

Node - Consumo de APIs Externas

Material Complementar



Marianne Salomão

Cloud Engineer na IBM Brasil, desenvolvedora de software full - stack e instrutora de TI.



linkedin.com/in/mariannesalomao



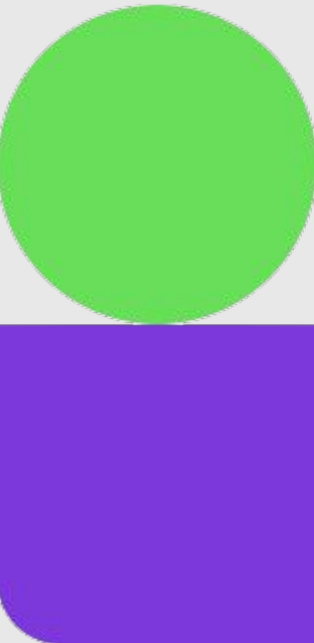
github.com/mariannesalomao



#PraCegoVer: Fotografia da autora Marianne Salomão



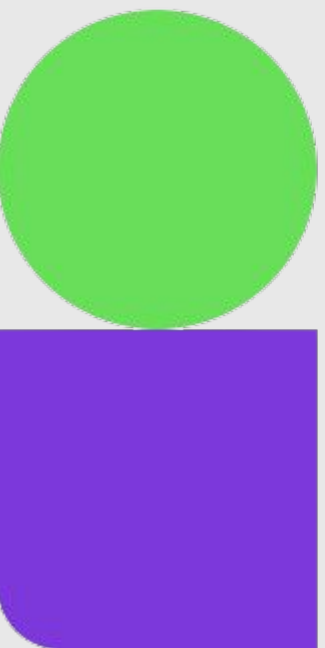
Índice

- + [Consumo de APIs externas](#)
 - + [O que é Node-Fetch?](#)
 - + [Axios](#)
- 

Introdução

"API (**Application Programming Interface - Interface entre Aplicativo e programação**) é um conjunto de instruções e padrões de programação para acesso a um aplicativo de software.

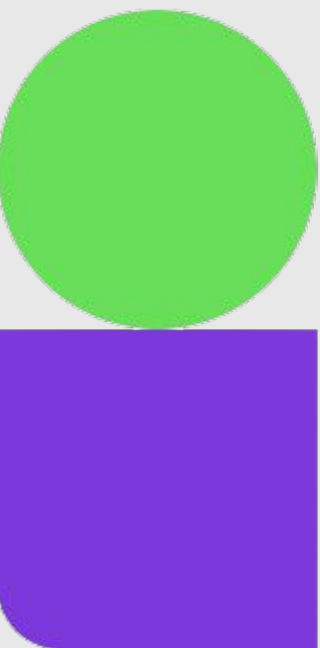
Uma empresa de software lança sua API para o público de modo que outros criadores de software possam desenvolver produtos acionados por esse serviço."



Introdução

A API tem o papel de conectar dois sistemas por meio de uma linguagem de programação comum.

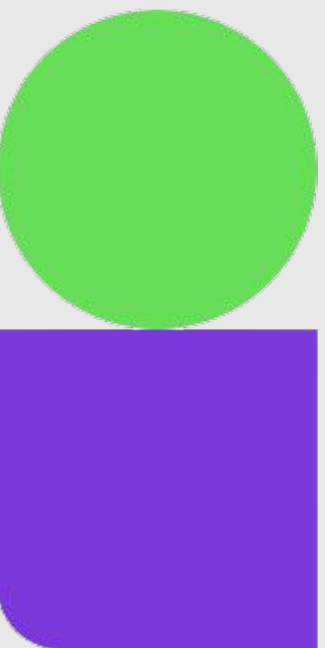
Por exemplo, fazendo com que aplicações, base de dados e serviços consigam se comunicar, evitando, assim, programações complexas e cansativas.



Introdução

O desenvolvimento **back-end cuida das engrenagens de uma aplicação web**, criando códigos para que as **funções** do site sejam executadas.

