

Documentação do desafio – Triágil

Desafios

- Desenvolver uma API
- Montar times de Pokémons utilizando a pokeapi.co
- API deve permitir o usuário criar um novo time passando alguns parâmetros (lista de Pokémons e um nome de usuário)
- API deve ler uma lista (usuário e Pokémon), buscar pelos pokémons na pokeAPI e caso não encontre nenhum erro salvar o time;
 - Time salvo: retorna ao usuário uma mensagem de validação e uma id única;
 - Guardar além do nome do Pokémon, ID do Pokémon (de acordo com a pokédex), sua altura e seu peso.
 - Time com erro: mensagem que o time não foi inserido corretamente;
- A API deve conter duas rotas para leitura dos dados:
 - busque todos os times registrados;
 - busque um time pela ID única
- Gerar um Dockerfile
- Gerar um compose
- Documentação

Rotas

- GET /api/teams - Deverá listar todos os times registrados
- GET /api/teams/{user} - Busca um time registrado por usuário
- POST /api/teams - Rota para criação de um time, que recebe um JSON

Exemplos

Output /api/teams/{user}

```
{
  "owner": "sleao",
  "pokemons": [
    {
      "id": 9,
      "name": "blastoise",
      "weight": 855,
      "height": 16
    },
    {
      "id": 25,
      "name": "pikachu",
      "weight": 60,
      "height": 4
    }
  ]
}
```



Output /api/teams

```
{
  "1": {
    "owner": "sleao",
    "pokemons": [
      {
        "id": 9,
        "name": "blastoise",
        "weight": 855,
        "height": 16
      },
      {
        "id": 25,
        "name": "pikachu",
        "weight": 60,
        "height": 4
      }
    ]
  },
  "2": {
    "owner": "sleao",
    "pokemons": [
      {
        "id": 9,
        "name": "blastoise",
        "weight": 855,
        "height": 16
      },
      {
        "id": 25,
        "name": "pikachu",
        "weight": 60,
        "height": 4
      },
      {
        "id": 3,
        "name": "venusaur",
        "weight": 1000,
        "height": 20
      },
      {
        "id": 6,
        "name": "charizard",
        "weight": 905,
        "height": 17
      },
      {
        "id": 131,
        "name": "lapras",
        "weight": 2200,
        "height": 25
      },
      {
        "id": 54,
        "name": "psyduck",
        "weight": 196,
        "height": 8
      }
    ]
  }
}
```



Input

```
{
  "user": "sleao",
  "team": [
    "blastoise",
    "pikachu",
    "charizard",
    "venusaur",
    "lapras",
    "dragonite"
  ]
}
```

Abordagem

Como o objetivo era testar os meus conhecimentos de programação (e de Google) o desafio foi analisado e dividido em duas partes, API e Dockerfile e dividido em sub-tarefas;

Linguagem de Programação

Para o desenvolvimento da API foi escolhida a linguagem de Programação Python.

API

Link da API: <https://pokeapi.co/>

Documentação da API: <https://pokeapi.co/docs/v2>

Fase de Testes

Exemplo de requisições feitas localmente,

- Para verificar os Teams existentes (GET)

```
http://localhost:5000/api/teams
```

- Para criar um Team (POST)

