Programación de Aplicaciones Móviles Nativas

Recomendaciones de arquitectura para aplicaciones Android

Raúl Cruz Ortega

Laura González Suárez

Ismael Ramos Alonso

Introducción

La arquitectura de las aplicaciones móviles desempeña un papel crítico en la eficacia y sostenibilidad de los proyectos. En este informe, exploramos las recomendaciones esenciales proporcionadas por la guía oficial para desarrolladores de Android. Estas directrices, centradas en aspectos como la gestión de datos, modularidad y pruebas, ofrecen un marco valioso para optimizar el rendimiento y la experiencia del usuario.

Seleccionaremos cinco recomendaciones fundamentales y evaluaremos su aplicabilidad en nuestro proyecto. Al seguir estas prácticas, anticipamos no solo una mayor eficiencia en el desarrollo, sino también una arquitectura más resistente y mantenible. Este informe detallará cada recomendación elegida, nuestras decisiones y las razones fundamentales detrás de estas elecciones.

Recomendación 1: No almacenar datos en los componentes de la aplicación

- **Descripción**: Esta recomendación sugiere evitar almacenar datos directamente en los puntos de entrada de la aplicación, como receptores de emisiones, servicios y actividades. En su lugar, se propone que estos componentes coordinen con otros para recuperar solo el subconjunto de datos necesario en ese momento.

- **Justificación**: Implementaremos esta recomendación para fomentar una mayor modularidad y claridad en el diseño. Al abstenernos de almacenar datos en los componentes de la aplicación, dirigiremos el almacenamiento hacia estructuras específicas, como bases de datos locales. Esta práctica optimizará la gestión del ciclo de vida de los componentes, evitando que estos retengan información innecesaria, mejorará la eficiencia en el manejo de datos y garantizará una experiencia de usuario fluida y eficaz.

Recomendación 2: Reduce las dependencias de clases de Android

- **Descripción**: Se aconseja que los componentes de la aplicación sean las únicas clases que dependan de las APIs del SDK del framework de Android, como Context o Toast. Abstraer otras clases fuera de estos componentes ayuda en la capacidad de prueba y reduce la interconexión entre diferentes partes de la aplicación (acoplamiento).
- **Justificación**: Reducir las dependencias de clases de Android es una estrategia esencial para fortalecer la robustez y adaptabilidad de nuestro proyecto. Al limitar las dependencias a los componentes principales, mejoramos la testabilidad de la aplicación. Esto se traduce en un proceso de prueba más eficiente, permitiéndonos identificar y corregir problemas con mayor facilidad.

  Además, nuestra aplicación se vuelve más resistente a posibles cambios en el framework de Android en el futuro, adaptándonos de manera más eficaz a nuevas actualizaciones o evoluciones en la plataforma.

  Por otra parte, contribuye a una arquitectura más modular y mantenible. Al evitar interconexiones innecesarias entre diferentes partes de la aplicación, facilitamos la comprensión del código y reducimos la posibilidad de efectos secundarios no deseados al realizar cambios o mejoras.

Recomendación 3: Crea límites de responsabilidad bien definidos entre varios módulos de tu app

- **Descripción**: Esta recomendación aboga por establecer límites de responsabilidad claros entre los diversos módulos de la aplicación, evitando la extensión de código entre clases o paquetes y manteniendo responsabilidades no relacionadas en módulos separados.
- **Justificación**: Seguiremos esta recomendación en nuestra aplicación para mejorar la organización del código y simplificar su mantenimiento. Al definir roles y responsabilidades específicos para cada componente, nuestra estructura se vuelve más comprensible y fácil de gestionar. Esta práctica busca reducir los desafíos asociados con la complejidad del código, permitiendo un desarrollo más eficiente y disminuyendo la probabilidad de errores. En última instancia, esta decisión contribuye a un entorno de desarrollo más ordenado y sostenible, lo que facilita la colaboración del

equipo y asegura que la aplicación evolucione de manera coherente y sin contratiempos.

Recomendación 4: Expón lo mínimo indispensable de cada módulo

- **Descripción**: Se aconseja no exponer detalles internos de implementación de un módulo mediante accesos directos. Gracias a esto se evita la generación de problemas técnicos a medida que evoluciona el código base.

- **Justificación**: Al exponer lo mínimo indispensable de cada módulo, se evita la divulgación de detalles internos, minimizando así posibles problemas técnicos al evolucionar el código.

  Optaremos por seguir esta directriz con el objetivo de promover una mayor encapsulación y reducir la dependencia de aspectos internos en cada módulo. Esta decisión no solo mejora la mantenibilidad del código, sino que también facilita futuras actualizaciones al proporcionar una capa de abstracción que preserva la integridad del sistema.

Recomendación 5: Conserva la mayor cantidad posible de datos relevantes y actualizados

- **Descripción**: Se aconseja conservar una cantidad significativa de datos relevantes y actualizados en la aplicación, permitiendo a los usuarios aprovechar su funcionalidad incluso en modo sin conexión.

- **Justificación**: Optamos por implementar esta recomendación para garantizar una experiencia continua y útil para los usuarios, independientemente de la conectividad. Al conservar datos localmente, aseguramos que la aplicación funcione eficientemente en situaciones de conectividad limitada o ausente.

  Esta decisión mejora la accesibilidad y usabilidad, adaptándose a escenarios comunes donde la conectividad puede ser intermitente. En concreto guardaremos los artículos favoritos del usuario para que pueda consultarlos incluso sin conexión.

Conclusión

La implementación de las recomendaciones de la guía oficial para desarrolladores de Android serán la base arquitectónica de nuestro proyecto. Al evitar el almacenamiento directo de datos en los componentes de la aplicación, optimizamos la gestión de recursos y aseguramos una experiencia de usuario fluida. La reducción de dependencias de clases de Android no solo mejora la testabilidad sino que también fortalece la capacidad de adaptación a futuras actualizaciones del framework. Establecer límites de responsabilidad, exponer lo mínimo indispensable y conservar datos relevantes contribuyen a una aplicación organizada, mantenible y eficiente.

Estas decisiones se alinean con nuestro compromiso de desarrollar una aplicación Android robusta, modular y adaptable, asegurando una experiencia de usuario superior y facilitando futuras evoluciones en el ecosistema Android.

Bibliografía

Android Developer. Guía de arquitectura de apps. [online] 13/11/23.

https://developer.android.com/topic/architecture?hl=es-419