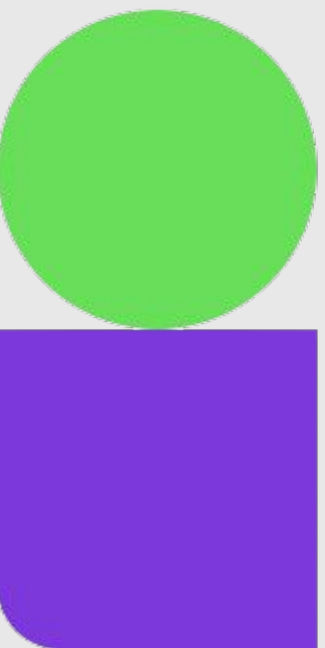
É um trabalho de bastidores.



Introdução

Simple processos como buscas ou ações mais complexas, como compras, dependem do **processamento** de dados no **servidor**, que busca e seleciona as informações. Tudo isso acontece no **back-end** e é responsabilidade do desenvolvedor fazer com que essas informações sejam encontradas.

Os usuários não têm acesso direto a esses dados, que são requisitados em linguagens de programação como PHP, Python e Ruby. **A maneira como essas informações são exibidas é responsabilidade do front-end.**



O que é um desenvolvedor back-end?

O desenvolvedor back-end é responsável por manter o **funcionamento dos websites**. Mesmo que os usuários não consigam visualizar o que é feito a olho nu, esse profissional coordena todas as tarefas relacionadas com códigos e linguagens de programação. Entre as **principais tarefas** do desenvolvedor back-end, temos:

- Fazer o domínio para sistemas operacionais de servidores;
- Analisar informações, relatórios de falhas, dados e estatística dos sites;
- Atuar com linguagens específicas, como Python, PHP, JavaScript e Ruby;
- Aplicar técnicas de segurança nos sites.
- Criar banco de dados e integrá-los com outras aplicações.

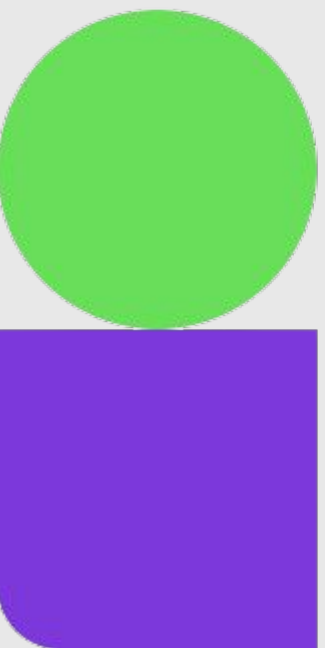
01. Consumindo uma API

Vamos demonstrar como consumir uma API externa via URL com NodeJS e retornar como JSON, uma API que retorna **Pokémons** 😁.

Crie uma pasta para esse projeto e o abra em uma IDE, por exemplo o VSCode.

Abra o terminal e digite o comando:

```
npm init
```



01. Consumo APIs Externas

Isso vai fazer com que ele crie seu arquivo de configuração **package.json** onde nele contem as informações para iniciar seu projeto.

Agora vamos criar nosso arquivo chamado **api.js** (nele terá nossas rotas para request dos dados da URL). Instale a dependência Request para que comuniquemos os dados:

```
npm install request — save
```



01. Consumo APIs Externas

Escreva o seguinte código no arquivo api.js:

```
const request = require('request')
```

```
const hostname = 'http://pokeapi.co/api/v2'
```

```
const path = '/pokemon/1/'
```

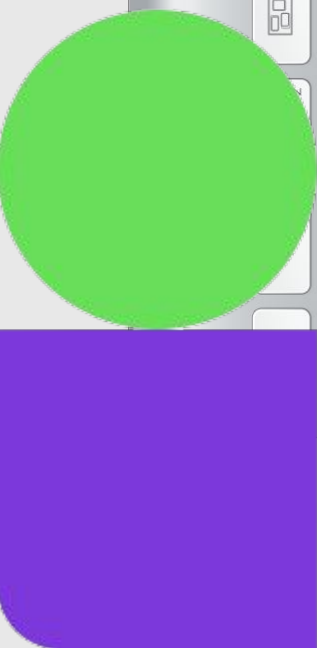
```
request(`${hostname}${path}`, (err, res, body) => {  
  console.log(body);  
});
```

Agora é só testar, no terminal execute o comando **node api**. Ele irá renderizar no terminal o json trazendo os pokémons.





02. O que é Node-Fetch?



02. Node-Fetch

Node - Fetch - É uma forma de fazer requisições HTTP.

Iremos testar o Fetch, uma alternativa que está sendo adotada e implementada nos navegadores para substituir a forma como se fazia requisições http no browser.



