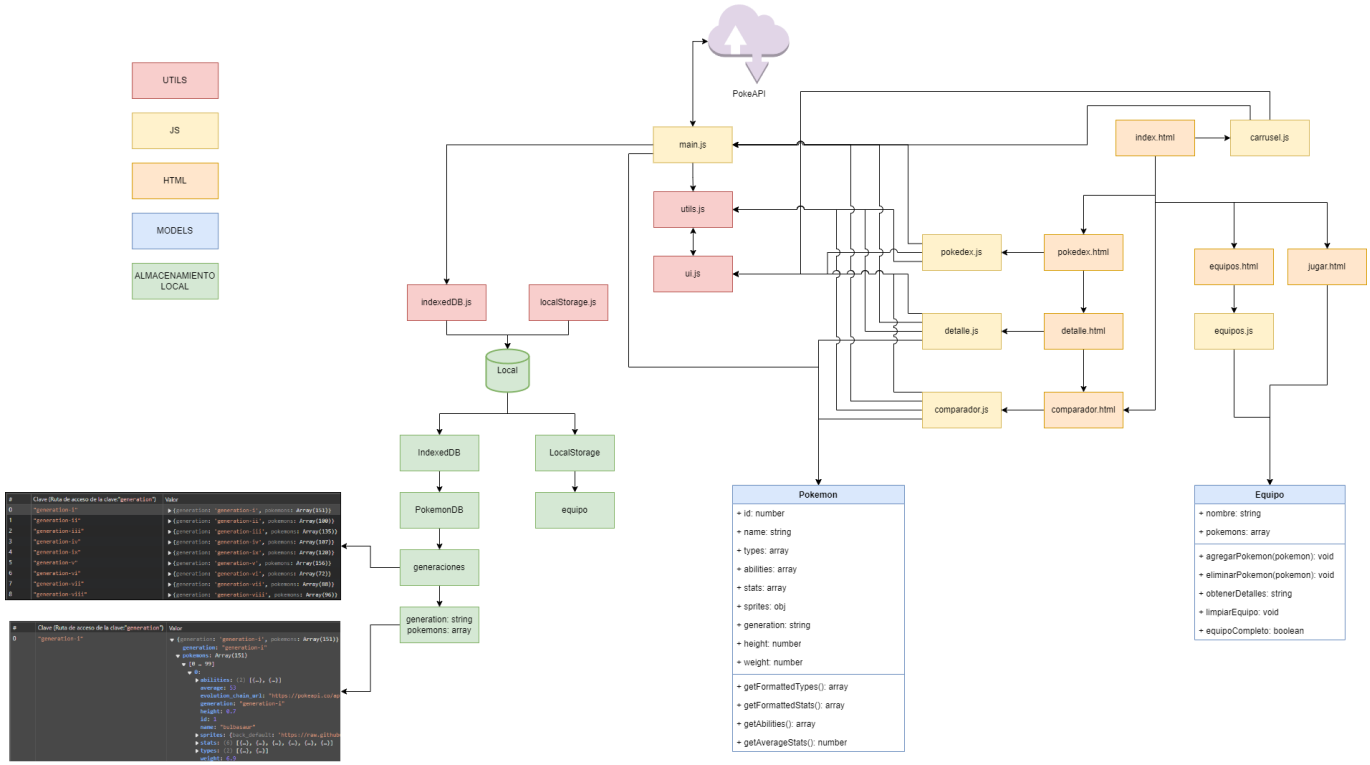
Proyecto Pokemon

Link de Github: Proyecto Pokemon

Link de la web: Web Pokemon

Arquitectura



Funcionalidades

Header y Footer

Este bloque define un encabezado con un logotipo animado que redirige a la página de inicio y un menú de navegación con enlaces a las secciones principales del sitio: Inicio, Pokédex, Comparador, Equipos y Jugar, facilitando el acceso rápido a las funcionalidades del sitio.

```
<header>
        <a href="./index.html">
            <div class="logo">
                <img src="./img/logotipo_pokemon.svg" alt="" width="150px">
                <img src="./img/logoPokeball.png" alt="Logo Pokémon"
class="animar-pokeball">
```

```
            </div>
        </a>
        <nav>
            <ul>
                <li><a href="./index.html">Inicio</a></li>
                <li><a href="./html/pokedex.html">Pokédex</a></li>
                <li><a href="./html/comparador.html">Comparador</a></li>
                <li><a href="./html/equipos.html">Equipos</a></li>
                <li><a href="./html/jugar.html">Jugar</a></li>
            </ul>
        </nav>
    </header>
```

```
Este bloque define un pie de página simple que muestra un mensaje de derechos
reservados con el año 2025 y el nombre "Pokémon Game", proporcionando un cierre
informativo para la página.

<footer>
        <div class="footer-content">
            <p>&copy; 2025 Pokémon Game. Todos los derechos reservados.</p>
        </div>
</footer>
```

Este HTML estructura una página de inicio para un sitio Pokémon, ofreciendo una bienvenida con enlace para jugar, destacando funciones clave como la Pokédex, comparador, creación de equipos y combates, mostrando una galería promocional, e incluyendo un llamado a la acción para unirse a la aventura.

