AEVO

Documentação do Desafio – AEVO por Iasmin Marques

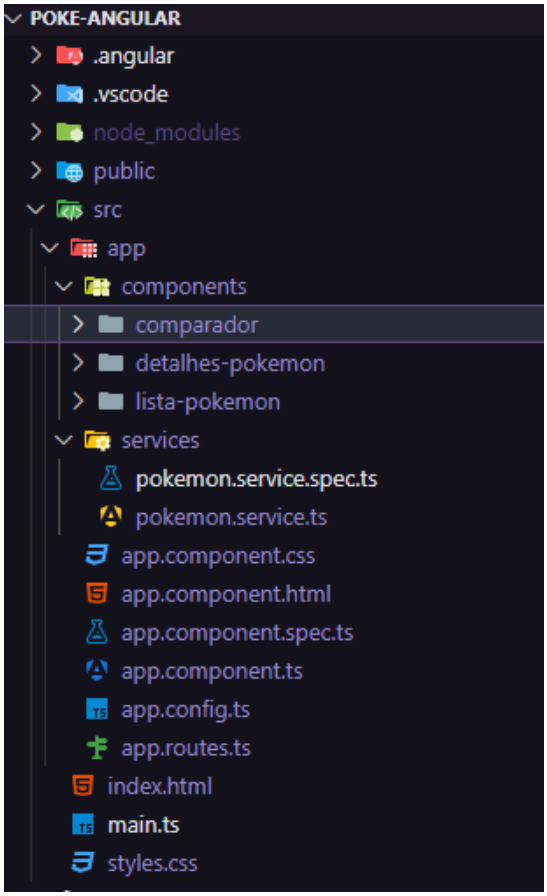
Desafios

- ✓ 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 requisicão 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

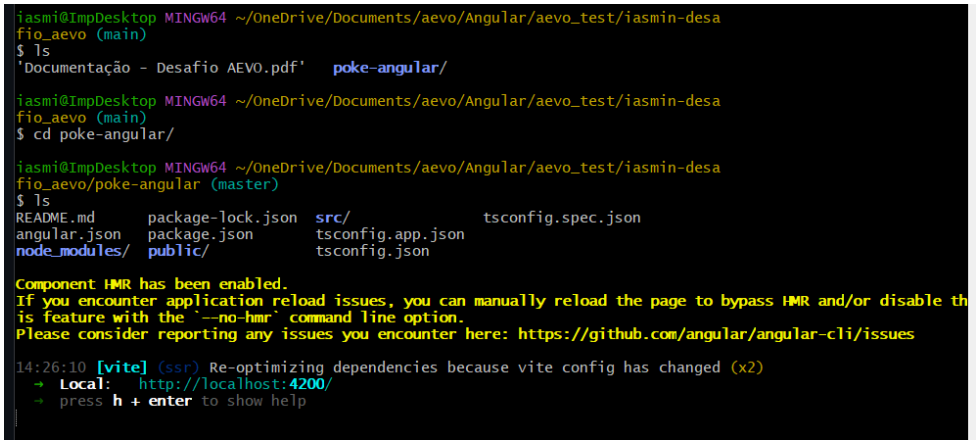
Estilização

- 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 dependencias: npm install
2. Rodar o projeto: ng serve
3. Acesse: http://localhost:4200

Exemplo:



Teste - Domínio

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

AEVO

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.
- *getImagemPokemon(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:

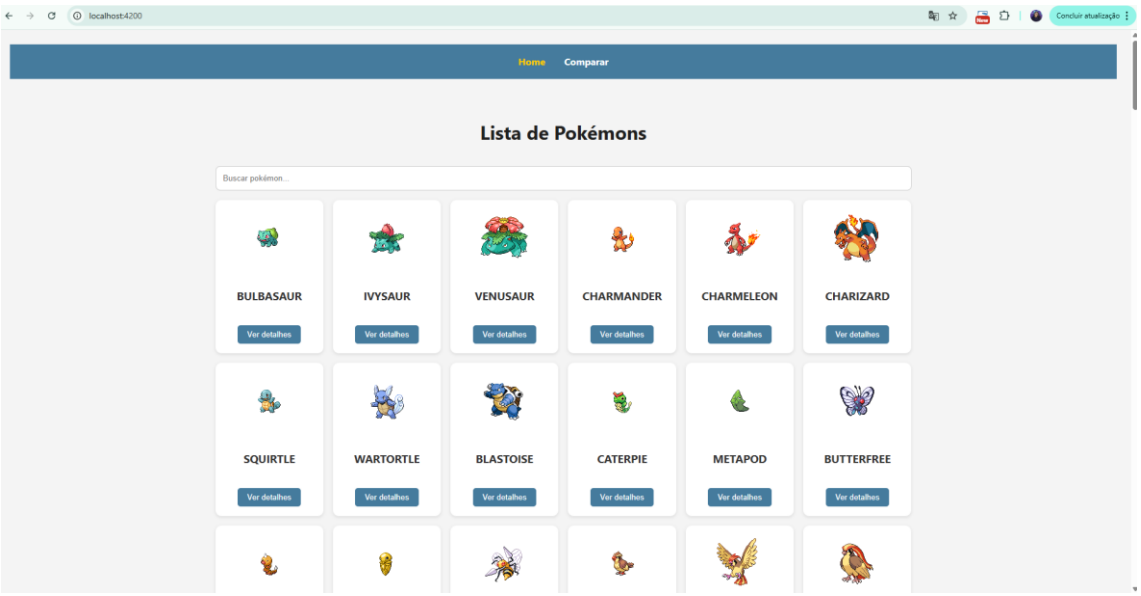
- *pokemon*: armazena os dados detalhados do Pokémon retornados pela API.
- *somaStats*: armazena a soma total dos stats base do Pokémon.
- *route.snapshot.paramMap.get('id')*: extrai o nome ou ID do Pokémon da rota (parâmetro da URL).
- *pokeService.getDetalhesPokemon(nomeOuId)*: método do serviço que 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:

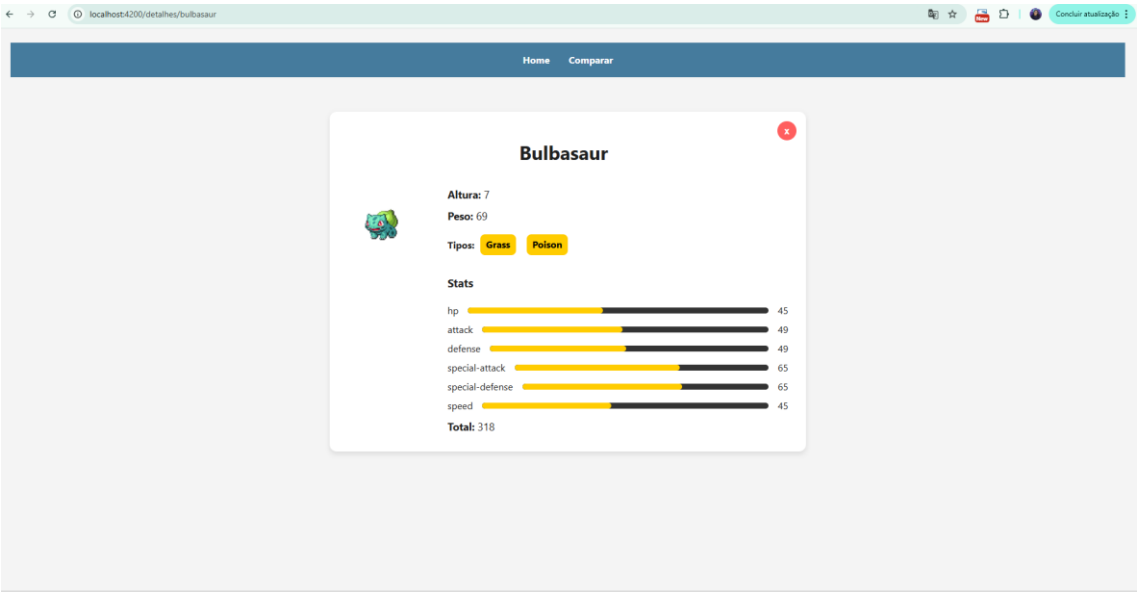
- Exibição da imagem do Pokémon, nome, número, tipos, altura, peso.
- Lista das habilidades e estatísticas base (como HP, ataque, defesa, etc.).