02. Node - Fetch

Iremos iniciar um novo projeto, em seguida adicione a dependência do node - fetch no projeto:

yarn add node-fetch

Agora precisa importar a dependência:

```
const express = require("express");
const fetch = require("node-fetch");
const app = express();
app.get('/', async function(req, res){
  const response = await fetch('https://dog.ceo/api/breeds/list/all')
  const app = await response.json()
  console.log(app)
})
app.listen(3000);
module.export = app
```



02. Node - Fetch

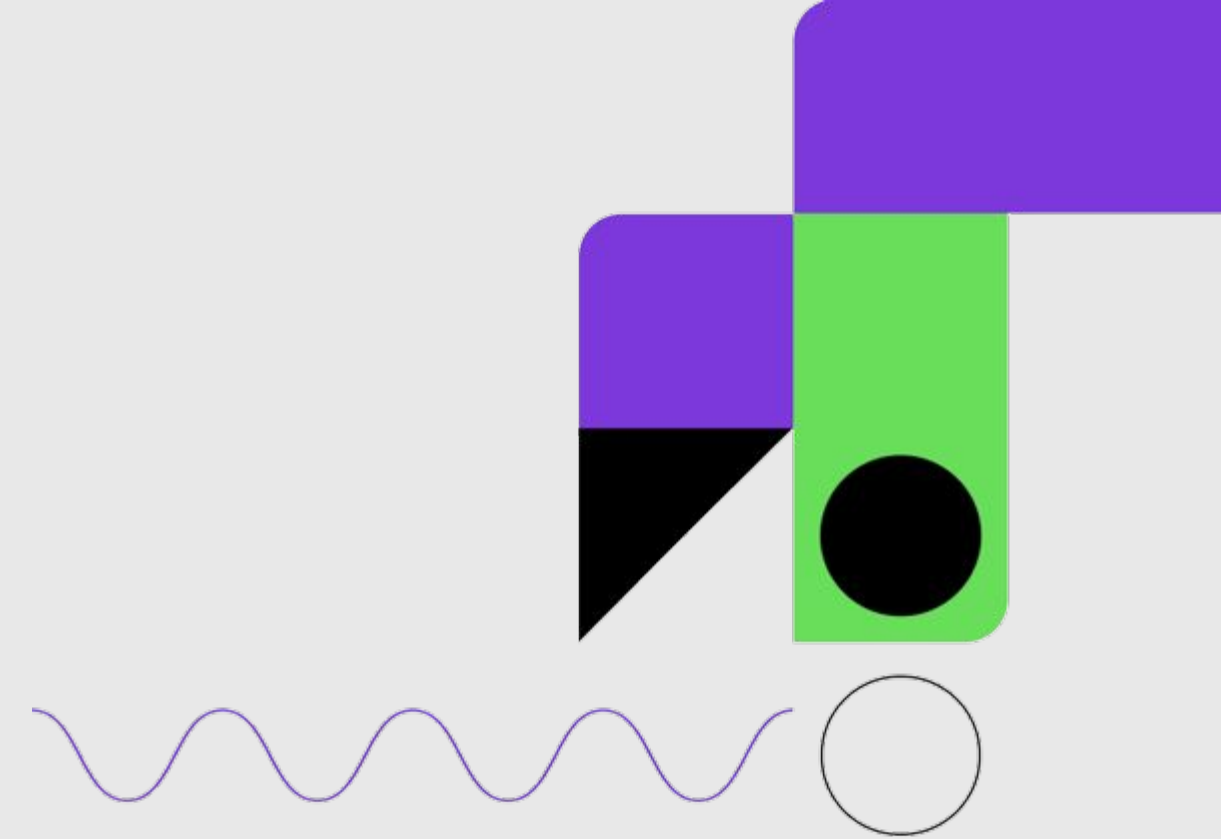
Aqui estou dando um **fetch** para pegar essa url e em seguida ele retorna o **res** (response). Ele retorna os resultados e também um json, que é uma promise.

03. Axios

Axios é outro cliente HTTP baseado em Promises que funciona no navegador e também no Node.js.

