Na outra requisição estavamos pegando apenas a lista dos pokemons, agora queremos pegar os detalhes deles também

Lembrando que a api que estamos fazendo é xxx.xxx.com

Agora vamos fazer para o xxx.xxx.com/bulbasaur por exemplo

E assim vamos retornar os dados dele

Para entender melhor o que é promisse.all

🚀 O que é Promise.all()?

Promise.all() é um método da API de Promises que permite **executar várias promessas em paralelo** e **aguardar todas elas serem resolvidas** (ou uma falhar). Ele recebe um **array de promessas** e retorna uma **única promessa** que:

* **Resolve** quando **todas** as promessas do array forem resolvidas.
* **Rejeita** assim que **uma** das promessas for rejeitada.

Queremos acessar individualmente a url de cada pokemon

.then((pokemon) => pokemon.map((pokemon) => fetch(pokemon.url)))

Dessa forma aqui

pokeApi.getPokemons = (offset = 0, limit = 10) => {

    const url = `https://pokeapi.co/api/v2/pokemon?offset=${offset}&limit=${limit}` // URL de onde vamos puxar.

    return fetch(url)

        .then((response) => response.json()) // Pegamos o response e estamos retornando em formato JSON

        .then((jsonBody) => jsonBody.results) // JsonBody é o body completo e results é onde esta nossa lista de pokemons

        .then((pokemon) => pokemon.map((pokemon) => fetch(pokemon.url)))

        .catch((erro) => console.log(erro)) // Se tiver um erro no codigo vai aparecer no console.

}

Estamos dizendo o que?

Primeiro acesse o URL

O nosso response vai ser convertido para JSON

O nosso response vai retornar o jsonBody e vamos retornar o jsonBody.results que é onde esta nossa lista de pokemons

Após isso vamos pegar esse results vai ser nossos pokemons, vai ser uma lista.

Vamos pegar essa lista de pokemons e vamos transformar ela

Vamos transformar essa nova lista nos detalhes do pokemon

Para facilitar vamos criar um método para organizar o código

Ficando dessa maneira o código

const pokeApi = {} // NOSSO OBJETO

pokeApi.getPokemonsDetalhes = (pokemon) => {

    return fetch(pokemon.url)

    .then((response) => response.json())

}

pokeApi.getPokemons = (offset = 0, limit = 10) => {

    const url = `https://pokeapi.co/api/v2/pokemon?offset=${offset}&limit=${limit}` // URL de onde vamos puxar.

    return fetch(url)

        .then((response) => response.json()) // Pegamos o response e estamos retornando em formato JSON

        .then((jsonBody) => jsonBody.results) // JsonBody é o body completo e results é onde esta nossa lista de pokemons

        .then((pokemons) => pokemons.map(pokeApi.getPokemonsDetalhes))

        .then((detalhesPokemon) => Promise.all(detalhesPokemon))

        .then((detalhesPokemon) => {

            debugger

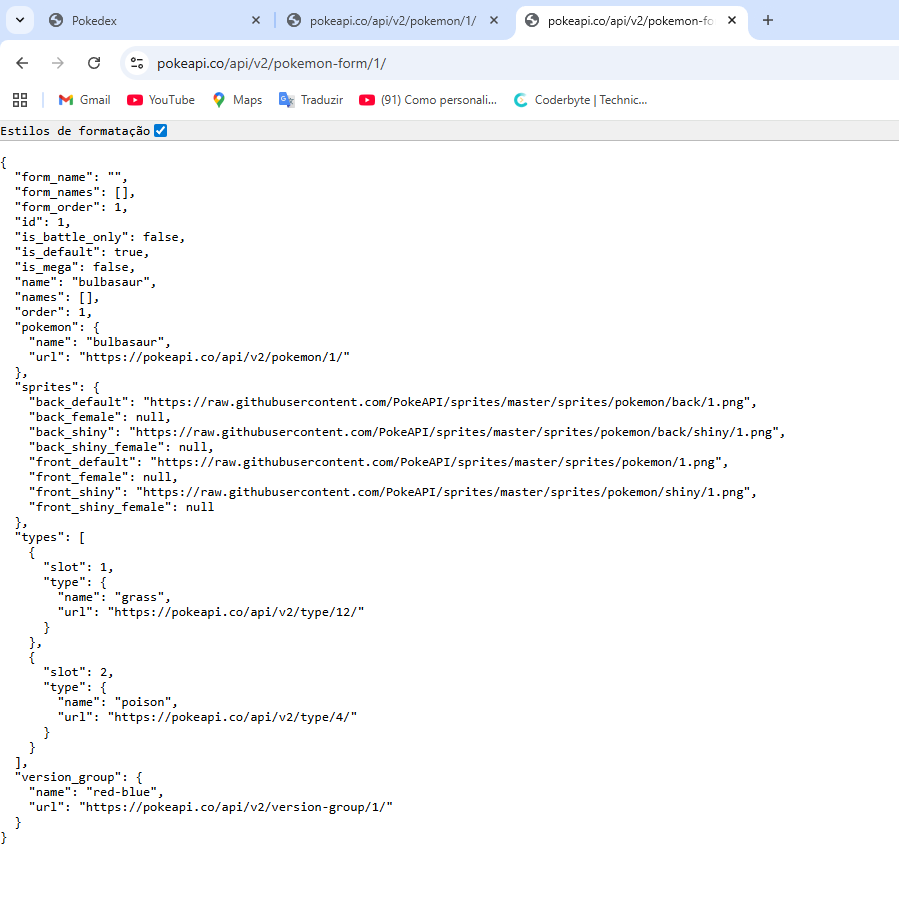
            console.log(detalhesPokemon)

