Pontificia Universidad Católica Madre y Maestra

CSTI-1870-5238 - Desarrollo Móvil

Tema:

Avance Pokédex - Desarrollo Móvil

Integrantes:

1014-3525 Randae Garcia Sanchez

1014-3611 Starlin Frías Luna

Profesor:

Freddy Peña

Santiago de los Caballeros, República Dominicana

Miércoles 4 de Diciembre del 2024

Índice

Introducción:

En el presente documento se encuentra la documentación con respecto al proyecto final de desarrollo móvil, el cual consiste en la implementación de una Pokédex utilizando el framework de Flutter. Este mismo debe ser desarrollado con el uso de las mejores prácticas, además, con una excelente calidad de código, sin mencionar la importancia de un buen diseño tanto en lo visual, como en el backend para lograr mayor fluidez y experiencia de los usuarios. El proyecto se desarrolló utilizando técnicas como Graphql para consultar el API de Poké API la cual contiene la información de miles de pokemones junto con sus evoluciones y cualquier información útil de los mismos. Esta es la consola utilizada para las consultas: https://beta.pokeapi.co/graphql/console/.

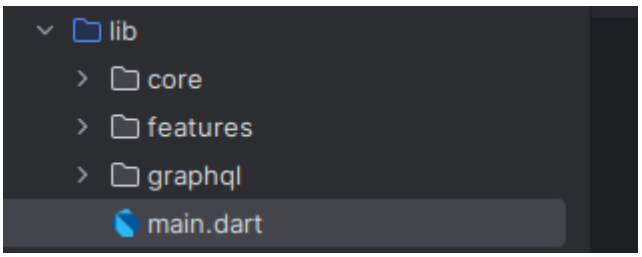
Diseño:

El diseño que utilizamos fue con el objetivo de separar y modular cada componente y widget lo más posible, todo con el propósito de lograr la reutilización de los mismos para así evitar las duplicaciones de código innecesario que lo único que hacen es complicar y dificultar la escalabilidad del proyecto. Por ello, creamos un archivo.dart por cada widget, además, del uso de diferentes modelos para pasear los datos del JSON obtenido del API a estos, así logramos un fácil acceso a los datos, y un manejo de una forma más organizada.

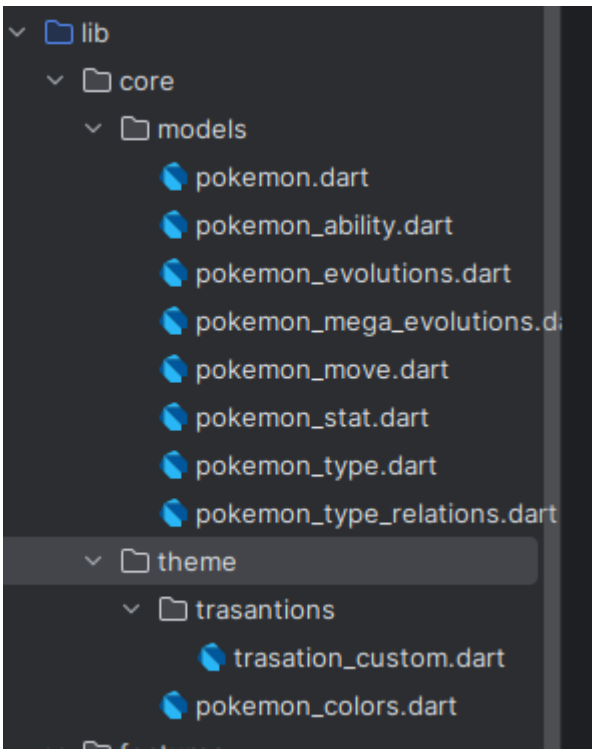
Estructura:

El proyecto cuenta con la siguiente estructura de archivos y directorios, esta misma facilita la integración de nuevos widgets, como además la escalabilidad a nuevas características que se deseen añadir, sin mencionar el fácil entendimiento para personas ajenas al código

En primer lugar contamos con 3 directorios principales y el archivo main.dart dentro de lib, los cuales se visualizan de la siguiente manera:

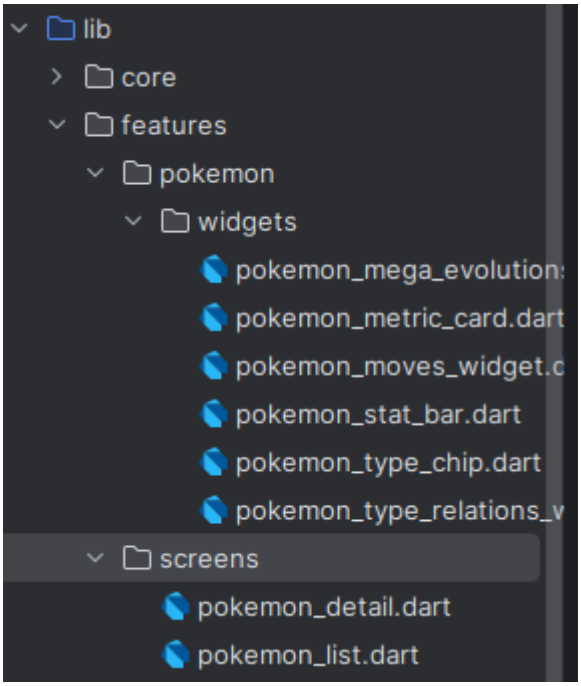


En el directorio de core tenemos 2 directorios mas que son models y themes, el primero cuenta con los diferentes modelos que se utilizan para encapsular los datos recibidos en el JSON desde el API, el segundo por otro lado contiene las transacciones para los cambios de pantalla y los colores para los diferentes tipos de Pokemones. De esta forma se visualiza:



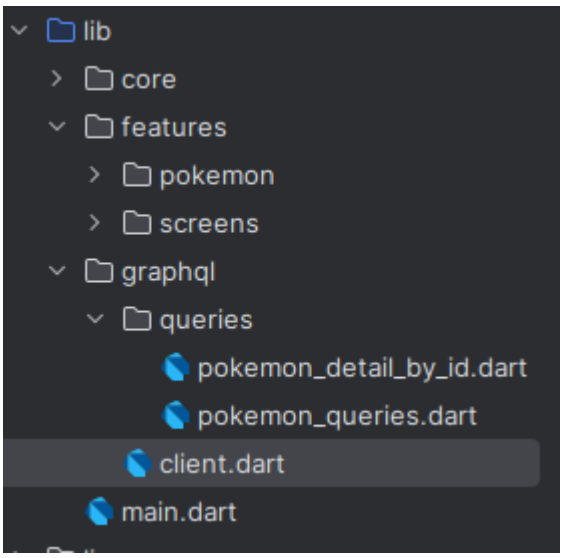
En el directorio principal de features contamos con un directorio llamado “Pokemon” el cual tiene otro llamado “widgets” el cual tiene dentro los diferentes componentes para ser utilizados tanto en la pantalla de detalle como en la lista en general con respecto a los

Pokemones. Además, aquí dentro de este directorio llamado “features” se tiene otra carpeta llamada “screens” la cual contiene las diferentes pantallas a mostrar al usuario, en este caso la lista y la pantalla de detalle.



Por último, tenemos el directorio de “graphql” la cual contiene una carpeta llamada “queries” en esta se encuentran las diferentes consulta a la API de:
“https://beta.pokeapi.co/graphql/v1beta”

De forma directa contamos con un archivo llamado “client.dart” el cual como su nombre lo indica es la creación del cliente para realizar las peticiones a la API.



Uso de la API GraphQL:

En esta sección se explicará el uso de la API con graphql en el proyecto, desde como funciona a cómo se realiza. Esta documentación cubre la implementación de GraphQL que se conecta con el endpoint GraphQL Beta de PokéAPI. La aplicación utiliza el paquete graphql_flutter para gestionar las operaciones GraphQL.