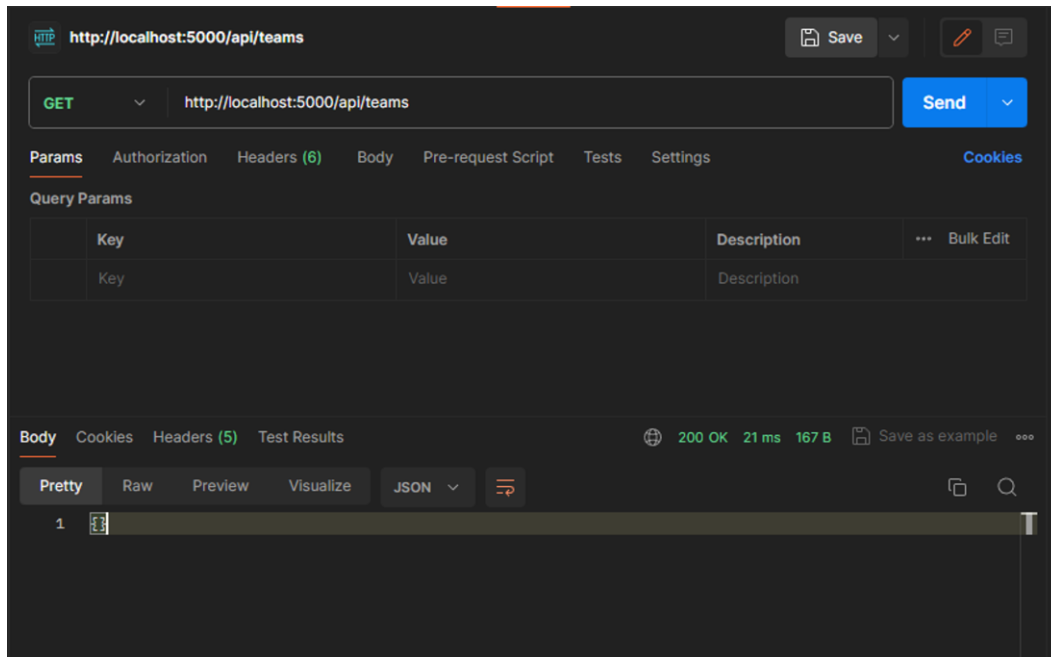
```
http://localhost:5000/api/teams
```



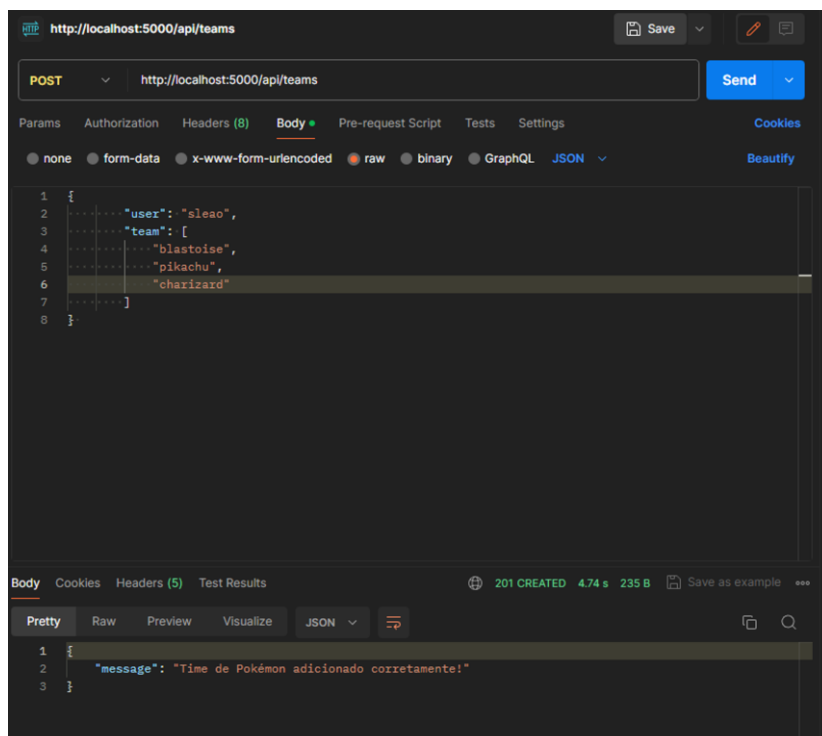
- Para verificar pelo Owner os Teams existentes (GET)

```
http://localhost:5000/api/teams/{owner}
```