index.html

```
 <main>
        <!-- Sección de bienvenida -->
        <section class="bienvenida">
            <div class="bienvenida-texto">
                <h1>¡Bienvenido al mundo de Pokémon!</h1>
                <p>Embárcate en una nueva aventura Pokémon, forma tu equipo y
realiza combates épicos.</p>
                <a href="./html/jugar.html" class="btn">¡Empezar a jugar!</a>
            </div>
            <div class="bienvenida-imagen">
                <img
src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/othe
r/official-artwork/25.png"
                    alt="Imagen de Pokémon(25) Pikachu">
            </div>
        </section>

        <!-- Características del juego -->
        <section class="caracteristicas">
```

```
            <h2>Características del juego</h2>
            <div class="caracteristicas-contenido">
                <div class="caracteristica">
                    <a href="./html/pokedex.html">
                        <h3>Explora toda la Pokédex</h3>
                        <p>Viaja a diferentes regiones y descubre nuevos Pokémon
en cada rincón del mapa.</p>
                    </a>
                </div>
                <div class="caracteristica">
                    <a href="./html/comparador.html">
                        <h3>Compara pokémons</h3>
                        <p>Elige entre dos pokémons para conocer sus diferencias.
</p>
                    </a>
                </div>
                <div class="caracteristica">
                    <a href="./html/equipos.html">
                        <h3>Crea tu equipo</h3>
                        <p>Crea tu equipo perfecto o haz combinaciones
inimaginables.</p>
                    </a>
                </div>
                <div class="caracteristica">
                    <a href="./html/jugar.html">
                        <h3>Combates épicos</h3>
                        <p>Realiza combates y demuestra tu habilidad como maestro
Pokémon.</p>
                    </a>
                </div>
            </div>
        </section>

        <!-- Galería o capturas de pantalla -->
        <section class="galeria">
            <h2>¡Descubre todas las especies!</h2>
            <div class="carrusel">
                <div class="carrusel-imagenes">
                    <img src="./img/logoPokeball.png" alt="Captura del juego">
                </div>
                <!--<div class="carrusel-controles">
                    <button class="prev">←</button>
                    <button class="next">→</button>
                </div>-->
            </div>
        </section>

        <!-- Call to Action -->
        <section class="cta">
            <h2>¡Únete a la aventura hoy mismo!</h2>
            <a href="./html/jugar.html" class="btn">¡Comienza ahora!</a>
        </section>
    </main>
```

main.js

Este módulo gestiona la obtención de datos de Pokémon desde la API oficial y IndexedDB. Proporciona funcionalidades para manejar especies, generaciones, evoluciones y datos específicos de los Pokémon.

1. Configuración Inicial

```
'use strict';

import { obtenerTodosLosPokemon, obtenerPokemonPorGeneracion, obtenerPokemon }
from "./helpers/indexedDB.js";
import { extraerID } from "./helpers/utils.js";
import Pokemon from "./models/Pokemon.js";

/**
 * @namespace app
 * @description Módulo principal de la aplicación que maneja la obtención de datos
de Pokémon desde la API y IndexedDB.
 */
export const app = (function () {
    /**
     * La URL base de la API de Pokémon.
     * @constant {string}
     */
    const urlAPI = 'https://pokeapi.co/api/v2';
```

Obtenemos los datos necesarios a través de una URL

```
    /**
     * Obtiene datos de una URL.
     * @param {string} url - La URL desde la cual obtener los datos.
     * @returns {Promise<Object>} Los datos obtenidos de la URL.
     * @throws {Error} Si hay un error al obtener los datos.
     * @memberof app
     */
    async function obtenerDatos(url) {
        try {
            const response = await fetch(url);

            if (!response.ok) {
                throw new Error();
            }

            return await response.json();
        } catch (error) {
            console.error(`Error obteniendo datos de ${url}:`, error);
            throw error;
        }
```

```
    }
```

Obtención del número máximo de pokemon

```
    /**
     * Obtiene el número máximo de Pokémon disponibles desde la API.
     * @returns {Promise<number>} El número máximo de Pokémon.
     * @throws {Error} Si hay un error al obtener los datos.
     * @memberof app
     */
    async function obtenerMaxPokemons() {
        try {
            const datosRespuesta = await obtenerDatos(`${urlAPI}/pokemon-
species`);
            return datosRespuesta.count;
        } catch (error) {
            console.error("Error obteniendo el número máximo de Pokémons:",
error);
            return 1025;
        }
    }
```

Obtenemos datos de las especies diferentes

```
    /**
     * Obtiene datos de las especies de Pokémon desde la API.
     * @param {number} desde - El índice inicial desde el cual obtener los datos.
     * @param {number} hasta - El número máximo de datos a obtener.
     * @returns {Promise<Array<Object>>} Una promesa que resuelve a un array de
objetos con los datos de las especies.
     * @throws {Error} Si hay un error al obtener los datos.
     * @memberof app
     */
    async function obtenerDatosEspecies(desde, hasta) {
        try {
            const datosRespuesta = await obtenerDatos(`${urlAPI}/pokemon-species?
limit=${hasta}&offset=${desde}`);
            const especiesURLs = datosRespuesta.results.map(specie => specie.url);

            const datosEspecies = await Promise.all(especiesURLs.map(async (url)
=> {
                const datosEspecie = await obtenerDatos(url);

                return {
                    generation: datosEspecie.generation.name,
                    pokemonUrl: datosEspecie.varieties[0].pokemon.url,
                    evolution_chain_url: datosEspecie.evolution_chain.url
```

```
                };
            }));

            return datosEspecies;
        } catch (error) {
            console.error("Error obteniendo datos de especies:", error);
        }
    }
```

