programmers to understand
- Contains English Words

Low Level Language

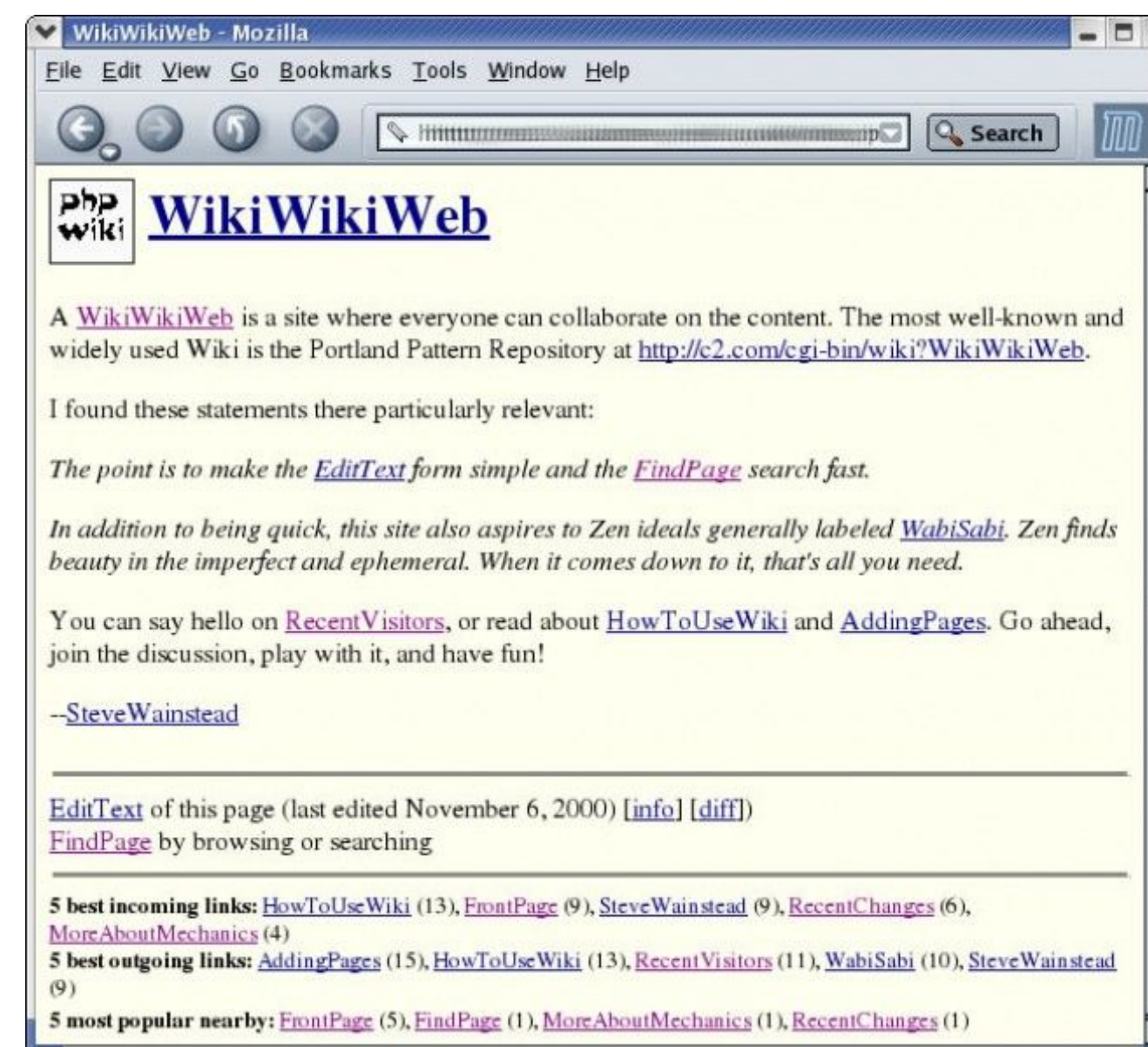
- The computer's own Language
- Binary numbers, in 1's and 0's



justcode.me

Como o computador entende o mundo?

{ reprograma }



{ reprograma }

Como funciona a Internet?

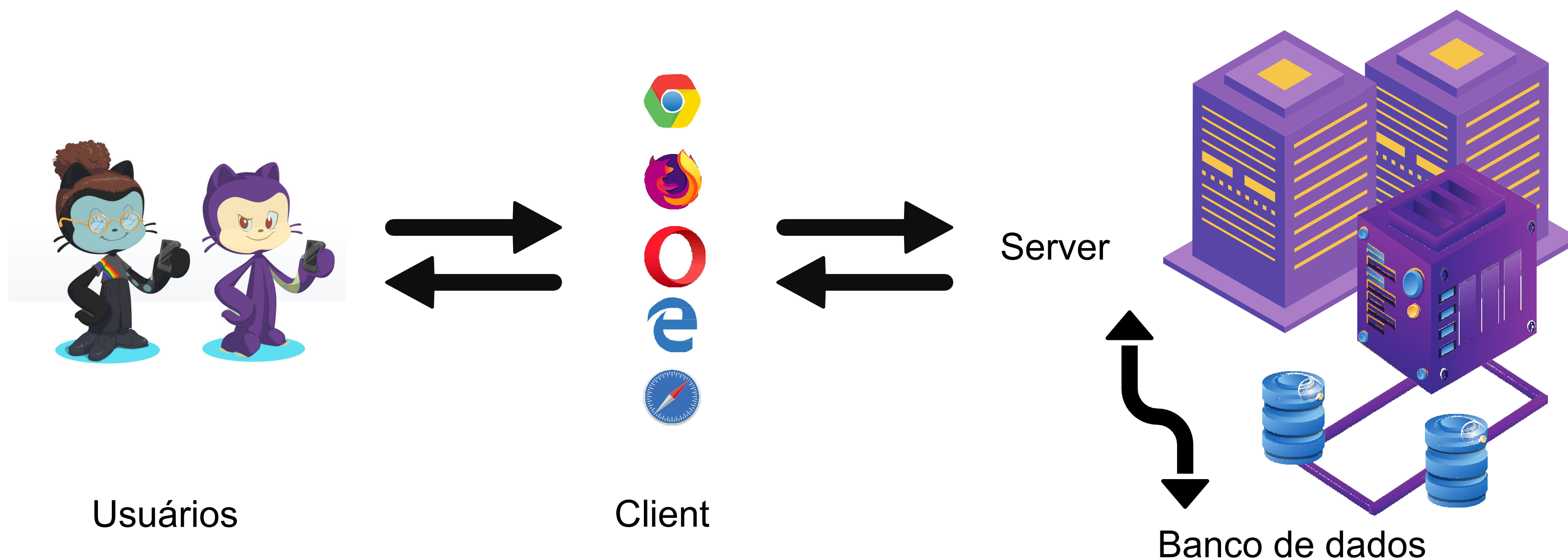
{ reprograma }

Comunicação



{ reprograma }

Servidor/Cliente



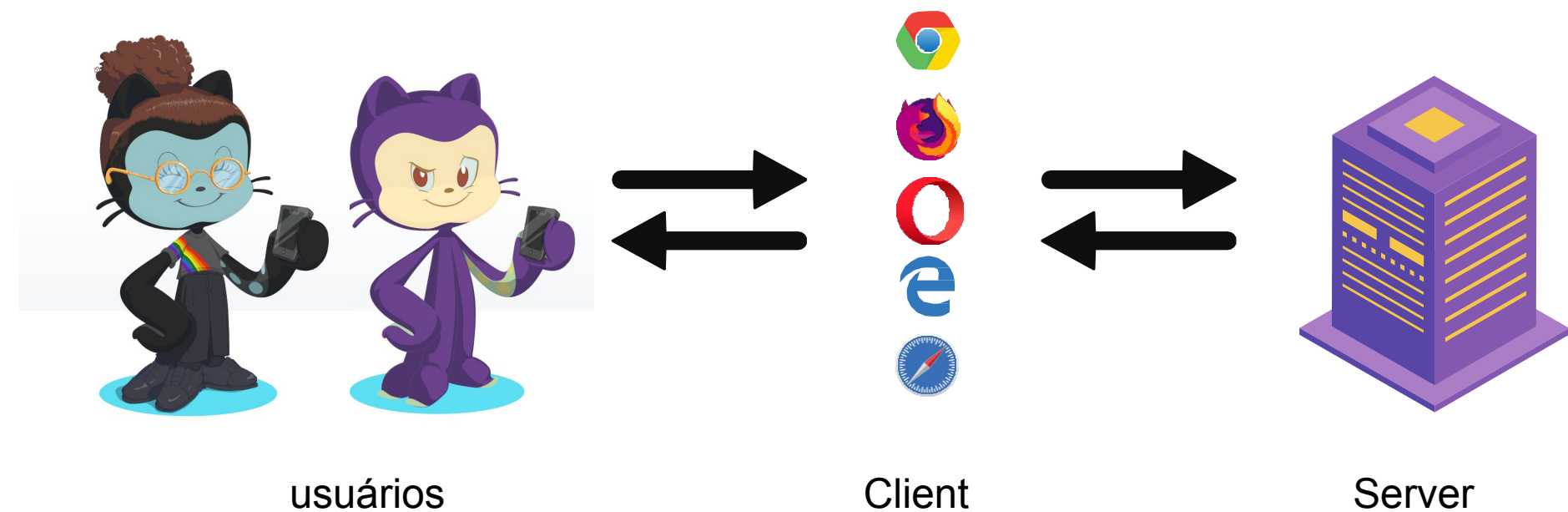
{ reprograma }

Client

Client é a interface que os usuários interagem, é essa camada que é responsável de **solicitar** 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)