Telas

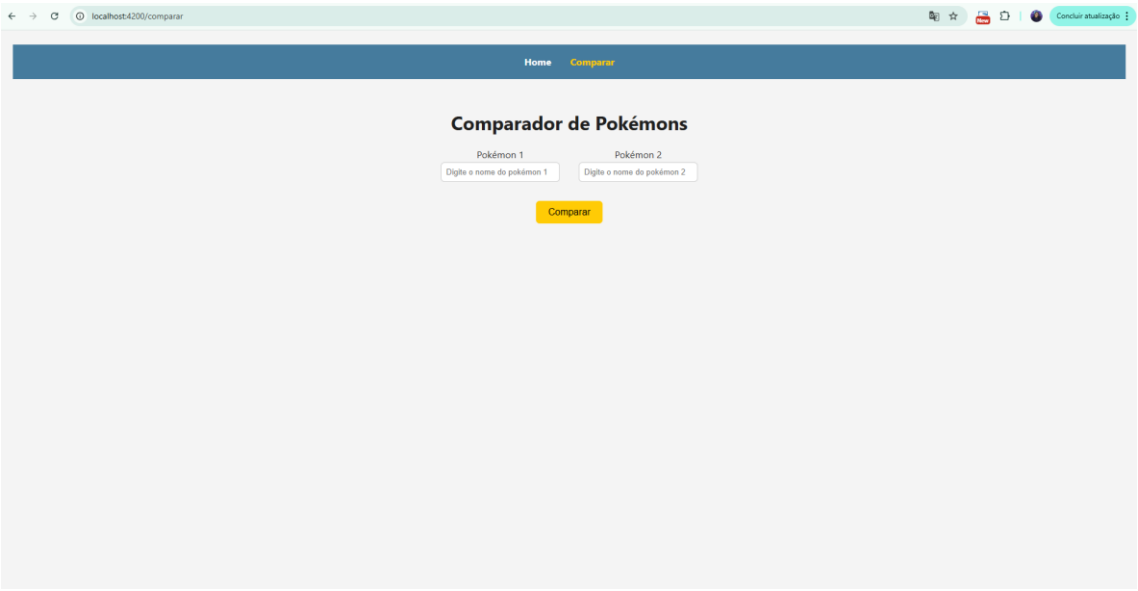
- **Tela Inicial/ Listagem**



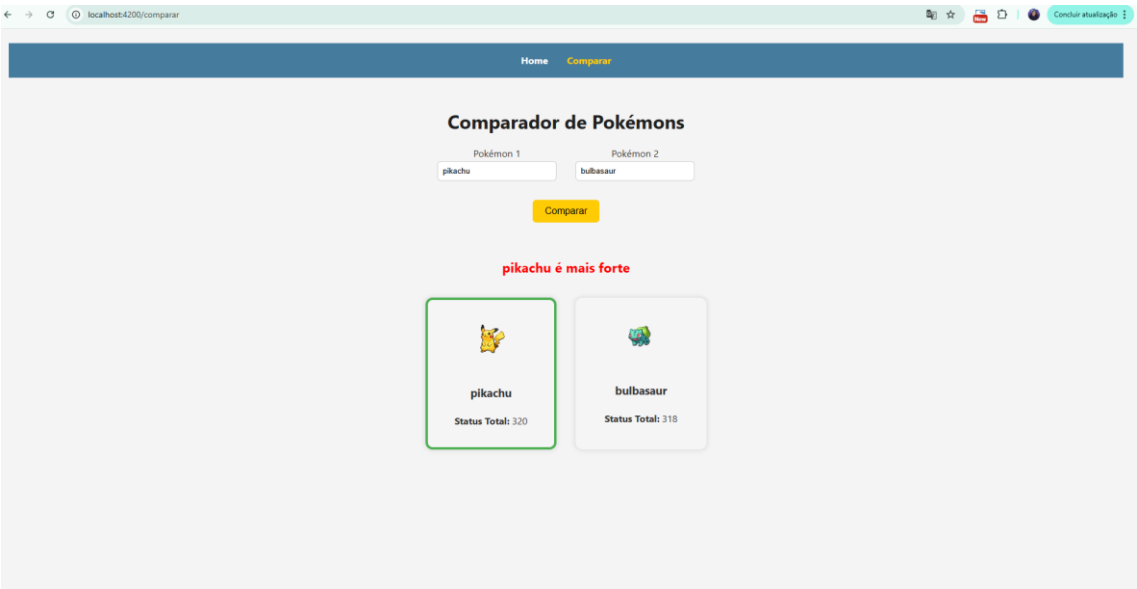
- Detalhes do Pokémon



- Tela de Comparação

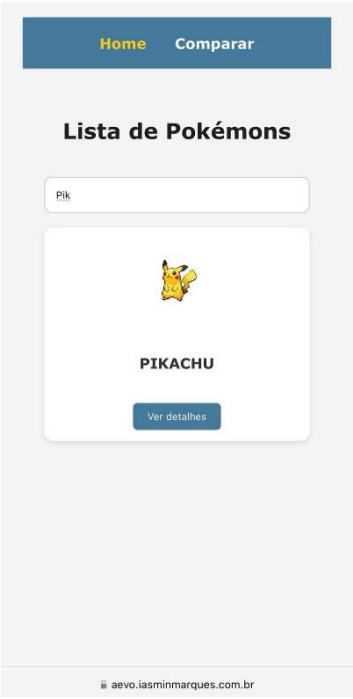


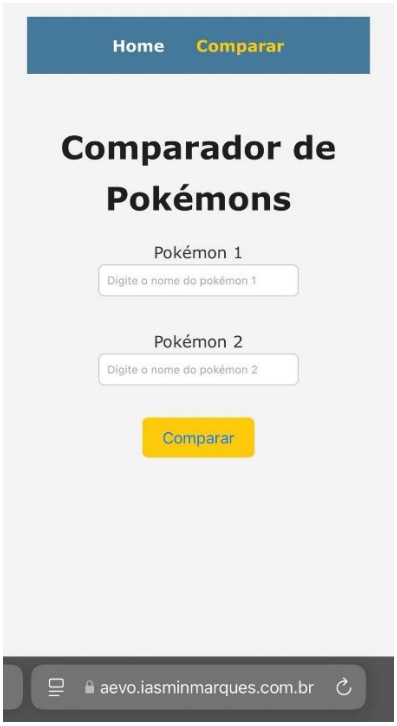
- Exemplo de Comparação