- Teste da rota Teams



- Teste adicionando Teams





- Teste visualizando o Team que foi adicionado

http://localhost:5000/api/teams

GET http://localhost:5000/api/teams

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| Key | Value | Description | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (5) Test Results 200 OK 8 ms 546 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "1": {
3     "owner": "sleao",
4     "pokemons": [
5       {
6         "height": 16,
7         "id": 9,
8         "name": "blastoise",
9         "weight": 855
10      },
11      {
12        "height": 4,
13        "id": 25,
14        "name": "pikachu",
15        "weight": 60
16      },
17      {
18        "height": 17,
19        "id": 6,
20        "name": "charizard",
21        "weight": 905
22      }
23    ]
24  }
25 }
```

- Teste visualizando os Teams pelo nome do owner

http://localhost:5000/api/teams/sleao

GET http://localhost:5000/api/teams/sleao

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

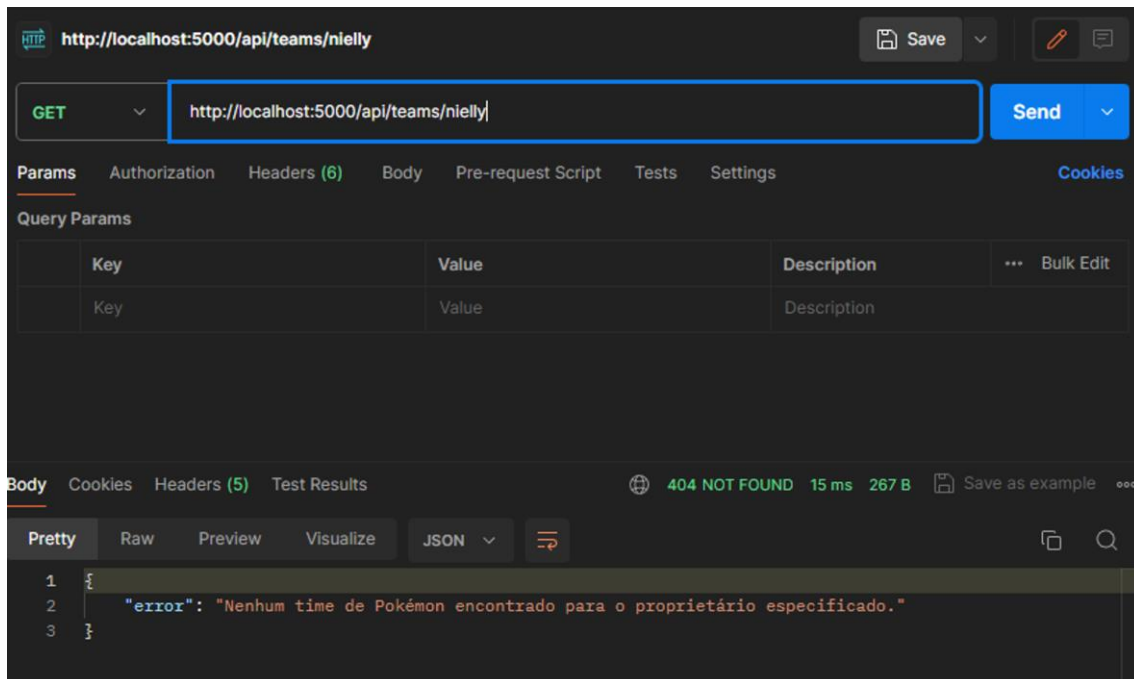
| Key | Value | Description | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (5) Test Results 200 OK 10 ms 541 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "owner": "sleao",
4     "pokemons": [
5       {
6         "height": 16,
7         "id": 9,
8         "name": "blastoise",
9         "weight": 855
10      },
11      {
12        "height": 4,
13        "id": 25,
14        "name": "pikachu",
15        "weight": 60
16      },
17      {
18        "height": 17,
19        "id": 6,
20        "name": "charizard",
21        "weight": 905
22      }
23    ]
24  }
25 ]
```