{ reprograma }

Server

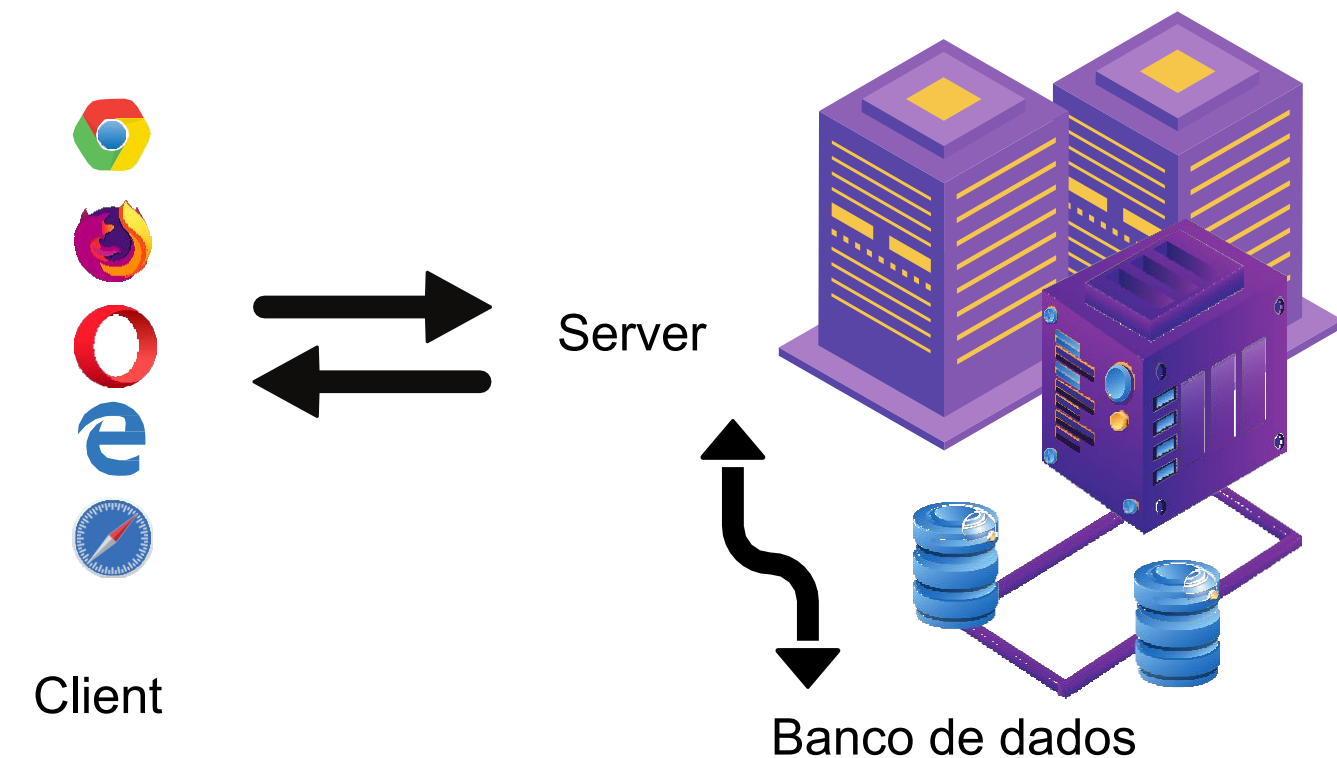
O Servidor é o responsável pelo processo, organização e gerenciamento das informações.

É ele que **responde às solicitações** feitas pelo Client.

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.



{ reprograma }

www.reprograma.com.br

O que acontece quando acessamos um site?



1

A URL é processada

2

É feita uma requisição

3

É dada uma Resposta

4

A página é renderizadas e aparece na tela

1

A URL é processada



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

Porém, no Server esse site não está registrado pelo nome de domínio, e sim pelo **endereço de IP**

Internet Protocol Address é o endereço exato de onde o site está dentro do servidor.

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

1

A URL é processada

URL - Uniform Resource Locator(localizador de recurso uniforme). Ela representa um recurso específico na web.

Cada as página, imagem, vídeo ou arquivo web tem um endereço dentro da internet, esse endereço é a **URL**



23.146.231.43

www.reprograma.com.br

http://

<protocolo>

www.reprograma.com.br

<endereço do servidor>

/courses

<recurso>

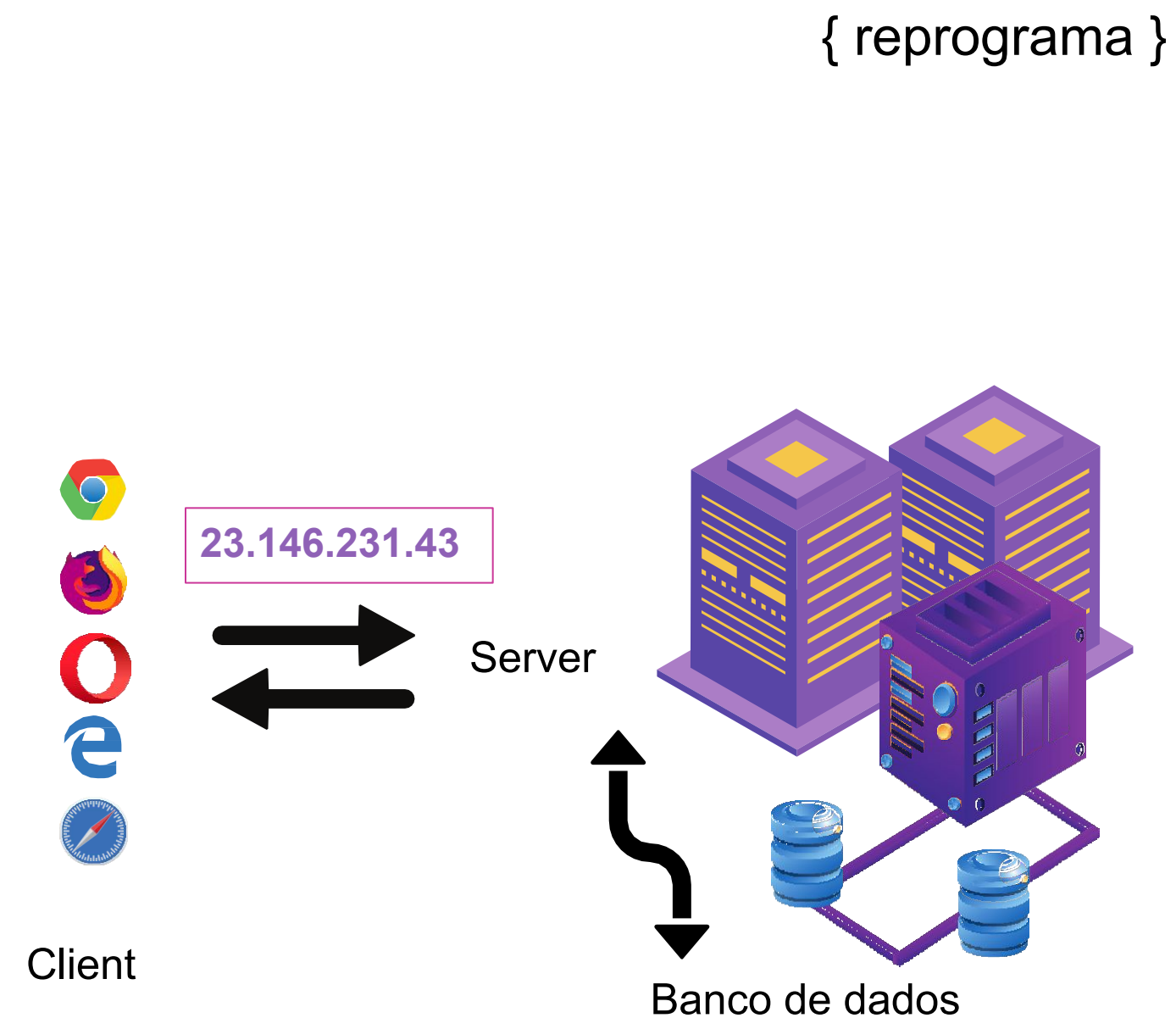
2

O Request é enviado

Agora com endereço certo, o Client faz uma requisição, ou Request, cheio de informações desejadas.

