Prueba técnica — Flutter Mobile + Web (Pokédex)

Fuente de datos: PokéAPI (https://pokeapi.co/)

Contexto del rol

Buscamos evaluar tu capacidad para construir y evolucionar un producto en Flutter con foco en:

- Calidad de frontend (UX/UI, estados, performance).
- Compatibilidad multi-plataforma: iOS/Android y Flutter Web.
- Arquitectura y criterio técnico (trade-offs razonables en un timebox corto).
- Robustez: manejo de red, errores y soporte offline.

Timebox

Máximo 1 día de desarrollo efectivo. Se valora más un alcance pequeño bien resuelto que muchas features incompletas.

Alcance funcional requerido

1) Listado de Pokémon

Construye una pantalla "Pokédex" que muestre un listado de Pokémon obtenido desde la PokéAPI.

Datos mínimos por ítem (cómo obtenerlos queda a tu criterio):

- Nombre
- Identificador (id o número en Pokédex)
- Imagen (thumbnail)

Requisitos de UX:

- Estados: loading / error / empty.
- Paginación: el listado debe poder navegarse por páginas (cargar más, scroll infinito o paginado clásico).

2) Pantalla de detalle

Al seleccionar un Pokémon desde el listado, muestra una pantalla de detalle con información relevante (a elección). Debe venir de un endpoint de detalle de PokéAPI.

3) Soporte online/offline

La app debe funcionar online (consumiendo API) y offline mostrando datos persistidos previamente.

4) Manejo de estado

La aplicación debe implementar el manejo de estado utilizando Cubit (flutter_bloc).

5) Flutter Web + Responsive + Landscape

Debe ejecutarse en iOS/Android y Flutter Web (Chrome).

Requisitos de UI:

- Responsive (al menos 2 layouts por ancho: mobile vs desktop/tablet).
- Landscape soportado.

Endpoints requeridos (PokéAPI v2)

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

La API es solo lectura (GET) y no requiere autenticación.

A) Endpoint para listado (resource list)

Objetivo: obtener una lista paginada de Pokémon.

Método	URL
GET	https://pokeapi.co/api/v2/pokemon?limit={n}&offset={m}

B) Endpoint para detalle

Objetivo: obtener el detalle de un Pokémon y su información para UI (incluye sprites).

Método	URL
GET	https://pokeapi.co/api/v2/pokemon/{id o name}/

Nota: el endpoint de listado no incluye imágenes. Decidir cómo resolver la imagen del listado sin disparar cientos de requests es parte del ejercicio.

Entregable

- Link a repositorio (GitHub/GitLab/Bitbucket) o archivo comprimido.
- Incluir un README con instrucciones para correr en mobile y web.

Preguntas obligatorias para el README

- Arquitectura y escalabilidad: ¿Qué arquitectura/patrón usaste y por qué es adecuado para escalar a un producto real (incluyendo Web)?
- ¿Qué trade-offs tomaste por el timebox de 1 día?

- Gestión de estado y side-effects: Describe tu flujo "UI → estado → datos" y cómo evitas acoplamiento entre capas.
- Offline y caché: Explica tu estrategia de persistencia: qué guardas, cómo versionas/invalidas, cómo resuelves conflictos entre "dato cacheado" y "dato remoto".
- Flutter Web: ¿Qué decisiones tomaste para que la experiencia en Web sea buena (responsive, navegación, performance, interacción tipo desktop)? ¿Qué limitaciones tuviste/anticipas y cómo las mitigarías?
- Calidad: Menciona 3 decisiones de "código limpio" aplicadas (con ejemplo puntual del repo).
- Testing: ¿Qué testeaste y por qué? Si no alcanzaste, ¿qué tests agregarías primero (prioridad) y qué asegurarían?
- Git: ¿Cómo estructuraste tus commits (granularidad, mensajes, convención) para facilitar review y mantenimiento?
- Pendientes: ¿Qué dejaste fuera? Lista priorizada (top 3-5) con cómo lo implementarías.