Obtenemos los datos del pokemon

```
    /**
     * Obtiene datos de los Pokémon desde la API.
     * @returns {Promise<Object>} Una promesa que resuelve a un objeto con los
datos de los Pokémon agrupados por generación.
     * @throws {Error} Si hay un error al obtener los datos.
     * @memberof app
     */
    async function obtenerDatosPokemon() {
        try {
            const maxPokemons = await obtenerMaxPokemons();
            const datosEspecies = await obtenerDatosEspecies(0, maxPokemons);

            const datosPokemons = await Promise.all(datosEspecies.map(async
(especie) => {
                const datosPokemon = await obtenerDatos(especie.pokemonUrl);
                // Corrección de unidades y generación
                datosPokemon.height = datosPokemon.height / 10;
                datosPokemon.weight = datosPokemon.weight / 10;
                datosPokemon.generation = especie.generation;
                datosPokemon.evolution_chain_url = especie.evolution_chain_url;

                return new Pokemon(datosPokemon);
            }));

            const generaciones = Object.groupBy(datosPokemons, pokemon =>
pokemon.generation);
            return generaciones;
        } catch (error) {
            console.error("Error obteniendo datos de Pokémon:", error);
        }
    }
```

Obtenemos los datos desde IndexedDB

```
/**
    * Obtiene datos de los Pokémon desde IndexedDB según el filtro proporcionado.
```

```
    * @param {string} filtrarPor - El criterio de filtrado (puede ser 'todos',
'generacion', 'name' o 'id').
    * @param {string|number|null} [valor=null] - El valor del filtro (puede ser el
nombre, ID o generación del Pokémon).
    * @returns {Promise<Array<Object>|Object>} Una promesa que resuelve a un array
de objetos con los datos de los Pokémon o un objeto con los datos de un Pokémon.
    * @throws {Error} Si hay un error al obtener los datos.
    * @memberof app
    */
   async function obtenerDatosDesdeIndexedDB(filtrarPor, valor = null) {
       try {
           let pokemons;

           switch (filtrarPor) {
               case 'todos':
                   pokemons = await obtenerTodosLosPokemon();
                   break;
               case 'generacion':
                   pokemons = await obtenerPokemonPorGeneracion(valor);
                   break;
               case 'name':
               case 'id':
                   pokemons = await obtenerPokemon(filtrarPor, valor);
                   break;
               default:
                   throw new Error(`Filtro desconocido: ${filtrarPor}`);
           }

           return pokemons;
       } catch (error) {
           console.error(`Error obteniendo datos desde IndexedDB con filtro
${filtrarPor}:`, error);
           throw error;
       }
   }
```

Obtenemos la líne evolutiva del pokemon

```
    /**
     * Obtiene la cadena evolutiva de un Pokémon desde IndexedDB y la API.
     * @param {number} pokemonID - El ID del Pokémon para el cual obtener la
cadena evolutiva.
     * @returns {Promise<Array<Object>>} Una promesa que resuelve a un array de
objetos con los datos de las evoluciones del Pokémon.
     * @throws {Error} Si hay un error al obtener los datos.
     * @memberof app
     */
    async function obtenerEvoluciones(pokemonID) {
        try {
            // Obtener datos de la especie del Pokémon
            //const datosEspecie = await obtenerDatos(`${urlAPI}/pokemon-
species/${pokemonID}`);
```

```
            //const urlCadenaEvolutiva = datosEspecie.evolution_chain.url;

            const pokemon = await obtenerDatosDesdeIndexedDB('id', pokemonID);
            const urlCadenaEvolutiva = pokemon.evolution_chain_url;

            // Obtener datos de la cadena evolutiva
            const datosCadenaEvolutiva = await obtenerDatos(urlCadenaEvolutiva);

            // Recorrer la cadena evolutiva
            const evoluciones = [];
            let actual = datosCadenaEvolutiva.chain;
            //console.log(actual);

            do {
                const idPokemon = parseInt(extraerID(actual.species.url));
                const nombrePokemon = actual.species.name;
                evoluciones.push({ idPokemon, nombrePokemon });
                actual = actual.evolves_to[0];
            } while (actual);

            return evoluciones;
        } catch (error) {
            console.error("Error obteniendo evoluciones del Pokémon:", error);
        }
    }

    return {
        obtenerDatosPokemon,
        obtenerDatosDesdeIndexedDB,
        obtenerEvoluciones,
        obtenerMaxPokemons
    };
```

carrusel.js

Descripción general

Este archivo implementa un carrusel de imágenes que muestra Pokémon de manera aleatoria. Utiliza la API de Pokémon (`PokeAPI`) para cargar imágenes de los Pokémon y permite el desplazamiento automático entre ellos. También incluye funcionalidad para modificar la imagen del encabezado al interactuar con el carrusel.

