PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA
FACULTAD DE CIENCIAS E INGENIERÍA
ESCUELA DE INGENIERÍA EN COMPUTACIÓN Y TELECOMUNICACIONES

DESARROLLO DE APLICACIONES MÓVILES
Proyecto Final
Desarrollo de una Pokédex con Flutter y GraphQL

Objetivo General:

El propósito de este proyecto es **diseñar, desarrollar y presentar una aplicación móvil multiplataforma** construida con **Flutter y GraphQL**, que funcione como una **Pokédex interactiva**, integrando aspectos de diseño UI/UX, manejo de datos remotos, almacenamiento local, accesibilidad, y características avanzadas de interacción y gamificación. Para obtener información detallada sobre diversas especies de Pokémon use la API de GraphQL

- Documentación: https://pokeapi.co
- GraphQL client: https://graphql.pokeapi.co/v1beta2/console

El proyecto será realizado en equipos de máximo dos personas y será evaluado en función de la funcionalidad, calidad del código, interfaz de usuario, creatividad, y documentación técnica.

Objetivos Específicos

- Aplicar los principios de **arquitectura limpia** (Clean Architecture) en Flutter.
- Consumir servicios **GraphQL** de la API de PokeAPI con **paginación** y **cache local**.
- Implementar **persistencia local avanzada** con Hive, Isar u otras.
- Desarrollar una **experiencia de usuario fluida, accesible y visualmente atractiva**.
- Incorporar **características de gamificación** para fomentar la interacción.
- Documentar y versionar correctamente el proyecto en GitHub, usando buenas prácticas de ingeniería.

Requerimientos Técnicos (Obligatorios)

1. **Interfaz de Usuario (UI/UX) — 25%**
    a. Mostrar una lista de Pokémon con su nombre, imagen, tipo y número.
    b. Barra de búsqueda con debounce (300–500 ms).
    c. Diseño moderno, accesible y responsivo
    d. Uso de Material Design 3 y temas dinámicos.
    e. Visualización de estadísticas con gráficos tipo Radar Chart
    f. Pantalla de error con mensajes y opción de “Reintentar”.

    Pantalla de detalles

    g. **Identidad:**
        i. Nombre, número (#dex nacional), tipos (1–2), sprite/official-artwork.
        ii. Criterios: si hay 2 tipos, mostrar ambos; si falta imagen oficial, usar sprite fallback.
    h. **Stats base** (HP, Atk, Def, SpA, SpD, Spe) + total
        i. Radar chart o barras.

ii. Criterios: cada stat muestra valor y porcentaje; total visible.

i. **Habilidades (Abilities)**
   i. Lista con: nombre, isHidden, breve efecto
   ii. Criterios: marcar “Oculta” cuando isHidden==true; efecto resumido (máx. 140–160 chars).

j. **Evoluciones (Chain/Branches)**
   i. Línea/árbol simple: pre-evo → evo1 → evo2 (soporta ramas).
   ii. Muestra trigger principal (nivel, objeto, intercambio, amistad, hora).
   iii. Criterios: si no tiene evoluciones (legendarios, míticos), mostrar estado vacío “No evoluciona”.

k. **Movimientos (Moves)**
   i. Ámbito acotado: por defecto solo version group/juego seleccionado y método level-up.
   ii. Filtros: método (level-up/TM/Tutor/Egg), orden por nivel o nombre.
   iii. Criterios: paginación local (virtual list) o por secciones; ver al menos 20 sin lag.

l. **Acciones**
   i. Favorito (toggle con animación), Compartir (card), botón “Abrir en mapa/regiones” si aplica.
   ii. Criterios: cambio de favorito persiste en <150 ms y se refleja al volver a la lista.

m. **Matchups (daño por tipo)**
   i. Debilidades/resistencias/inmunidades (x4, x2, x0.5, x0.25, x0).

n. **Peso/altura/egg groups (metadatos rápidos).**

o. **Variantes/formas**:
   i. dropdown (Alolan, Galarian, etc.) → cambia sprite/tipos/moveset.

p. Shiny toggle (si hay assets).

2. **Uso de GraphQL — 10%**
   a. Integración completa con la API GraphQL de PokeAPI.
   b. Consultas optimizadas y paginación basada en cursor.
   c. Uso de cache local y control de errores (timeout, rate limit, sin conexión).

3. **Gestión de Estado y Arquitectura — 15%**
   a. Implementar una arquitectura de 3 capas: *data/, domain/, presentation*/.
   b. Uso de Riverpod o BLoC para la gestión del estado (puede ser otro).
   c. Separación clara de responsabilidades, modelos, DTOs y mapeos.

4. **Filtrado y Ordenación — 10%**
   a. Filtrar Pokémon por nombre, dropdown (tipo, generación, región, habilidades o poder).
   b. Ordenar por nombre, número o poder.
   c. Mantener los filtros activos entre sesiones.

5. **5. Favoritos y Persistencia Local — 10%**
   a. Guardar Pokémon favoritos utilizando persistencia local
   b. Mostrar lista de favoritos en vista dedicada.
   c. Habilitar modo offline para acceder a los favoritos y al último listado visto.

6. **Animaciones y Transiciones — 5%**
    a. Implementar animaciones Hero, microinteracciones y transiciones suaves.
    b. Animación al agregar/quitar favoritos o cambiar entre vistas.

7. **Compartir Pokémon — 5%**
    a. Generar una “Pokémon Card” y compartirla como imagen.

8. **Mapa Interactivo — 5%**
    a. Mostrar en qué regiones o juegos aparece cada Pokémon.
    b. Usar mapas interactivos (flutter_map, leaflet, o similar).

9. **Accesibilidad e Internacionalización — 5%**
    a. Incorporar etiquetas Semantics y tamaños táctiles adecuados.
    b. Implementar soporte multilenguaje (español/inglés), por lo menos en la trivia.

10. **Sección Interactiva: “¿Quién es este Pokémon?” — 10%**
    a. Modo de juego tipo trivia:
        i. Mostrar la silueta del Pokémon y pedir al usuario que adivine su nombre.
        ii. Incluir sistema de puntuación, tiempo límite y ranking local.
        iii. Persistir local los resultados.
        iv. Desbloquear logros visuales al alcanzar puntajes altos.

Requisitos Adicionales (Opcional)

- Modo Oscuro y Claro.
- Onboarding introductorio animado al iniciar la app.

Documentación

- Proporcionar una documentación técnica clara que explique el funcionamiento de la aplicación, el uso de la API GraphQL y las decisiones de diseño, esta debe ser en el archivo ***readme.md***.
- La documentación debe incluir:
    - Explicación del uso de GraphQL en la aplicación.
    - Detalles sobre la organización del código y las características adicionales implementadas.

Evaluación y Entrega

Entrega:

- Repositorio GitHub público o privado compartido.
- Demo presencial obligatoria (10–12 minutos).
- Evaluación grupal e individual basada en aportes reales (commits, PRs, issues).
- Requisitos mínimos: 80 % de cumplimiento total.

Valoración:

Criterio	**Ponderación**
Interfaz (UI/UX)	20 %
GraphQL (paginación, cache, eficiencia)	15 %
Arquitectura limpia y gestión de estado	15 %
Filtrado y Ordenación	10%
Favoritos y Persistencia Local	10 %
Animaciones, transiciones y performance	5 %
Compartir Pokémon	5%
Accesibilidad e Internacionalización	5%
Mapas y visualizaciones interactivas	5 %
Sección Interactiva	10 %

Total: 100 puntos