Pra que isso aconteça, tanto o Server quando o Client devem "falar a mesma língua".

Na maioria dos casos, essa comunicação entre Server e Client é feita a partir do **Protocolo HTTP**



2

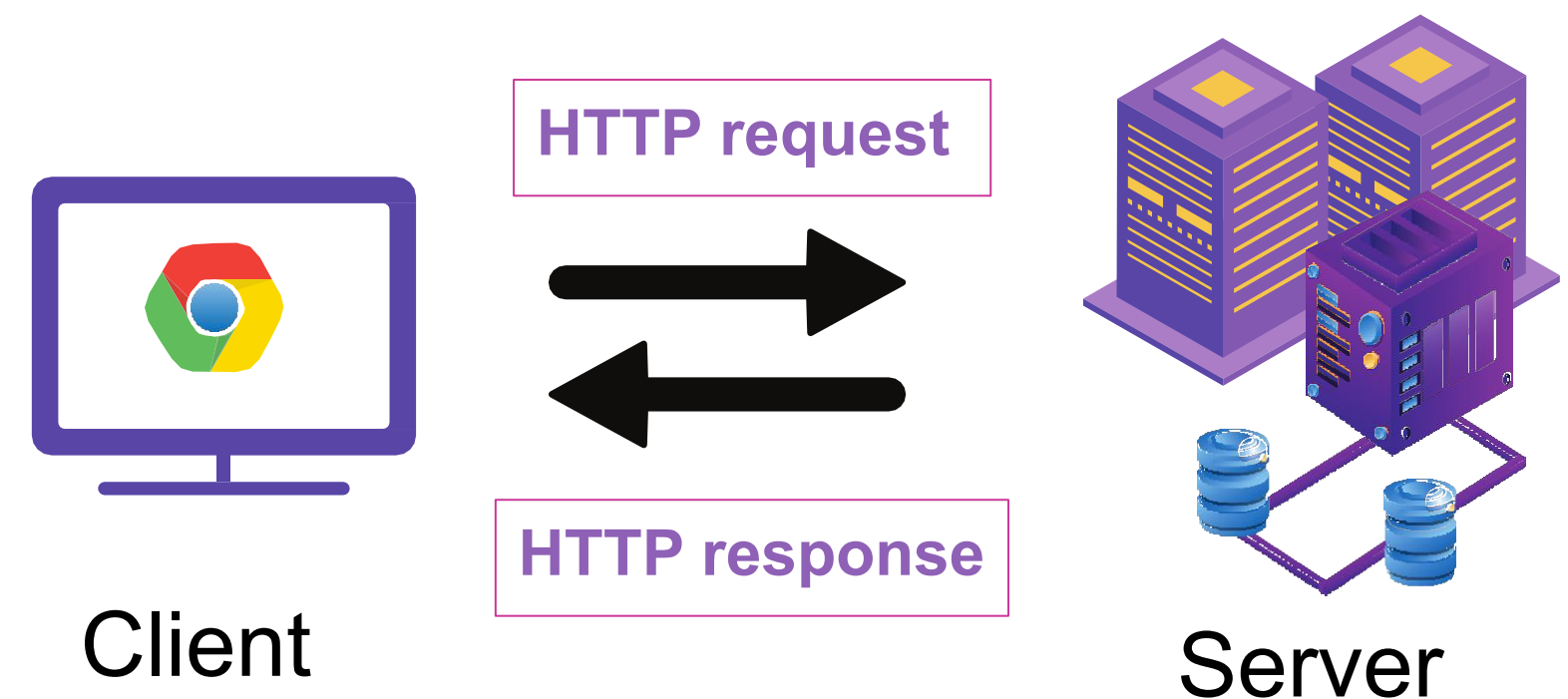
O Request é enviado

Protocolo de Transferência de Hipertexto, o HTTP, é um protocolo usado dentro do modelo Client/Server é baseado em pedidos (requests) e respostas (responses).

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**.

{ reprograma }



{ reprograma }

2

O Request é enviado

Os verbos HTTP mais utilizados são:

- GET
- POST
- PUT
- PATCH
- DELETE

Cada um deles corresponde a uma ação real no banco de dados.

GET

ler

POST

criar

PUT

substituir

PATCH

modificar

DELETE

excluir

{ reprograma }

3

O Server responde

Quando o Client faz um Request o Server envia um **Response**.
E na resposta tem, além do resultado do que foi pedido, um código de status numerico padronizado

| código | tipo de resposta |
|---------|------------------|
| 100-199 | informação |
| 200-299 | sucesso |
| 300-399 | redirecionamento |
| 400-499 | erro do cliente |
| 500-599 | erro de servidor |

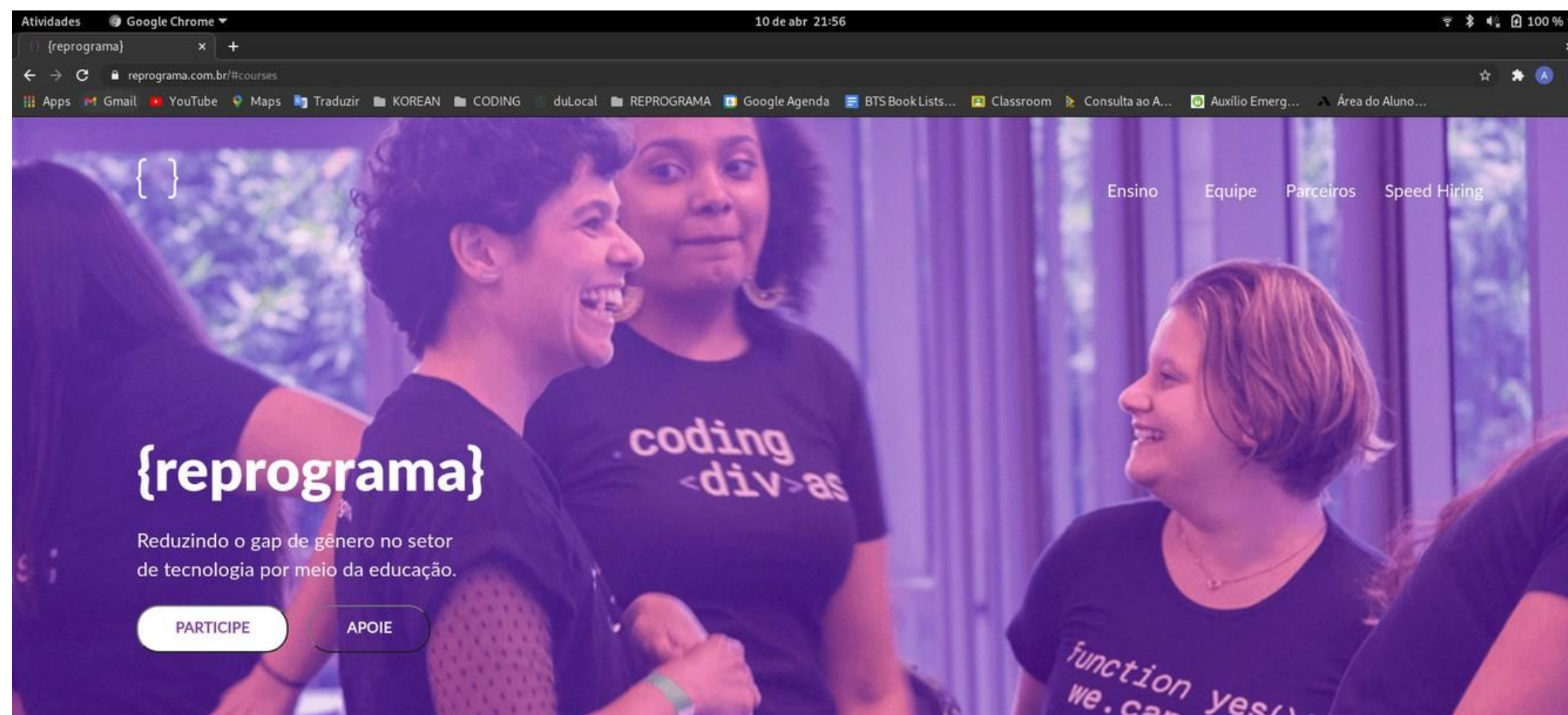
{ reprograma }

4

O site aparece na tela



Finalmente!