        })

        .catch((erro) => console.log(erro)) // Se tiver um erro no codigo vai aparecer no console.

}

Agora já estamos retornando a lista de pokemons com detalhes, agora podemos ir no nosso HTML e colocar mais dados também.



Estamos usando esse end point, vamos pegar por exemplo o id

function converterPokemonParaHTML(pokemon) {

    return `

        <li class="pokemon">

                <span class="number">${pokemon.order}</span>

                <span class="name">${pokemon.name}</span>

                <div class="detail">

                    <ol class="types">

                        <li class="type">grass</li>

                        <li class="type">poison</li>

                    </ol>

                    <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/dream-world/1.svg"

                         alt="${pokemon.name}">

                </div>

        </li>

    `

}

Agora vamos fazer uma função para pegar o types também, eles são uma lista

function convertPokemonTypesToLi(pokemonTypes){

    return pokemonTypes.map((typeSlot) => `<li class="type">${typeSlot.type.name}</li>`)

}

function converterPokemonParaHTML(pokemon) {

    return `

        <li class="pokemon">

                <span class="number">${pokemon.order}</span>

                <span class="name">${pokemon.name}</span>

                <div class="detail">

                    <ol class="types">

                        ${convertPokemonTypesToLi(pokemon.types).join('')}

                    </ol>

                    <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/dream-world/1.svg"

                         alt="${pokemon.name}">

                </div>

        </li>

    `

}

const pokemonList = document.getElementsByClassName('pokemons')

/\*\* Vamos garantir que se a API não retonar nenhum pokemom, ela vai ser uma lista vazia por padrao. \*/

pokeApi.getPokemons().then((listaPokemons = []) => {

    /\*\* Atenção

     \* Lembrar que se na arrow function tiver apenas um retorno e uma unica linha, podemos remover ele e nao precisamos das {}

     \*/

    const novaListaDePokemons = listaPokemons.map((pokemon) => converterPokemonParaHTML(pokemon)) // Aqui estamos retornando o HTML

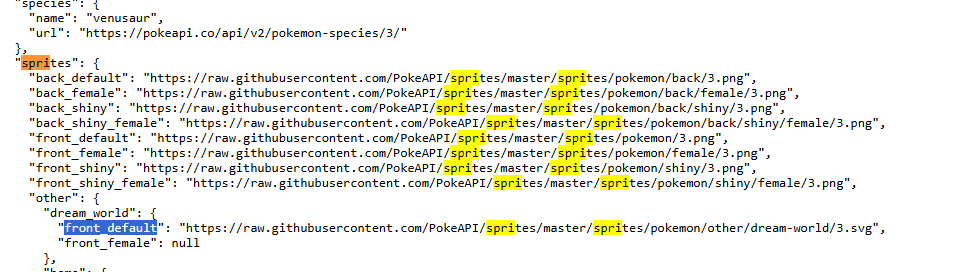
    const newHTML = novaListaDePokemons.join('') // Estamos juntando os dados

    pokemonList[0].innerHTML += newHTML // Juntando tudo no meu HTML

})

Agora vamos mudar a imagem também

Se você reparar essa é a estrutura que esta a imagem do pokemon



E para acessar fomos acessando objeto por objeto dessa forma

function converterPokemonParaHTML(pokemon) {

    return `

        <li class="pokemon">

                <span class="number">${pokemon.order}</span>

                <span class="name">${pokemon.name}</span>

                <div class="detail">

                    <ol class="types">

                        ${convertPokemonTypesToLi(pokemon.types).join('')}

                    </ol>

                    <img src="${pokemon.sprites.other.dream\_world.front\_default}"

                         alt="${pokemon.name}">

                </div>

        </li>

    `

}