POKETOURNAMENT

Elsa Ferreira, Adrian Tresgallo ,Pablo Mata, Iker Garcia, Angel Fernandez y Joel Martinez

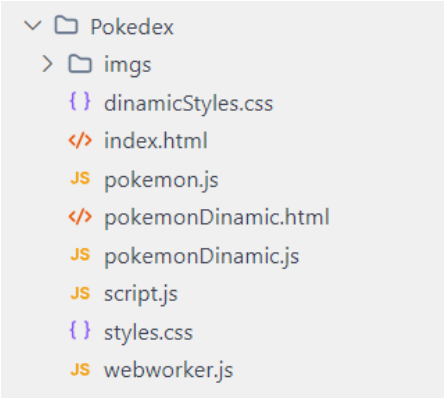
PokemonApi

Pokedex 3
Clase script.js 4
Clase webworker.js 9
Clase pokemonDinamic.html 12
Clase pokemonDinamic.js 12
Clase pokemon.js 15
Clase estilos.css 16
Equipo 18
Api.js 18
App.js 20
Pokemon.js 27
Index.html 28
Estilos.css 29
Comparador 31
funcionesComparador.js 31
Comparador.html 36
Combate 1vs1 37
LogicaCombate.js 37
PaginaInicio 43
CargaDatos.js 44
Index.html 45
style.css 45
Difcultades/Resumen 46

PokemonApi


APIPOKEMON-MAIN
Combate
Comparador
Equipos
PaginaInicio
Pokedex

Pokedex



Index.htlm

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pokedex</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
    <header>
        <img  src="imgs/charizard.gif">
        <a href="../index.html"><h1>Pokedex</h1></a>
        </header>
    <div id="filters">
        <input type="text" id="pokemon-search" placeholder="Buscar Pokémon..." />
        <select id="generation-filter">
            <option value="">Todas las generaciones</option>
            <option value="1">Generación I</option>
            <option value="2">Generación II</option>
            <option value="3">Generación III</option>
            <option value="4">Generación IV</option>
            <option value="5">Generación V</option>
            <option value="6">Generación VI</option>
            <option value="7">Generación VII</option>
            <option value="8">Generación VIII</option>
        </select>
    </div>
    <div id="pokemon-gallery"></div>
    <div id="pagination">
        <button id="prev-page">Anterior</button>
        <button id="next-page">Siguiente</button>
    </div>
    <script src="script.js" type="module"></script>
</body>
```

Clase script.js

```
import { PokemonAPI } from './webworker.js';

const gallery = document.getElementById('pokemon-gallery');
const generationFilter = document.getElementById('generation-filter');
const searchInput = document.getElementById('pokemon-search');
const prevButton = document.getElementById('prev-page');
const nextButton = document.getElementById('next-page');
const uriImages = 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/';

let currentPage = 1;
const limit = 42; //divisible entre 3 y 6(quedan bien als paginas)
let currentData = []; // Datos de la generación o filtro actual
let totalData = []; // Todos los datos cacheados
```

Esta clase implementa una aplicación que muestra una galería de Pokémon utilizando la PokeAPI. Permite explorar Pokémon por generaciones, buscarlos por nombre y navegar entre páginas con un límite de 42 Pokémon por página

Carga inicial de datos:

- Al cargarse la página, verifica si los datos de los Pokémon están almacenados en localStorage.
- Si no están, utiliza la API para descargarlos y almacenarlos localmente.
- Una vez cargados, renderiza la galería con los datos disponibles

```
// Cargar todas las generaciones y Pokémon al inicio
document.addEventListener('DOMContentLoaded', async () => {
    const storedData = localStorage.getItem('allPokemon');
    if (storedData) {
        totalData = JSON.parse(storedData);
        currentData = totalData;
        renderGallery(currentData);
        updatePaginationButtons(currentData.length);
    } else {
        await pokemonAPI.saveAllPokemonToLocalStorage();
        totalData = getAllPokemonFromLocalStorage();
        currentData = totalData;
        renderGallery(currentData);
        updatePaginationButtons(currentData.length);
    }
    const generations = await pokemonAPI.fetchGenerations();
    populateGenerationFilter(generations);
});
```

Filtro por generación:

- Ofrece un menú desplegable con generaciones (e.g., Generación 1, Generación 2, etc.).

- Al seleccionar una generación, filtra los Pokémon correspondientes utilizando datos de la API.

```
// Evento para el filtro de generación
generationFilter.addEventListener('change', async () => {
    const selectedGenerationURL = generationFilter.value;
    if (selectedGenerationURL === '') {
        loadAllPokemon(); // Mostrar todos los Pokémon
    } else {
        const pokemonNames = await pokemonAPI.fetchGenerationData(selectedGenerationURL);
        filterByNames(pokemonNames);
    }
});
```

Búsqueda dinámica:

- Permite buscar Pokémon por nombre en el filtro actual (usando input).
- Actualiza la galería en tiempo real mientras el usuario escribe.

```
// Evento para el buscador
searchInput.addEventListener('input', () => {
    const query = searchInput.value.toLowerCase();
    const filtered = currentData.filter(pokemon =>
        pokemon.name.toLowerCase().includes(query)
    );
    currentPage = 1; // Restablecer a la primera página
    renderGallery(filtered);
    updatePaginationButtons(filtered.length);
});
```

Paginación:

- Divide la galería en páginas con un límite de 42 Pokémon por página.
- Los botones "Anterior" y "Siguiente" permiten moverse entre páginas.
- Habilita o deshabilita los botones según la página actual.

```
// Eventos para la paginación
prevButton.addEventListener('click', () => changePage(-1));
nextButton.addEventListener('click', () => changePage(1));
```

Traducción de tipos:

- Mapea los tipos de Pokémon a español (e.g., "fire" a "Fuego") y les asigna un color característico para mostrarlo en la tarjeta del Pokémon.

```
// Rellenar el filtro de generaciones
function populateGenerationFilter(generations) {
    generationFilter.innerHTML = '<option value="">Todas las generaciones</option>';
    generations.forEach((generation, index) => {
        const option = document.createElement('option');
        option.value = generation.url;
        option.textContent = `Generación ${index + 1}`;
        generationFilter.appendChild(option);
    });
}
```

Cargar todos los pokemons

```
// Cargar todos los Pokémon
function loadAllPokemon() {
    totalData = getAllPokemonFromLocalStorage();
    currentData = totalData;
    currentPage = 1;
    renderGallery(currentData);
    updatePaginationButtons(currentData.length);
}
```

- Carga todos los Pokémon almacenados en localStorage.
- Actualiza las variables totalData y currentData con todos los datos.
- Reinicia la página actual a la primera y renderiza la galería completa.
- Actualiza los botones de paginación.

Obtener Pokemons localStorge

```
// Obtener todos los Pokémon desde localStorage
function getAllPokemonFromLocalStorage() {
    const allPokemon = [];
    for (let i = 1; i <= 1025; i++) {
        const pokemonData = localStorage.getItem(`pokemon_${i}`);
        if (pokemonData) {
            allPokemon.push(JSON.parse(pokemonData));
        }
    }
    localStorage.setItem('allPokemon', JSON.stringify(allPokemon));
    return allPokemon;
}
```

getAllPokemonFromLocalStorage()

- Recupera los datos de todos los Pokémon almacenados en localStorage con claves del formato pokemon_1, pokemon_2, etc.
- Combina los datos en un solo array allPokemon.

- Almacena el resultado combinado en localStorage bajo la clave allPokemon.
- Devuelve el array de Pokémon.

filterByNames(names)

```
// Filtrar por nombres (generación seleccionada)
function filterByNames(names) {
    currentData = totalData.filter(pokemon => names.includes(pokemon.name));
    currentPage = 1; // Restablecer a la primera página
    renderGallery(currentData);
    updatePaginationButtons(currentData.length);
}
```

- Filtra los Pokémon en totalData que coincidan con los nombres en el array names.
- Actualiza currentData con los resultados filtrados.
- Reinicia la página actual a la primera.
- Renderiza la galería filtrada y actualiza los botones de paginación.

changePage(direction)

```
// Cambiar página
function changePage(direction) {
    currentPage += direction;
    renderGallery(currentData);
    updatePaginationButtons(currentData.length);
}
```

- Cambia la página actual sumando o restando según la dirección (-1 o +1).
- Renderiza la galería de la página nueva y actualiza los botones de paginación.

renderGallery(data)

```
// Renderizar galería con paginación
function renderGallery(data) {
    gallery.innerHTML = '';
    const startIndex = (currentPage - 1) * limit;
    const endIndex = startIndex + limit;

    const pageData = data.slice(startIndex, endIndex);
    pageData.forEach(pokemon => {
        const card = document.createElement('div');
        card.classList.add('pokemon-card');

        const img = document.createElement('img');
        img.classList.add('pokemon-img');
        img.src = uriImages + pokemon.id + '.png' || '';
        img.alt = pokemon.name;

        const name = document.createElement('p');
        name.textContent = pokemon.name;
        name.classList.add('pokemon-name');

        const numero = document.createElement('p');
        numero.textContent = '#' + pokemon.id;
        numero.classList.add('pokemon-number');

        const typesContainer = document.createElement('div');
        typesContainer.classList.add('pokemon-types-container');
```

```
        pokemon.types.forEach(typeInfo => {
            const type = typeInfo.name;
            const typeData = typeTranslations[type];
            if (typeData) {
                const typeElement = document.createElement('span');
                typeElement.textContent = typeData.name;
                typeElement.classList.add('pokemon-type');
                typeElement.style.backgroundColor = typeData.color;
                typesContainer.appendChild(typeElement);
            }
        });

        card.appendChild(img);
        card.appendChild(name);
        card.appendChild(typesContainer);
        card.appendChild(numero);

