Pruebas de perfilamiento – Sprint 2

MISW 4203 – Ingeniería de software para aplicaciones móviles
Grupo 1

Metodología

Para la ejecución de la prueba de perfilamiento se preparó el siguiente ambiente de pruebas:

1. +100 álbumes cargados en la base de datos
2. +200 artistas en la base de datos, combinando bandas y músicos
3. +100 coleccionistas creados
4. Ejecutar el perfilador en el modo de información completa (complete data)
5. Las pruebas se realizaron en 3 dispositivos físicos diferentes:
    a. Xiaomi Redmi 10 – Android 12
    b. Samsung Galaxy S21 – Android 14
    c. Samsung A34 5G – Android 14
6. Los pasos por seguir durante la prueba son los siguientes:
    a. Abrir la aplicación y cargar la vista principal (lista álbumes)
    b. Hacer scroll hasta el fin de lista álbumes
    c. Hacer scroll hasta el inicio, abriendo diferentes álbumes de por medio. Se deben abrir todos los álbumes diferentes por lo menos 3 veces
    d. Cambiar a la vista de artistas
    e. Hacer scroll hasta el fin de lista artista
    f. Hacer scroll hasta el inicio, abriendo diferentes artistas de por medio. Se deben abrir todos los artistas diferentes por lo menos 3 veces
    g. Cambiar a la vista de coleccionistas
    h. Hacer scroll hasta el fin de la lista de coleccionistas, revelando los detalles de algunas de las cartas de coleccionista
    i. Cambiar repetidamente entre la vista de opciones y la vista de álbumes
    j. Cambiar repetidamente entre la vista de opciones y la vista de artista

