



BENEMÉRITA UNIVERSIDAD AUTÓNOMA
DE PUEBLA

FACULTAD DE CIENCIAS DE LA
COMPUTACIÓN

INGENIERÍA EN TECNOLOGÍAS DE LA
INFORMACIÓN

MATERIA: PROGRAMACIÓN DE
DISPOSITIVOS MÓVILES

PRÁCTICA 4: POKEDEX APP

M.C. LUIS Yael MENDEZ – SANCHEZ

ALUMNOS: JORGE LEON VIVANCO

MATRICULA: 202129258

INTERPERIODO 2025

INTRODUCCIÓN

En el mundo actual, las aplicaciones móviles se han convertido en herramientas imprescindibles para facilitar el acceso a información, entretenimiento y servicios. En este contexto, la creación de aplicaciones con Ionic, un framework basado en Angular, permite desarrollar soluciones altamente interactivas y optimizadas para diversas plataformas, desde dispositivos móviles hasta escritorios. Este proyecto tiene como objetivo principal la creación de una aplicación de Pokédex que permita a los usuarios explorar el mundo de los Pokémon de una manera sencilla y visualmente atractiva.

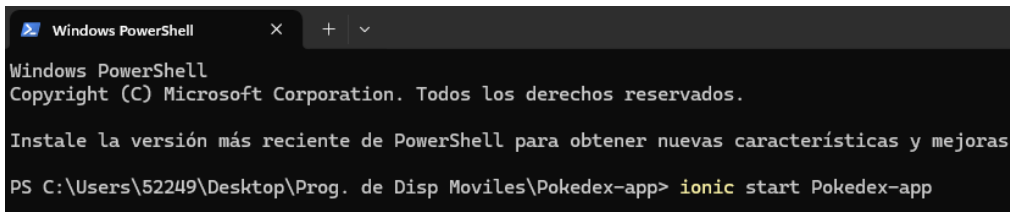
La idea de desarrollar una aplicación Pokédex surge de la necesidad de proporcionar una interfaz accesible para que los usuarios puedan acceder a información detallada de cada Pokémon disponible en la PokeAPI, una API gratuita que ofrece datos actualizados sobre los Pokémon. Esta aplicación se construyó con Ionic 6/7, aprovechando las características de este framework como su capacidad para crear interfaces altamente responsivas, su integración con Angular, y su compatibilidad con Capacitor, que permite el acceso a funcionalidades nativas.

A lo largo de este proyecto, se implementaron una serie de características clave para garantizar una experiencia de usuario fluida y atractiva. Entre ellas, se incluye el consumo de la API de Pokémon, la implementación de un scroll infinito para cargar más pokemons conforme el usuario se desplaza, el almacenamiento local con Ionic Storage para mejorar el rendimiento de la aplicación, y un diseño responsivo que asegura que la app se vea y funcione perfectamente en dispositivos móviles, tabletas y escritorios.

DESARROLLO

- **Creación del Proyecto Ionic**

La aplicación se construyó utilizando **Ionic 6/7**. Para ello, creamos un nuevo proyecto con el comando `ionic start`, especificando el tipo de proyecto y la estructura necesaria.



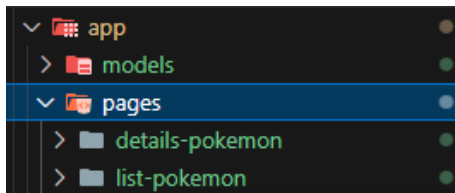
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras.