Objetivos



Modelo Server/Client



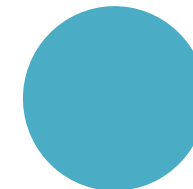
URL, Dominio, IP e DNS



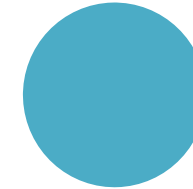
Protocolo HTTP



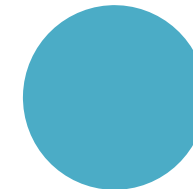
Request e Response



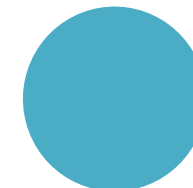
API e API Rest



JSON



Consumindo APIs



Para o lar

{ reprograma }

Dúvidas?

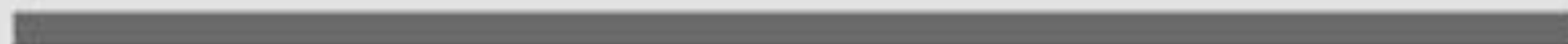
{ reprograma }

Intervalo!



15min

15:00



Dev Front

São as pessoas que são responsáveis muito mais do que construir as telas bonitas e funcionais.

Elas tem que criar aplicações preparadas para enviar Requests corretamente e receber as Responses, também disponibilizar elas para os usuários



Dev Back

Dentro desse fluxo, são as pessoas que constroem toda a dinâmica do recebimento de Requests, o envio das Responses corretas, o tratamento das Responses, as execuções de ação no Banco de Dados e a disponibilização para a Dev Front



{ reprograma }

As devs backend \o/



{ reprograma }

API

tá, mas o que realmente é isso?