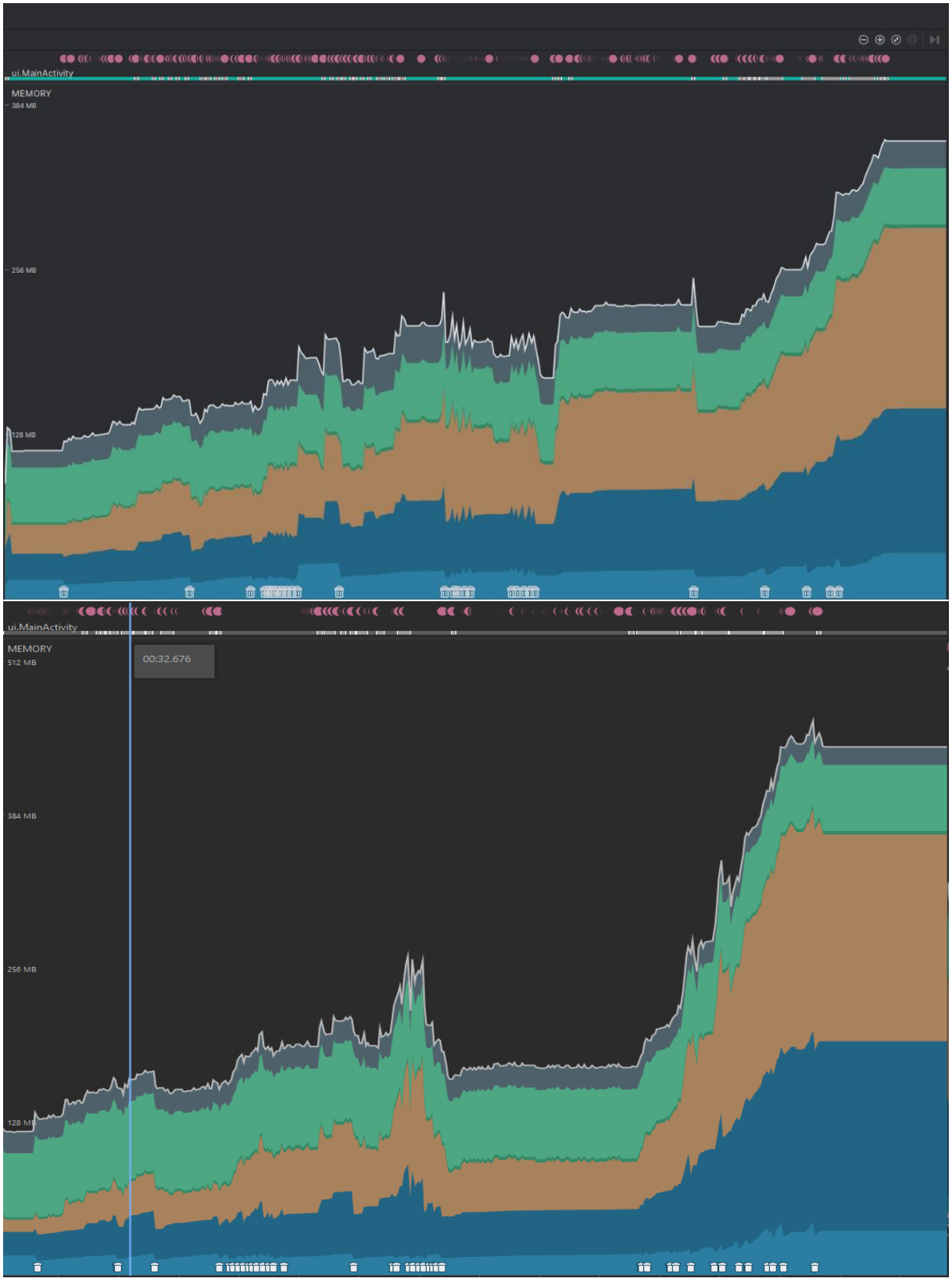
Resultados

El análisis de los resultados se enfoca en las 2 variables principales relacionadas con el rendimiento de la aplicación, consumo de CPU y consumo de memoria RAM. En particular, desde la perspectiva del usuario, no se detectaron interrupciones en el uso de la aplicación, la cual respondió de forma fluida a los inputs generados durante la prueba. Tampoco se encontraron errores de tipo OOM ni ANR

En cuanto al consumo de CPU, no se observó en ninguna de las pruebas un consumo superior al 30% de uso de CPU en los tres dispositivos. Esto es explicado porque la aplicación no requiere de la ejecución de procesos complejos y por la existencia de una API REST desplegada en una nube remota, la cual se encarga de la lógica de negocio. Este comportamiento se puede observar en la figura 1

Figura 1 - Vista general de las pruebas realizadas en los 3 dispositivos. Arriba Samsung Galaxy S21, en el centro Samsung A34 5G y abajo Xiaomi Redmi 10

Sin embargo, en la misma figura es posible identificar un elevado consumo de memoria RAM, el cuál incrementa significativamente durante la ejecución del paso J dentro del escenario de pruebas. Este comportamiento se puede identificar en la figura 2, donde se presentan para los 3 dispositivos el detalle del consumo de memoria RAM y el crecimiento descrito anteriormente hacia el final del perfilamiento