Código completo y explicación

```
'use strict';

import { app } from "./main.js";
import { modificarImagenHeader } from "./helpers/ui.js";
```

```
const carrusel = document.querySelector('.carrusel-imagenes');
const image = document.querySelector('.carrusel-imagenes img');

// const prevButton = document.querySelector('.prev');
// const nextButton = document.querySelector('.next');

const maxPokemons = await app.obtenerMaxPokemons();
```

- **Importaciones:**
  - `app` se importa desde el archivo `main.js` y se utiliza para obtener el número máximo de Pokémon disponibles.
  - `modificarImagenHeader` es una función importada desde `ui.js` que modifica el encabezado visual.
- **Selección de elementos HTML:**
  - Se selecciona el contenedor del carrusel y la imagen interna que muestra los Pokémon.

```
/**
 * Actualiza el carrusel mostrando un Pokémon aleatorio.
 * Cambia temporalmente la imagen a una Pokéball mientras se carga la imagen del
Pokémon.
 */
function actualizarCarrusel() {
    const numeroAleatorio = Math.floor(Math.random() * maxPokemons) + 1; // Genera
un número aleatorio entre 1 y el máximo de pokémons disponibles
    image.src = '../img/logoPokeball.png'; // Cambia la imagen a la Pokéball
    carrusel.style.transform = 'scale(0.5)';
    setTimeout(() => {
        image.src =
`https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/of
ficial-artwork/${numeroAleatorio}.png`;
        carrusel.style.transform = 'scale(1)';
    }, 700); // Cambia a la nueva imagen después de 700ms
}
```

- **Función `actualizarCarrusel`:**
  - Genera un número aleatorio para seleccionar un Pokémony muestra temporalmente la imagen de una Pokéball mientras carga la imagen del Pokémon.

```
/**
 * Cambia la imagen del carrusel a otra aleatoria.
 * Llama internamente a `actualizarCarrusel`.
 */
function cambiarSlide() {
    actualizarCarrusel();
}
```

- **Función `cambiarSlide`:**
  - Invoca la función `actualizarCarrusel` para cambiar la imagen a otro Pokémon aleatorio.

```
// Desplazamiento automático
let autoSlide = setInterval(cambiarSlide, 3000); // Cambia cada 3 segundos

/**
 * Reinicia el temporizador del desplazamiento automático.
 * Se utiliza cuando el usuario interactúa manualmente con el carrusel.
 */
function reiniciarAutoSlide() {
    clearInterval(autoSlide);
    autoSlide = setInterval(cambiarSlide, 3000);
}
```

- **Desplazamiento automático:**
  - Se define un intervalo que cambia la imagen del carrusel cada 3 segundos.
  - La función `reiniciarAutoSlide` reinicia el temporizador automático.

```
// Modificar la imagen del header al hacer hover
modificarImagenHeader(await app.obtenerMaxPokemons());

// Inicializa el carrusel con una imagen aleatoria al cargar la página
actualizarCarrusel();
```

- **Modificación del encabezado:**
  - Cambia la imagen del encabezado cuando se realiza un evento de hover.
- **Inicialización del carrusel:**
  - Muestra una imagen aleatoria de Pokémon al cargar la página.

`pokedex.html`

Este archivo presenta la Pokédex del proyecto, donde se listan todos los Pokémon disponibles. Incluye:

- Un encabezado con el título "Pokédex".
- Una sección principal para mostrar la lista de Pokémon, generada dinámicamente con `pokedex.js`.

```
<main>
        <h1 class="centrado">Pokédex</h1>

        <div class="contenedor-filtros">
            <div class="contenedor-filtro-generacion">
                <label for="generationFilter">Filtrar por generación:</label>
                <select id="generationFilter">
                    <option value="all">Todas</option>
```

```
                    <option value="generation-i">Generación I</option>
                    <option value="generation-ii">Generación II</option>
                    <option value="generation-iii">Generación III</option>
                    <option value="generation-iv">Generación IV</option>
                    <option value="generation-v">Generación V</option>
                    <option value="generation-vi">Generación VI</option>
                    <option value="generation-vii">Generación VII</option>
                    <option value="generation-viii">Generación VIII</option>
                    <option value="generation-ix">Generación IX</option>
                </select>
            </div>

            <div class="contenedor-busqueda-nombre">
                <label for="buscarNombre">Buscar Pokémon: </label>
                <input type="text" name="buscarNombre" id="buscarNombre"
placeholder="Pikachu" title="Nombre del Pokémon">
                <input type="button" value="Buscar" title="Buscar Pokémon" data-
accion="buscar">
                <input type="button" value="Limpiar" title="Limpiar filtros" data-
accion="limpiar">
            </div>
        </div>

        <div class="contenedor-fichas">

        </div>

        <div class="contenedor-boton-cargar-mas">
            <input type="button" value="Cargar más Pokémons" id="cargarMas"
style="display: none;" class="boton-cargar-mas" title="Cargar más Pokémons"/>
        </div>

        <input type="button" value="↑" id="volverArriba" title="Volver arriba"
class="boton-volver-arriba"/>
    </main>
```

pokedex.js

Carga Inicial de los datos

```
async function cargaInicial() {
    mostrarSpinner();
    const datosExisten = await existeDatosEnGeneraciones();

    if (!datosExisten) {
        const generaciones = await app.obtenerDatosPokemon();

        if (generaciones) {
            await guardarDatosPokemon(generaciones);
        }
```

```
    }

    pokemons = await app.obtenerDatosDesdeIndexedDB('todos');

    if (pokemons.length) {
        mostrarFichaPokemon(pokemons, inicio, cantidad);
        inicio += cantidad;

        if (pokemons.length > cantidad) {
            document.getElementById('cargarMas').style.display = 'block';
        }
    }

    ocultarSpinner();
}
```

