

Documentação do Desafio – AEVO por Iasmin Marques

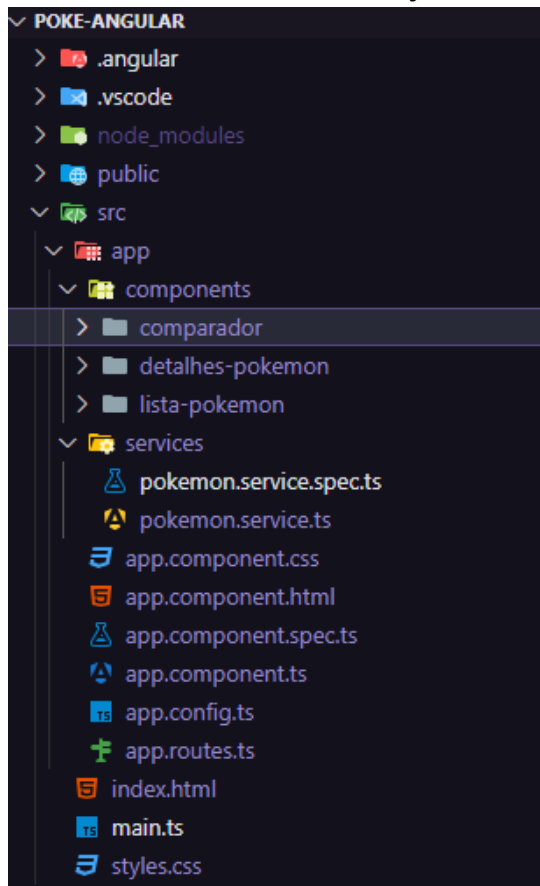
Desafios

- ✓ Explorar a Documentação da API (<https://pokeapi.co/docs/v2>), para detalhes de utilização;
- ✓ Elaborar uma página para consultar e exibir as informações (lista de pokemons) da requisição da API na página;
- ✓ Adicionar um input na página para permitir buscas;
- ✓ Selecionar um dos pokemons listados para ver informações detalhadas (Informações detalhadas vem de outra requisição na API);
- ✓ Realizar a soma de todos status (Atributo base_stats que fica dentro de stats) do pokemon selecionado e exibir esse valor;
- ✓ Selecionar dois pokemons e exibir qual dos dois tem a soma de status (Atributo base_stats que fica dentro de stats) maior;

Tecnologias Utilizadas

- **Angular:** Framework JavaScript utilizado para a construção da aplicação.
- **TypeScript:** Linguagem utilizada para o desenvolvimento da lógica da aplicação.
- **PokeAPI:** API pública utilizada para buscar informações sobre os Pokémons.
- **CSS:** Estilização da interface.

Estrutura de Diretórios/ Projeto





Componentes

- **ListaPokemonComponent**
 - a. Busca dinâmica por nome
 - b. Exibe nome e imagem dos Pokémons
 - c. Navega para a tela de detalhes
- **DetalhesPokemonComponent**
 - a. Exibe altura, peso, tipos e todos os stats
 - b. Mostra a soma total de stats
 - c. Botão para voltar à home (nav-bar e “X”)
- **ComparadorComponent**
 - a. Busca e seleciona dois Pokémons
 - b. Exibe seus status totais e imagem
 - c. Indica o vencedor com destaque visual
 - d. Exibe mensagem de erro caso não selecione os dois

Acessibilidade e UX

- Inputs com feedback
- Mensagens de erro claras
- Navegação fluida e responsiva

Estilização

- CSS modular para cada componente
- Design responsivo com *flex* e *grid*
- Variáveis CSS para reutilização de estilos
- Feedback visual com *:hover*, *:focus*, mensagens de erro e destaque no vencedor