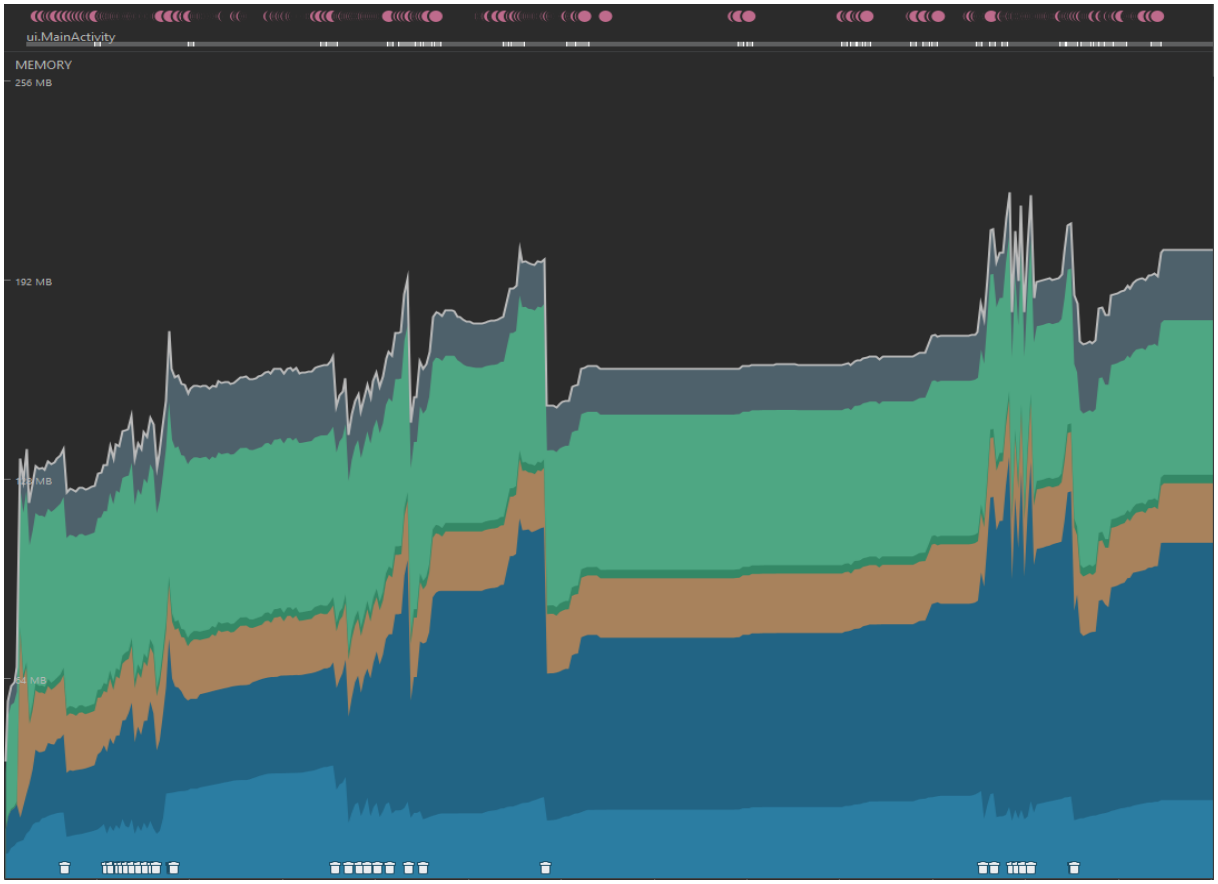


Figura 2 - Vista de detalle del consumo de memoria RAM de las pruebas realizadas en los 3 dispositivos. Arriba Samsung Galaxy S21, en el centro Samsung A34 5G y abajo Xiaomi Redmi 10

Otro comportamiento interesante se puede observar en los eventos de garbage collection (GC) ejecutados por el sistema operativo. Estos se observan en la parte inferior con el ícono de bote de basura. Es posible observar una ejecución numerosa de estos eventos en los dispositivos Samsung durante la ejecución de los pasos d – e – f, que corresponden a la carga de la lista de artistas. En cuanto al celular Xiaomi, el cual tiene menor capacidad, se observa el mismo comportamiento, sin embargo, también se presenta un elevado número de procesos de GC durante la visualización de la lista de álbumes. A continuación, se expondrán los hallazgos y suposiciones para cada uno de los escenarios

Listado de álbumes

Respecto al comportamiento presentado en el celular Xiaomi, los eventos de GC generados corresponden a una menor disponibilidad de memoria RAM en este dispositivo que obligan a la ejecución más agresiva de estos tipos de eventos para garantizar la operación de la app. Esta hipótesis podría explicar el por qué de que este comportamiento no se presente en los celulares Samsung.