        // Añadir evento de clic para redirigir a la página de detalles
        card.addEventListener('click', () => {
            window.location.href = `pokemonDinamic.html?id=${pokemon.id}`;
        });

        gallery.appendChild(card);
```

- Limpia la galería HTML (gallery.innerHTML = '').
- Calcula los índices de inicio y fin según la página actual y el límite de Pokémon por página.
- Genera y agrega al DOM las tarjetas de Pokémon correspondientes a la página actual:
- Muestra la imagen, nombre, número y tipos con colores.
- Añade un evento de clic para redirigir a la página de detalles del Pokémon.

updatePaginationButtons(totalItems)

```
// Actualizar botones de paginación
function updatePaginationButtons(totalItems) {
    const totalPages = Math.ceil(totalItems / limit);
    prevButton.disabled = currentPage === 1;
    nextButton.disabled = currentPage === totalPages;
}
```

- Calcula el número total de páginas según el total de elementos (totalItems) y el límite por página.
- Habilita o deshabilita los botones "Anterior" y "Siguiente" dependiendo de si la página actual es la primera o la última.

Clase webworker.js

```
import { Pokemon } from './pokemon.js';

class PokemonAPI {
    constructor() {
        this.apiBaseURL = 'https://pokeapi.co/api/v2/';
        this.pokemonURL = `${this.apiBaseURL}pokemon/`;
        this.generationURL = `${this.apiBaseURL}generation/`;
    }

    async fetchPokemon(id) {
        try {
            const response = await fetch(`${this.pokemonURL}${id}`);
            const data = await response.json();
            return new Pokemon(
                data.id,
                data.name,
                data.types.map(typeInfo => typeInfo.type),
                data.height,
                data.weight,
                data.abilities.map(abilityInfo => abilityInfo.ability.name),
                data.stats.map(statInfo => ({ name: statInfo.stat.name, value: statInfo.base_stat }))
            );
        } catch (error) {
            console.error('Error fetching Pokémon:', error);
        }
    }
```

1. Constructor constructor()

- Inicializa la clase definiendo las URLs base de la API:
    - **this.apiBaseURL**: URL base de la PokéAPI (https://pokeapi.co/api/v2/).
    - **this.pokemonURL**: URL para obtener datos de Pokémon individuales.
    - **this.generationURL**: URL para obtener datos de generaciones.

2. **async fetchPokemon(id)**

- **Descripción**: Obtiene los datos de un Pokémon específico usando su ID.
- **Parámetros**:
  - `id`: El ID del Pokémon (número o nombre).
- **Funcionamiento**:
  - Hace una petición `GET` a la PokéAPI (`https://pokeapi.co/api/v2/pokemon/{id}`).
  - Transforma la respuesta JSON en una instancia de la clase `Pokemon`.
  - Extrae y organiza los datos relevantes, como tipos, altura, peso, habilidades y estadísticas.
- **Retorna**:
  - Un objeto de la clase `Pokemon`.
- **Manejo de errores**: Imprime un error en consola si la petición falla.

```
async saveAllPokemonToLocalStorage() {
    try {
        const promises = [];
        for (let i = 1; i <= 1025; i++) {
            promises.push(this.fetchPokemon(i));
        }
        const allPokemon = await Promise.all(promises);
        allPokemon.forEach(pokemon => {
            if (pokemon) {
                localStorage.setItem(`pokemon_${pokemon.id}`, JSON.stringify(pokemon));
            }
        });
    } catch (error) {
        console.error('Error saving Pokémon to localStorage:', error);
    }
}
```

Descripción: Descarga los datos de todos los Pokémon (IDs del 1 al 1025) y los guarda en localStorage.

Funcionamiento:

- Usa un bucle para iterar del 1 al 1025 y genera una promesa por cada Pokémon utilizando fetchPokemon(id).
- Espera a que se resuelvan todas las promesas con Promise.all(promises).
- Almacena cada Pokémon en localStorage con una clave del tipo pokemon_{id}.

Manejo de errores: Imprime un error en consola si algo falla durante la descarga o el almacenamiento.

```
async fetchGenerations() {
    try {
        const response = await fetch(this.generationURL);
        const data = await response.json();
        return data.results; // Devuelve todas las generaciones
    } catch (error) {
        console.error('Error fetching generations:', error);
    }
}
```

async fetchGenerations()

- **Descripción**: Obtiene la lista de todas las generaciones disponibles en la PokéAPI.
- **Funcionamiento**:
    - Hace una petición GET a la URL https://pokeapi.co/api/v2/generation/.
    - Retorna un array de objetos con información básica de las generaciones (name y url).
- **Manejo de errores**: Imprime un error en consola si la petición falla.

```
async fetchGenerationData(url) {
    try {
        const response = await fetch(url);
        const data = await response.json();
        return data.pokemon_species.map(species => species.name);
    } catch (error) {
        console.error('Error fetching generation data:', error);
    }
}
```

async fetchGenerationData(url)

- **Descripción**: Obtiene la lista de especies de Pokémon pertenecientes a una generación específica.
- **Parámetros**:
    - url: URL de una generación específica (e.g., https://pokeapi.co/api/v2/generation/1/).
- **Funcionamiento**:
    - Hace una petición GET a la URL proporcionada.

- Retorna un array de nombres de especies de Pokémon que pertenecen a esa generación.

Manejo de errores: Imprime un error en consola si la petición falla.

```
export { PokemonAPI };
```

Marca la clase PokemonAPI como exportable desde el archivo donde está definida.

Clase pokemonDinamic.html

Pokedex > pokemonDinamic.html > html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Detalles del Pokémon</title>
    <link rel="stylesheet" href="dinamicStyles.css">
    <link href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap" rel="stylesheet">

</head>
<body id="pokemon-dinamic">
    <header>
        <img  src="imgs/charizard.gif">
        <a href="index.html" ><h1 id="header-title"></h1></a>
    </header>
    <div id="pokemon-details"></div>
    <script src="pokemonDinamic.js" type="module"></script>
</body>
</html>
```

Clase pokemonDinamic.js

```
import { PokemonAPI } from './webworker.js';
const pokemonAPI = new PokemonAPI();
const detailsContainer = document.getElementById('pokemon-details');
const uriImages = 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/';
const uriSprites = 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/';

// Obtener el ID del Pokémon de la URL
const urlParams = new URLSearchParams(window.location.search);
const pokemonId = urlParams.get('id');
const typeTranslations = {
    normal: { name: 'Normal', color: '#A8A77A' },
    fire: { name: 'Fuego', color: '#EE8130' },
    water: { name: 'Agua', color: '#6390F0' },
    electric: { name: 'Eléctrico', color: '#F7D02C' },
    grass: { name: 'Planta', color: '#7AC74C' },
    ice: { name: 'Hielo', color: '#96D9D6' },
    fighting: { name: 'Lucha', color: '#C22E28' },
    poison: { name: 'Veneno', color: '#A33EA1' },
    ground: { name: 'Tierra', color: '#E2BF65' },
    flying: { name: 'Volador', color: '#A98FF3' },
    psychic: { name: 'Psíquico', color: '#F95587' },
    bug: { name: 'Bicho', color: '#A6B91A' },
    rock: { name: 'Roca', color: '#B6A136' },
    ghost: { name: 'Fantasma', color: '#735797' },
    dragon: { name: 'Dragón', color: '#6F35FC' },
    dark: { name: 'Siniestro', color: '#705746' },
    steel: { name: 'Acero', color: '#B7B7CE' },
    fairy: { name: 'Hada', color: '#D685AD' }
};
```

PokemonApi

```
document.addEventListener('DOMContentLoaded', async () => {
    if (pokemonId) {
        const pokemonData = await fetchPokemonFromLocalStorage(pokemonId);
        if (pokemonData) {
            renderPokemonDetails(pokemonData);
        } else {
            const pokemon = await pokemonAPI.fetchPokemon(pokemonId);
            renderPokemonDetails(pokemon);
        }
    }
});
```

Eventos y flujo principal

1. **Evento DOMContentLoaded**:
    - Al cargarse la página, se verifica si hay un pokemonId en la URL.
    - Si existe, primero intenta cargar el Pokémon desde localStorage mediante la función fetchPokemonFromLocalStorage.
    - Si no está en localStorage, utiliza la clase PokemonAPI para obtener los datos de la API y luego renderiza esos datos.

```
function fetchPokemonFromLocalStorage(id) {
    const pokemonData = localStorage.getItem(`pokemon_${id}`);
    return pokemonData ? JSON.parse(pokemonData) : null;
}
```

fetchPokemonFromLocalStorage(id):

- Intenta recuperar los datos del Pokémon desde el almacenamiento local (localStorage).
- Si el Pokémon está en localStorage, los devuelve como un objeto; si no, devuelve null

```
function renderPokemonDetails(pokemon) {
    let titulo = document.getElementById('header-title');
    titulo.textContent = pokemon.name;
    detailsContainer.innerHTML = `
        <img src="${uriImages}${pokemon.id}.png" alt="${pokemon.name}">
        <p>ID: #${pokemon.id}</p>
        <p>Altura: ${pokemon.height} dm</p>
        <p>Peso: ${pokemon.weight} hg</p>
        <div class="pokemon-types-container">
            ${pokemon.types.map(typeInfo => {
                const typeData = typeTranslations[typeInfo.name];
                return `<span class="pokemon-type" style="background-color: ${typeData.color};">${typeData.name}</span>`;
            }).join('')}
        </div>
        <h2>Habilidades</h2>
        <ul>
            ${pokemon.abilities.map(ability => `<li>${ability}</li>`).join('')}
        </ul>
```

```
<h2>Habilidades</h2>
<ul>
    ${pokemon.abilities.map(ability => `<li>${ability}</li>`).join('')}
</ul>
<h2>Estadísticas</h2>
<ul>
    ${pokemon.stats.map(stat => `<li>${stat.name}: ${stat.value}</li>`).join('')}
</ul>
<h2>Imágenes y Sprites</h2>
<div class="pokemon-sprites">
    <img src="${uriSprites}${pokemon.id}.png" alt="${pokemon.name} sprite">
    <img src="${uriSprites}back/${pokemon.id}.png" alt="${pokemon.name} back sprite">
    <img src="${uriSprites}shiny/${pokemon.id}.png" alt="${pokemon.name} shiny sprite">
    <img src="${uriSprites}back/shiny/${pokemon.id}.png" alt="${pokemon.name} back shiny sprite">
</div>
```

```
<h2>Tabla de Daños</h2>
<div class="damage-table">
    <div class="damage-row header">
        <div class="damage-cell">Tipo</div>
        <div class="damage-cell">Daño a</div>
        <div class="damage-cell">Débil contra</div>
    </div>
    ${pokemon.types.map(typeInfo => {
        const typeData = typeTranslations[typeInfo.name];
        return `
            <div class="damage-row">
                <div class="damage-cell" style="background-color: ${typeData.color};">${typeData.name}</
                <div class="damage-cell">${getDamageTo(typeInfo.name)}</div>
                <div class="damage-cell">${getWeakAgainst(typeInfo.name)}</div>
            </div>
        `;
    }).join('')}
```

renderPokemonDetails(pokemon):

- Renderiza los detalles de un Pokémon en la página, incluyendo:
    - Imagen oficial.
    - Información básica (ID, altura, peso).
    - Tipos con sus traducciones y colores.
    - Habilidades.
    - Estadísticas (ataque, defensa, etc.).
    - Diferentes sprites (normal, shiny, trasero, etc.).