Interface de Programação de Aplicativos

{ reprograma }

Interface

o I da API

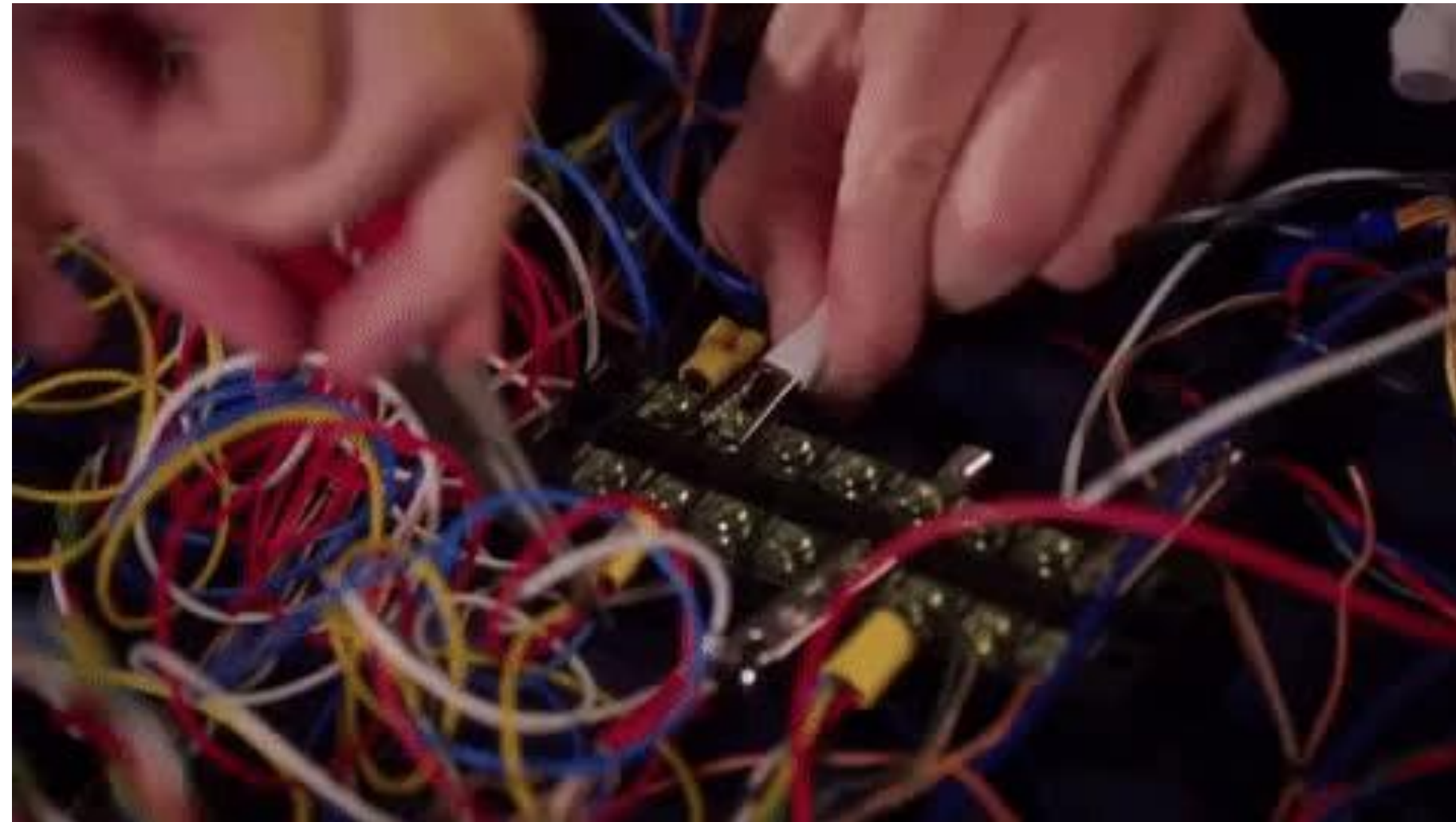


Imagine um
rádio

{ reprograma }

Interface

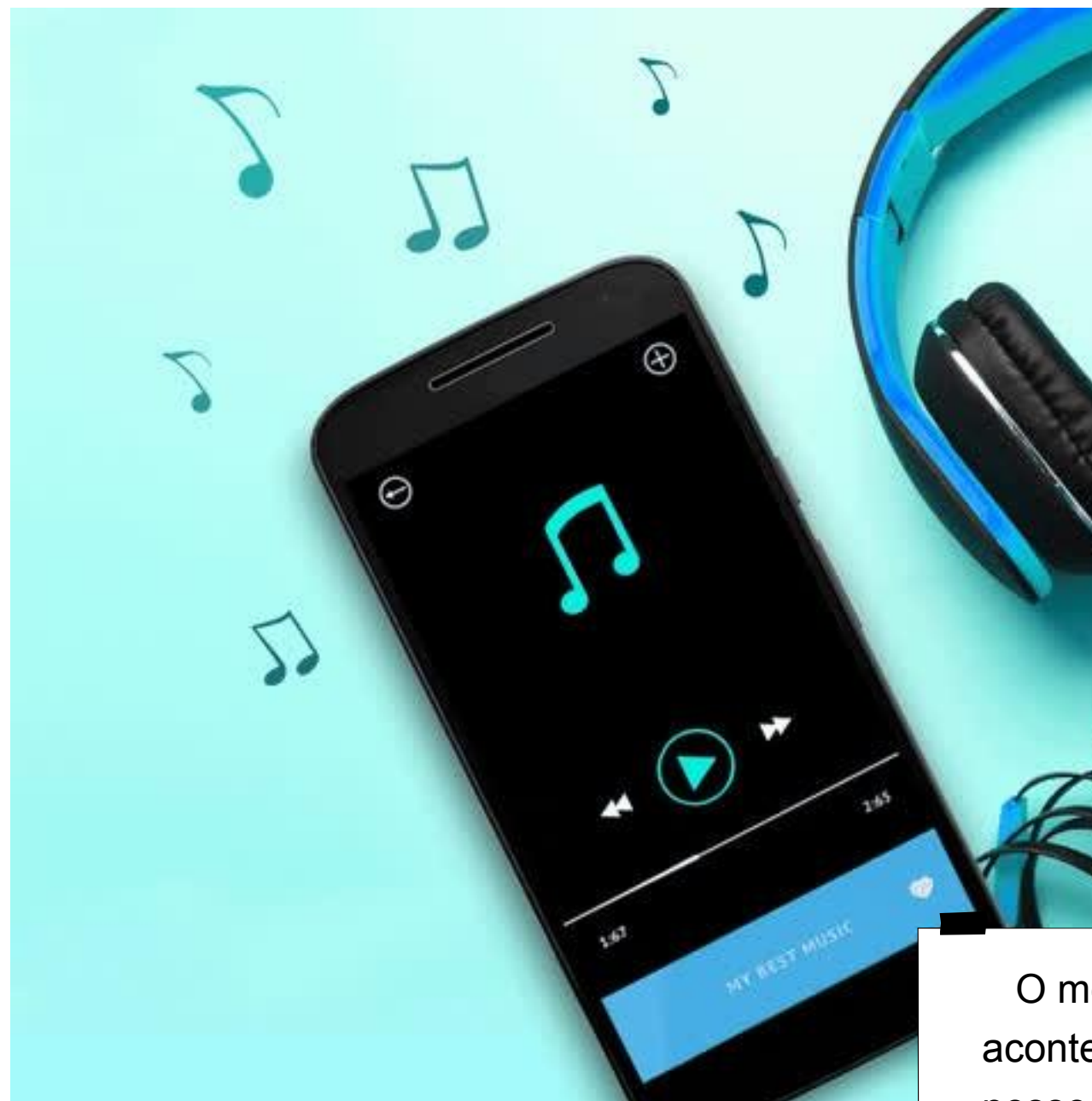
o I da API



{ Reprograma }

Interface

o I da API



O mesmo
acontece com
nosso celular

{ reprograma }

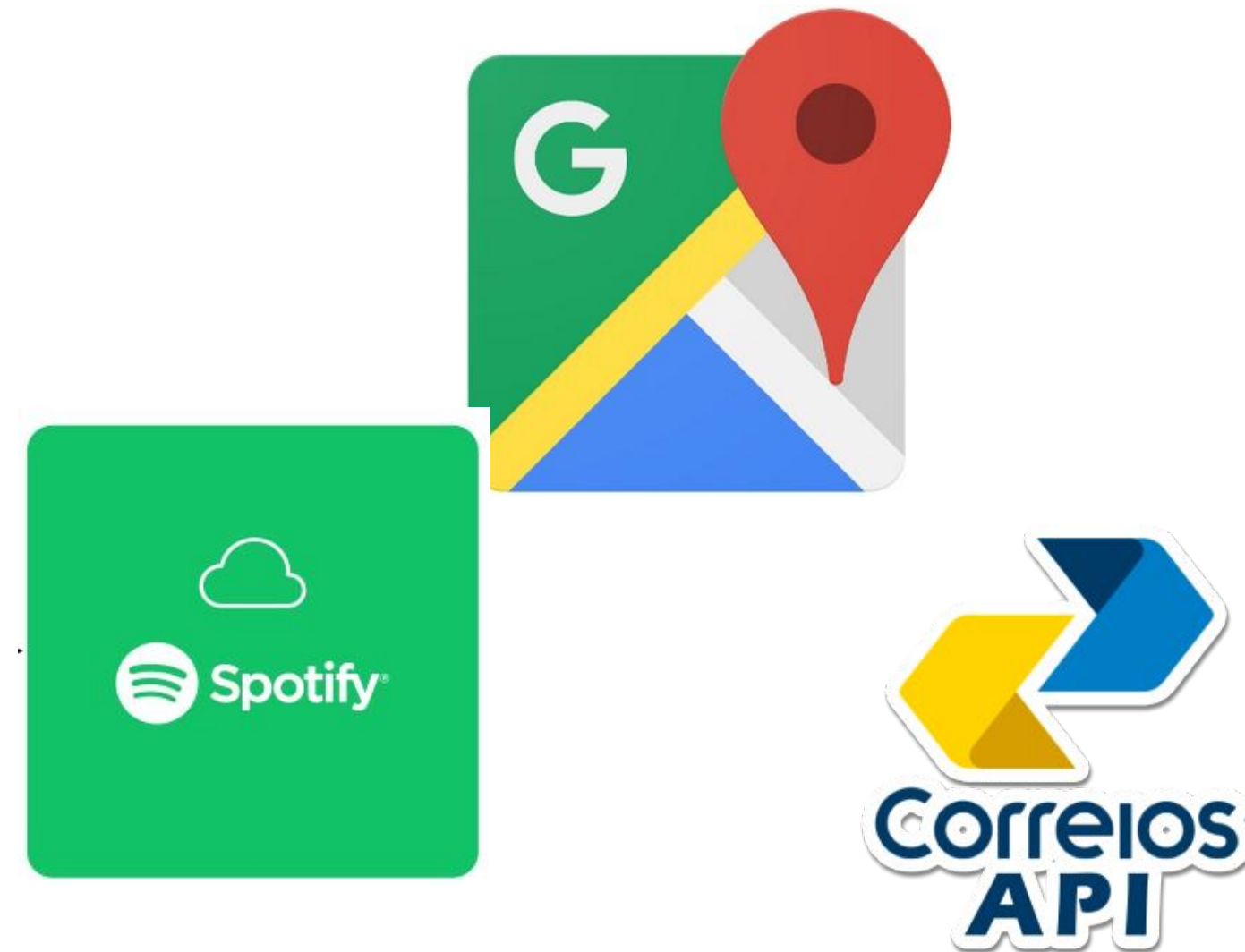
Interface de Programação de Aplicativos

Assim como a interface do rádio, a API busca criar formas e ferramentas de se usar uma funcionalidade ou uma informação sem realmente ter que "reinventar a tal função."

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

Web APIs

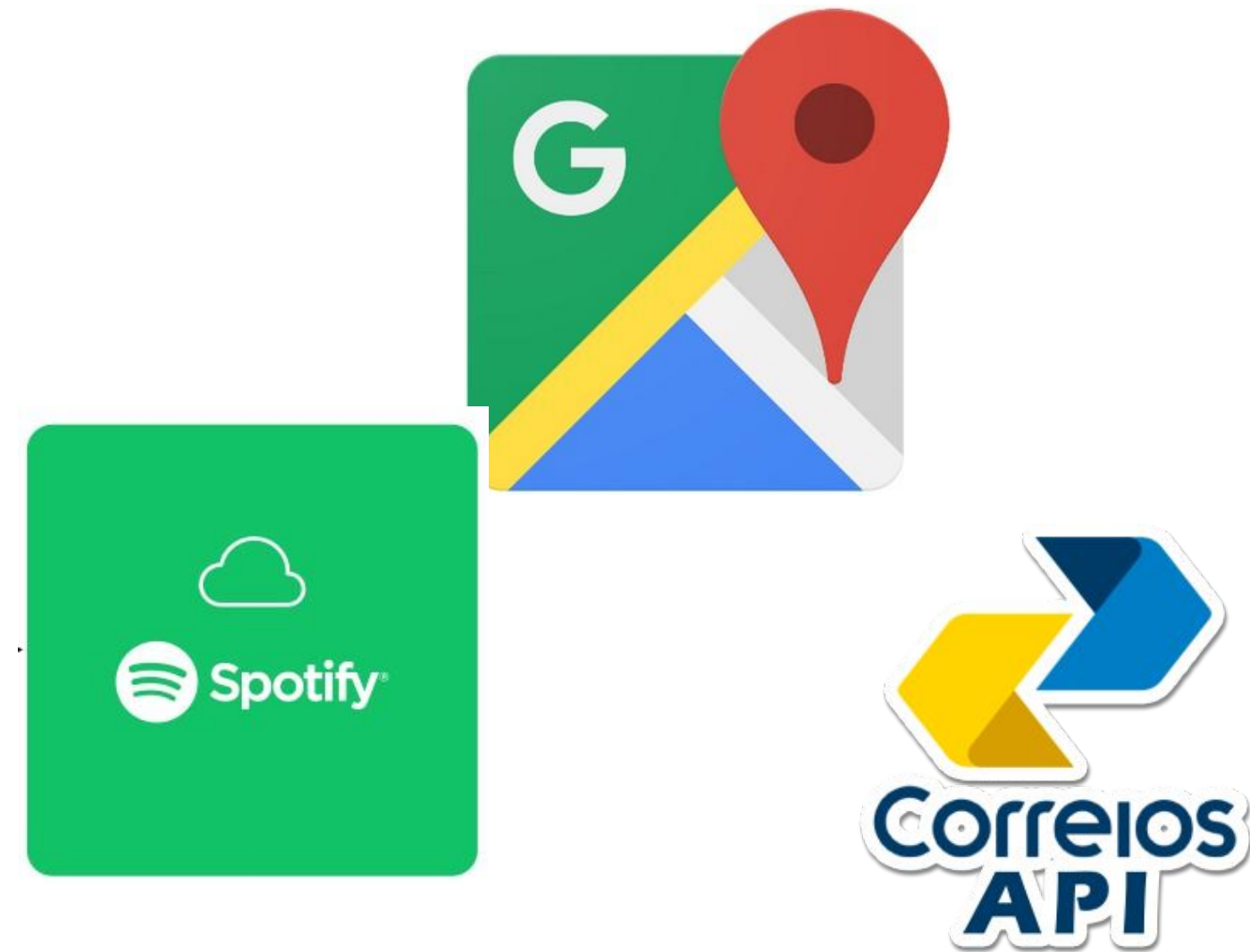
São um conjunto de instruções e padrões de programação para acesso a um aplicativo de software. Uma empresa de software lança sua API para o público de modo que outros criadores de software possam desenvolver produtos acionados por esse serviço.



APIs Publicas

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.