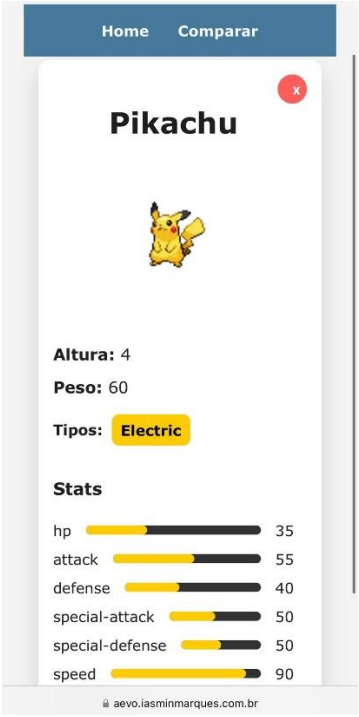


- Exemplo de erro de comparação

- **Telas do celular (Acessibilidade de UX)**







API

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

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

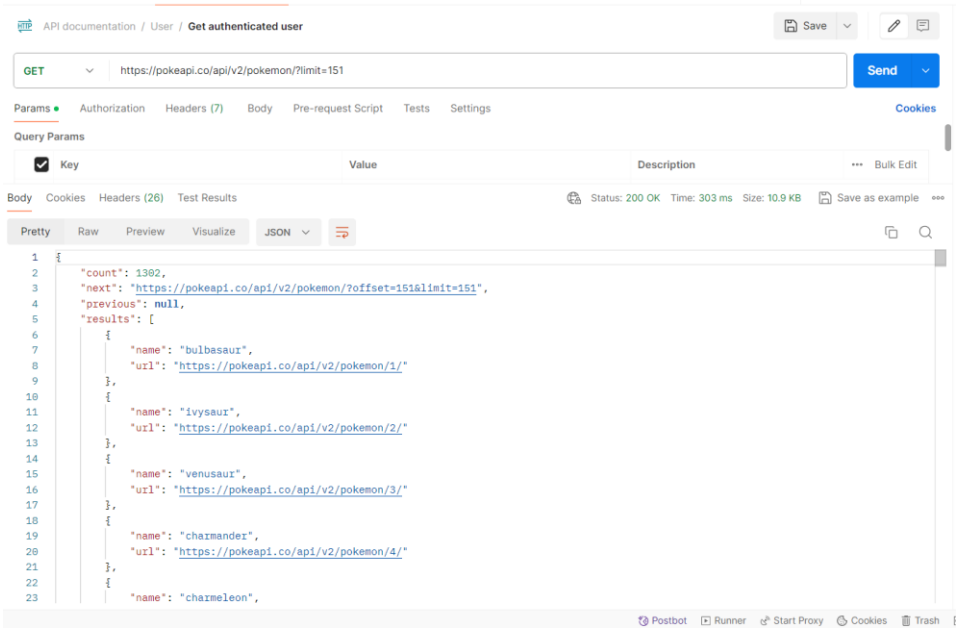