Como Executar o Projeto Localmente

1. Instalar as dependências: `npm install`
2. Rodar o projeto: `ng serve`
3. Acesse: <http://localhost:4200>

Exemplo:

```
iasmi@ImpDesktop MINGW64 ~/OneDrive/Documents/aevo/Angular/aevo_test/iasmin-desafio_aevo (main)
$ ls
'Documentação - Desafio AEVO.pdf'  poke-angular/

iasmi@ImpDesktop MINGW64 ~/OneDrive/Documents/aevo/Angular/aevo_test/iasmin-desafio_aevo (main)
$ cd poke-angular/

iasmi@ImpDesktop MINGW64 ~/OneDrive/Documents/aevo/Angular/aevo_test/iasmin-desafio_aevo/poke-angular (master)
$ ls
README.md      package-lock.json  src/              tsconfig.spec.json
angular.json    package.json       tsconfig.app.json
node_modules/  public/            tsconfig.json

Component HMR has been enabled.
If you encounter application reload issues, you can manually reload the page to bypass HMR and/or disable this feature with the '--no-hmr' command line option.
Please consider reporting any issues you encounter here: https://github.com/angular/angular-cli/issues

14:26:10 [vite] (ssr) Re-optimizing dependencies because vite config has changed (x2)
→ Local: http://localhost:4200/
→ press h + enter to show help
```

Teste - Domínio

Foi criado um subdomínio dentro do site de hospedagens Hostgator para testes, segue o Link: <http://aevo.iasminmarques.com.br>



IASMIN MARQUES PEREIRA

Detalhes dos Componentes

1. ComparadorComponent

O componente *ComparadorComponent* permite que os usuários comparem dois Pokémons lado a lado, visualizando suas principais estatísticas, tipos e outros atributos relevantes. Esse componente é parte central da funcionalidade de comparação no projeto e promove uma experiência interativa e visualmente clara para o usuário.

Arquivos:

- [comparador.component.ts](#): Lógica e controle do componente.
- [comparador.component.html](#): Estrutura visual (template).
- [comparador.component.css](#): Estilização do componente.

Lógica:

- *PokemonService*: serviço responsável por buscar dados da API.
- *listaPokemons*: armazena todos os Pokémons obtidos da API (utilizada para autocompletar os campos).
- *nomePokemon1* e *nomePokemon2*: armazenam os nomes digitados pelo usuário para selecionar e comparar.
- *filtro1* e *filtro2*: armazenam os resultados filtrados com base nos nomes digitados (includes), usados para sugestões automáticas.
- *detalhesPokemon1* e *detalhesPokemon2*: armazenam os dados completos (stats, nome, etc.) dos dois Pokémons selecionados.
- *mensagemErro*: exibe mensagens de erro para o usuário, como validação de entrada.
- *vencedor*: armazena o número do vencedor (1 ou 2), ou null em caso de empate.

HTML:

- Campos de input para digitar o nome dos Pokémons.
- Botão para comparar dois pokemons (caso selecionados corretamente).
- Exibição dos dados dos Pokémons lado a lado:
 - Imagem (sprite).
 - Nome.
 - Tipos.
 - Estatísticas base (HP, Ataque, Defesa, etc.).
- Botões do nav-bar para navegar entre a tela de comparação e home/ listagem;

2. ListaPokemonComponent

O componente *ListaPokemonComponent* exibe uma lista paginada de Pokémons com um campo de busca para filtragem por nome. O usuário pode visualizar os dados básicos e clicar para ver os detalhes de um Pokémon específico.

Arquivos:

- [lista-pokemon.component.ts](#): Lógica e controle do componente.
- [lista-pokemon.component.html](#): Estrutura visual (template).



- [lista-pokemon.component.css](#): Estilização do componente.

Lógica Utilizada:

- *pokemons*: armazena a lista completa de Pokémons obtidos da API.
- *pokemonsFiltrados*: armazena a lista filtrada com base na busca do usuário (exibida na tela).
- *busca*: armazena o texto digitado no campo de busca.
- *filtrarPokemons()*: filtra a lista de Pokémons pelo nome (case insensitive).
- *getIdFromUrl(url: string)*: extrai o ID de um Pokémon a partir da URL.
- *getImagePokemon(pokemon: any)*: retorna a URL da imagem do Pokémon com base no ID extraído.