PS C:\Users\52249\Desktop\Prog. de Disp Moviles\Pokedex-app> ionic start Pokedex-app
```

Una vez creado el proyecto, se configuraron los módulos necesarios y se instalaron las dependencias de Ionic Angular.

- **Estructura de la Aplicación**



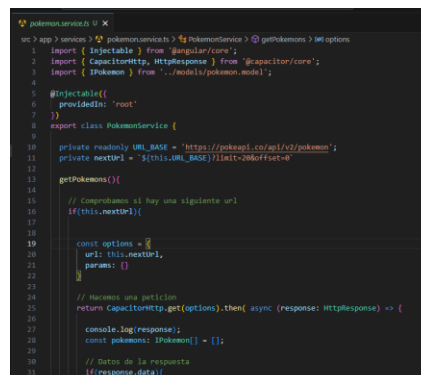
La aplicación está organizada en dos pantallas principales:

- List-Pokemon: Muestra la lista de pokemons con scroll infinito.
- Details-Pokemon: Muestra información detallada sobre un pokemon seleccionado.

Además, se creó un servicio `PokemonService` para gestionar la obtención de los datos desde la PokeAPI.

- **Consumo de la API de Pokémon**

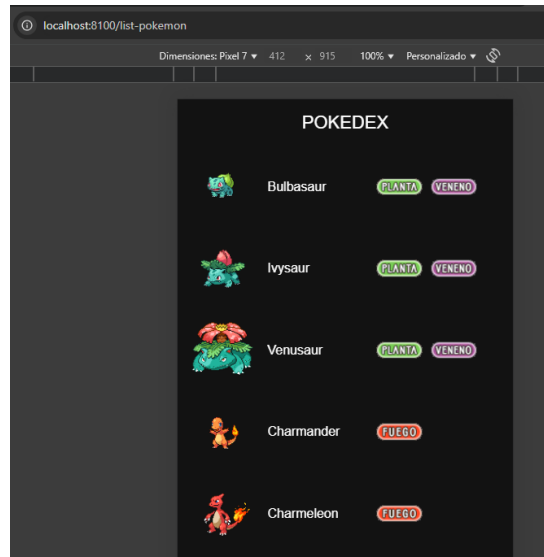
La PokeAPI se utiliza para obtener información de los pokemons. Para ello, se realiza una petición HTTP a la URL base de la API y luego, utilizando `CapacitorHttp`, se recuperan los detalles de cada pokemon.