- **Requisições a API**

As chamadas à API são feitas via: pokemon.service.ts

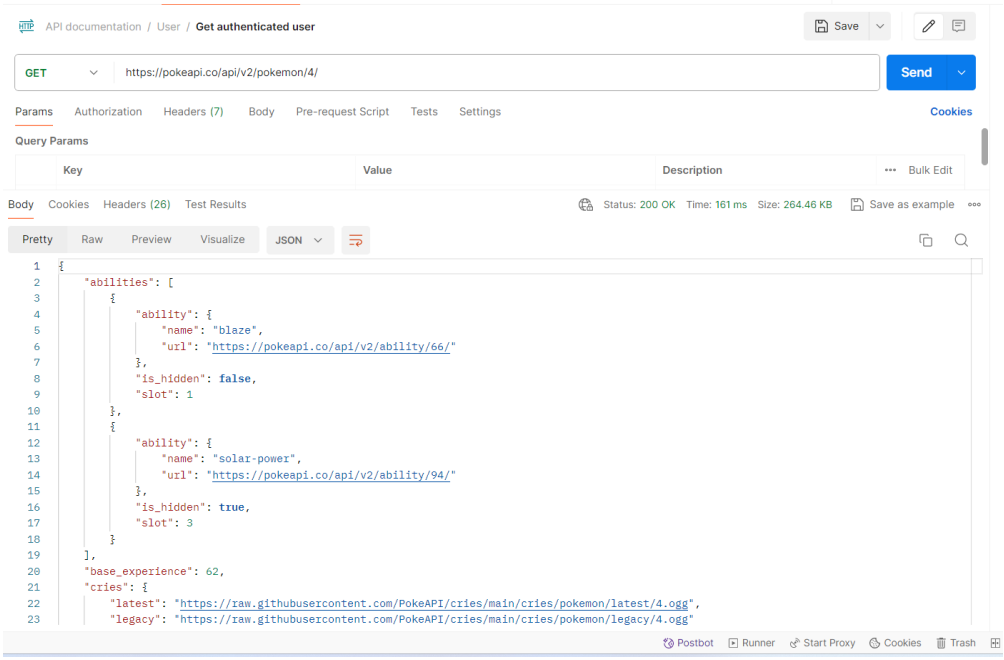
```ts
export class PokemonService {
  private apiUrl = 'https://pokeapi.co/api/v2';

  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

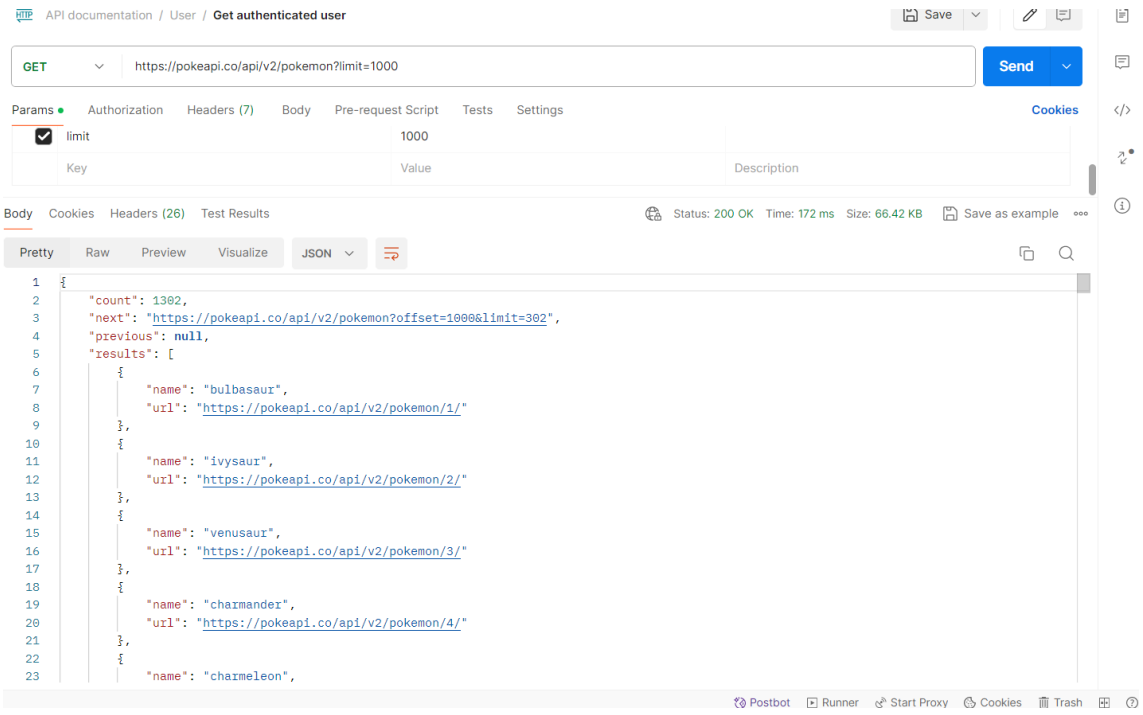