- Una tabla de daños mostrando ventajas y debilidades del Pokémon.

```
function getDamageTo(type) {
    const damageTo = {
        fire: ['Grass', 'Ice', 'Bug', 'Steel'],
        water: ['Fire', 'Ground', 'Rock'],
        electric: ['Water', 'Flying'],
        grass: ['Water', 'Ground', 'Rock'],
        ice: ['Grass', 'Ground', 'Flying', 'Dragon'],
        fighting: ['Normal', 'Ice', 'Rock', 'Dark', 'Steel'],
        poison: ['Grass', 'Fairy'],
        ground: ['Fire', 'Electric', 'Poison', 'Rock', 'Steel'],
        flying: ['Grass', 'Fighting', 'Bug'],
        psychic: ['Fighting', 'Poison'],
        bug: ['Grass', 'Psychic', 'Dark'],
        rock: ['Fire', 'Ice', 'Flying', 'Bug'],
        ghost: ['Psychic', 'Ghost'],
        dragon: ['Dragon'],
        dark: ['Psychic', 'Ghost'],
        steel: ['Ice', 'Rock', 'Fairy'],
        fairy: ['Fighting', 'Dragon', 'Dark'],

    };
    return damageTo[type] ? damageTo[type].join(', ') : 'N/A';
}

function getWeakAgainst(type) {
    const weakAgainst = {
        fire: ['Water', 'Rock', 'Fire'],
        water: ['Electric', 'Grass'],
        // Añadir más tipos según sea necesario
    };
    return weakAgainst[type] ? weakAgainst[type].join(', ') : 'N/A';
}
```

getDamageTo(type) y **getWeakAgainst(type)**:

- Estas funciones determinan los tipos contra los que el tipo actual tiene ventaja (damageTo) o desventaja (weakAgainst).
- La información está configurada manualmente en objetos damageTo y weakAgainst

Clase pokemon.js

```
export class Pokemon {
    constructor(id, name, types, height, weight, abilities, stats) {
        this.id = id;
        this.name = name;
        this.types = types;
        this.height = height;
        this.weight = weight;
        this.abilities = abilities;
        this.stats = stats;
    }
}
```

Partes del código:

1. **export class Pokemon**:
    - Declara una clase llamada Pokemon.

- Se utiliza export para que esta clase pueda ser importada en otros archivos JavaScript.

2. **constructor(id, name, types, height, weight, abilities, stats)**:
   - Define un constructor que permite inicializar las propiedades de un Pokémon cuando se crea una instancia de la clase.
   - Los parámetros del constructor coinciden con las propiedades típicas de un Pokémon que se obtiene desde la PokeAPI.

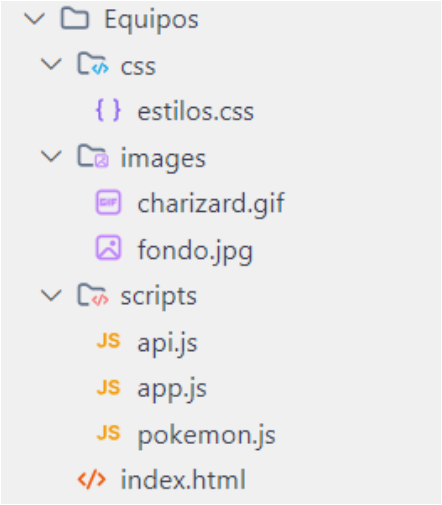
Clase estilos.css

```css
/* General Styles */
body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f9;
    margin: 0;
    padding: 0;
    color: #333;
}

/*Cabecera*/
header{
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2); /* Sombra */
    justify-content: center;
    display: flex;
    background: linear-gradient(90deg, #ff0000, #0000ff); /* Gradiente de colores */
}
header h1{
    font-size: 3rem;
    justify-content: center;
    display: flex;
    padding: 20px;
    margin: 20px 0; /* Espaciado */
    text-align: center; /* Centrado */
    color: white;
    text-shadow: 2px 2px 8px rgba(0, 0, 0, 0.7); /* Sombra de texto para darle profundidad */
    font-weight: bold;

}
header a {

    text-decoration: none;
}
```

```css
/* estilos para filtros*/
#filters{
    margin-top: 10px;
    display: flex;
    justify-content: center;
    align-items: center;
    margin-right: 20px;

}
/* Estilos para el buscador */
#pokemon-search {
    padding: 12px 20px;
    font-size: 16px;
    border-radius: 30px;
    background-color: #fff;
    width: 300px;
    transition: all 0.3s ease;
    border: 2px solid #00000030;
}

/* Estilos para el filtro de generación */
#generation-filter {
    padding: 12px 20px;
    font-size: 16px;
    border-radius: 30px;
    background-color: #fff;
    transition: all 0.3s ease;
    margin-left: 20px;
    border: 2px solid #00000030;
}
```

```css
/* Pokemon Card */
.pokemon-card {
    background-color: #ffffff;
    border: 1px solid #ddd;
    border-radius: 15px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
    width: 220px;
    text-align: center;
    padding: 15px;
    position: relative;
    border: 2px solid #00000030;
}

.pokemon-card:hover {
    transform: scale(1.1);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    .pokemon-number{
        background-color: #999;
        color: white;
    }
}
```

Equipo



Api.js

```
export async function obtenerPokemonPorGeneracion(generacion, pagina = 1, limite = 20) {
    const desplazamiento = (pagina - 1) * limite;  // Cálculo de desplazamiento para paginación
    let url = '';
    if (generacion) {
      // Si hay generación seleccionada, traemos los Pokémon de esa generación
      url = `https://pokeapi.co/api/v2/generation/${generacion}`;
    } else {
      // Si no hay generación seleccionada, traemos los Pokémon globales
      url = `https://pokeapi.co/api/v2/pokemon?offset=${desplazamiento}&limit=${limite}`;
    }
    const respuesta = await fetch(url);
    const datos = await respuesta.json();
    return datos;
  }
```

obtenerPokemonPorGeneracion

Propósito:

Obtener una lista de Pokémon, ya sea de una generación específica o de forma global, con soporte para paginación.

Parámetros:

1. **generacion**: (Opcional) Un identificador de generación (por ejemplo, 1 para la primera generación, 2 para la segunda, etc.).
2. **pagina**: (Opcional) Número de la página que se desea cargar. Por defecto, es la página 1.
3. **limite**: (Opcional) Número de Pokémon a obtener por página. Por defecto, se establece en 20.

Funcionamiento:

1. Calcula el **desplazamiento** usando la fórmula (pagina - 1) * limite. Esto determina desde qué punto comenzar a obtener datos para la paginación.
2. Decide la **URL** de la API:
    - Si se proporciona un valor para generacion, utiliza la URL específica para obtener los Pokémon de esa generación (/generation/{generacion}).
    - Si no se especifica una generación, utiliza la URL global para listar todos los Pokémon con paginación (/pokemon?offset={desplazamiento}&limit={limite}).
3. Realiza una solicitud fetch a la URL calculada.
4. Convierte la respuesta en un objeto JSON y la devuelve.

```
export async function obtenerDetallesPokemon(url) {
  const respuesta = await fetch(url);
  const detalles = await respuesta.json();
  return detalles;
}
```

Propósito:

Obtener los detalles completos de un Pokémon específico, dado su URL.

Parámetros:

- **url**: La URL de un Pokémon específico proporcionada por la PokeAPI.

Funcionamiento:

1. Realiza una solicitud fetch a la URL proporcionada.
2. Convierte la respuesta en un objeto JSON que contiene los detalles del Pokémon.
3. Devuelve los datos obtenidos.

Relación entre ambas funciones

1. **obtenerPokemonPorGeneracion**: Proporciona una lista de Pokémon (ya sea de una generación o de forma global). Cada entrada de esta lista generalmente incluye un campo url que apunta a los detalles del Pokémon.

2. **obtenerDetallesPokemon**: Puede utilizarse para obtener la información completa de cualquier Pokémon usando la url obtenida de obtenerPokemonPorGeneracion.

App.js

Selección de elementos del DOM

```
const cuadros = document.querySelectorAll('.cuadro');
const ventanaEmergente = document.getElementById('ventanaEmergente');
const listaPokemon = document.getElementById('listaPokemon');
const cerrarVentanaEmergenteBtn = document.getElementById('cerrarVentanaEmergente');
const paginadorAnterior = document.getElementById('anterior');
const paginadorSiguiente = document.getElementById('siguiente');
const filtroGeneracion = document.getElementById('filtro-generacion');
```

Estos elementos son referencias a elementos HTML de la interfaz gráfica:

- **cuadros**: Son los cuadros donde se mostrarán los Pokémon seleccionados.
- **ventanaEmergente**: Modal que muestra la lista de Pokémon para seleccionar.
- **listaPokemon**: Contenedor donde se listan las tarjetas de Pokémon.
- **cerrarVentanaEmergenteBtn**: Botón para cerrar el modal.
- **Paginadores**: Botones para cambiar entre páginas de Pokémon.
- **filtroGeneracion**: Filtro para seleccionar Pokémon por generación.