- **Configuración del Cliente GraphQL**

  La aplicación inicializa un cliente GraphQL en `client.dart`:

```dart
import 'package:flutter/material.dart';
import 'package:graphql_flutter/graphql_flutter.dart';

ValueNotifier<GraphQLClient> setupGraphQLClient() {
  final HttpLink httpLink = HttpLink(
    'https://beta.pokeapi.co/graphql/v1beta',
  );

  return ValueNotifier(
    GraphQLClient(
      link: httpLink,
      cache: GraphQLCache(store: InMemoryStore()),
    ),  // GraphQLClient
  );  // ValueNotifier
}
```

Utiliza HttpLink para conectarse al endpoint GraphQL de PokéAPI. Implementa caché en memoria con GraphQLCache para lograr mayor velocidad. Devuelve un cliente envuelto en ValueNotifier para actualizaciones reactivas.

- **Integración en la Aplicación**

El cliente GraphQL se integra en la raíz de la aplicación en main.dart:

```dart
import 'package:flutter/material.dart';
import 'package:graphql_flutter/graphql_flutter.dart';
import 'graphql/client.dart';
import 'features/screens/home_page.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return GraphQLProvider(
      client: setupGraphQLClient(),

      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: "Pokedex Flutter",
        theme: ThemeData(
            primarySwatch: Colors.red
        ),  // ThemeData

        home: const HomePage(),
      ),  // MaterialApp
    );  // GraphQLProvider
  }
}
```

El widget GraphQLProvider hace que el cliente esté disponible en todo el árbol de widgets.

Sistema de Listado de Pokémon

El sistema implementa un desplazamiento infinito eficiente para mostrar los datos de los Pokémon. La aplicación obtiene los Pokémon en lotes paginados de 20 elementos para optimizar el rendimiento y el uso de memoria. Cada Pokémon se muestra en una cuadrícula de dos columnas, presentando información esencial como el número del Pokémon, nombre, tipos y su imagen sprite. Los elementos de la cuadrícula presentan un fondo con degradado que refleja los colores del tipo del Pokémon, creando una interfaz visualmente atractiva e intuitiva.

El sistema de listado utiliza consultas GraphQL para obtener datos del backend, implementando varias políticas de obtención para optimizar la recuperación de datos y el almacenamiento en caché. El sistema mantiene la capacidad de respuesta mostrando indicadores de carga durante la obtención de datos y estados de error apropiados cuando es necesario.

Funcionalidad de Ordenamiento

La aplicación proporciona un sistema integral de ordenamiento que permite a los usuarios organizar los Pokémon según varios criterios:

- Criterios de Ordenamiento
    1. Número de Identificación (ID): Organiza los Pokémon en su orden numérico oficial
    2. Nombre: Ordena los Pokémon alfabéticamente por sus nombres
    3. Tipo: Agrupa y ordena los Pokémon por sus tipos primarios
    4. Estadísticas Base Totales: Ordena los Pokémon según sus estadísticas base combinadas
    5. Habilidad: Organiza los Pokémon por sus habilidades

Cada opción de ordenamiento admite orden ascendente y descendente, implementado a través de consultas GraphQL que manejan la lógica de ordenamiento a nivel de base de datos para un rendimiento óptimo.

Sistema de Filtrado

La aplicación cuenta con un sistema robusto de filtrado que permite a los usuarios refinar su búsqueda de Pokémon basándose en múltiples criterios simultáneamente:

Categorías de Filtros

1. Generaciones: Los usuarios pueden filtrar Pokémon de generaciones específicas (I a IX), cada una representando diferentes eras de los juegos Pokémon
2. Tipos: Permite filtrar por cualquiera de los 18 tipos de Pokémon (Normal, Fuego, Agua, etc.)
3. Habilidades: Proporciona una funcionalidad de búsqueda con autocompletado para filtrar Pokémon por sus habilidades específicas

El sistema de filtrado implementa actualizaciones en tiempo real, refrescando automáticamente la lista de Pokémon a medida que se aplican o eliminan filtros. Los usuarios pueden combinar múltiples filtros entre diferentes categorías, y el sistema incluye una opción conveniente de "Limpiar filtros" para restablecer todas las selecciones.