Para instalar o Axios pelo npm, digite o seguinte comando no seu terminal:

```
npm install axios
```



03. Axios

O código a seguir vai realizar a mesma tarefa de exibir a URL e a explicação da foto astronômica do dia:

```
const axios = require('axios');
```

```
(async () => {
```

```
  try {
```

```
    const response = await
```

```
    axios.get('https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY')
```

```
    console.log(response.data.url);
```

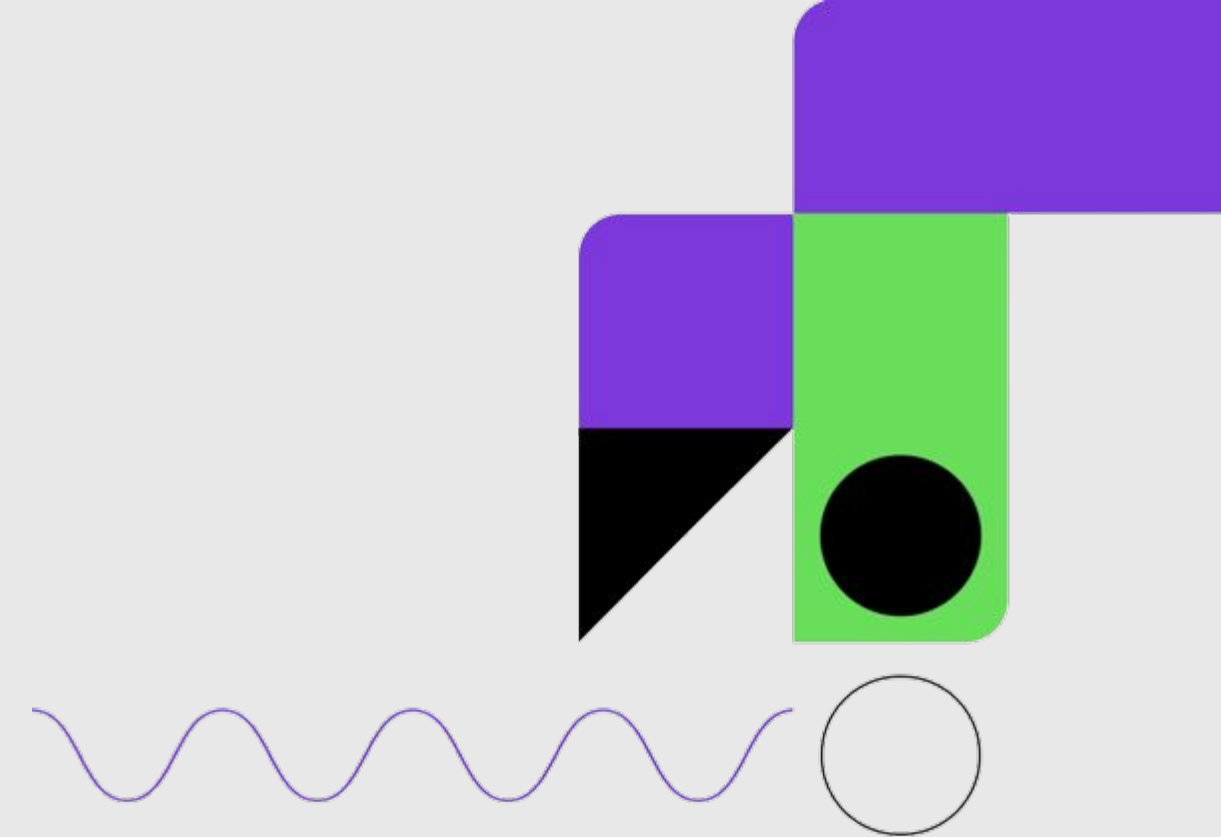
```
    console.log(response.data.explanation);
```

```
  } catch (error) {
```

```
    console.log(error.response.body);
```

```
  }
```

```
})();
```



03. Axios

Você também pode fazer múltiplas requisições simultâneas com `axios.all`, se você quiser fazer algo como carregar a foto astronômica de dois dias diferentes ao mesmo tempo:

```
const axios = require('axios');

(async () => {
  try {
    const [response1, response2] = await axios.all([
      axios.get('https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY&date=2020-03-18'),
      axios.get('https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY&date=2020-03-17')
    ]);
    console.log(response1.data.url);
    console.log(response1.data.explanation);

    console.log(response2.data.url);
    console.log(response2.data.explanation);
  } catch (error) {
    console.log(error.response.body);
  }
})();
```




Fechamento

Vimos diversos assuntos até aqui, ainda há muito a ser abordado. Esse material pode servir como consulta e guia de estudos.

Até a próxima pessoal!