Pokédex

Aplicación Android basada en el patrón Modelo - Vista - Controlador (MVC)

Autora: Ana Núñez

Curso: Desarrollo de Aplicaciones Multiplataforma (DAM)

Fecha: Octubre 2025

Proyecto Android Studio – Lenguaje Java

`github.com/AnaNunezRejon/Pokedex`$_{AndroidStudio_{Java}}$

1. Descripción general

Aplicación Android basada en el patrón **Modelo - Vista - Controlador (MVC)**.

Autora: Ana Núñez
Curso: Desarrollo de Aplicaciones Multiplataforma (DAM)
Fecha: Octubre 2025
Proyecto: Android Studio – Lenguaje Java
Repositorio: github.com/AnaNunezRejon/Pokedex$_{A}ndroidStudio_{j}ava$

—

2. Descripción de la aplicación

Pokédex es una app Android desarrollada en **Java** siguiendo el patrón **MVC**. Consume la API pública PokeAPI para mostrar información de los Pokémon: nombre, número, imagen, tipos, altura, peso, habilidad, categoría y descripción (todo en español).

La interfaz incluye:

- Buscador con sugerencias en tiempo real (nombre o número).
- Chips de tipos y debilidades.
- Carga por bloques con paginación.
- Traducción al español de habilidades, categorías y descripciones.

—

3. Objetivo general

Permitir **consultar** y **explorar** Pokémon como ejemplo práctico de:

- Uso del patrón MVC en Android.
- Consumo de APIs REST (HTTP + JSON).
- Diseño de interfaces dinámicas en XML.
- Buenas prácticas de UI y asincronía (uso de hilos, `runOnUiThread`, `ExecutorService`).

—

4. Requisitos funcionales

- **RF1:** Pantalla de inicio – Mostrar el logo de Pokédex y redirigir a la pantalla de tipos.
- **RF2:** Pantalla de tipos – Permitir seleccionar un tipo de Pokémon y realizar búsquedas por nombre o número.
- **RF3:** Pantalla de lista – Mostrar los Pokémon del tipo seleccionado en cuadrícula con paginación (30 por bloque).
- **RF4:** Pantalla de detalle – Mostrar ficha completa con imagen, número, altura, peso, habilidad, categoría, descripción, tipos y debilidades.
- **RF5:** Buscador – Realizar búsquedas dinámicas (nombre o número) con sugerencias automáticas.
- **RF6:** Traducción – Mostrar todos los textos traducidos al español.
- **RF7:** Carga asíncrona – Las llamadas a la API deben ejecutarse en hilos separados para no bloquear la interfaz.
- **RF8:** Interfaz dinámica – Generar chips de tipo y debilidad de forma programática con color por tipo.
- **RF9:** Manejo de errores – Mostrar mensajes adecuados si no hay conexión o el Pokémon no existe.
- **RF10:** Fluidez – Cargar imágenes y datos sin retrasos visibles mediante asincronía.

—

5. Funcionamiento general

1. **Pantalla de inicio:** muestra el logo de Pokédex.
2. **Pantalla de tipos:** botones de tipos (agua, fuego, planta, etc.) y buscador superior.
3. **Pantalla de lista:** muestra los Pokémon del tipo elegido en un `GridLayout`.
4. **Pantalla de detalle:** muestra la ficha completa del Pokémon: imagen, número, altura, peso, habilidad, categoría, descripción, tipos y debilidades.

—

6. Arquitectura interna (MVC)

6.1. Modelo (com.example.intentopokedex3.model)

- **Pokemon.java:** Clase que representa un Pokémon individual. Contiene: nombre, número, imagenUrl, tipos, altura, peso y habilidad.
- **PokedexApi.java:** Gestiona las peticiones HTTP a la API.

—

6.2. Controlador (com.example.intentopokedex3.controller)

Controla la lógica entre las vistas y el modelo (`PokedexApi`). Gestiona hilos y comunicación entre UI y datos. En este proyecto, las *Activities* realizan funciones de controlador directo.

—

6.3. Vista (com.example.intentopokedex3.view)

Archivo	Descripción
ActivityInicio.java	Pantalla inicial con el logo.
ActivityTipos.java	Muestra botones de tipos y buscador.
ActivityLista.java	Lista de Pokémon del tipo seleccionado, con paginación.
ActivityDetalle.java	Ficha detallada del Pokémon con información traducida al español.