Variables globales

```
let equipoActual = new Equipo(Date.now());  // Usamos el timestamp como ID único para el equipo
let cuadroSeleccionado = null;
let paginaActual = 1;
let totalPokemons = 0;
const limitePorPagina = 20;
```

Función CargarPokemon

PokemonApi

```
async function cargarPokemon(pagina) {
  const generacion = filtroGeneracion.value; // Obtener generación seleccionada
  listaPokemon.innerHTML = ''; // Limpiar la lista de Pokémon
  let pokemonData = [];

  if (generacion) {
    // Obtener todos los Pokémon de la generación seleccionada
    const datos = await obtenerPokemonPorGeneracion(generacion);
    pokemonData = datos.pokemon_species;

    // Dividir los Pokémon de la generación en páginas
    const inicio = (pagina - 1) * limitePorPagina;
    const fin = inicio + limitePorPagina;
    pokemonData = pokemonData.slice(inicio, fin);
  } else {
    // Obtener Pokémon globales con paginación
    const datos = await obtenerPokemonPorGeneracion(null, pagina, limitePorPagina);
    pokemonData = datos.results;
  }
  // Mostrar tarjetas de Pokémon
  for (const pokemon of pokemonData) {
    const urlPokemon = generacion ? pokemon.url.replace('-species', '') : pokemon.url;
    const detalles = await obtenerDetallesPokemon(urlPokemon);
    const tarjeta = document.createElement('div');
    tarjeta.classList.add('tarjeta-pokemon');
    tarjeta.innerHTML = `
      <img src="${detalles.sprites.front_default}" alt="${detalles.name}">
      <h4>${detalles.name}</h4>
    `;
    tarjeta.onclick = () => seleccionarPokemon(detalles);
    listaPokemon.appendChild(tarjeta);
  }
```

cargarPokemon(pagina)

Obtiene y muestra una lista de Pokémon en la ventana emergente:

1. Obtiene los datos desde PokeAPI dependiendo de la generación seleccionada o en modo global.
2. Realiza paginación dividiendo los Pokémon de una generación o usando el soporte nativo de la API.
3. Crea tarjetas de Pokémon con la imagen y el nombre y las agrega al contenedor listaPokemon.
4. Configura los botones de paginación (paginadorAnterior y paginadorSiguiente) según la página actual y el total de Pokémon disponibles.

Función seleccionarPokemon

PokemonApi

```
function seleccionarPokemon(pokemon) {
  const cuadro = cuadros[cuadroSeleccionado];

  // Crear un objeto de Pokémon con todos los detalles (nombre, sprite y URL)
  const pokemonDetalle = {
    id: pokemon.id,  // Asegúrate de que estás usando el ID correcto
    nombre: pokemon.name,
    sprite: pokemon.sprites.front_default,
    url: pokemon.url
  };

  // Actualiza el Pokémon en el equipo actual en el cuadro correspondiente
  equipoActual.pokemon[cuadroSeleccionado] = pokemonDetalle;  // Reemplazamos el Pokémon en el equipo

  // Mostrar la imagen y el nombre del Pokémon en el cuadro seleccionado
  cuadro.innerHTML = `
    <img src="${pokemonDetalle.sprite}" alt="${pokemonDetalle.nombre}">
    <span>${pokemonDetalle.nombre}</span>
  `;

  // Ocultar la ventana emergente
  ventanaEmergente.classList.add('hidden');
}
```

seleccionarPokemon(pokemon)

Permite al usuario elegir un Pokémon desde la lista:

1. Asocia un Pokémon seleccionado con un cuadro (cuadroSeleccionado).
2. Actualiza la vista del cuadro con la imagen y el nombre del Pokémon.
3. Lo agrega al equipo actual (equipoActual).
4. Cierra la ventana emergente.

Función mostrarVentanaEmergente

```
function mostrarVentanaEmergente(index) {
  cuadroSeleccionado = index;
  paginaActual = 1;
  cargarPokemon(paginaActual);
  ventanaEmergente.classList.remove('hidden');
}
```

Muestra la ventana emergente y carga los Pokémon correspondientes para seleccionar.

Función genererEquipoAleatorio

PokemonApi

```
async function generarEquipoAleatorio() {
  const respuesta = await fetch('https://pokeapi.co/api/v2/pokemon?limit=898');
  const datos = (await respuesta.json()).results.sort(() => 0.5 - Math.random()).slice(0, 6);

  equipoActual = new Equipo(Date.now()); // Reiniciar el equipo con un nuevo ID único

  datos.forEach(async (pokemon, index) => {
    const detalles = await obtenerDetallesPokemon(pokemon.url);
    const pokemonInstancia = new Pokemon(detalles.id, detalles.name, detalles.sprites.front_default, pokemon.url);

    // Mostrar Pokémon en los cuadros
    cuadros[index].innerHTML = `
      <img src="${pokemonInstancia.sprite}" alt="${pokemonInstancia.nombre}">
      <span>${pokemonInstancia.nombre}</span>
    `;

    // Agregar Pokémon al equipo actual
    equipoActual.agregarPokemon(pokemonInstancia);
  });
}
```

generarEquipoAleatorio()

Crea un equipo aleatorio:

1. Obtiene una lista global de Pokémon desde PokeAPI y selecciona 6 al azar.
2. Muestra estos Pokémon en los cuadros.
3. Crea un nuevo equipo con estos Pokémon.

limpiarEquipo()

```
function limpiarEquipo() {
  cuadros.forEach(c => (c.innerHTML = 'Seleccionar'));
  equipoActual = new Equipo(Date.now()); // Crear un nuevo equipo con ID único
}
```

Limpia los cuadros y reinicia el equipo actua

guardarEquipo()

```
function guardarEquipo() {
  const nombreEquipo = document.getElementById('nombreEquipo').value.trim();

  if (!nombreEquipo) {
    alert('Por favor, asigna un nombre al equipo antes de guardarlo.');
    return;
  }
  if (!equipoActual.esValido()) {
    alert('Tu equipo debe tener 6 Pokémon antes de guardarlo.');
    return;
  }
  // Guardamos los detalles completos de cada Pokémon (nombre, sprite y URL) junto con el ID del equipo
  const equipoParaGuardar = {
    id: equipoActual.id,  // Guardamos el ID único del equipo
    nombre: nombreEquipo,
    pokemons: equipoActual.pokemon.map(pokemon => ({
      id: pokemon.id,  // Guardamos el ID del Pokémon
      nombre: pokemon.nombre,
      sprite: pokemon.sprite,
      url: pokemon.url
    }))
  };
```

Guarda el equipo actual en el almacenamiento local (localStorage):

1. Valida que el equipo tenga 6 Pokémon y que se le haya asignado un nombre.
2. Convierte el equipo actual en un objeto serializable.
3. Lo guarda en localStorage.
4. Limpia el equipo y actualiza la lista de equipos guardados

mostrarEquiposGuardados()

```
function mostrarEquiposGuardados() {
  const listaEquipos = document.getElementById('listaEquipos');
  listaEquipos.innerHTML = '';  // Limpiar la lista antes de mostrar los equipos

  const equiposGuardados = JSON.parse(localStorage.getItem('equiposPokemon')) || [];

  if (equiposGuardados.length === 0) {
    listaEquipos.innerHTML = '<p>No tienes equipos guardados.</p>';
    return;
  }
```

Muestra los equipos guardados en localStorage:

1. Recupera los equipos guardados y los muestra en una lista.
2. Cada equipo tiene botones para editar o elimina

```
  equiposGuardados.forEach((equipo, index) => {
    const item = document.createElement('li');
    item.innerHTML = `
      <h4>${equipo.nombre}</h4>
      <div>
        ${equipo.pokemons.map(pokemon => `
          <img src="${pokemon.sprite}" alt="${pokemon.nombre}" title="${pokemon.nombre}">
        `).join('')}
      </div>
      <button id="editarBtn${equipo.id}">Editar</button>
      <button id="eliminarBtn${equipo.id}">Eliminar</button>
    `;
    listaEquipos.appendChild(item);
    // Añadir los eventos para los botones de editar y eliminar
    document.getElementById(`editarBtn${equipo.id}`).addEventListener('click', () => editarEquipo(equipo.id));
    document.getElementById(`eliminarBtn${equipo.id}`).addEventListener('click', () => eliminarEquipo(equipo.id));
  });
  console.log(equiposGuardados);
}
```

editarEquipo(id)

```
function editarEquipo(id) {
  const equiposGuardados = JSON.parse(localStorage.getItem('equiposPokemon')) || [];
  const equipo = equiposGuardados.find(equipo => equipo.id === id);

  // Asignamos el nombre del equipo al input de nombre
  document.getElementById('nombreEquipo').value = equipo.nombre;

  // Restauramos las imágenes y nombres de los Pokémon
  equipo.pokemons.forEach((pokemon, i) => {
    cuadros[i].innerHTML = `
      <img src="${pokemon.sprite}" alt="${pokemon.nombre}">
      <span>${pokemon.nombre}</span>
    `;
  });

  // Creamos un nuevo equipo con los Pokémon editados
  equipoActual = new Equipo(equipo.id, equipo.nombre);
  equipo.pokemons.forEach(pokemon => equipoActual.agregarPokemon(pokemon));

  // Actualizamos el botón para guardar cambios
  document.getElementById('guardarEquipo').textContent = 'Guardar cambios';
  document.getElementById('guardarEquipo').onclick = () => guardarCambiosEquipo(id);
}
```

Permite cargar un equipo guardado para modificarlo:

1. Carga los datos del equipo seleccionado.
2. Rellena los cuadros con los Pokémon del equipo.
3. Cambia el botón de guardar para que guarde los cambios en lugar de crear un nuevo equipo.