APIs Privadas

São oposto das APIs públicas. Elas estão ligadas à serviços sigilosos, dados sensíveis, transações de empresas privadas, comunicação e ferramentas interna da empresa, etc



APIs REST e RESTfull

Trata-se de um conjunto de princípios e definições necessários para a criação de um projeto com interfaces bem definidas, **Rest**, que é a abreviatura de **Representational State Transfer**, é um conjunto de restrições utilizadas para que as requisições HTTP atendam as diretrizes definidas na arquitetura.

{ reprograma }

Exemplos

[APIs Windows](#)

[APIs Públicas](#)

[As 50 APIs mais famosas](#)

[API Twitter](#)

[API Spotify](#)

Objetivos



Modelo Server/Client



URL, Dominio, IP e DNS



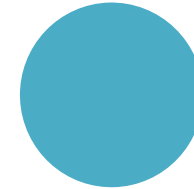
Protocolo HTTP



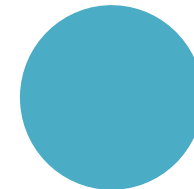
Request e Response



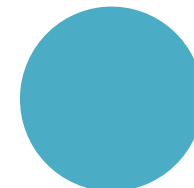
API e API Rest



JSON



Consumindo APIs



Para o lar

{ reprograma }

Dúvidas?

{ reprograma }

Almoço



JSON

- Notação de Objetos JavaScript.
- JSON é uma formatação leve de troca de dados.
- 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

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.

JSON

Você pode incluir os mesmos tipos de dados básicos como em um objeto JavaScript padrão.

Porém, diferente das Arrays e Objetos os nomes das propriedades devem ser **strings** com **aspas duplas** e as vírgulas à direita são proibidas.

```
const data = [{
  "name": "Bulbasaur",
  "description": "Bulbasaur can be seen napping in bright sunlight. There is a seed on its back. By soaking up the sun's rays, the seed grows progressively larger. Bulbasaur can be seen napping in bright sunlight. There is a seed on its back. By soaking up the sun's rays, the seed grows progressively larger.",
  "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/001.png",
  "types": ["poison", "grass"]
},
{
  "name": "Charmander",
  "description": "The flame that burns at the tip of its tail is an indication of its emotions. The flame wavers when Charmander is enjoying itself. If the Pokémon becomes enraged, the flame burns fiercely. The flame that burns at the tip of its tail is an indication of its emotions. The flame wavers when Charmander is enjoying itself. If the Pokémon becomes enraged, the flame burns fiercely.",
  "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/004.png",
  "types": ["fire"]
},
{
  "name": "Charmeleon",
  "description": "Charmeleon mercilessly destroys its foes using its sharp claws. If it encounters a strong foe, it turns aggressive. In this excited state, the flame at the tip of its tail flares with a bluish white color. Charmeleon mercilessly destroys its foes using its sharp claws. If it encounters a strong foe, it turns aggressive. In this excited state, the flame at the tip of its tail flares with a bluish white color.",
  "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/005.png",
  "types": ["fire"]
}]
```


1-exe-ghibli

Apresente no console, de cada objeto:

- título
- descrição

```
exercicios-JSON > 01-exe-ghibli > JS script.js > ...
1 //Apresente no console cada um dos atributos desse JSON
2
3 const obj = [
4   {
5     "title": "Castle in the Sky",
6     "description": "The orphan Sheeta inherited a mysterious crystal that I
7   },
8   {
9     "title": "Grave of the Fireflies",
10    "description": "In the latter part of World War II, a boy and his siste
11  },
12  {
13    "title": "My Neighbor Totoro",
14    "description": "Two sisters move to the country with their father in or
15  },
16  {
17    "title": "Kiki's Delivery Service",
18    "description": "A young witch, on her mandatory year of independent lif
19  },
20  {
21    "title": "Only Yesterday",
22    "description": "It's 1982, and Taeko is 27 years old, unmarried, and ha
23  }
24 ]
25
26 // COMEÇA O EXERCÍCIO
27
```

{ reprograma }

2-exe-pokemon

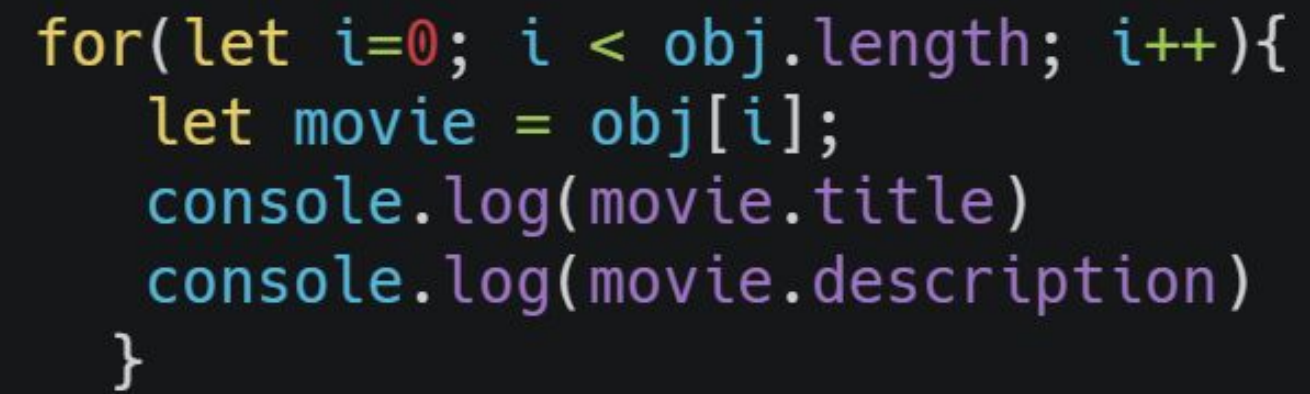
Apresente no console, de cada objeto:

- **name**
- **descrição**
- **tipos** (separadamente)

```
exercicios-JSON > 02-exe-pokemon > JS script.js > ...
1  const data = [{
2    "name": "Bulbasaur",
3    "description": "Bulbasaur can be seen napping in bright sunlight. There is a seed on its back. By soaking
4    "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/001.png",
5    "types": ["poison", "grass"]
6  },
7  {
8    "name": "Ivysaur",
9    "description": "There is a bud on this Pokémon's back. To support its weight, Ivysaur's legs and trunk gr
10   "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/002.png",
11   "types": ["poison", "grass"]
12 },
13 {
14   "name": "Venusaur",
15   "description": "There is a large flower on Venusaur's back. The flower is said to take on vivid colors if
16   "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/003.png",
17   "types": ["poison", "grass"]
18 },
19 {
20   "name": "Charmander",
21   "description": "The flame that burns at the tip of its tail is an indication of its emotions. The flame w
22   "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/004.png",
23   "types": ["fire"]
24 },
25 {
26   "name": "Charmeleon",
27   "description": "Charmeleon mercilessly destroys its foes using its sharp claws. If it encounters a strong
28   "art_url": "http://assets22.pokemon.com/assets/cms2/img/pokedex/full/005.png",
29   "types": ["fire"]
30 }]
31
32 // COMEÇA O EXERCÍCIO
```

{ reprograma }

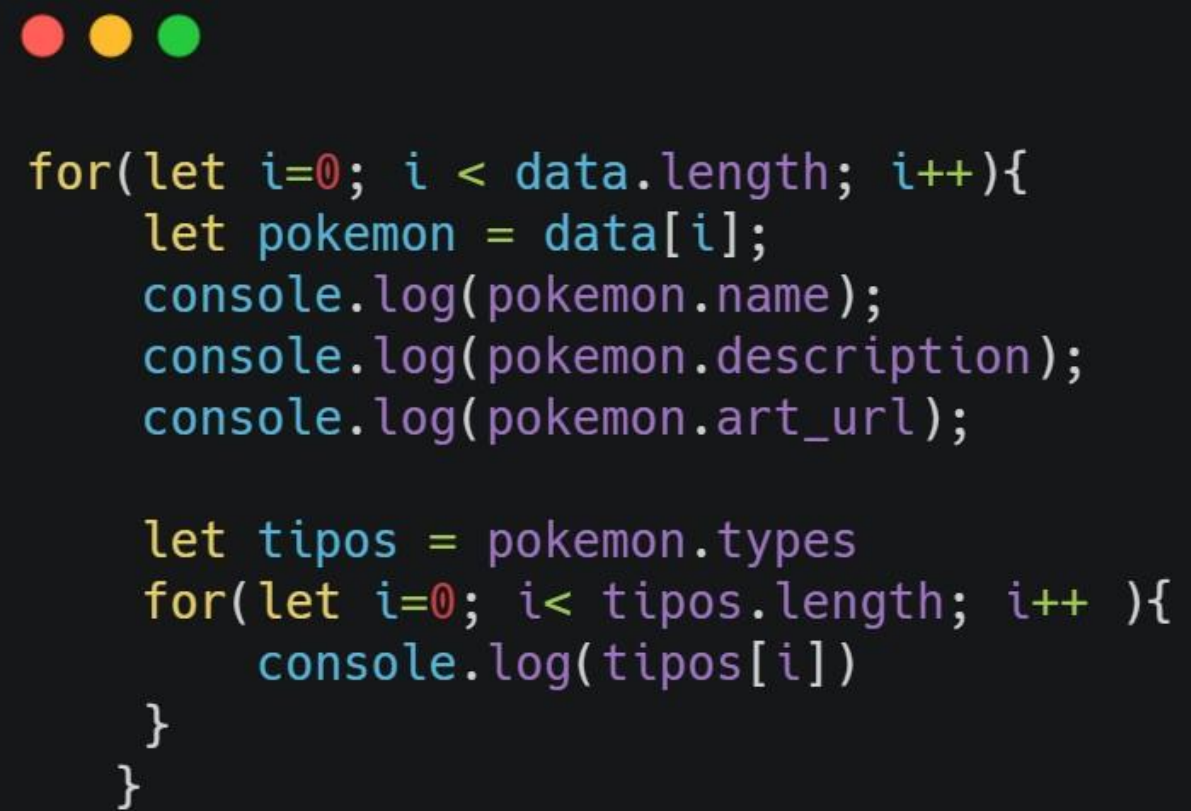
1-exe-ghibli



```
for(let i=0; i < obj.length; i++){  
  let movie = obj[i];  
  console.log(movie.title)  
  console.log(movie.description)  
}
```