—

7. Conexión con la API

Base URL: `https://pokeapi.co/api/v2/`

Endpoints utilizados:

- `type/{tipo}` → lista de Pokémon por tipo.
- `pokemon/{nombre|id}` → datos principales.
- `pokemon-species/{id}` → categoría y descripción.
- `ability/{nombre}` → nombre de la habilidad en español.

Formato: JSON – Conversión manual mediante `JSONObject` y `JSONArray`.

—

8. Explicación del código y sus clases

En este apartado se detalla la función de cada clase, pantalla (Activity) y los métodos principales implementados en la aplicación.

—

8.1. Modelo

Pokemon.java

Clase que representa un Pokémon individual. Contiene los atributos:

- `nombre` – nombre del Pokémon.
- `numero` – identificador numérico.
- `imagenUrl` – enlace a la imagen oficial.
- `tipos` – lista de tipos (agua, fuego, planta...).
- `altura`, `peso`, `habilidad` – datos básicos del Pokémon.

—

PokedexApi.java

Controlador que gestiona la comunicación con la **PokeAPI**. Realiza todas las peticiones HTTP y devuelve los datos al resto de la aplicación.

- `obtenerPokemonPorTipo(tipo, listener)` Ejecuta la búsqueda en un hilo independiente y devuelve los Pokémon del tipo indicado (fuego, agua, planta, etc.). Llama internamente a `obtenerPokemonPorTipoSync()`.
- `obtenerPokemonPorTipoSync(tipo)` Descarga hasta 30 Pokémon del tipo seleccionado desde la API y crea los objetos `Pokemon` con sus datos básicos.
- `obtenerPokemonPorTipoSync(tipo, offset, limite)` Versión mejorada con paginación manual. Permite cargar más Pokémon por bloques de 30 usando el botón "Ver más".

- `obtenerDetallePokemon(url)` Recupera los datos completos de un Pokémon concreto (nombre, id, imagen, tipos, altura, peso, habilidad).
- `buscarPorNombre(texto, listener)` Busca Pokémon por nombre o número introducido en el buscador. Si el texto coincide parcialmente (por ejemplo "char"), muestra todos los que comienzan así. Utiliza `obtenerDetallePokemon()` para completar los datos.
- `descargarImagen(url, listener)` Descarga y convierte la imagen del Pokémon a formato `Bitmap` para mostrarla en pantalla. Se usa en las pantallas de lista y detalle.

—

8.2. Vista (Activities)

ActivityInicio.java

Primera pantalla que se muestra al abrir la app. Contiene el logo de Pokédex y un botón que redirige a la pantalla de tipos mediante un `Intent`. Sirve como introducción o pantalla de carga.

—

ActivityTipos.java

Pantalla que muestra todos los botones de tipos (Agua, Fuego, Planta, etc.). Al pulsar un tipo:

- Traduce el nombre al inglés para la API (por ejemplo, "planta" → "grass").
- Abre `ActivityLista` con el tipo seleccionado.

También incluye un buscador superior que permite buscar Pokémon por nombre o número, mostrando sugerencias en un `ListView` flotante.

—

ActivityLista.java

Pantalla que muestra una cuadrícula con los Pokémon del tipo seleccionado. Usa un `GridLayout` y carga los resultados en bloques de 30 mediante el botón "Ver más".

Principales métodos:

- `cargarPokemonsPorTipo(tipo)` – inicia la descarga inicial del tipo elegido.

- `cargarMasPokemons()` – carga el siguiente bloque de 30 Pokémon (paginación).
- `mostrarPokemons(lista)` – crea dinámicamente las tarjetas con la imagen y nombre de cada Pokémon.
- `configurarBuscador()` – permite buscar Pokémon por nombre o número y abrir su detalle.

—

ActivityDetalle.java

Pantalla que muestra la ficha completa de un Pokémon concreto. Recibe los datos desde la lista mediante un `Intent`.

Flujo principal:

1. Recupera el nombre, número e imagen del Pokémon recibido.
2. Descarga los datos completos desde la API (altura, peso, habilidad, tipos).
3. Muestra la información en un cuadro central con bordes redondeados.
4. Genera los chips de **tipo** y **debilidad** dinámicamente con colores personalizados.
5. Permite volver a la lista o al inicio mediante los botones inferiores.