guardarCambiosEquipo(index)

```
function guardarCambiosEquipo(index) {
  const nombreEquipo = document.getElementById('nombreEquipo').value.trim();
  if (!nombreEquipo) {
    alert('Por favor, asigna un nombre al equipo antes de guardarlo.');
    return;
  }
  if (!equipoActual.esValido()) {
    alert('Tu equipo debe tener 6 Pokémon antes de guardarlo.');
    return;
  }
  const equiposGuardados = JSON.parse(localStorage.getItem('equiposPokemon')) || [];
  // Encontrar el equipo original por su id y actualizarlo
  const equipoEditado = equiposGuardados.find(equipo => equipo.id === equiposGuardados[index].id);
  // Actualizamos el equipo editado con el nuevo nombre y Pokémon
  equipoEditado.nombre = nombreEquipo;
  equipoEditado.pokemons = equipoActual.pokemon;
  // Guardar el equipo editado en el localStorage
  localStorage.setItem('equiposPokemon', JSON.stringify(equiposGuardados));
  alert(`Equipo "${nombreEquipo}" actualizado con éxito.`);
  limpiarEquipo(); // Limpiar los cuadros
  mostrarEquiposGuardados(); // Volver a mostrar los equipos actualizados
  // Restaurar el texto del botón a 'Guardar Equipo'
  document.getElementById('guardarEquipo').textContent = 'Guardar Equipo';
  document.getElementById('guardarEquipo').onclick = guardarEquipo;
}
```

Guarda los cambios realizados en un equipo existente.

eliminarEquipo(id)

```
function eliminarEquipo(id) {
  const equiposGuardados = JSON.parse(localStorage.getItem('equiposPokemon')) || [];
  const index = equiposGuardados.findIndex(equipo => equipo.id === id);
  equiposGuardados.splice(index, 1); // Eliminar el equipo seleccionado
  localStorage.setItem('equiposPokemon', JSON.stringify(equiposGuardados));

  alert('Equipo eliminado con éxito.');
  mostrarEquiposGuardados(); // Actualizar la lista de equipos guardados
}
```

Elimina un equipo guardado:

1. Lo busca en localStorage.
2. Lo elimina y actualiza la lista de equipos guardados.

```
// Inicialización
document.getElementById('guardarEquipo').addEventListener('click', guardarEquipo);
document.getElementById('limpiarEquipo').addEventListener('click', limpiarEquipo);
document.getElementById('generarAleatorio').addEventListener('click', generarEquipoAleatorio);
document.addEventListener('DOMContentLoaded', mostrarEquiposGuardados);

cuadros.forEach((cuadro, index) => cuadro.addEventListener('click', () => mostrarVentanaEmergente(index)));
cerrarVentanaEmergenteBtn.addEventListener('click', () => ventanaEmergente.classList.add('hidden'));
```

Pokemon.js

```
export class Pokemon {
  constructor(id, nombre, sprite, url) {
    this.id = id;  // ID único
    this.nombre = nombre;
    this.sprite = sprite;
    this.url = url;
  }
  static async crearDesdeAPI(datos) {
    const { name, url } = datos;
    const detalles = await fetch(url).then(res => res.json());
    const id = detalles.id;  // ID del Pokémon desde la API
    return new Pokemon(id, name, detalles.sprites.front_default, url);
  }
}
```

Método estático crearDesdeAPI(datos):

- Recibe un objeto datos con información básica de un Pokémon, como su name y url para detalles.
- Realiza una consulta a la API para obtener datos más detallados (como su id y sprite).
- Devuelve una instancia de Pokemon con toda la información necesaria.

```
export class Equipo {
  constructor(id, nombre = '') {
    this.id = id;  // ID único para el equipo
    this.nombre = nombre;
    this.pokemon = [];
  }
  agregarPokemon(pokemon) {
    if (this.pokemon.length < 6) {
      this.pokemon.push(pokemon);
    }
  }

  esValido() {
    return this.pokemon.length === 6;
  }
}
```

agregarPokemon(pokemon):

- Añade un objeto Pokemon al equipo, siempre y cuando haya espacio (máximo 6 Pokémon).

esValido():

- Verifica si el equipo tiene exactamente 6 Pokémon.
- Devuelve true si el equipo está completo, de lo contrario false.

Relación entre las clases

- La clase Pokemon representa a un Pokémon individual con sus detalles.
- La clase Equipo organiza varios objetos Pokemon en un grupo que puede ser gestionado como una unidad (por ejemplo, un equipo de batalla).
- Puedes usar crearDesdeAPI para generar instancias de Pokemon desde los datos de la API y luego añadirlos al equipo mediante agregarPokemon.

Index.html

```
<!-- Header -->
<header>
  <img  src="./images/charizard.gif">
  <a href="index.html"></a><h1>Equipos Pokemon</h1>
</header>
```

```
<!-- Main -->
<main>
  <!-- Sección para crear equipos -->
  <section id="crear-equipo">
    <h2>Crear Equipo</h2>
    <input type="text" id="nombreEquipo" placeholder="Nombre del equipo" />
    <div id="opciones">
      <label for="filtro-generacion">Generación:</label>
      <select id="filtro-generacion">
        <option value="">Todas</option>
        <option value="1">Generación 1</option>
        <option value="2">Generación 2</option>
        <option value="3">Generación 3</option>
        <option value="4">Generación 4</option>
        <option value="5">Generación 5</option>
        <option value="6">Generación 6</option>
        <option value="7">Generación 7</option>
        <option value="8">Generación 8</option>
      </select>
      <button id="generarAleatorio">Generar Equipo Aleatorio</button>
      <button id="limpiarEquipo">Limpiar Equipo</button>
      <button id="guardarEquipo">Guardar Equipo</button>
    </div>
```

```
    <div class="cuadros">
      <!-- 6 cuadros para los Pokémon del equipo -->
      <div class="cuadro" data-index="0">Seleccionar</div>
      <div class="cuadro" data-index="1">Seleccionar</div>
      <div class="cuadro" data-index="2">Seleccionar</div>
      <div class="cuadro" data-index="3">Seleccionar</div>
      <div class="cuadro" data-index="4">Seleccionar</div>
      <div class="cuadro" data-index="5">Seleccionar</div>
    </div>
  </section>

  <!-- ventanaEmergente para seleccionar Pokémon -->
  <div id="ventanaEmergente" class="hidden">
    <div class="ventanaEmergente-content">
      <h3>Selecciona un Pokémon</h3>
      <div id="listaPokemon" class="lista-pokemon"></div>
      <div id="paginador">
        <button id="anterior" disabled>Anterior</button>
        <button id="siguiente">Siguiente</button>
      </div>
      <button id="cerrarVentanaEmergente">Cerrar</button>
    </div>
  </div>

  <!-- Sección de equipos creados -->
  <section id="equipos-creados">
    <h2>Equipos Creados</h2>
    <ul id="listaEquipos"></ul>
  </section>
</main>
```

```
<!-- Footer -->
<footer>
  <p>Desarrollado por - Basado en PokéAPI</p>
</footer>
```

Estilos.css

```
main {
    background: url(../images/fondo.jpg);
    background-attachment: fixed;
    background-position: center;
    background-size: cover;
    background-repeat: no-repeat;
    flex: 1;
    padding: 20px;
    display: flex;
    flex-direction: column;
    gap: 20px;
}

section {
    background: rgba(255, 255, 255, 0.685);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
}
```

```css
#ventanaEmergente {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 1000;
}

#ventanaEmergente.hidden {
    display: none;
}

.ventanaEmergente-content {
    background: white;
    padding: 20px;
    border-radius: 10px;
    width: 80%;
    max-width: 800px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
    text-align: center;
}
```

```css
/* Estilo para la paginación */
#paginador {
    display: flex;
    justify-content: space-between;
    margin: 10px 0;
}

#paginador button {
    padding: 10px 20px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

#paginador button:disabled {
    background-color: #ccc;
    cursor: not-allowed;
}
```

```css
.lista-pokemon {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
    gap: 10px;
}

.listaEquipos li {
    margin-bottom: 20px;
}
```

Comparador

funcionesComparador.js

mostrarPokemons()

```
'use strict'
/**
 * Función para mostrar los Pokémon comparados
 */
async function mostrarPokemons(listaPokemon, pokemonId1, pokemonId2) {
    // Buscar los Pokémon por ID
    const pokemon1 = listaPokemon.find(pokemon => pokemon.id == pokemonId1);
    const pokemon2 = listaPokemon.find(pokemon => pokemon.id == pokemonId2);
```

```
    // Mostrar los detalles de ambos Pokémon en la consola
    console.log(pokemon1, pokemon2);
```

```
// Verificar si se encontró el primer Pokémon
const pokemonImagen1 = document.getElementById('imagenPokemon1');
const quitarPokemon1 = document.getElementById('quitarPokemon1');
if (pokemon1) {
    if (pokemonImagen1) {
        pokemonImagen1.innerHTML = `
            <div class="pokemon-name">${pokemon1.name}</div>
            <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/${pokemon1.id}.pr
        `;
        pokemonImagen1.classList.remove('hidden');
        quitarPokemon1.classList.remove('hidden');
    } else {
        console.error('No se encontró el elemento con ID "imagenPokemon1"');
    }
} else {
    if (pokemonImagen1) {
        pokemonImagen1.classList.add('hidden');
        quitarPokemon1.classList.add('hidden');
    }
    console.error(`No se encontró el Pokémon con ID ${pokemonId1}`);
}
```