Gestión de Favoritos

La aplicación incluye un sofisticado sistema de favoritos que permite a los usuarios mantener su colección personal de Pokémon preferidos:

Características Principales

1. Almacenamiento Persistente: Utiliza SharedPreferences para mantener la lista de favoritos del usuario entre sesiones de la aplicación
2. Funcionalidad de Alternancia: Los usuarios pueden agregar o eliminar Pokémon de sus favoritos mediante un gesto de doble toque o a través de la pantalla de detalles
3. Retroalimentación Visual: Presenta un ícono de corazón animado que proporciona retroalimentación visual inmediata del estado de favorito de un Pokémon
4. Vista Dedicada: Incluye una pestaña separada para ver los Pokémon favoritos, manteniendo el mismo diseño de cuadrícula y funcionalidad que la lista principal
5. Estado Sincronizado: El estado de favorito se sincroniza en todas las vistas de la aplicación, asegurando la consistencia en la interfaz de usuario

El sistema de favoritos mantiene el orden de clasificación basado en el ID del Pokémon e implementa su propio sistema de paginación para garantizar un rendimiento fluido incluso con grandes colecciones de Pokémon favoritos.

Sistema de Compartir Card:

Esta funcionalidad se agregó en la pantalla de detalle, con el objetivo de permitir a los usuarios la capacidad de estos compartir cartas de Pokémon en cuestión. Al presionar en este botón de compartir se extraen las cartas del API de TCG, pero también se crea una personalizada mediante un widget que extrae la información de la pantalla de detalle.

Para la creación de la carta personalizada se creó una clase que recibe dos parámetros, el Pokemon y el card key (identificador de la carta), esta clase es utilizada por los diferentes widget para formar la sección de la imagen, como el contenedor del tipo, debilidad y otros aspectos del mismo.

También como se comenta, esta funcionalidad cuenta con la extracción de las cartas de Trading Collection Game (TCG), el cual ofrece un API donde permite hacer uso de las mismas. Esta es la API de la cual es extraída https://api.pokemontcg.io/v2. Para esto se implementó un servicio el cual consulta dicha API recibiendo como parámetro el nombre del Pokémon en cuestión.

Este es el fragmento del código encargado de esto:

```dart
import 'dart:convert';
import 'package:http/http.dart' as http;

import '../models/pokemon_tcg_models.dart';

class PokemonTCGService {
  static const String baseUrl = 'https://api.pokemontcg.io/v2';

  static Future<List<PokemonCard>> searchCardsByPokemonName(String pokemonName) async {
    try {
      final response = await http.get(
        Uri.parse('$baseUrl/cards?q=name:"$pokemonName"'),
      );

      if (response.statusCode == 200) {
        final data = json.decode(response.body);
        return (data['data'] as List)
            .map((card) => PokemonCard.fromJson(card))
            .toList();
      } else {
        throw Exception('Failed to load cards');
      }
    } catch (e) {
      print('Error fetching cards: $e');
      return [];
    }
  }
}
```

Pantalla de Comparaciones:

La aplicación cuenta con un screen con el objetivo de permitir la comparación entre 2 Pokemones. La pantalla cuenta con un buscador por ID de cada Pokémon, donde estos id son enviados a una función Graphql para extraer los detalles de los mismos. Esta es la declaración de la función “Future<Pokemon> fetchPokemonDetails(int id) async”, la cual nos permite extraer los detalles.

Una vez que los datos del Pokémon son extraídos, se utiliza una librería en Flutter la cual nos permite la construcción de gráficos. El nombre de esta librería es “fl_chart”, usamos la versión 0.66.0.

La librería la ayudamos de ciertos widget que nos facilitan la construcción de forma atractiva de la pantalla para permitir mayor visualización al usuario.

Voces de los Pokemones:

Para hacer uso de las voces y los gritos de cada Pokemon y poder integrarlos a la pantalla de detalle fue necesario consultar otra API diferente de la Poké Api, contamos con que Poké Api no contenía las voces de los mismo de la forma en la cual la buscábamos.

Para la implementación de esto se utilizaron 2 librerías diferentes, las cuales permitieron dicha funcionalidad, estas fueron :

1. just_audio: ^0.9.36
2. just_audio_background: ^0.0.1-beta.11

Cada una realiza su tarea, una nos permite escuchar el audio en la pantalla ahi mismo, mientras que otra de background.

La URL que utilizamos para extraer las voces y gritos de los mismo es esta: https://pokemoncries.com/cries/${widget.pokemonId}.mp3';

Esta misma le pasamos el id del Pokémon, el cual es extraído del widget que contiene al mismo.