HTML:

- Campo de input com autocomplete para filtrar Pokémons.
- Lista de Pokémons exibida com nome e imagem.
- Botões do nav-bar para navegar entre a tela de home/ listagem e comparação;

3. DetalhesComponent

O componente *DetalhesPokemonComponent* é responsável por exibir informações detalhadas de um Pokémon selecionado, acessado a partir da lista de Pokémons;

Arquivos:

- [detalhes.component.ts](#): Lógica e controle do componente.
- [detalhes.component.html](#): Estrutura visual (template).
- [detalhes.component.css](#): Estilização do componente.

Lógica Utilizada:

- *pokemon*: armazena os dados detalhados do Pokémon retornados pela API.
- *somaStats*: armazena a soma total dos stats base do Pokémon.
- *route.snapshot.paramMap.get('id')*: extrai o nome ou ID do Pokémon da rota (parâmetro da URL).
- *pokeService.getDetalhesPokemon(nomeOuId)*: método do serviço que busca os detalhes do Pokémon com base no nome ou ID.
- *reduce()*: calcula a soma dos stats base ao iterar sobre o array stats.

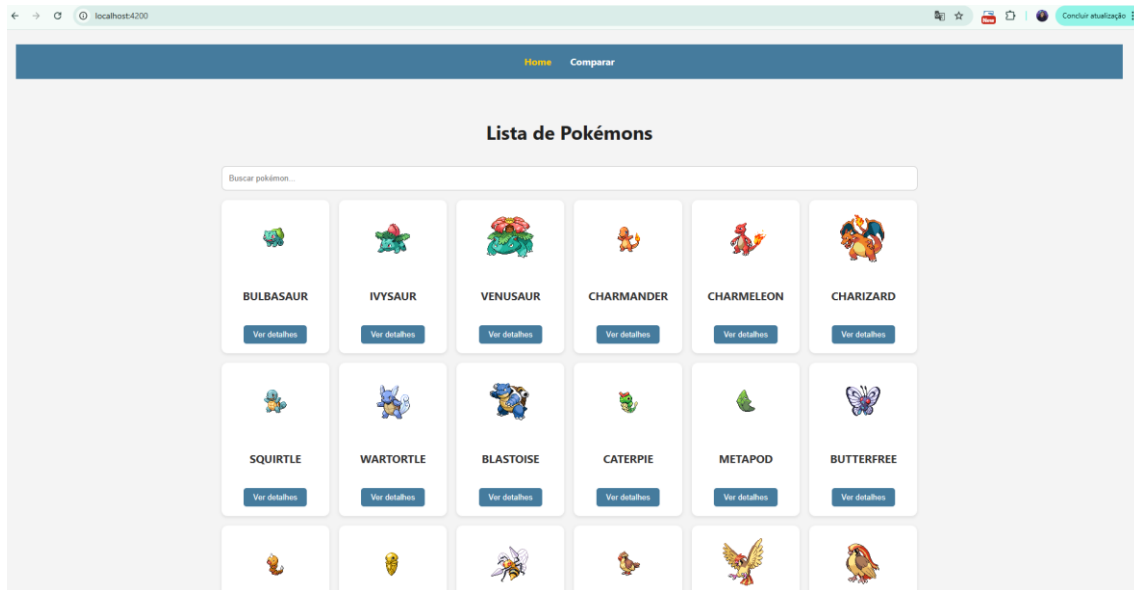
HTML:

- Exibição da imagem do Pokémon, nome, número, tipos, altura, peso.
- Lista das habilidades e estatísticas base (como HP, ataque, defesa, etc.).