Resetea el índice de inicio y establece el filtro de generación en "all" y obtiene los datos del Pokémon según el nombre o ID ingresado. Si se encuentran varios resultados, muestra los primeros cantidad Pokémon y habilita el botón "Cargar más" si hay más datos.Si el resultado es un único Pokémon, lo muestra sin paginación.

```
async function desencadenadorEventoBuscarPokemon(inputNombrePokemon,
selectGeneracion, botonCargarMas = null) {
    inicio = 0;
    selectGeneracion.value = 'all';
    const idPokemon = parseInt(inputNombrePokemon.value);
    let camposFiltro = [];

    if (idPokemon) {
        camposFiltro = ['id', idPokemon];
    } else {
        camposFiltro = ['name', inputNombrePokemon.value];
    }

    pokemons = await app.obtenerDatosDesdeIndexedDB(camposFiltro[0],
camposFiltro[1]);

    if (Array.isArray(pokemons)) {
        mostrarFichaPokemon(pokemons, inicio, cantidad);
        inicio += cantidad;

        if (botonCargarMas) {
            if (pokemons.length > cantidad) {
                botonCargarMas.style.display = 'block';
            } else {
                botonCargarMas.style.display = 'none';
            }
        }
    } else {
        mostrarFichaPokemon(pokemons);
```

```
            if (botonCargarMas) {
                botonCargarMas.style.display = 'none';
            }
        }
    }
```

Limpia el contenedor de fichas de Pokémon, el input de búsqueda, y el filtro de generación.Resetea el índice de inicio.Llama a la función cargaInicial para recargar los datos iniciales.

```
async function desencadenadorEventoLimpiar(contenedorFichas, inputNombrePokemon,
selectGeneracion) {
    inicio = 0;
    contenedorFichas.innerHTML = '';
    inputNombrePokemon.value = '';
    selectGeneracion.value = 'all';
    await cargaInicial();
}
```

Ejecuta los bloques de código cuando se carga completamente el DOM e inicializa la imagen del encabezado según el número máximo de Pokémon disponibles.

```
document.addEventListener('DOMContentLoaded', async () => {
    modificarImagenHeader(await app.obtenerMaxPokemons());
```

`detalle.html`

Este archivo proporciona la estructura para mostrar los detalles de un Pokémon específico. Sus características principales incluyen:

- Un encabezado con el título "Detalle del Pokémon".
- Una sección principal donde se cargan los datos detallados del Pokémon de manera dinámica mediante `detalle.js`.

```
<main>
        <section class="detalles-pokemon">
            <div class="info-general">
                <div class="info-basica">
                    <p id="id-pokemon">N.º ID</p>
                    <h2 id="nombre-pokemon">Nombre del Pokémon</h2>
                    <img id="imagen-pokemon" src="" alt="Imagen del Pokémon">
                    <p>
                        <a id="comparar" name="comparar" class="btn">Comparar</a>
```

```
                    </p>
                    <p id="generacion"></p>
                    <p id="altura"></p>
                    <p id="peso"></p>
                </div>

                <div class="info-tipos">
                    <h3>Tipos</h3>
                    <div id="contenedor-tipos" class="tipos">
                        <!-- Aquí se agregarán dinámicamente los tipos -->
                    </div>
                </div>

                <div class="info-sprites">
                    <h3>Sprites</h3>
                    <div id="contenedor-sprites" class="sprites">
                        <!-- Aquí se agregarán dinámicamente los sprites -->
                    </div>
                </div>



            </div>

            <div class="info-stats-hab">
                <div class="estadisticas">
                    <h3>Estadísticas</h3>
                    <ul id="lista-estadisticas">
                        <!-- Aquí se agregarán dinámicamente las estadísticas -->
                    </ul>
                </div>

                <div class="habilidades">
                    <h3>Habilidades</h3>
                    <ul id="lista-habilidades">
                        <!-- Aquí se agregarán dinámicamente las habilidades -->
                    </ul>
                </div>

                <div class="info-evoluciones">
                    <h3>Evoluciones</h3>
                    <div id="contenedor-evoluciones" class="evoluciones">
                        <!-- Aquí se agregarán dinámicamente las evoluciones -->
                    </div>
                </div>
            </div>
        </section>
    </main>
```

detalle.js

Este código carga y muestra información detallada de un Pokémon al abrir su página específica. La funcionalidad está diseñada para ofrecer interactividad con datos del Pokémon, incluyendo estadísticas, evoluciones, habilidades y sprites. Al cargar el DOM, se configura la página con información básica, evoluciones, y opciones interactivas.

```
document.addEventListener('DOMContentLoaded', async () => {
    modificarImagenHeader(await app.obtenerMaxPokemons()); // Modifica la imagen
del header.

    const main = document.querySelector('body');
    main.classList.add('fondo'); // Agrega estilo al fondo.

    const params = new URLSearchParams(window.location.search);
    const id = parseInt(params.get('id'), 10); // Obtiene el ID del Pokémon desde
la URL.
```

Valida y carga la información básica del Pokémon desde IndexedDB.

```
    const datosPokemon = await app.obtenerDatosDesdeIndexedDB('id', id);

    if (Array.isArray(datosPokemon) && !datosPokemon.length) {
        alert('Pokémon no encontrado');
        window.location.href = 'pokedex.html';
        return;
    }

    const pokemon = new Pokemon(datosPokemon);
```