- Uma chamada da API para trazer 151 pokemons
  Rota: https://pokeapi.co/api/v2/pokemon/?limit=151



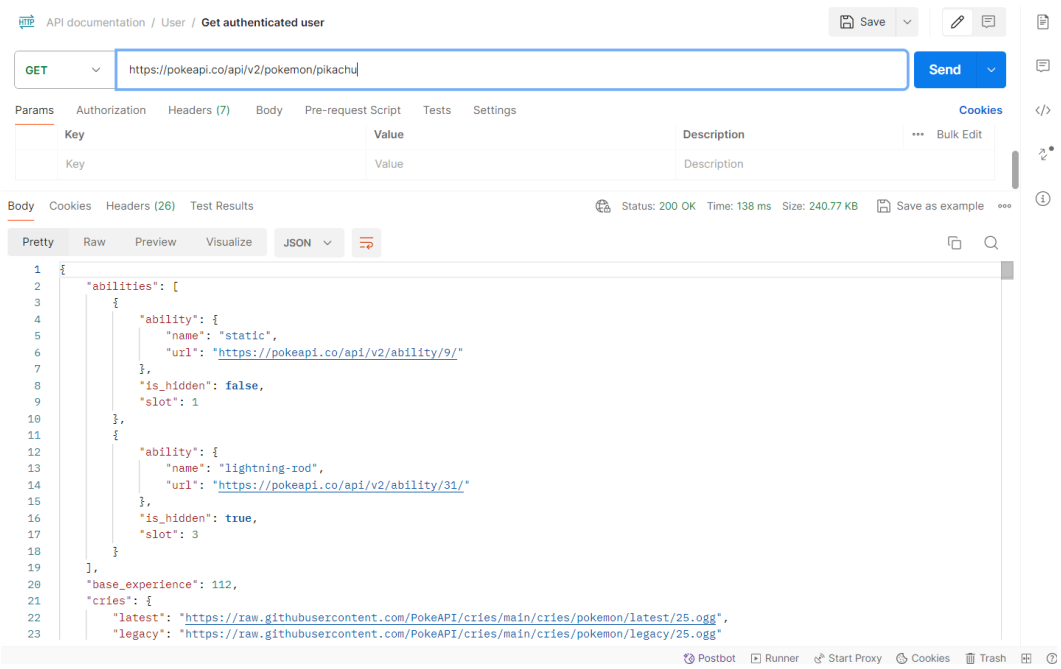
- Uma chamada da API para trazer mais informações do pokemon 4 - Charmander
  Rota: https://pokeapi.co/api/v2/pokemon/4/

