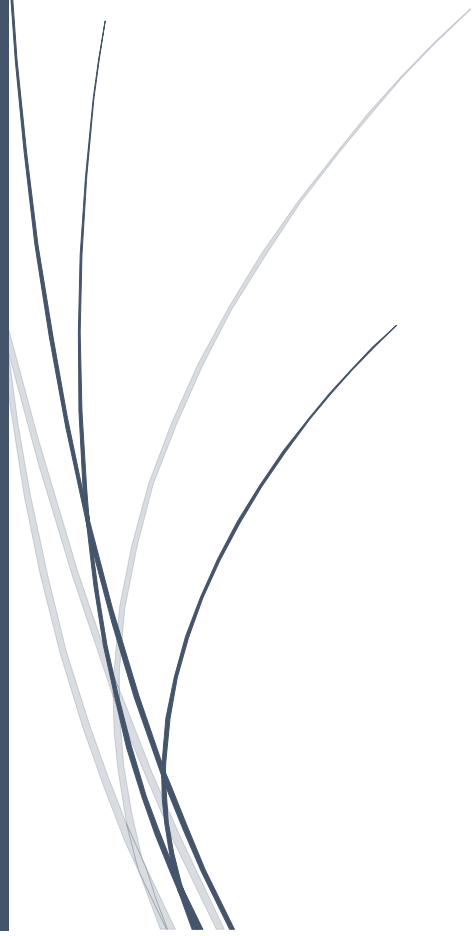
8-11-2023

Bloque II – S8 Recomendaciones de arquitectura para aplicaciones Android

Grupo 05: Acorán González, Javier Ocón y Santiago Emilio



Universidad de Las Palmas de Gran Canaria

PROGRAMACIÓN DE APLICACIONES MÓVILES NATIVAS

Contenido

1. Recomendaciones que consideramos más importantes o relevantes.

***Documentación**: para cada una de las 5 recomendaciones seleccionadas:*

- ***Descripción:** proporciona una breve descripción de la recomendación.*
- ***Decisión:** indica si van a seguir esta recomendación en el proyecto de la asignatura.*
- ***Justificación:** explica las razones detrás de tu decisión. ¿Por qué consideras que esta recomendación es esencial? ¿Qué beneficios aporta a tu proyecto?*

1.1 Usa una capa de datos claramente definida.

- Descripción: Esta recomendación sugiere establecer una capa de datos bien definida, lo que implica separar claramente la lógica de acceso a datos de otras capas de la aplicación.
- Decisión: Sí, esta recomendación sería seguida en el proyecto de la asignatura.
- Justificación: Al emplear una capa de datos claramente definida, se mejora la mantenibilidad, la legibilidad del código y la escalabilidad del proyecto. Además, permite futuras modificaciones y actualizaciones de la lógica de acceso a datos sin afectar otras partes de la aplicación.

1.2 La capa de datos debería exponer los datos de la aplicación mediante un repositorio.

- Descripción: La recomendación implica utilizar un repositorio para exponer los datos de la aplicación a través de la capa de datos.
- Decisión: Sí, esta recomendación sería seguida en el proyecto de la asignatura.
- Justificación: Al emplear un repositorio para exponer los datos, se establece una interfaz coherente y simplificada para acceder y manipular los datos. Esto mejora la abstracción y permite un acceso más estructurado a los datos, lo que facilita el mantenimiento y la comprensión del flujo de datos en la aplicación.

1.3 Usa la inserción de dependencias.

- Descripción: La recomendación indica utilizar la técnica de inyección de dependencias, que implica suministrar los objetos necesarios a una clase en lugar de que la clase misma los cree.
- Decisión: Probablemente esta recomendación sería seguida en el proyecto de la asignatura, aunque no es definitivo.
- Justificación: La inyección de dependencias facilita la modularidad y la reutilización del código al reducir el acoplamiento entre componentes. Esto hace que el código sea más mantenible, testeable y permite cambios más sencillos en el futuro al tener componentes más independientes y desacoplados entre sí.

1.4 Usa corrutinas y flujos

- Descripción: Esta recomendación se refiere al uso de corrutinas y flujos (coroutines and flows en inglés), que son patrones de programación utilizados en entornos de desarrollo de aplicaciones, principalmente en Android para operaciones asincrónicas. 'Usa corrutinas y flujos para establecer la comunicación entre capas.'
- Decisión: Sí, esta recomendación sería seguida en el proyecto de la asignatura, siempre y cuando sea oportuno
- Justificación: Las corrutinas y flujos permiten realizar operaciones asincrónicas de manera eficiente y concurrente, lo que mejora la capacidad de respuesta de la aplicación y evita bloqueos innecesarios. Las corrutinas en particular son muy útiles en entornos donde se realizan operaciones de red, bases de datos u otras tareas que puedan bloquear el hilo principal de la interfaz de usuario. Los flujos son una forma reactiva y eficiente de trabajar con secuencias de datos

1.5 Usa ViewModels a nivel de la pantalla.

- Descripción: Esta recomendación se refiere a la arquitectura de ViewModels en el desarrollo de aplicaciones, donde se propone utilizar ViewModels específicos para cada pantalla o sección de la interfaz de usuario.
- Decisión: Es bastante probable que esta recomendación será seguida en el proyecto de la asignatura, si la elección lo requiere.
- Justificación: La utilización de ViewModels a nivel de la pantalla brinda ventajas significativas en la separación de la lógica de la interfaz de usuario, permitiendo una gestión más clara y organizada de los datos y acciones asociadas a cada pantalla. Los ViewModels se encargan de manejar la lógica de presentación y proporcionan datos a la interfaz de usuario, lo que resulta en una separación clara de las responsabilidades y una interfaz de usuario más robusta y fácil de mantener.