Muestra detalles como nombre, número, generación, peso y altura.

```
    document.getElementById('nombre-pokemon').textContent =
pokemon.name.toUpperCase();
    document.getElementById('id-pokemon').textContent = `N.º ${pokemon.id}`;
    document.getElementById('generacion').textContent = `Generación:
${pokemon.generation}`;
    document.getElementById('peso').textContent = `Peso: ${pokemon.weight}
kilogramos`;
    document.getElementById('altura').textContent = `Altura: ${pokemon.height}
metros`;
```

Al pasar el ratón, cambia entre el sprite normal y shiny.

```
    const imagenPokemon = document.getElementById('imagen-pokemon');
```

```
    const defaultSrc = pokemon.sprites.other['official-artwork'].front_default;
    const hoverSrc = pokemon.sprites.other['official-artwork'].front_shiny;

    imagenPokemon.src = defaultSrc;

    imagenPokemon.addEventListener('mouseover', () => {
        imagenPokemon.src = hoverSrc;
    });

    imagenPokemon.addEventListener('mouseout', () => {
        imagenPokemon.src = defaultSrc;
    });
```

Muestra los tipos asociados al Pokémon como imágenes.

```
    const tiposContenedor = document.getElementById('contenedor-tipos');
    pokemon.types.forEach(tipo => {
        const imgTipo = document.createElement('img');
        const id = extraerID(tipo.type.url);
        imgTipo.src =
`https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/types/generation
-viii/sword-shield/${id}.png`;
        tiposContenedor.appendChild(imgTipo);
    });
```

Carga y muestra las evoluciones del Pokémon.

```
    const evolucionesContenedor = document.getElementById('contenedor-
evoluciones');
    const evoluciones = await app.obtenerEvoluciones(pokemon.id);

    for (const evolucion of evoluciones) {
        const aDetallePokemon = document.createElement('a');
        aDetallePokemon.href = `detalle.html?id=${evolucion.idPokemon}`;

        const divNombre = document.createElement('div');
        divNombre.textContent = capitalizarPrimeraLetra(evolucion.nombrePokemon);

        const evolucionImg = document.createElement('img');
        const evolucionData = await app.obtenerDatosDesdeIndexedDB('id',
evolucion.idPokemon);
        evolucionImg.src = evolucionData.sprites.other['official-
artwork'].front_default;

        aDetallePokemon.appendChild(evolucionImg);
        aDetallePokemon.appendChild(divNombre);
        evolucionesContenedor.appendChild(aDetallePokemon);
```

```
    }
```

Lista las estadísticas base y el promedio.

```
    const listaEstadisticas = document.getElementById('lista-estadisticas');
    pokemon.stats.forEach(stat => {
        const li = document.createElement('li');
        li.textContent = `${capitalizarPrimeraLetra(stat.stat.name)}:
${stat.base_stat}`;
        listaEstadisticas.appendChild(li);
    });

    const media = document.createElement('li');
    media.textContent = `Average: ${pokemon.getAverageStats()}`;
    listaEstadisticas.appendChild(media);
```

Lista las habilidades con sus descripciones.

```
    const listaHabilidades = document.getElementById('lista-habilidades');
    const habilidadesTexto = await pokemon.getAbilities();

    habilidadesTexto.forEach(habilidad => {
        const li = document.createElement('li');
        li.textContent = `${capitalizarPrimeraLetra(habilidad.name)}:
${capitalizarPrimeraLetra(habilidad.description)}`;
        listaHabilidades.appendChild(li);
    });
```

Redirige al usuario a la página de comparación.

```
    const btnComparar = document.querySelector('#comparar');
    btnComparar.addEventListener('click', function () {
        window.location.href = `comparador.html?pokemon1=${pokemon.id}`;
    });
```

`comparador.html`

Este archivo define la estructura de la página donde se pueden comparar dos Pokémon. Contiene:

- Un encabezado con el título "Comparador de Pokémon".
- Una sección principal con dos contenedores (`div`) para mostrar la información de los Pokémon seleccionados.

- Incluye el archivo de estilos `style.css` para el diseño y el archivo `comparador.js` para la lógica de comparación.

```
<main>
        <h1 class="centrado">Comparador</h1>

        <div class="layout-comparador">
            <div id="pokemon1" class="comparador">
                <p>Selecciona un pokémon</p>
            </div>

            <div id="pokemon2" class="comparador">
                <p>Selecciona un pokémon</p>
            </div>

            <div class="contenedor-principal">
                <div class="contenedor-filtros">
                    <div class="contenedor-filtro-generacion">
                        <label for="generationFilter">Filtrar por generación:
</label>
                        <select id="generationFilter">
                            <option value="all">Todas</option>
                            <option value="generation-i">Generación I</option>
                            <option value="generation-ii">Generación II</option>
                            <option value="generation-iii">Generación III</option>
                            <option value="generation-iv">Generación IV</option>
                            <option value="generation-v">Generación V</option>
                            <option value="generation-vi">Generación VI</option>
                            <option value="generation-vii">Generación VII</option>
                            <option value="generation-viii">Generación
VIII</option>
                            <option value="generation-ix">Generación IX</option>
                        </select>
                    </div>

                    <div class="contenedor-busqueda-nombre">
                        <label for="buscarNombre">Buscar Pokémon: </label>
                        <input type="text" name="buscarNombre" id="buscarNombre"
placeholder="Pikachu"
                            title="Nombre del Pokémon">
                        <input type="button" value="Buscar" title="Buscar Pokémon"
data-accion="buscar">
                        <input type="button" value="Limpiar" title="Limpiar
filtros" data-accion="limpiar">
                    </div>
                </div>

                <div class="contenedor-fichas contenedor-scroll">

                </div>
            </div>
        </div>
```

```
    </main>
```

Este módulo gestiona la lógica para cargar, buscar, comparar y mostrar información sobre Pokémon utilizando `IndexedDB`.