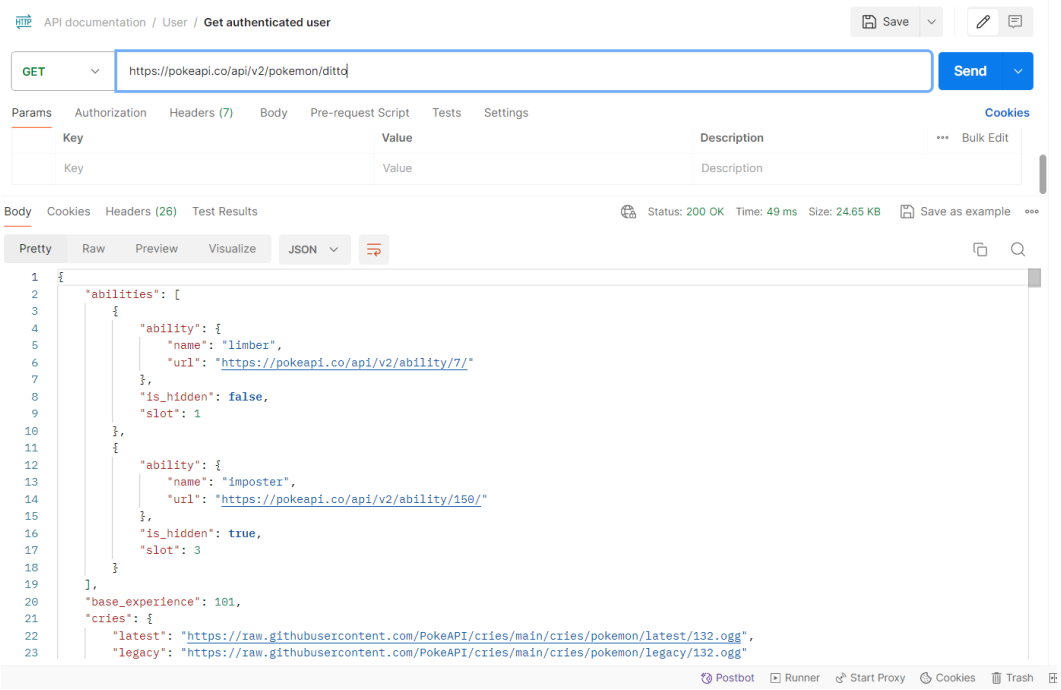


- Listagem de todos os Pokémon (limit=1000 define o número máximo de resultados retornados)
  Rota: https://pokeapi.co/api/v2/pokemon?limit=1000



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





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;