# Prueba Técnica Flutter

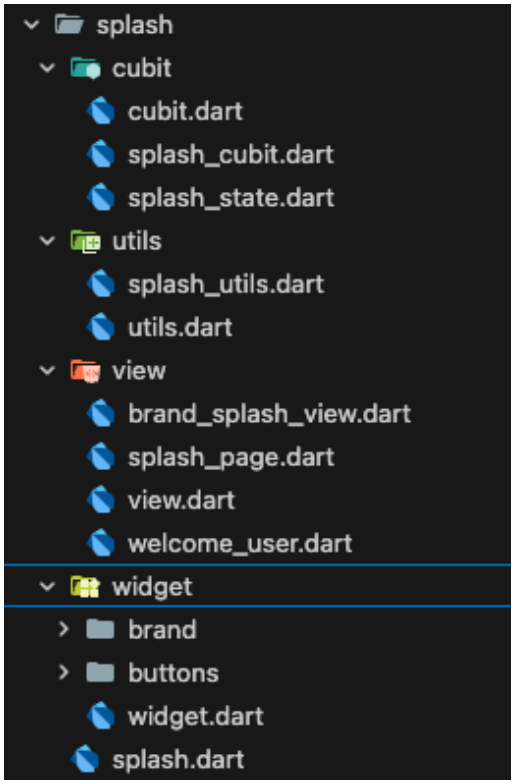
## Descripción del Proyecto

Desarrollar una aplicación móvil en Flutter que permita al usuario gestionar una lista de ítems obtenidos desde una API pública.
La aplicación debe cumplir con las siguientes funcionalidades principales:
1. Consultar y mostrar ítems provenientes de una API pública.
2. Seleccionar un ítem y asignarle un nombre personalizado, almacenando la información de forma local.
3. Listar todos los elementos guardados en la base de datos local.
4. Editar o eliminar cualquier elemento guardado.
5. Visualizar el detalle completo de cada elemento proveniente de la API.

## Alcance y Tiempo de Entrega

- Duración máxima: 72 horas desde la recepción del enunciado.
- Entrega: En un repositorio Git, que contenga:
  - Código fuente completo y funcional.
  - Archivo README.md con instrucciones claras de instalación, configuración y ejecución.
  - Diagrama de arquitectura o estructura de carpetas del proyecto.
  - Breve justificación de las decisiones técnicas clave tomadas durante el desarrollo.

## Estructura de Carpetas Sugerida



## Requisitos Funcionales

1. Clean Code & Buenas Prácticas
  - Código legible y modular.
  - Funciones cortas y con nombres descriptivos.
  - Sin duplicación de lógica.
  - Null Safety en todo el proyecto.

2. Gestión de Estado – Cubit
  - Implementar el patrón BLoC/Cubit, con al menos dos Cubits principales:
    • ApiCubit: para la obtención y gestión de datos desde la API.
    • PreferenceCubit: para el manejo de los ítems locales (CRUD).
  - Cada Cubit debe manejar correctamente los estados de loading, success y error.

3. Persistencia Local
  - Utilizar una base de datos local (Hive, Drift o SQLite).
  - Definir correctamente los modelos y adaptadores.
  - Implementar operaciones CRUD confiables y persistentes.

4. Consumo de API Pública
  - API de libre elección.
  - Mapear el JSON a modelos Dart utilizando fromJson / toJson.
  - Implementar estados: loading, success, error.
  - Indicador visual de carga (CircularProgressIndicator).
  - Manejo de errores con mensajes descriptivos y opción de reintento.

5. Interfaz de Usuario (UI)
  - Diseño responsive adaptable a distintos tamaños de pantalla.
  - Uso coherente de márgenes, paddings y tamaños relativos en todas las vistas.

6. Navegación
  - Usar rutas nombradas con Navigator o go_router.
  - Rutas requeridas:
    /api-list → Listado de ítems desde la API
    /prefs → Listado de ítems guardados
    /prefs/new → Crear nuevo gusto
    /prefs/:id → Detalle de un gusto específico

7. Documentación Mínima
  - Archivo README.md con descripción, requisitos, instalación y ejecución.
  - Comentarios en puntos clave del código.

## Pantallas Mínimas

1. Listado de Ítems de la API (/api-list)

- ListView de elementos, spinner de carga y manejo de error con botón “Reintentar”.

2. Crear Nuevo elemento en la BD (/prefs/new)
   - Selector para elegir ítem de la API, campo de texto para nombre personalizado, botones “Guardar” y “Cancelar”.

3. Lista de elementos Guardados (/prefs)
   - ListView de ítems guardados (nombre + mini imagen/ícono), opción de eliminar.

4. Detalle de elemento (/prefs/:id)
   - Mostrar nombre personalizado, imagen y datos relevantes. Botones “Eliminar” y “Volver”.

5. Pantalla Global de Carga
   - Widget centrado con CircularProgressIndicator.

6. Pantalla Global de Error
   - Mensaje amigable y botón “Reintentar”.

7. Buscador de Ítems
   - Campo de texto que permita buscar en la API en tiempo real con estados visuales.

## Criterios de Evaluación

Se evaluará principalmente:
- Claridad, legibilidad y organización del código.
- Uso correcto de Cubit y principios de arquitectura limpia.
- Implementación adecuada de persistencia local.
- Diseño UI coherente y adaptable.
- Documentación completa y facilidad de ejecución del proyecto.

Una vez finalice responder nuevamente el correo con el enlace a github y agregar todo lo que considere necesario.