PokemonApi

```
// Verificar si se encontró el segundo Pokémon
const pokemonImagen2 = document.getElementById('imagenPokemon2');
const quitarPokemon2 = document.getElementById('quitarPokemon2');
if (pokemon2) {
    if (pokemonImagen2) {
        pokemonImagen2.innerHTML = `
            <div class="pokemon-name">${pokemon2.name}</div>
            <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemo
        `;
        pokemonImagen2.classList.remove('hidden');
        quitarPokemon2.classList.remove('hidden');
    } else {
        console.error('No se encontró el elemento con ID "imagenPokemon2"');
    }
} else {
    if (pokemonImagen2) {
        pokemonImagen2.classList.add('hidden');
        quitarPokemon2.classList.add('hidden');
    }
    console.error(`No se encontró el Pokémon con ID ${pokemonId2}`);
}
```

```
if (pokemon1 && pokemon2) {
    const phPokemons = document.getElementById('hpPokemons');
    const atPokemons = document.getElementById('atPokemons');
    const dfPokemons = document.getElementById('dfPokemons');
    const vlPokemons = document.getElementById('vlPokemons');
    const atEPokemons = document.getElementById('atEPokemons');
    const dfEPokemons = document.getElementById('dfEPokemons');
    // Función auxiliar para comparar valores y agregar clase de color
    const compararConEstilo = (valor1, valor2) => {
        if (valor1 > valor2) {
            return `<span style="background-color: green; color: white; padding: 2px;">${valor1}</span> vs
                <span style="background-color: red; color: white; padding: 2px;">${valor2}</span>`;
        } else if (valor1 < valor2) {
            return `<span style="background-color: red; color: white; padding: 2px;">${valor1}</span> vs
                <span style="background-color: green; color: white; padding: 2px;">${valor2}</span>`;
        }
        return `<span style="background-color: gray; color: white; padding: 2px;">${valor1}</span> vs
            <span style="background-color: gray; color: white; padding: 2px;">${valor2}</span>`;
    };
```

```
// Actualizar contenido con estilos
phPokemons.innerHTML = compararConEstilo(pokemon1.stats[0].Ps, pokemon2.stats[0].Ps);
atPokemons.innerHTML = compararConEstilo(pokemon1.stats[1].Ataque, pokemon2.stats[1].Ataque);
dfPokemons.innerHTML = compararConEstilo(pokemon1.stats[2].Defensa, pokemon2.stats[2].Defensa);
vlPokemons.innerHTML = compararConEstilo(pokemon1.stats[5].Velocidad, pokemon2.stats[5].Velocidad);
atEPokemons.innerHTML = compararConEstilo(pokemon1.stats[3].Ataque_Especial, pokemon2.stats[3].Ataque_Especial);
dfEPokemons.innerHTML = compararConEstilo(pokemon1.stats[4].Defensa_Especial, pokemon2.stats[4].Defensa_Especial);
```

mostrarListaPaginada()

```
async function mostrarListaPaginada(pokemonSection) {
    const listaPokemon = JSON.parse(localStorage.getItem('pokemons')) || [];
    const section = document.querySelector(`section.${pokemonSection}`);
    const container = document.getElementById(`pokemon-list-container-${pokemonSection}`);
    container.innerHTML = ''; // Limpiar contenido previo

    const searchInput = document.createElement('input');
    searchInput.type = 'text';
    searchInput.placeholder = 'Buscar por nombre';
    searchInput.addEventListener('input', () => filtrarPokemons(listaPokemon, searchInput.value, pokemonSection));
    container.appendChild(searchInput);

    const lista = document.createElement('div');
    lista.id = `pokemon-list-${pokemonSection}`;
    container.appendChild(lista);

    const paginacion = document.createElement('div');
    paginacion.className = 'paginacion';
    container.appendChild(paginacion);

    const flechaIzquierda = document.createElement('button');
    flechaIzquierda.innerHTML = '&larr;';
    flechaIzquierda.addEventListener('click', () => cambiarPagina(listaPokemon, -1, pokemonSection));
    paginacion.appendChild(flechaIzquierda);

    const flechaDerecha = document.createElement('button');
    flechaDerecha.innerHTML = '&rarr;';
    flechaDerecha.addEventListener('click', () => cambiarPagina(listaPokemon, 1, pokemonSection));
    paginacion.appendChild(flechaDerecha);

    paginarPokemons(listaPokemon, 1, pokemonSection);
}
```

- **Propósito**: Mostrar una lista paginada de Pokémon para seleccionar en una sección específica.
- **Pasos principales**:
    1. Crea un campo de búsqueda y un contenedor para la lista de Pokémon.
    2. Agrega botones para navegar entre páginas.
    3. Llama a paginarPokemons para cargar la primera página de la lista

```
let paginaActual = { pokemon1: 1, pokemon2: 1 };
let seleccionados = { pokemon1: null, pokemon2: null };
```

```
// Función para filtrar Pokémon por nombre
function filtrarPokemons(listaPokemon, query, pokemonSection) {
    const lista = document.getElementById(`pokemon-list-${pokemonSection}`);
    lista.innerHTML = '';
    const pokemonsFiltrados = listaPokemon.filter(pokemon => pokemon.name.toLowerCase().includes(query.toLowerCase()));
    paginarPokemons(pokemonsFiltrados, 1, pokemonSection);
}
```

- **Propósito**: Filtrar Pokémon por nombre en base al texto ingresado.
- **Pasos principales**:
    1. Filtra la lista de Pokémon en base a la consulta (query).

2. Llama a paginarPokemons para mostrar los resultados filtrados.

```
// Función para paginar Pokémon
function paginarPokemons(listaPokemon, pagina, pokemonSection) {
    const lista = document.getElementById(`pokemon-list-${pokemonSection}`);
    lista.innerHTML = '';
    const itemsPorPagina = 20;
    const inicio = (pagina - 1) * itemsPorPagina;
    const fin = inicio + itemsPorPagina;
    const pokemonsPaginados = listaPokemon.slice(inicio, fin);

    pokemonsPaginados.forEach(pokemon => {
        if (pokemon.id !== seleccionados.pokemon1 && pokemon.id !== seleccionados.pokemon2) {
            const item = document.createElement('div');
            item.className = 'pokemon-item';
            item.innerHTML = `<img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/$
            item.addEventListener('click', () => seleccionarPokemon(pokemon.id, pokemonSection));
            lista.appendChild(item);
        }
    });
}
```

- **Propósito**: Mostrar un subconjunto de Pokémon en la página actual.
- **Pasos principales**:

1. Divide la lista de Pokémon en bloques de 20 elementos por página.
2. Renderiza los Pokémon visibles en el contenedor de la sección (pokemonSection).
3. Excluye Pokémon ya seleccionados (pokemon1, pokemon2).

```
// Función para seleccionar Pokémon
function seleccionarPokemon(id, pokemonSection) {
    if (seleccionados.pokemon1 === id || seleccionados.pokemon2 === id) {
        console.error('No se pueden comparar Pokémon iguales');
        return;
    }

    seleccionados[pokemonSection] = id;
    mostrarPokemons(JSON.parse(localStorage.getItem('pokemons')), seleccionados.pokemon1, seleccionados.pokemon2);
    // Ocultar la lista de Pokémon seleccionada
    document.getElementById(`pokemon-list-container-${pokemonSection}`).style.display = 'none';
}
```

- **Propósito**: Seleccionar un Pokémon para compararlo.
- **Pasos principales**:

1. Verifica que el Pokémon seleccionado no esté ya en comparación.
2. Actualiza la selección en el objeto seleccionados.
3. Llama a mostrarPokemons para visualizar los detalles y comparaciones.
4. Oculta la lista de selección para evitar cambios adicionales.

quitarPokemon(pokemonSection)

```
// Función para quitar Pokémon seleccionado
function quitarPokemon(pokemonSection) {
    seleccionados[pokemonSection] = null;
    const pokemonImagen = document.getElementById(`imagenPokemon${pokemonSection === 'pokemon1' ? '1' : '2'}`);
    pokemonImagen.innerHTML = '';
    pokemonImagen.classList.add('hidden');
    document.getElementById(`quitarPokemon${pokemonSection === 'pokemon1' ? '1' : '2'}`).classList.add('hidden')
    document.getElementById(`pokemon-list-container-${pokemonSection}`).style.display = 'block';
    paginarPokemons(JSON.parse(localStorage.getItem('pokemons')), paginaActual[pokemonSection], pokemonSection);
    if (!seleccionados.pokemon1 || !seleccionados.pokemon2) {
        document.getElementById('hpPokemons').innerHTML = '';
        document.getElementById('atPokemons').innerHTML = '';
        document.getElementById('dfPokemons').innerHTML = '';
        document.getElementById('vlPokemons').innerHTML = '';
        document.getElementById('atEPokemons').innerHTML = '';
        document.getElementById('dfEPokemons').innerHTML = '';
    }
}
document.getElementById('quitarPokemon1').addEventListener('click', () => quitarPokemon('pokemon1'));
document.getElementById('quitarPokemon2').addEventListener('click', () => quitarPokemon('pokemon2'));
document.getElementById('volverMain').addEventListener('click', () => {
    window.location.href = '../PaginaInicio/index.html';
```

- **Propósito**: Eliminar la selección de un Pokémon en una sección específica.
- **Pasos principales**:

1. Limpia los elementos visuales relacionados con el Pokémon eliminado.
2. Restaura la lista de selección visible.
3. Borra las comparaciones estadísticas si falta algún Pokémon seleccionado

iniciar()

```
async function iniciar() {
    let listaPokemon1 = JSON.parse(localStorage.getItem('pokemons')) || [];

    if (listaPokemon1.length > 0) {
        console.log('Pokémon cargados en localStorage');
        mostrarListaPaginada('pokemon1');
        mostrarListaPaginada('pokemon2');
    } else {
        console.error('No se encontraron Pokémon en localStorage');
    }
}
```

- **Propósito**: Inicializar la aplicación.
- **Pasos principales**:

1. Verifica si hay una lista de Pokémon almacenada en localStorage.

2. Si hay datos, llama a mostrarListaPaginada para ambas secciones de Pokémon.
3. Si no hay datos, muestra un mensaje de error.