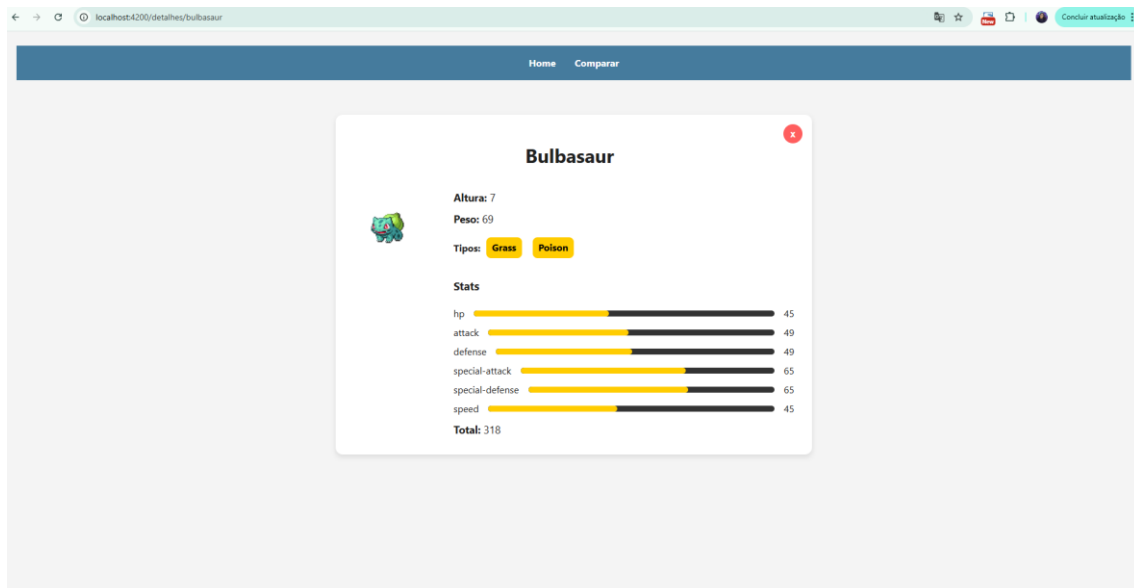


Telas

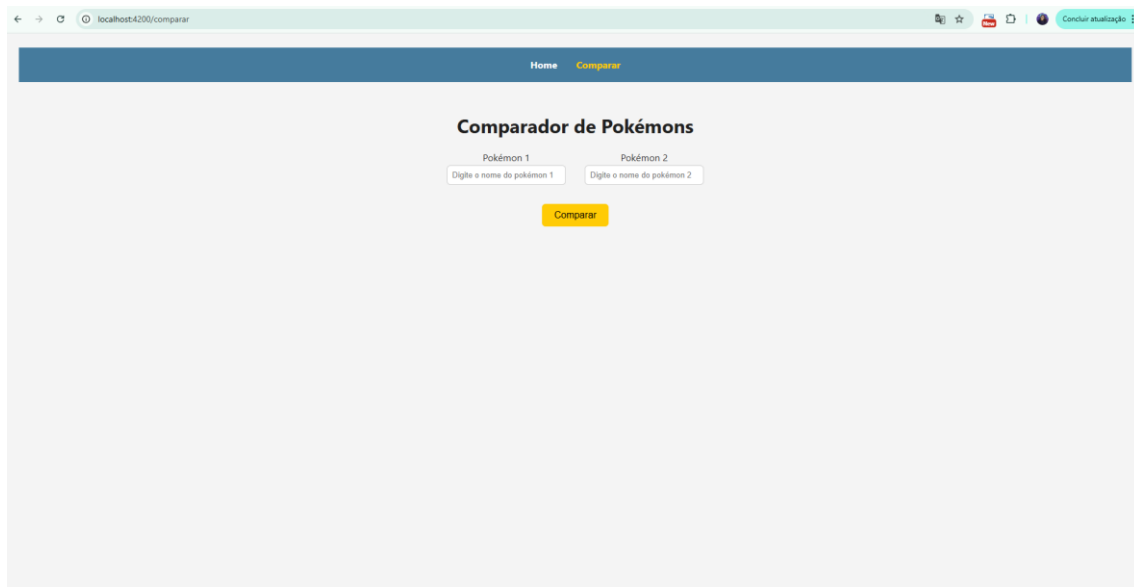
- Tela Inicial/ Listagem



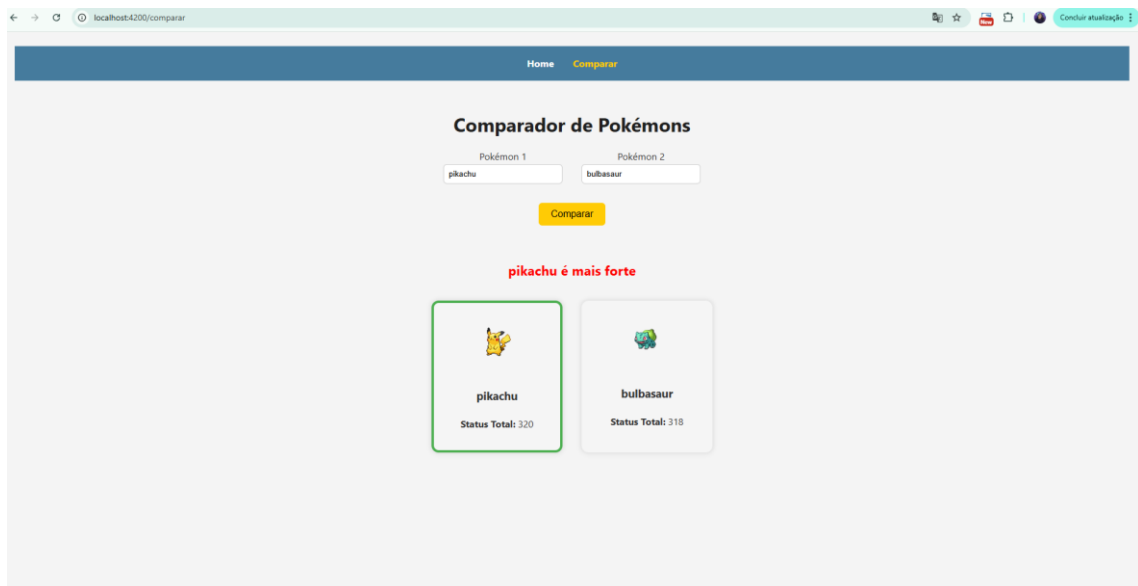
- Detalhes do Pokémon



- Tela de Comparação



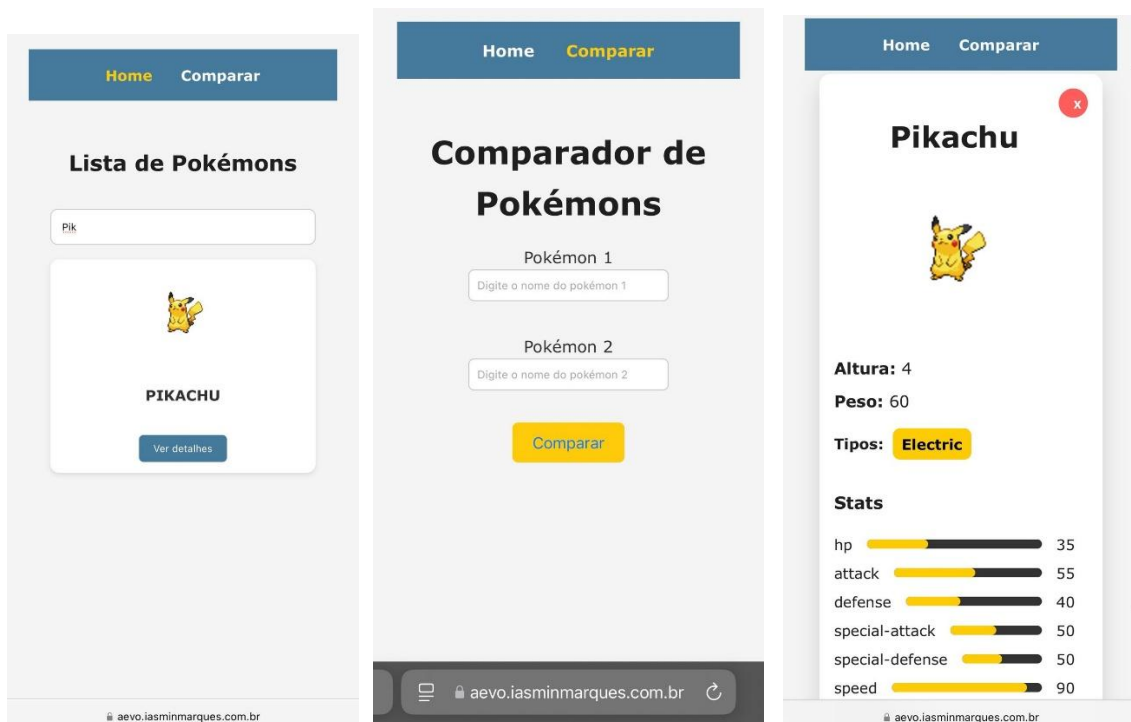
- Exemplo de Comparação



- Exemplo de erro de comparação



- Telas do celular (Acessibilidade de UX)



API

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

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

- **Requisições a API**

As chamadas à API são feitas via: [pokemon.service.ts](#)

```
export class PokemonService {
  private apiUrl = 'https://pokeapi.co/api/v2';

  constructor(private http: HttpClient) {}

  getListaPokemon(limit: number = 151): Observable<any> {