Funcionalidades Principales

Carga datos de Pokémon desde una API o `IndexedDB` y los muestra.

```
async function cargaInicial() {
    mostrarSpinner();
    const datosExisten = await existeDatosEnGeneraciones();

    if (!datosExisten) {
        const generaciones = await app.obtenerDatosPokemon();
        if (generaciones) await guardarDatosPokemon(generaciones);
    }

    pokemons = await app.obtenerDatosDesdeIndexedDB('todos');
    if (pokemons) mostrarFichaPokemon(pokemons, 0, pokemons.length);
    ocultarSpinner();
}
```

Permite buscar Pokémon por nombre o ID.

```
async function desencadenadorEventoBuscarPokemon(inputNombrePokemon,
selectGeneracion) {
    selectGeneracion.value = 'all';
    const idPokemon = parseInt(inputNombrePokemon.value);
    const camposFiltro = idPokemon ? ['id', idPokemon] : ['name',
inputNombrePokemon.value];

    pokemons = await app.obtenerDatosDesdeIndexedDB(camposFiltro[0],
camposFiltro[1]);
    if (pokemons) mostrarFichaPokemon(pokemons);
}
```

Carga y compara uno o dos Pokémon según los parámetros de la URL.

```
async function cargarTarjetas() {
    const params = new URLSearchParams(window.location.search);
    const idPokemon1 = parseInt(params.get('pokemon1'));
    const idPokemon2 = parseInt(params.get('pokemon2'));
```

```
    if (params.has('pokemon1') && idPokemon1) {
        const datosPokemon1 = await app.obtenerDatosDesdeIndexedDB('id',
idPokemon1);
        const pokemon1 = new Pokemon(datosPokemon1);
        divPokemon1.innerHTML = devolverDetallePokemon(pokemon1);
    }

    if (params.has('pokemon2') && idPokemon2) {
        const datosPokemon2 = await app.obtenerDatosDesdeIndexedDB('id',
idPokemon2);
        const pokemon2 = new Pokemon(datosPokemon2);
        divPokemon2.innerHTML = devolverDetallePokemon(pokemon2);
    }

    if (idPokemon1 && idPokemon2) comparar(pokemon1, pokemon2);
}
```

Filtra Pokémon según la generación seleccionada en un menú desplegable.

```
selectGeneracion.addEventListener('change', async (event) => {
    const generacion = event.target.value;
    pokemons = generacion === 'all'
        ? await app.obtenerDatosDesdeIndexedDB('todos')
        : await app.obtenerDatosDesdeIndexedDB('generacion', generacion);

    if (pokemons) mostrarFichaPokemon(pokemons);
});
```

Incluye funciones rápidas para limpiar la interfaz y vaciar la base de datos.

```
// Limpia pantalla con F1
document.addEventListener('keydown', async (event) => {
    if (event.key === 'F1') {
        event.preventDefault();
        await desencadenadorEventoLimpiar(contenedorFichas, inputNombrePokemon,
selectGeneracion);
    }
});

// Vacía IndexedDB con F2
document.addEventListener('keydown', (event) => {
    if (event.key === 'F2') limpiarDatosPokemon();
});
```

Planificación

El diagrama de Gantt al que nos referimos se encuentra en el readme de nuestro github.

El diagrama de Gantt muestra la planificación del proyecto entre los días 10 y 20. Incluye tareas como análisis de requisitos, diseño de arquitectura, implementación de funcionalidades (buscador, comparador, gestor de equipos, batallas), optimización, pruebas, documentación y preparación de la presentación. Las tareas se solapan estratégicamente para optimizar el tiempo, finalizando todo el día 20 con la entrega y defensa del proyecto.

Arquitectura

El documento al que hacemos alusión se encuentra dentro del github que hemos utilizado para seguir el progreso del proyecto y donde hemos guardado todos los documentos sobre el proyecto.

Explicación del Diagrama de Arquitectura

El diagrama representa la arquitectura y el flujo de datos de un proyecto basado en una aplicación Pokémon. A continuación, se describen las principales partes y su funcionalidad:

1. Conexión con PokeAPI

La aplicación utiliza **PokeAPI** como fuente de datos externa para obtener información sobre Pokémon, que posteriormente se procesa y almacena localmente para su uso.

2. Componentes de Almacenamiento Local

- **IndexedDB**: Maneja datos estructurados como información de generaciones y detalles de Pokémon específicos.
- **LocalStorage**: Almacena datos simples o persistentes, como los equipos creados por los usuarios.
- **Clases relacionadas**:
  - `PokemonDB`: Gestiona los datos almacenados en IndexedDB.
  - `generaciones`: Organiza la información de Pokémon por generación.