Comparador.html

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="estilosComparador.css">
    <title>Comparador de Pokémon</title>
    <script src="funcionesComparador.js" type="module"></script>
</head>
```

```
<main>
    <section class="pokemon1">
        <div id="imagenPokemon1"></div>
        <button id="quitarPokemon1">Quitar Pokémon Izquierda</button>
        <div id="pokemon-list-container-pokemon1" class="pokemon-list-container"></div>
    </section>
    <section class="estadisticas">
        <h1>COMPARATIVA ESTADISTICAS</h1>
        <div class="estadisticas">
            <div class="table">
                <div class="fila">
                    <div>HP</div>
                </div>
                <div class="fila" id="hpPokemons"></div>
                <div class="fila">
                    <div>Ataque</div>
                </div>
                <div class="fila" id="atPokemons"></div>
                <div class="fila">
                    <div>Defensa</div>
                </div>
                <div class="fila" id="dfPokemons"></div>
                <div class="fila">
                    <div>Velocidad</div>
                </div>
                <div class="fila" id="vlPokemons"></div>
                <div class="fila">
                    <div>Ataque Especial</div>
                </div>
                <div class="fila" id="atEPokemons"></div>
                <div class="fila">
                    <div>Defensa Especial</div>
                </div>
                <div class="fila" id="dfEPokemons"></div>
            </div>
```

```
</section>
<section class="pokemon2">
    <div id="imagenPokemon2"></div>
    <button id="quitarPokemon2">Quitar Pokémon Derecha</button>
    <div id="pokemon-list-container-pokemon2" class="pokemon-list-container"></div>
</section>
```

Combate 1vs1

LogicaCombate.js

calcularDanio(atacante, defensor)

```
function calcularDanno(ataqueAtacante, defensaDefensor) {
    danno = (ataqueAtacante * generaAleatorio(0.2, 1) - defensaDefensor* generaAleatorio(0.4, 1)) + 2
    // Tabla de multiplicadores de daño según los tipos
    const efectividadTipos = {
        'normal': { 'rock': 2, 'ghost': 0, 'steel': 0.5 },
        'fire': { 'grass': 2, 'fire': 0.5, 'water': 0.5, 'rock': 2, 'bug': 2, 'ghost': 1 },
        'water': { 'fire': 2, 'water': 0.5, 'electric': 2, 'grass': 0.5, 'ghost': 1 },
        'electric': { 'water': 2, 'electric': 0.5, 'ground': 0, 'flying': 2, 'ghost': 1 },
        'grass': { 'water': 2, 'fire': 0.5, 'grass': 0.5, 'bug': 2, 'flying': 0.5, 'ghost': 1 },
        'ice': { 'fire': 0.5, 'water': 0.5, 'ice': 0.5, 'steel': 2, 'ghost': 1 },
        'fighting': { 'normal': 2, 'rock': 2, 'steel': 2, 'ghost': 0, 'flying': 0.5, 'psychic': 0.5, 'fairy': 0.5 },
        'poison': { 'grass': 2, 'poison': 0.5, 'ghost': 1, 'ground': 0.5, 'steel': 0 },
        'ground': { 'fire': 2, 'electric': 2, 'water': 0.5, 'grass': 0.5, 'ice': 2, 'flying': 0 },
        'flying': { 'fighting': 2, 'bug': 2, 'grass': 2, 'electric': 0.5, 'steel': 1, 'ghost': 1 },
        'psychic': { 'fighting': 2, 'poison': 2, 'psychic': 0.5, 'ghost': 1 },
        'bug': { 'grass': 2, 'fire': 0.5, 'fighting': 0.5, 'ghost': 1 },
        'rock': { 'fire': 2, 'electric': 1, 'grass': 1, 'ice': 2, 'fighting': 0.5, 'bug': 2, 'ghost': 1 },
        'ghost': { 'ghost': 2, 'normal': 0, 'psychic': 2, 'dark': 1 },
        'dragon': { 'dragon': 2, 'fairy': 0, 'steel': 0.5, 'ghost': 1 },
        'dark': { 'psychic': 2, 'ghost': 2, 'fairy': 0.5, 'fighting': 0.5, 'dark': 1 },
        'steel': { 'rock': 2, 'bug': 2, 'grass': 1, 'fire': 0.5, 'steel': 0.5, 'ghost': 1 },
        'fairy': { 'fighting': 2, 'dragon': 2, 'dark': 2, 'steel': 0.5, 'ghost': 1 }
    };
    let efectividad = 1;
    if (efectividadTipos[atacante.type] && efectividadTipos[atacante.type][defensor.type]) {
        efectividad = efectividadTipos[atacante.type][defensor.type];
    }
    danno *= efectividad;
    return Math.max(danno, 1);
}
```

- **Propósito**: Calcular el daño que realiza cada Pokemon cada vez que ataca.
- **Pasos principales**:
    1. Usa las estadísticas de ataque del atacante y de defensa del defensor para generar el daño introduciendo un factor aleatorio.
    2. Modifica la efectividad del ataque según la tabla de tipos
    3. Establece un daño mínimo de 1.

```
// Función principal para el combate
async function combate(pokemon1, pokemon2) {
    if (!selectedPokemon || !enemyPokemon) {
        alert('Se deben seleccionar ambos Pokémon.');
        return;
    }
    while (pokemon1.hp > 0 && pokemon2.hp > 0) {
        if (compararVelocidades(pokemon1.velocidad, pokemon2.velocidad)) {
            realizarAtaque(pokemon1, pokemon2)
            if (pokemon2.hp > 0) {
                realizarAtaque(pokemon2, pokemon1)
            }
        } else {
            realizarAtaque(pokemon2, pokemon1)
            if (pokemon1.hp > 0) {
                realizarAtaque(pokemon1, pokemon2)
            }
        }
    }
    if (pokemon1.hp > 0) {
        agregarParrafoBattleLog(`${pokemon1.nombre} ganó el combate!`)
    } else {
        agregarParrafoBattleLog(`${pokemon2.nombre} ganó el combate!`)
    }
}
```

```
// Función para seleccionar el siguiente Pokémon del equipo humano
function seleccionarPokemonHumano(equipoHumano) {
    // Aquí puedes implementar la lógica para permitir que el humano seleccione el próximo Pokémon
    // Por ejemplo, podrías mostrar un prompt o una interfaz gráfica para la selección
    // Para fines de este ejemplo, simplemente tomaremos el primer Pokémon con salud
    for (let i = 0; i < equipoHumano.length; i++) {
        if (equipoHumano[i].hp > 0) {
            return equipoHumano[i];
        }
    }
    return null;
}
```

```
// Función para seleccionar aleatoriamente el siguiente Pokémon de la máquina
function seleccionarPokemonMaquina(equipoMaquina) {
    let pokemonConSalud = equipoMaquina.filter(pokemon => pokemon.hp > 0);
    if (pokemonConSalud.length > 0) {
        let indiceAleatorio = Math.floor(generaAleatorio(0, pokemonConSalud.length - 1));
        return pokemonConSalud[indiceAleatorio];
    }
    return null;
}
```

```
// Función para el combate por equipos
function combatePorEquipos(equipoHumano, equipoMaquina) {
    let pokemonHumano = seleccionarPokemonHumano(equipoHumano);
    let pokemonMaquina = seleccionarPokemonMaquina(equipoMaquina);

    while (pokemonHumano && pokemonMaquina) {
        combate(pokemonHumano, pokemonMaquina);
        if (pokemonHumano.hp <= 0) {
            agregarParrafoBattleLog(`${pokemonHumano.nombre} ha sido derrotado.`);
            pokemonHumano = seleccionarPokemonHumano(equipoHumano);
        }
        if (pokemonMaquina.hp <= 0) {
            agregarParrafoBattleLog(`${pokemonMaquina.nombre} ha sido derrotado.`);
            pokemonMaquina = seleccionarPokemonMaquina(equipoMaquina);
        }
    }

    if (equipoHumano.every(pokemon => pokemon.hp <= 0)) {
        agregarParrafoBattleLog("¡La máquina ha ganado el combate por equipos!");
    } else if (equipoMaquina.every(pokemon => pokemon.hp <= 0)) {
        agregarParrafoBattleLog("¡Has ganado el combate por equipos!");
    }
}
```

```
// Función para agregar un <p> con el texto proporcionado
function agregarParrafoBattleLog(texto, tiempoEspera = 1000) {
    setTimeout(() => {
        var battleLog = document.getElementById("battle-log");
        if (battleLog) {
            var nuevoParrafo = document.createElement("p");
            nuevoParrafo.innerText = texto;
            battleLog.appendChild(nuevoParrafo);
        } else {
            console.error("No se encontró el elemento con id 'battle-log'");
        }
    }, tiempoEspera);
}
```

```
// Función para borrar todos los elementos hijos de battle-log
function borrarBattleLog() {
    var battleLog = document.getElementById("battle-log");
    if (battleLog) {
        battleLog.innerHTML = '';
    } else {
        console.error("No se encontró el elemento con id 'battle-log'");
    }
}

let selectedPokemon = null;
let enemyPokemon = null;
```

PokemonApi

```
// Inicializa el juego
document.addEventListener('DOMContentLoaded', () => {
    document.getElementById('select-player-pokemon').addEventListener('click', () => mostrarSeleccionPokemon(false));
    document.getElementById('select-enemy-pokemon').addEventListener('click', () => mostrarSeleccionPokemon(true));
    document.getElementById('reset-selection').addEventListener('click', reiniciarSelecciones);
    document.getElementById('start-battle').addEventListener('click', iniciarBatalla);
});

function reiniciarSelecciones() {
    selectedPokemon = null;
    enemyPokemon = null;

    document.getElementById('player-name').textContent = 'Ningún Pokémon seleccionado';
    document.getElementById('enemy-name').textContent = 'Ningún Pokémon seleccionado';
    document.getElementById('player-image').style.display = 'none';
    document.getElementById('enemy-image').style.display = 'none';
}
```

```
async function mostrarSeleccionPokemon(esEnemigo) {
    const modal = document.createElement('div');
    modal.className = 'pokemon-modal';

    const grid = document.createElement('div');
    grid.className = 'pokemon-grid';

    for (const pokemon of pokemonList) {
        const card = document.createElement('div');
        card.className = 'pokemon-card';
        const data = await obtenerPokemon(pokemon);
        card.innerHTML = `
            <img src="${uriSprites.uri}${data.id}${uriSprites.extension}" alt="${pokemon}">
            <h3>${pokemon}</h3>
        `;
        card.addEventListener('click', () => {
            seleccionaPokemon(data, esEnemigo);
            modal.remove();
        });
        grid.appendChild(card);
    }
    modal.appendChild(grid);
    document.body.appendChild(modal);
}
```

- **Propósito**: Mostrar las tarjetas de los Pokemon para seleccionar el Pokemon propio y el del rival.
- **Pasos principales**:
    1. Construye el modal de Pokemon
    2. Recorre la lista de Pokemon (obtenida del localStorage) y los agrega a las tarjetas del modal
    3. esEnemigo sirve para discriminar (bool) si el Pokemon seleccionado es del rival o no. La función se usa tanto en el botón de selección del Pokemon propio como del Pokemon enemigo.

PokemonApi