{ reprograma }

2-exe-pokemon



```
for(let i=0; i < data.length; i++){  
  let pokemon = data[i];  
  console.log(pokemon.name);  
  console.log(pokemon.description);  
  console.log(pokemon.art_url);  
  
  let tipos = pokemon.types  
  for(let i=0; i< tipos.length; i++){  
    console.log(tipos[i])  
  }  
}
```

Objetivos



Modelo Server/Client



URL, Dominio, IP e DNS



Protocolo HTTP



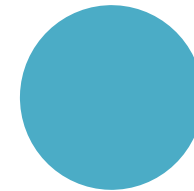
Request e Response



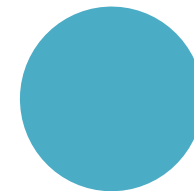
API e API Rest



JSON



Consumindo APIs



Para casa

{ reprograma }

Dúvidas?

{ reprograma }

Consumindo API

[Documentação](#)

スタジオジブリ作品 STUDIO GHIBLI

🔍 Search

STUDIO GHIBLI API

GETTING STARTED

BASE URL

AUTHENTICATION

WORKFLOW

USE CASE

HELPER LIBRARIES >

ENDPOINTS

FILMS >

PEOPLE >

LOCATIONS >

SPECIES >

VEHICLES >

Studio Ghibli API (v1.0.1)

Download OpenAPI specification: [Download](#)

Studio Ghibli API

The Studio Ghibli API catalogs the people, places, and things found in the worlds of Ghibli. It was created to help users discover resources, consume them via HTTP requests, and interact with them in whatever way makes sense. Navigation can be found on the left sidebar, and the right sidebar shows examples of returned objects for successful calls.

Users can raise an issue, ask for help, or find a contribution guide at the main repo: <https://github.com/janaipakos/ghibliapi>

Getting Started

Requests can be made with `curl` or other helper libraries by following regular REST calls. For example, here is how to GET the resource for the film *My Neighbor Totoro*.

```
curl https://ghibliapi.herokuapp.com/films/58611129-2dbc-4a81-a72f-77ddfc1b1b49
```

Calling this resource will respond with the following object:

```
{
  "id": "58611129-2dbc-4a81-a72f-77ddfc1b1b49",
  "title": "My Neighbor Totoro",
  "description": "Two sisters move to the country with their father in order to be closer to their hospitali",
  "director": "Hayao Miyazaki",
  "producer": "Hayao Miyazaki",
  "release_date": "1988",
  "rt_score": "93",
  ...
}
```

Consumindo API

curl

[Docs Overview](#) [Project](#) [Protocols](#) [Releases](#) [Tool](#) [Who and Why](#)

[curl](#) / [Docs](#) / [Tool Documentation](#) / [Man Page](#)

curl.1 the man page

NAME

curl - transfer a URL

SYNOPSIS

curl [options / URLs]

DESCRIPTION

curl is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command is designed to work without user interaction.

curl offers a busload of useful tricks like proxy support, user authentication, FTP upload, HTTP post, SSL connections, cookies, file transfer resume, Metalink, and more. As you will see below, the number of features will make your head spin!

curl is powered by libcurl for all transfer-related features. See *libcurl(3)* for details.

URL

The URL syntax is protocol-dependent. You'll find a detailed description in [RFC 3986](#).

You can specify multiple URLs or parts of URLs by writing part sets within braces and quoting the URL as in:

```
"http://site.{one,two,three}.com"
```

Related:

[File a bug about this man page](#)
[Manual](#)
[FAQ](#)
[HTTP Scripting](#)

Consumindo API

Consumindo com curl



Requests podem ser feitas com `curl` ou outras libs que seguem o protocolo REST. Por exemplo, aqui tem como fazer um GET para o filme Me Visinho Totoro:

```
curl https://ghibliapi.herokuapp.com/films/58611129-2dbc-4a81-a72f-77ddfc1b1b49
```



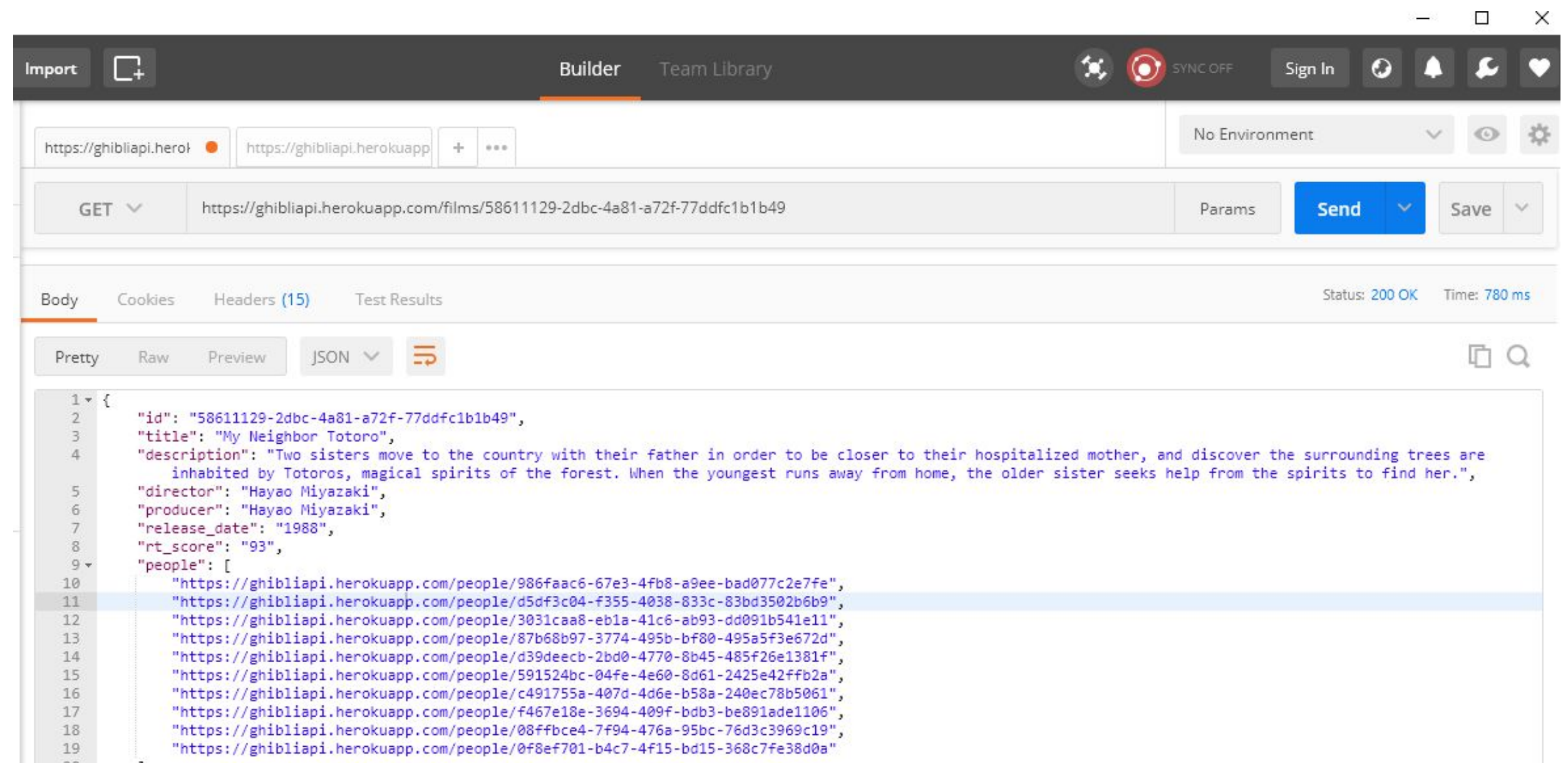
```
$ curl GET https://ghibliapi.herokuapp.com/films/58611129-2dbc-4a81-a72f-77ddfc1b1b49
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed

  0     0     0     0     0     0     0     0  0:00:01 0:00:01 --:--:--    0curl: (6) Could not
resolve host: GET
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100  1868  100  1868    0     0  2587     0 --:--:-- --:--:-- --:--:--  2587{
  "id": "58611129-2dbc-4a81-a72f-77ddfc1b1b49",
  "title": "My Neighbor Totoro",
  "description": "Two sisters move to the country with their father in order to be closer to their
hospitalized mother, and discover the surrounding trees are inhabited by Totoros, magical spirits of
the forest. When the youngest runs away from home, the older sister seeks help from the spirits to find
her.",
  "director": "Hayao Miyazaki",
  "producer": "Hayao Miyazaki",
  "release_date": "1988",
  "rt_score": "93",
  "people": [
    "https://ghibliapi.herokuapp.com/people/986faac6-67e3-4fb8-a9ee-bad077c2e7fe",
    "https://ghibliapi.herokuapp.com/people/d5df3c04-f355-4038-833c-83bd3502b6b9",
    "https://ghibliapi.herokuapp.com/people/3031caa8-eb1a-41c6-ab93-dd091b541e11",
    "https://ghibliapi.herokuapp.com/people/87b68b97-3774-495b-bf80-495a5f3e672d",
    "https://ghibliapi.herokuapp.com/people/d39deecb-2bd0-4770-8b45-485f26e1381f",
    "https://ghibliapi.herokuapp.com/people/591524bc-04fe-4e60-8d61-2425e42ffb2a",
    "https://ghibliapi.herokuapp.com/people/c491755a-407d-4d6e-b58a-240ec78b5061",
    "https://ghibliapi.herokuapp.com/people/f467e18e-3694-409f-bdb3-be891ade1106",
    "https://ghibliapi.herokuapp.com/people/08fffbce4-7f94-476a-95bc-76d3c3969c19",
    "https://ghibliapi.herokuapp.com/people/0f8ef701-b4c7-4f15-bd15-368c7fe38d0a"
  ],
  "species": [
    "https://ghibliapi.herokuapp.com/species/af3910a6-429f-4c74-9ad5-dfe1c4aa04f2",
    "https://ghibliapi.herokuapp.com/species/603428ba-8a86-4b0b-a9f1-65df6abef3d3",
    "https://ghibliapi.herokuapp.com/species/74b7f547-1577-4430-806c-c358c8b6bcf5"
  ],
  "locations": [
    "https://ghibliapi.herokuapp.com/locations/"
  ],
  "vehicles": [
    "https://ghibliapi.herokuapp.com/vehicles/"
  ],
  "url": "https://ghibliapi.herokuapp.com/films/58611129-2dbc-4a81-a72f-77ddfc1b1b49",
  "length": null
}
```