- Teste consultando um owner que não existe



HTTP **GET** `http://localhost:5000/api/teams/nielly` **Send**

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

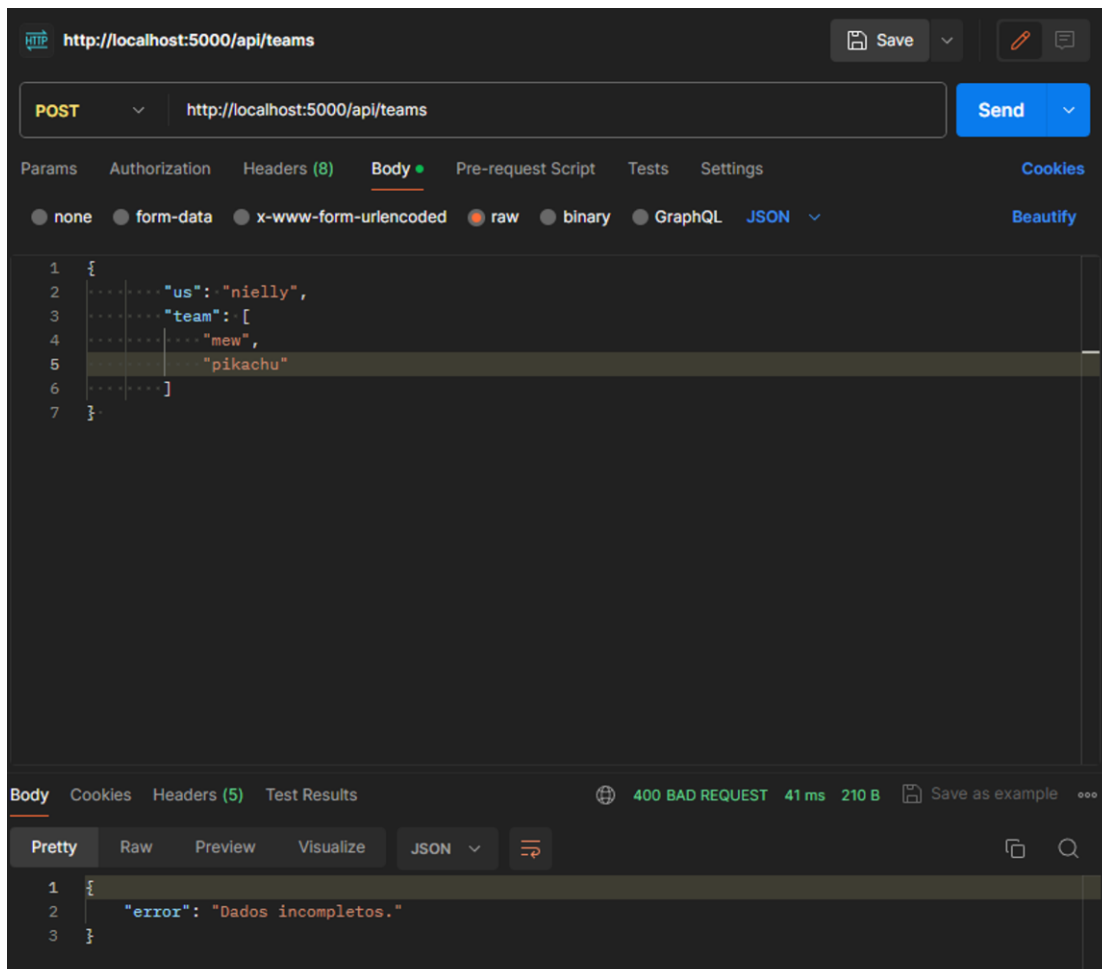
| Key | Value | Description | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (5) Test Results **404 NOT FOUND** 15 ms 267 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "Nenhum time de Pokémon encontrado para o proprietário especificado."
3 }
```

- Teste tentando adicionar um team - com erro



HTTP **POST** `http://localhost:5000/api/teams` **Send**

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

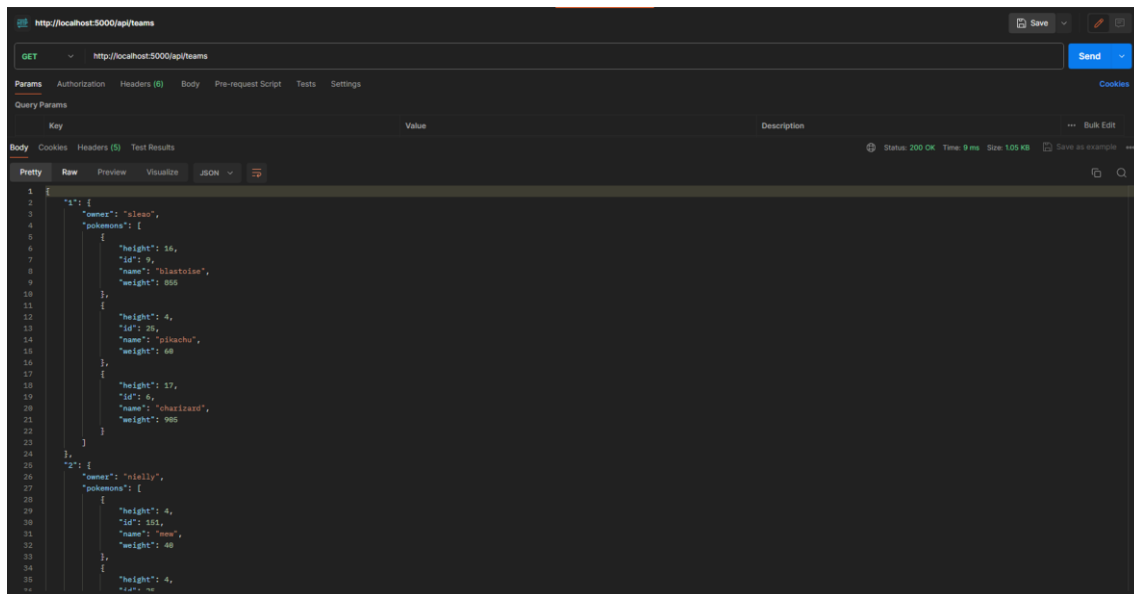
```
1 {
2   "us": "nielly",
3   "team": [
4     "mew",
5     "pikachu"
6   ]
7 }
```

Body Cookies Headers (5) Test Results **400 BAD REQUEST** 41 ms 210 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "Dados incompletos."
3 }
```