    return this.http.get(`${this.apiUrl}/pokemon?limit=${limit}`);
  }

  getDetalhesPokemon(urlOuId: string | number): Observable<any> {
    return this.http.get(`${this.apiUrl}/pokemon/${urlOuId}`);
  }

  getMultiplosPokemons(ids: (number | string)[]): Observable<any[]> {
    return forkJoin(ids.map((id) => this.getDetalhesPokemon(id)));
  }
}
```

Exemplos de chamadas – fase de testes

- Uma chamada da API para trazer 151 pokemons

Rota: <https://pokeapi.co/api/v2/pokemon/?limit=151>

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** <https://pokeapi.co/api/v2/pokemon/?limit=151>
- Status:** 200 OK
- Time:** 303 ms
- Size:** 10.9 KB
- Response Body (JSON):**

```
{
  "count": 1392,
  "next": "https://pokeapi.co/api/v2/pokemon/?offset=151&limit=151",
  "previous": null,
  "results": [
    {
      "name": "bulbasaur",
      "url": "https://pokeapi.co/api/v2/pokemon/1/"
    },
    {
      "name": "ivysaur",
      "url": "https://pokeapi.co/api/v2/pokemon/2/"
    },
    {
      "name": "venusaur",
      "url": "https://pokeapi.co/api/v2/pokemon/3/"
    },
    {
      "name": "charmander",
      "url": "https://pokeapi.co/api/v2/pokemon/4/"
    },
    {
      "name": "charmeleon",

```



- Uma chamada da API para trazer mais informações do pokemon 4 - Charmander
Rota: <https://pokeapi.co/api/v2/pokemon/4/>

API documentation / User / Get authenticated user

GET <https://pokeapi.co/api/v2/pokemon/4/> Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description