```
src > app > services > pokemon.service.ts | cat
1 import { Injectable } from '@angular/core';
2 import { CapacitorHttp, HttpResponse } from '@capacitor/core';
3 import { IPokemon } from '../models/pokemon.model';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class PokemonService {
9
10   private readonly URL_BASE = 'https://pokeapi.co/api/v2/pokemon';
11   private nextUrl = `${this.URL_BASE}?limit=20&offset=0`;
12
13   getPokemons(): IPokemon[] {
14     // Comprobamos si hay una siguiente url
15     if (this.nextUrl) {
16       const options = {
17         url: this.nextUrl,
18         params: {}
19       };
20
21       // Hacemos una petición
22       return CapacitorHttp.get(options).then(async (response: HttpResponse) => {
23         console.log(response);
24         const pokemons: IPokemon[] = [];
25         // Datos de la respuesta
26         if (response.data) {
27           pokemons.push(...response.data.pokemon);
28           this.nextUrl = response.data.next;
29         }
30       });
31     }
32   }
33 }
```

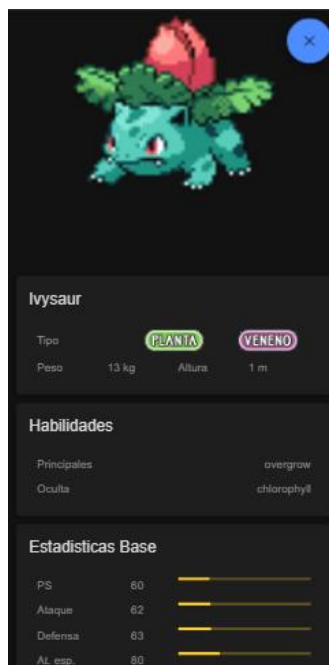
- **Procesamiento de los Datos**

Una vez obtenidos los datos de la API, los procesamos en el servicio PokemonService, mapeando la respuesta de la API a un modelo de datos estructurado.



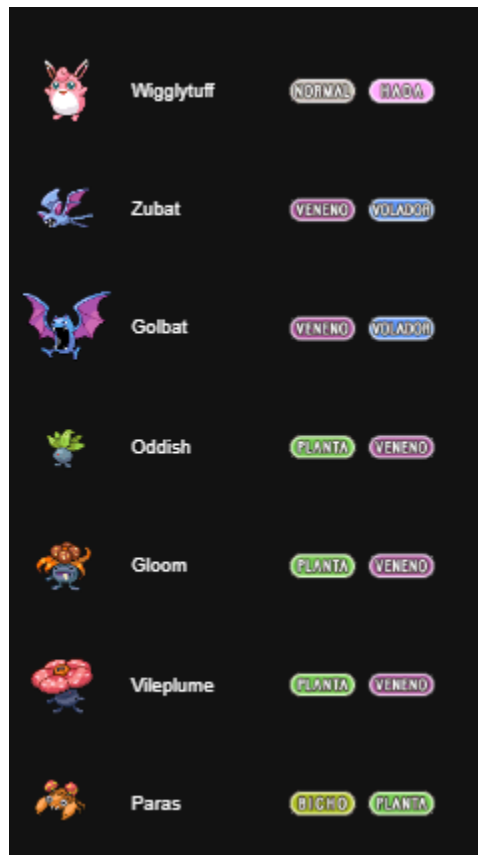
- **Pipes**

Se crearon **pipes** personalizados para formatear la visualización de las estadísticas de los pokemons, como transformar los valores de las estadísticas en un formato más legible y presentable.



- **Scroll Infinito con IonInfiniteScroll**

Para mejorar la experiencia del usuario, implementé un **scroll infinito** que permite cargar más pokemons a medida que el usuario se desplaza hacia abajo. Usamos el componente IonInfiniteScroll y IonInfiniteScrollContent de Ionic.



CONCLUSION

El desarrollo de esta aplicación de Pokédex con Ionic ha sido una experiencia enriquecedora, que ha permitido profundizar en el uso de tecnologías modernas como Angular, Capacitor y el propio Ionic Framework. Cada etapa del desarrollo fue una oportunidad para aprender nuevas técnicas y mejores prácticas para optimizar el rendimiento de la aplicación y mejorar la experiencia del usuario.

Uno de los logros más importantes de este proyecto fue la integración efectiva con la PokeAPI, que proporcionó un acceso fluido a la información de los Pokémon, permitiendo a la aplicación actualizarse constantemente con nuevos datos. El uso de scroll infinito y Ionic Storage ayudó a mejorar la interacción del usuario, asegurando que la aplicación se cargara de manera eficiente sin comprometer el rendimiento.

Además, se puso un énfasis especial en el diseño responsivo, lo que permitió que la aplicación se adaptara automáticamente a diferentes tamaños de pantalla, garantizando una experiencia óptima tanto en móviles como en tabletas y escritorios. Esta adaptación es crucial, dado que los usuarios de hoy esperan que las aplicaciones se comporten de manera consistente sin importar el dispositivo.

En términos de arquitectura, el proyecto siguió un enfoque modular, utilizando componentes reutilizables para separar las diferentes funcionalidades de la aplicación. Esto no solo mejoró la mantenibilidad del código, sino que también facilitó la implementación de futuras mejoras, como la adición de nuevas características o la optimización de la interfaz de usuario.

REFERENCIAS BIBLIOGRÁFICAS

- Ionic Framework. (s.f.). *Ionic Documentation*. Recuperado el 10 de junio de 2025, de <https://ionicframework.com/docs>
- Angular. (s.f.). *Angular Documentation*. Recuperado el 10 de junio de 2025, de <https://angular.io/docs>
- CapacitorJS. (s.f.). *Capacitor Documentation*. Recuperado el 10 de junio de 2025, de <https://capacitorjs.com/docs>
- Ionic Team. (2022). *Ionic 6.0 Release Notes*. Recuperado de <https://ionicframework.com/blog/announcing-ionic-v6>
- Wieruch, R. (2019). *The Road to React: Your journey to master plain yet pragmatic React.js*. Independently published. (*Referente a buenas prácticas en componentes reutilizables, aplicables también en Angular/Ionic*)
- Freeman, A., & Sanderson, P. (2020). *Pro Angular 9*. Apress. <https://doi.org/10.1007/978-1-4842-6037-2>
- Hartl, M. (2020). *Web Development with JavaScript and the Ionic Framework*. Learn Enough. Recuperado de <https://www.learnenough.com/>