Prueba - App Perritos

- Para realizar esta prueba debes haber estudiado previamente todo el material disponibilizado correspondiente al módulo.
- Una vez terminada la prueba, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el `.zip` en el LMS..
- Desarrollo prueba:
    - La prueba se debe desarrollar de manera Individual o Grupal.

Habilidades

- Crear una aplicación Android nativa en lenguaje Java.
- Utilizar un patrón de diseño MVP.
- Utilizar un sistema de control de versiones(Git). Como mínimo un commit por parte.
- Respetar las convenciones de nombres Java y seguir las buenas prácticas recomendadas.
- Seguir las recomendaciones SOLID.

Descripción

La empresa EvilCorp, que se dedica a prestar soluciones de desarrollo tecnológico a grandes empresas del retail, comenzó un nuevo proyecto junto con una empresa de alimentos para perro llamada “Perritos” que requiere una aplicación de razas de perro como Mínimo Producto Viable (MVP) para esta app.

Esta aplicación debe mostrar un listado de razas de perros y que cuando el usuario haga click en una raza en específico, muestre un listado de las fotos de perros de esa raza. Al desplegarse el listado de fotos de perros debemos añadir una función que, al hacer clic largo sobre una imagen, se guarde como ”Favorita”.

Requerimientos

- Consumir datos desde una API. Se sugiere dog.ceo (si decide consumir otra API es su responsabilidad realizar la conexión y mapeo de datos correspondiente, el requerimiento mínimo es un listado de atributos y listado de fotografías).
- Utilizar como EndPoints:
    - `breeds/list/` Para el listado de razas.
    - `breed/{breed}/images/` Para el listado de imágenes basándonos en una raza.
- Utilizar Single Activity y las vistas en Fragmentos.
- Crear un botón que muestre el listado de favoritos.
- Usar librerías externas para mostrar las imágenes (como Picasso o Glide).
- Utilizar Retrofit para la conexión a la API y Gson para el mapeo de datos.
- Hacer uso de Firebase(FireStore) para almacenar datos de favoritos.
- Utilizar el método que estime conveniente para unir las vistas (como **findViewById, ViewBinding o DataBinding**).
- Realizar test unitarios en el presentador.
- Almacenar la referencia a la imagen favorita en FireStore. (opcional)

Parte I - Modelo de la app:

1. Crear el modelo de la aplicación (ver “Imagen 1”):
    - Crear los POJOS necesarios para recibir la información de la API.
    - Crear el POJOS necesario para subir la colección de favoritos a `FireStore (raza, url, timeStamp)`.
        - **Observación:** Obtener el TimeStamp en formato String.

2. Crear `item_list_XXX.xml` que correspondan a cada elemento a mostrar.

3. Crear los Fragmentos necesarios:
    - Listado de razas.
    - Detalles.
    - Listado de favoritos (opcional).

4. Mostrar en un fragmento el `RecyclerView` con el listado de razas.

5. Mostrar en un fragmento el `RecyclerView` con el listado de fotos de la raza seleccionada.

6. Crear los adapters necesarios para transformar los distintos DataSet.

Parte II - Consumo y muestra de información:

1. Crear el Cliente Retrofit y la interface necesaria para la conexión.

2. Realizar la conexión a la API.

3. Crear el presentador que servirá para conectar la vista con el modelo. Se necesita más de un presentador.

4. Implementar los métodos en las vista correspondientes.

5. Instanciar los adapters y pasar los dataSets necesarios para cada uno de ellos.

Parte III - Guardar favoritos usando FireStore:

1. Implementar la funcionalidad para que, al hacer un clic largo en la imagen, este lleve los datos a FireStore.

2. Mostrar el detalle de los favoritos en un Fragmento de detalles (Opcional).

Pauta de Evaluación

- Parte I
- Parte II
- Parte III
- Utilización de buenas prácticas de código

Diagrama de clases

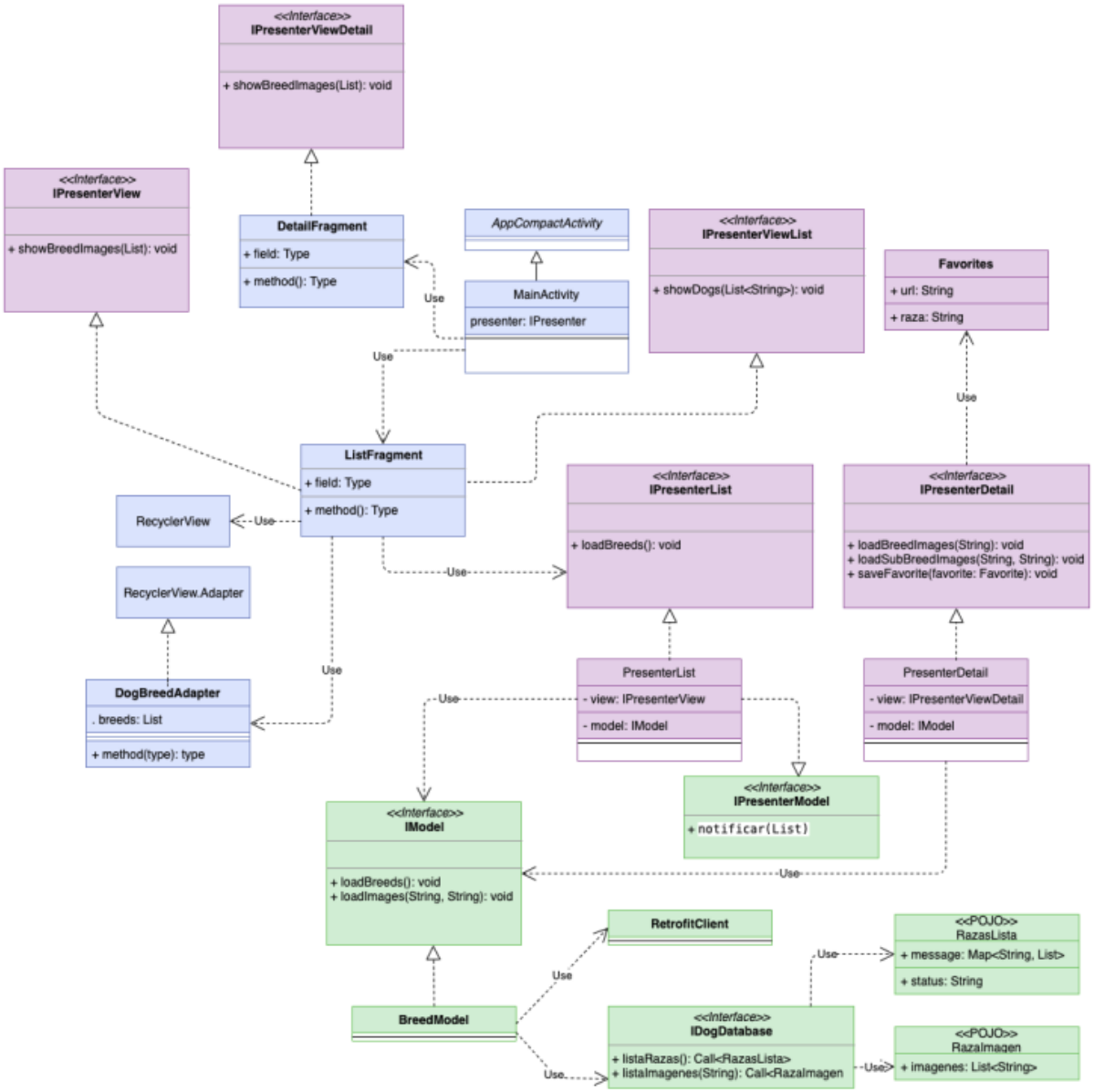

Imagen 1. Diagrama del modelo
Referencia: Desafío Latam