Body Cookies Headers (26) Test Results Status: 200 OK Time: 161 ms Size: 264.46 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "abilities": [
3     {
4       "ability": {
5         "name": "blaze",
6         "url": "https://pokeapi.co/api/v2/ability/66/"
7       },
8       "is_hidden": false,
9       "slot": 1
10    },
11    {
12      "ability": {
13        "name": "solar-power",
14        "url": "https://pokeapi.co/api/v2/ability/94/"
15      },
16      "is_hidden": true,
17      "slot": 3
18    }
19  ],
20  "base_experience": 62,
21  "cries": {
22    "latest": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/4.ogg",
23    "legacy": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/legacy/4.ogg"
  }
}

```

Postbot Runner Start Proxy Cookies Trash

- Listagem de todos os Pokémon (limit=1000 define o número máximo de resultados retornados)
Rota: <https://pokeapi.co/api/v2/pokemon?limit=1000>

API documentation / User / Get authenticated user

GET <https://pokeapi.co/api/v2/pokemon?limit=1000> Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

☒ limit 1000

Key	Value	Description

Body Cookies Headers (26) Test Results Status: 200 OK Time: 172 ms Size: 66.42 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "count": 1302,
3   "next": "https://pokeapi.co/api/v2/pokemon?offset=1000&limit=302",
4   "previous": null,
5   "results": [
6     {
7       "name": "bulbasaur",
8       "url": "https://pokeapi.co/api/v2/pokemon/1/"
9     },
10    {
11      "name": "ivysaur",
12      "url": "https://pokeapi.co/api/v2/pokemon/2/"
13    },
14    {
15      "name": "venusaur",
16      "url": "https://pokeapi.co/api/v2/pokemon/3/"
17    },
18    {
19      "name": "charmander",
20      "url": "https://pokeapi.co/api/v2/pokemon/4/"
21    },
22    {
23      "name": "charmeleon",

```

Postbot Runner Start Proxy Cookies Trash

- Detalhes de um Pokémon específico (Exemplo: pikachu e ditto)
Rota: <https://pokeapi.co/api/v2/pokemon/{nome-do-pokemon}>



API documentation / User / Get authenticated user

GET Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Key	Value	Description
Key	Value	Description

Body Cookies Headers (26) Test Results Status: 200 OK Time: 138 ms Size: 240.77 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "abilities": [
3     {
4       "ability": {
5         "name": "static",
6         "url": "https://pokeapi.co/api/v2/ability/9/"
7       },
8       "is_hidden": false,
9       "slot": 1
10    },
11    {
12      "ability": {
13        "name": "lightning-rod",
14        "url": "https://pokeapi.co/api/v2/ability/31/"
15      },
16      "is_hidden": true,
17      "slot": 3
18    }
19  ],
20  "base_experience": 112,
21  "cries": {
22    "latest": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/25.ogg",
23    "legacy": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/legacy/25.ogg"
  }
}

```

Postbot Runner Start Proxy Cookies Trash

API documentation / User / Get authenticated user

GET Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Key	Value	Description
Key	Value	Description

Body Cookies Headers (26) Test Results Status: 200 OK Time: 49 ms Size: 24.65 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "abilities": [
3     {
4       "ability": {
5         "name": "limber",
6         "url": "https://pokeapi.co/api/v2/ability/7/"
7       },
8       "is_hidden": false,
9       "slot": 1
10    },
11    {
12      "ability": {
13        "name": "imposter",
14        "url": "https://pokeapi.co/api/v2/ability/150/"
15      },
16      "is_hidden": true,
17      "slot": 3
18    }
19  ],
20  "base_experience": 101,
21  "cries": {
22    "latest": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/132.ogg",
23    "legacy": "https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/legacy/132.ogg"
  }
}

```

Postbot Runner Start Proxy Cookies Trash

Funções para se adicionar futuramente

- Infinity scroll (adicionar paginação): atualmente a pesquisa está limitada com um numero de Pokémon
- Melhorar o designer/ padronizar: questões de UX/ UI da aplicação;
- Loading Spinner/ carregando: estiver buscando Pokémons, exibir um loading para dar uma maior fluidez;
- Em detalhes do pokemon: gráficos de radar ou barra para as estatísticas, a exibição de evoluções do Pokémon;