{ reprograma }

Consumindo API

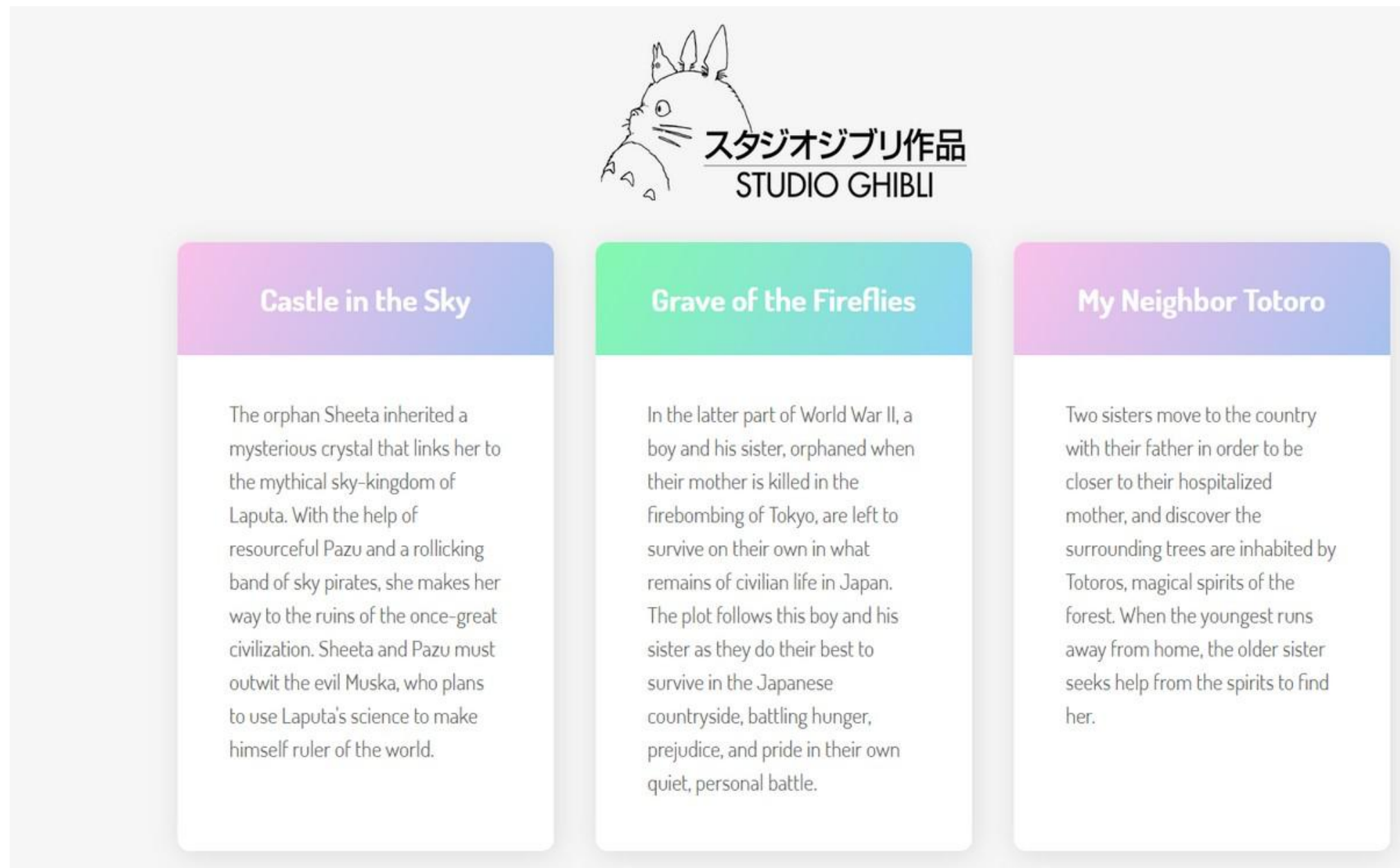
Consumindo com Postman



{ reprograma }

Consumindo API

Consumindo como um Front



```
const app = document.getElementById('root');
const container = document.createElement('div');
container.setAttribute('class', 'container');

app.appendChild(container);

const request = new XMLHttpRequest();

request.open('GET', 'https://ghibliapi.herokuapp.com/films', true);

request.onload = function(){
  const data = JSON.parse(this.response);

  console.log(data)
  console.log(data[0].title)

  if(request.status >= 200 && request.status < 400){
    console.log("SUCESSO!!!")
    data.forEach(movie => {
      const card = document.createElement('div');
      card.setAttribute('class', 'card');

      const h1 = document.createElement('h1');
      h1.textContent = movie.title;

      const p = document.createElement('p');
      movie.description = movie.description;
      p.textContent = movie.description;

      container.appendChild(card);
      card.appendChild(h1);
      card.appendChild(p);
    });
  }else{
    console.log("erro :c")
  }
}
request.send()
```

Objetivos



Modelo Server/Client



URL, Dominio, IP e DNS



Protocolo HTTP



Request e Response



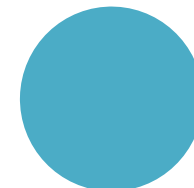
API e API Rest



JSON



Consumindo APIs



Para casa

{ reprograma }

Dúvidas?

{ reprograma }



Para casa



{ reprograma }



Para casa

- 1) Qual a relação entre os métodos HTTP e o CRUD?
- 2) Comente, com exemplos, a diferença entre o PUT e o PATCH.
- 3) Assim como na aula, apresente os dados dos JSONs no console
- 4) Defina o conceito de idempotência e como uma API pode ser idempotente
- 5) Cite alguns diferentes padrões de projetos de software