- Teste - Exibindo todos os Teams criados

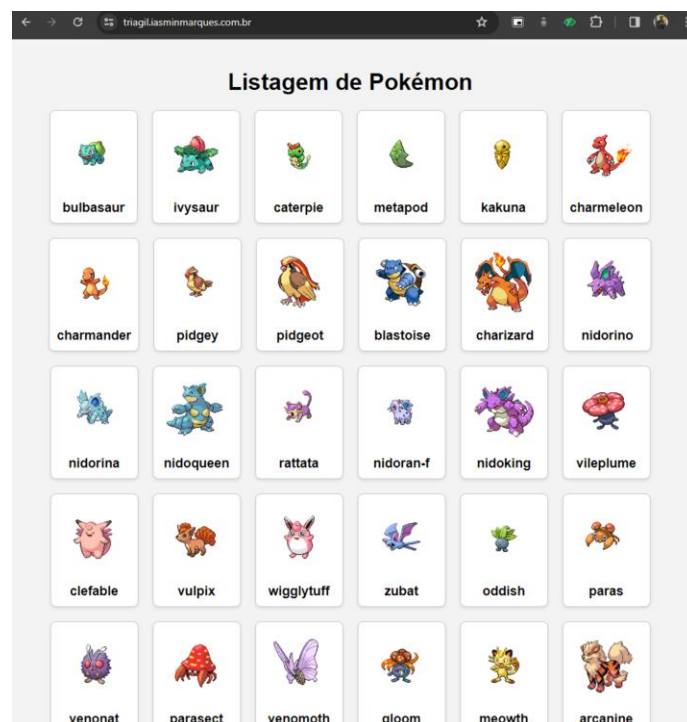


Front-End

Para verificar todos os pokemons existentes e verificar se o pokemon existe criei duas páginas web, uma que visualiza todos pokemons e outro para fazer uma consulta através do nome do pokémon. Utilizei para desenvolver as duas páginas as tecnologias: HTML, CSS, Javascript. E fiz o upload para o subdomínio:

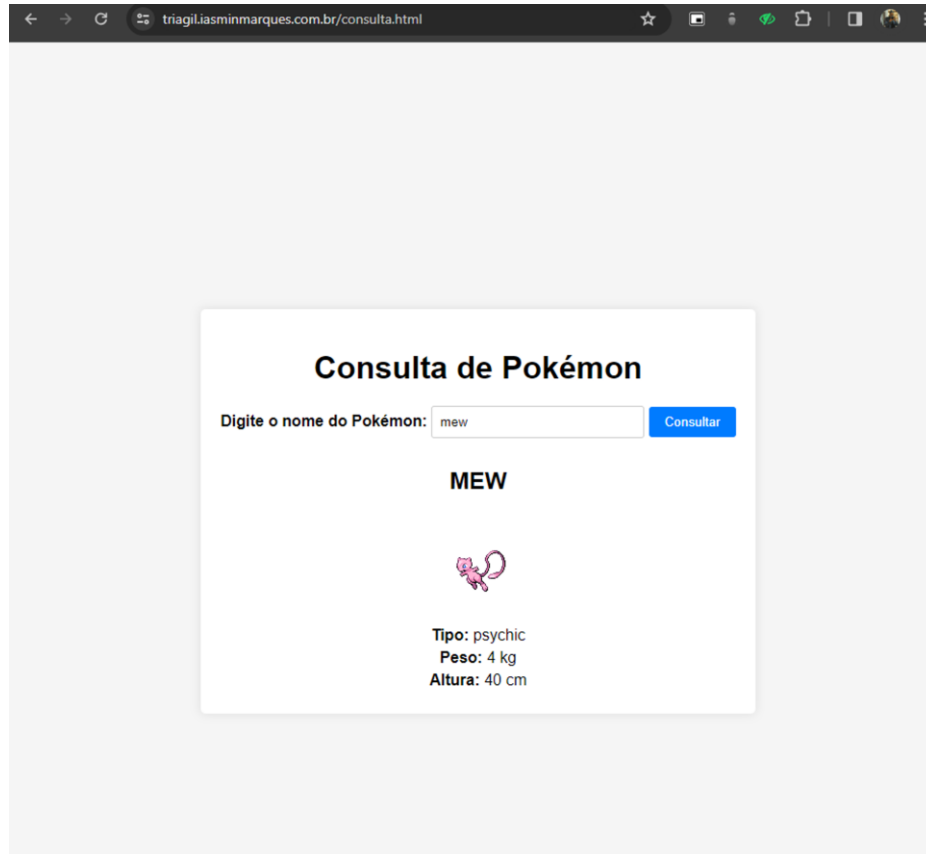
- Mostra todos os pokemons

Link: <http://triagil.iasminmarques.com.br/>



- Consulta o pokemon

<https://triagil.iasminmarques.com.br/consulta.html>



Docker/ Container

Dockerfiles and docker-compose made for Database I classes. Depois de instalar o docker desktop, entre na pasta que você baixou do github e modifique o **docker-compose.yml** com seu nome e email do github.

- Windows

```
docker-compose up --build
```

- Linux

```
docker compose up --build
```




Comandos

Depois disso, abram outro terminal (comando pra usar o terminal)

```
docker exec -it ubuntu bash
```

Para parar o container

```
docker stop ubuntu
```

Para rodar ele depois de parar

```
docker start -i ubuntu
```

Verificar os containers ativos

```
docker ps -a
```

Funções para se adicionar futuramente

- Adicionar um id específico para cada Team criado e exibir na tela para o usuário na hora da criação dos Teams
- Implementar o fron-end das outras páginas para consumir a API dos Teams que desenvolvi.