Métodos destacados:

- `cargarDetallesPokemon(nombre)` – obtiene y muestra todos los datos del Pokémon.
- `mostrarDetalles(pokemon)` – rellena los campos del layout con los valores del objeto `Pokemon`.
- `mostrarTipos(tipos)` – crea chips para cada tipo (por ejemplo, Agua o Planta).
- `cargarDebilidades(tipos)` – consulta las debilidades del Pokémon y las muestra como chips.
- `crearChip(tipo)` – genera visualmente cada chip de color con un fondo redondeado.

—

8.3. Relación entre pantallas

- **ActivityInicio** → abre **ActivityTipos**.
- **ActivityTipos** → abre **ActivityLista** con el tipo seleccionado.
- **ActivityLista** → abre **ActivityDetalle** con los datos del Pokémon pulsado.
- **ActivityDetalle** → puede volver a Inicio o a la Lista anterior.

Cada transición usa la clase `Intent` para enviar datos entre pantallas (nombre, número e imagen del Pokémon).

—

8.4. Gestión de asincronía

Todas las peticiones a la API se realizan en hilos separados usando `ExecutorService` o `Thread`, y los resultados se devuelven al hilo principal mediante `Handler` o `runOnUiThread()`. Esto garantiza que la interfaz gráfica nunca se bloquee durante la carga de datos o imágenes.

—

8.5. Resumen visual de flujo de datos

```
ActivityInicio
   -> ActivityTipos
   -> (selección de tipo)
   -> ActivityLista
   -> (selección de Pokémon)
   -> ActivityDetalle
      muestra datos completos
```

—

9. Internacionalización (Español)

Elemento	Fuente	Traducción
Habilidad	/ability/{nombre}	names[language=es].name
Categoría	/pokemon-species/{id}	genera[language=es].genus
Descripción	/pokemon-species/{id}	flavor_text_entries[language=es].flavor_text
Tipos (UI)	Mapeo interno	fire→Fuego, grass→Planta, etc.

—

10. Características técnicas

- **Lenguaje:** Java
- **IDE:** Android Studio
- **Arquitectura:** MVC
- **API:** PokeAPI
- **Diseño:** XML + Drawables personalizados
- **Permisos:** INTERNET
- **Librerías:** AndroidX, FlexboxLayout

—

11. Funcionamiento del buscador

Basado en `EditText` + `TextWatcher`. Realiza llamadas a `PokedexApi.buscarPorNombre()` en un hilo separado. Si el texto cambia durante la búsqueda, se descarta el resultado anterior. Permite buscar por nombre parcial o número. Muestra una lista flotante (`ListView`) con resultados.

—

12. Rendimiento y asincronía

Uso de `ExecutorService` y `Handler` para evitar bloqueos en la UI. Carga por bloques (paginación) y descarga de imágenes en hilos independientes.

—

13. Diseño visual

Temática Pokédex: fondo morado degradado y logo inferior. Cajas de detalle con esquinas redondeadas. Botones personalizados y chips de tipo/debilidad generados dinámicamente con colores.

—

14. Estructura de carpetas

```
com.example.intentopokedex3
|-- model
|   |-- Pokemon.java
|-- controller
|   |-- PokedexApi.java
|-- view
|   |-- ActivityInicio.java
|   |-- ActivityTipos.java
|   |-- ActivityLista.java
|   |-- ActivityDetalle.java
-- res
    |-- layout/
    |-- drawable/
    -- values/
```

—

15. Pruebas realizadas

Búsqueda por nombre (ej. “char” → Charmander, Charizard). Búsqueda por número (ej. “25” → Pikachu). Traducciones correctas (habilidad, descripción, categoría). Paginación fluida (bloques de 30). Carga de debilidades exacta.

—

16. Mejoras futuras

- Implementar RecyclerView en lugar de GridLayout.
- Añadir favoritos offline (Room Database).
- Cachear imágenes (Glide o LruCache).
- Modo oscuro y accesibilidad.

—

17. Mockups y referencias visuales

Figura 1: *
Pantallas de la aplicación Pokédex Android

—

18. Créditos

Proyecto académico desarrollado por **Ana Núñez**, como parte del módulo *Desarrollo de Aplicaciones Multiplataforma (DAM)*. Datos obtenidos de la API oficial PokeAPI.

Pokédex Android – Octubre 2025
Desarrollado en Java + Android Studio.
Diseño y código original por Ana Núñez.