3. Archivos de Utilidad

- **indexedDB.js** y **localStorage.js**: Manejan las operaciones de almacenamiento y recuperación de datos.
- **utils.js** y **ui.js**:
  - `utils.js`: Proporciona funciones de soporte como cálculos y transformaciones.
  - `ui.js`: Maneja la manipulación de la interfaz de usuario, incluyendo la visualización de datos y la interacción con eventos.

4. Flujo Principal

- **main.js**: Es el núcleo de la lógica de la aplicación. Coordina el flujo de datos entre los distintos componentes, desde la obtención de datos de la API hasta su uso en la interfaz.

5. HTML y JS Relacionados con la Interfaz

- Archivos HTML (`pokedex.html`, `comparador.html`, `equipos.html`, etc.) y sus respectivos scripts JS (`pokedex.js`, `comparador.js`, etc.) manejan funcionalidades específicas:

- **Exploración de la Pokédex**.
- **Comparación de Pokémon**.
- **Gestión de equipos de Pokémon**.
- **Combates y juego**.

6. Modelos Principales

- **Pokemon**:
  - Define la estructura de datos de un Pokémon, con propiedades como `id`, `name`, `types`, `stats`, entre otros.
  - Métodos incluidos: `getFormattedTypes()`, `getAbilities()`, `getAverageStats()`.
- **Equipo**:
  - Representa un equipo de Pokémon.
  - Métodos incluidos: `agregarPokemon()`, `eliminarPokemon()`, `limpiarEquipo()`.

7. Visualización de Datos

- Tablas del diagrama muestran cómo los datos se organizan en **IndexedDB**, usando claves como `generation` y detalles como listas de Pokémon de cada generación.

Resumen

El diagrama presenta una estructura modular y escalable, en la que se conectan el backend (API y almacenamiento local) y el frontend (interfaz y lógica específica) de manera clara y eficiente. Esto permite manejar grandes cantidades de datos y proporcionar una experiencia de usuario intuitiva.

Ejemplos de uso

Página principal

Como podemos ver en la página principal tenemos un menú que nos lleva a los demás apartados además de información de nuestra web. Hemos añadido animaciones para hacer más interesante su navegación.

Si pinchamos en el primer link, que sería "Pokedex", nos llevará a esta como su propio nombre indica.

Pokédex

En ella podemos ver una lista de todos los pokemon existentes actualmente. Podemos ver también un buscador que podemos filtrarlo por generación. Este por defecto esta en "Todas" por lo que recoge todas las generaciones y por tanto todos los pokemon que hay. En cambio si cambiamos ese filtro a cualquier generación solo nos aparecerán los pokemon de dicha generación. Podemos apreciar también un buscador que funcionan por ID (número del pokemon en la pokedex) y por nombre. En la parte inferior de la pokedex contamos con un botón "Cargar más Pokemons" que nos sacará los 12 pokemon siguientes para ir viéndolos poco a poco. También cuneta con un botón que nos lleva al inicio de la pokedex directamente. Al pasar el ratón por encima de los pokemon se produce una animación para destacar dicho pokemon.

Lo siguiente que vamos a ver es que al hacer clic en un pokemon nos dirigimos a otra web que es la ficha de datos del pokemon en cuestión.

Ficha del pokemon(ejemplo)

Dentro de esta ficha podemos ver los datos del pokemon, como sus estadísticas, peso, altura, evoluciones, sprites, tipos y habilidades, es decir, una vista más ampliada del pokemon. Podemos apreciar también un botón "Comparar" que nos llevará a otra parte de la web que explicaremos ahora. A esta parte podemos acceder tanto desde dentro de la ficha como desde el menú de navegación clicando a "Comparador".

Comparador

En este apartado vemos la lista de pokemon en el mismo formato que la pokedex, es decir, contamos con los pokemon de forma visual como con el filtro por generación y el buscador por ID y nombre. Este apartado nos permite elegir dos pokemon(los que queramos) y compararlos entre ellos para ver cual es la diferencia entre uno y otro. Si elegimos dos pokemon lo que vemos es uno a la izquierda y otro a la derecha para ver la diferencia entre ellos de forma más clara. Lo que llama la atención es que en el apartado de las estadísticas sale en verde sila estadística de un pokemon es mayor que la del otro, y roja si es más débil que la del otro pokemon. Debajo de todos los datos que de los pokemon hay un botón "Eliminar" que nos permite quitar un pokemon de la comparación y así poder elegir otro sin tener que borrar ambos.

Por último, para los apartados de "Equipos" y "Jugar" no tenemos funcionalidad aún.

Dificultades/Resumen

Los principales problemas que nos hemos encontrado son la dificultad a la hora de planear como llevar a cabo el proyecto, ya que se podía realizar de muchas maneras diferentes lo que nos llevo un tiempo ponernos de acuerdo. Otro problema fue el de la organización ya que en algún momento no estuvo todo el equipo al completo para realizar las tareas necesarias, lo que llevó a una pequeña falta de comunicación en diferentes partes del proyecto. El manejo de la información del tema del proyecto también fue una complicación, porque la ser muy amplia tardamos en ver como poder manejarla y cual sería la manera más adecuada de hacerlo. Más adelante, durante la elaboración del proyecto fueron surgiendo dudas respecto a la visualización y el formato en el que queríamos mostrar toda esa información.