Recomendaciones de arquitectura para aplicaciones Android

Sirius

Protectora animal

Selene González Curbelo
Mariana Surós Álvarez

12/11/2023

ULPGC
Escuela de Ingeniería Informática

Contents

Selene González Curbelo
Mariana Surós Álvarez

Introducción

Este informe presenta cinco recomendaciones fundamentales para mejorar y optimizar el proyecto de la asignatura, que se centra en el desarrollo del proyecto final. Estas recomendaciones se basan en las directrices proporcionadas en el sitio web oficial de Android [1]. Cada recomendación se analiza en detalle, y se proporciona una decisión sobre si se seguirá en el proyecto, junto con una justificación que destaca por qué es esencial incorporarla en la aplicación.

1. Control de dependencias: inserción de dependencias

Descripción

Esta recomendación sugiere utilizar la inserción de dependencias, en particular, la inserción del constructor, siempre que sea posible.

Decisión

Seguir esta recomendación.

Justificación

La inserción de dependencias es crucial para garantizar que los componentes estén bien estructurados y sean fácilmente reemplazables. Esto permitirá que, por ejemplo, los módulos relacionados con la adopción, donaciones o apadrinamiento de animales sean independientes y se puedan cambiar o mantener sin afectar otras partes de la aplicación.

Esto promueve la modularidad y la mantenibilidad del código. Al utilizar la inserción del constructor, se facilita la inyección de dependencias y se promueve la cohesión y el desacoplamiento entre los diferentes componentes de la aplicación. Mejorando así, las pruebas unitarias y la detección de errores.

2. Convenciones de nombres: nombrar métodos

Descripción

Esta recomendación sugiere nombrar los métodos de manera descriptiva utilizando frases verbales, por ejemplo, "makePayment()".

Decisión

Seguir esta recomendación.

Justificación

Nombrar los métodos de manera descriptiva es fundamental para que el código sea comprensible y mantenible. Esto ayudará a identificar fácilmente las operaciones, como solicitar adopción, hacer donaciones o pedir apadrinamiento de un animal, lo que facilitará el desarrollo y la colaboración en el proyecto.

3. Capa de la IU: no enviar eventos del ViewModel a la IU

Descripción

Esta recomendación sugiere procesar los eventos directamente en el ViewModel y generar una actualización de estado con el resultado del control del evento en lugar de enviar eventos a la IU.

Decisión

Seguir esta recomendación.

Justificación

En nuestra aplicación, es fundamental procesar eventos en el ViewModel para mantener un flujo de datos unidireccional y simplificar la lógica de manejo de eventos. Por ejemplo, cuando un usuario solicita la adopción de un animal, procesar este evento en el ViewModel y actualizar el estado de adopción del animal será más eficiente y coherente.

Al seguir esta recomendación, se evita la necesidad de comunicación directa entre el ViewModel y la IU a través de eventos, lo que reduce la complejidad y acoplamiento en el sistema.

4. Arquitectura en capas: la capa de datos debería exponer los datos de la aplicación mediante un repositorio

Descripción

Esta recomendación sugiere que la capa de datos exponga los datos de la aplicación a través de un repositorio, evitando que los componentes de la capa de IU interactúen directamente con las fuentes de datos.

Decisión

Seguir esta recomendación.

Justificación

Utilizar un repositorio para exponer los datos de la aplicación es esencial en nuestro proyecto pues nos permitirá gestionar los datos de los animales en adopción, los usuarios, las donaciones y el apadrinamiento de manera centralizada. Además, si necesitamos cambiar la fuente de datos en el futuro, se podrá hacer sin afectar la capa de IU, logrando así, una mayor modularidad y flexibilidad en el sistema.

5. Arquitectura en capas: usar una capa de datos claramente definida

Descripción

Esta recomendación nos empuja a utilizar una capa de datos claramente definida que exponga los datos de la aplicación y contenga la mayoría de la lógica empresarial.

Decisión

Seguir esta recomendación.

Justificación

Una capa de datos bien definida es esencial para mantener organizada la lógica empresarial en la aplicación, pues al separar claramente los datos de la lógica de la IU, se puede mantener un

código limpio y escalable, lo que es fundamental para gestionar la información sobre los animales, los usuarios y las interacciones de adopción, donación y apadrinamiento en el refugio de animales, de esta manera, se logra una mayor claridad y cohesión en la estructura de la aplicación.

Bibliografía y fuentes de información

[1] https://developer.android.com/topic/architecture/recommendations?hl=es-419&authuser=1#handle-dependencies