```
function seleccionarPokemon(pokemon) {
    const cuadro = cuadros[cuadroSeleccionado];

    // Crear un objeto de Pokémon con todos los detalles (nombre, sprite y URL)
    const pokemonDetalle = {
      id: pokemon.id,  // Asegúrate de que estás usando el ID correcto
      nombre: pokemon.name,
      sprite: pokemon.sprites.front_default,
      url: pokemon.url
    };
    // Actualiza el Pokémon en el equipo actual en el cuadro correspondiente
    equipoActual.pokemon[cuadroSeleccionado] = pokemonDetalle;  // Reemplazamos el Pokémon en el equipo

    // Mostrar la imagen y el nombre del Pokémon en el cuadro seleccionado
    cuadro.innerHTML = `
      <img src="${pokemonDetalle.sprite}" alt="${pokemonDetalle.nombre}">
      <span>${pokemonDetalle.nombre}</span>
    `;
    // Ocultar la ventana emergente
    ventanaEmergente.classList.add('hidden');
}
```

```
async function obtenerPokemon(nombrePokemon) {

    const response = await fetch(`${pokeAPI}pokemon/${nombrePokemon.toLowerCase()}`);
    const data = await response.json();
    return {
        id: data.id,
        name: data.name,
        hp: data.stats[0].base_stat,
        attack: data.stats[1].base_stat,
        defense: data.stats[2].base_stat,
        speed: data.stats[5].base_stat,
        type: data.types[0].type.name
    };
}
```

```
function seleccionaPokemon(pokemon, esEnemigo) {
    if (esEnemigo) {
        enemyPokemon = pokemon;
        actualizarUIPokemon(enemyPokemon, 'enemy');
        document.getElementById('start-battle').style.display = 'block';
    } else {
        selectedPokemon = pokemon;
        actualizarUIPokemon(selectedPokemon, 'player');
        document.getElementById('select-enemy-pokemon').disabled = false;
    }

    document.getElementById('reset-selection').disabled = false;
}
```

```
function actualizarUIPokemon(pokemon, idElemento) {
    // Actualiza el nombre del Pokémon
    document.getElementById(`${idElemento}-name`).textContent = pokemon.name;

    // Actualiza la imagen del Pokémon
    const image = document.getElementById(`${idElemento}-image`);
    image.src = `${uriSprites.uri}${data.id}${uriSprites.extension}`;
    image.style.display = 'block'; // Asegúrate de que la imagen esté visible

    // Actualiza la barra de HP
    const barraHp = document.getElementById(`${idElemento}-hp`).querySelector('.hp-bar');
    barraHp.style.width = '100%'; // Establece la barra a 100% al inicio
}
```

- **Propósito**: Actualizar la barra de vida del Pokemon en función del daño recibido.
- **Pasos principales**:
    1. Resta a la barra de vida el daño recibido por el Pokemon.

```
function mostrarVentanaEmergente(index) {
    cuadroSeleccionado = index; // Seleccionar el cuadro actual
    paginaActual = 1; // Reiniciar siempre a la primera página al abrir
    cargarPokemon(paginaActual); // Cargar Pokémon de la primera página
    ventanaEmergente.classList.remove('hidden'); // Mostrar la ventana emergente
}

cerrarVentanaEmergenteBtn.addEventListener('click', () => ventanaEmergente.classList.add('hidden'));
```

reiniciarSelecciones()

```
function reiniciarSelecciones() {
    selectedPokemon = null;
    enemyPokemon = null;

    document.getElementById('player-name').textContent = 'Ningún Pokémon seleccionado';
    document.getElementById('enemy-name').textContent = 'Ningún Pokémon seleccionado';
    document.getElementById('player-image').style.display = 'none';
    document.getElementById('enemy-image').style.display = 'none';


}
```

Propósito: Limpiar la selección de Pokémon a combatir.

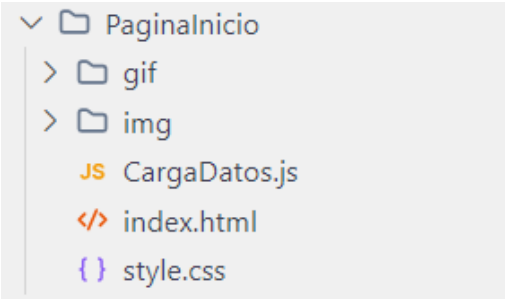
iniciarBatalla()

PokemonApi

```
async function iniciarBatalla() {
    if (!selectedPokemon || !enemyPokemon) {
        alert('Ambos Pokémon deben ser seleccionados.');
        return;
    }

    let atacante = selectedPokemon.speed > enemyPokemon.speed ? selectedPokemon : enemyPokemon;
    let defensor = atacante === selectedPokemon ? enemyPokemon : selectedPokemon;

    const logContainer = document.getElementById('battle-log');
    logContainer.innerHTML = '<h3>Registros de batalla</h3>';

    while (selectedPokemon.hp > 0 && enemyPokemon.hp > 0) {
        const danio = calcularDanio(atacante, defensor);
        defensor.hp -= danio;

        const barraHpDefensor = document.getElementById(`${defensor === selectedPokemon ? 'player' : 'enemy'}-hp`).querySelector('.hp-bar');
        barraHpDefensor.style.width = `${Math.max((defensor.hp / 100) * 100, 0)}%`;

        const entradaLog = document.createElement('p');
        entradaLog.textContent = `${atacante.name} le causó ${danio} de daño a ${defensor.name}`;
        logContainer.appendChild(entradaLog);

        const vidaRestante = document.createElement('p');
        vidaRestante.textContent = `${defensor.name} tiene ${Math.max(defensor.hp, 0)} de vida restante.`;
        logContainer.appendChild(vidaRestante);

        logContainer.scrollTop = logContainer.scrollHeight;

        await new Promise(resolve => setTimeout(resolve, 1000));

        [atacante, defensor] = [defensor, atacante];
    }

    const mainDiv = document.querySelector('main');
    const ganador = selectedPokemon.hp > 0 ? selectedPokemon : enemyPokemon;
    const entradaLog = document.createElement('div');
    entradaLog.classList.add('entradaLog');
    entradaLog.textContent = `${ganador.name} gana la batalla!`;
    mainDiv.appendChild(entradaLog);
}
```

- **Propósito**: Realización de la batalla.
- **Pasos principales**:
    1. Controla que ambos Pokemon sean seleccionados
    2. Mientras ambos Pokemon sea superior a 0 el bucle principal se ejecuta.
    3. Durante el bucle se calcula el daño y se actualiza la barra de salud y el log de combate.
    4. Cuando se acaba el bucle se informa de quién ganó la batalla.

PaginaInicio



CargaDatos.js

Funciones para cargar la api

```
/**
 * Funcion que obtiene los tipos
 *
 */
async function cargarTipo() {
    try {
        const response = await fetch("https://pokeapi.co/api/v2/type/?offset=0&limit=21");
        if (!response.ok) {
            throw new Error(`Error HTTP: ${response.status}`);
        }
        const habilidades = await response.json();
        return habilidades;
    } catch (error) {
        console.error(error);
    }
}
```

```
/**
 * Funciones que obtiene las generaciones
 * @
 */
async function cargarGeneraciones() {
    try {
        const response = await fetch("https://pokeapi.co/api/v2/generation/?offset=0&limit=50");
        if (!response.ok) {
            throw new Error(`Error HTTP: ${response.status}`);
        }
        const habilidades = await response.json();
        return habilidades;
    } catch (error) {
        console.error(error);
    }
}
```

```
/**Funcion que obtiene todas las regiones */
async function cargarRegion() {
    try {
        const response = await fetch("https://pokeapi.co/api/v2/region?offset=0&limit=0");
        if (!response.ok) {
            throw new Error(`Error HTTP: ${response.status}`);
        }
        const habilidades = await response.json();
        return habilidades;
    } catch (error) {
        console.error(error);
    }
}
```

PokemonApi

```
/**
 * Funcion que obtiene los pokemons
 */
async function cargarPokemon(offset, limit) {
    try {
        const response = await fetch(`https://pokeapi.co/api/v2/pokemon/?offset=${offset}&limit=${limit}`);

        if (!response.ok) {
            throw new Error(`Error HTTP: ${response.status}`);
        }

        const habilidades = await response.json();

        return habilidades;

    } catch (error) {
        console.error(error);
    }
}
```

Index.html

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Poketournamet</title>
    <link rel="stylesheet" href="style.css">
    <script type="module" src="CargaDatos.js"></script>
</head>
```

```
<header>
    <img  src="gif/charizard.gif">
    <h1>Poketournamet</h1>
    <img src="img/pokeball.png">
</header>
```

```
<main>
    <section id="menu">
            <ul>
                <li><a href="../Pokedex/index.html">Pokedex</a></li>
                <li><a href="../Equipos/index.html">Equipos</a></li>
                <li><a href="../Comparador/ComparadorPokemons.html">Comparador</a></li>
            </ul>
    </section>
    <section id="botones">
        <section id="elementoBotones">
            <button id="" ><a href="../Combate/Combate.html"">Combate 1 vs 1</a></button>
            <button id="" ><a href="../Combate/CombateEquipo.html"">Combate en equipos</a> </button>
        </section>
    </section>
</main>
```

style.css

Difcultades/Resumen

-En la pokedex al buscar en el buscador y tener en el filtro alguna generación buscaba en todas las generaciones y no en la generación puesta en el filtro

-Reducir el tiempo de carga de datos