Listado de artistas

Una particularidad del conjunto de datos escogido es que la URL de las imágenes de algunas bandas no contenían una imagen que pudiera ser cargada, tal como se muestra en la figura 3

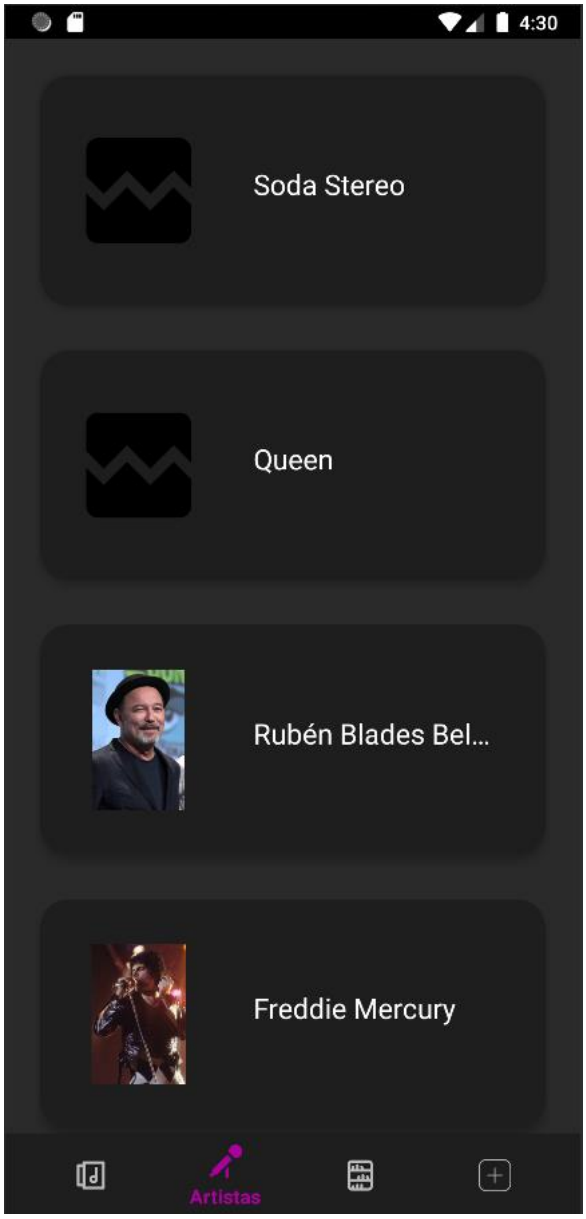


Figura 3 - Cartas de artista con imagen no funcional

Dado que la implementación se realizó con Glide, solo las imágenes cargadas exitosamente son almacenadas en caché. Por tal razón, cualquier imagen dañada generaba una petición adicional para poder cargar la fuente, lo que se traduce en un incremento significativo del consumo de memoria RAM. Entendiendo que alrededor de la mitad de los artistas en el ambiente de pruebas no tenía una imagen apta, se puede dar a entender que la prevalencia de enlaces dañados para las imágenes puede tener un impacto significativo en el rendimiento de una aplicación, si el manejo no es el adecuado.

Por último, el incremento elevado en el consumo de memoria RAM para el paso J está explicado por la misma razón, sin embargo, entendiendo que parte del fragmento en el cual se realiza el cambio se destruye al realizar la navegación a otro fragmento, es posible incrementar el gasto de memoria RAM hasta un punto en el que se pueda generar un error de tipo OOM (Out of Memory)

Conclusiones y siguientes pasos

Por una parte, las pruebas permitieron exponer que tener un gran número de información cargada en una sola vista impacta significativamente el rendimiento de una aplicación. Si esta, además, presenta

información incompleta u orígenes de imagen dañados, la aplicación intentará continuamente realizar solicitudes hasta poder almacenar una imagen valida en el caché implementado.

Sin embargo, también es importante destacar la estabilidad del proceso en escenarios donde la información es estable e íntegra. A pesar del consumo excesivo de RAM durante la ejecución de la vista de artistas, en el resto de las vistas no se observa el mismo comportamiento, y se refleja en un número reducido de operaciones de GC y en valles de estabilidad en el consumo de memoria.

Para el siguiente sprint se debe buscar una manera de reducir el impacto de objetos en la base de datos cuya información no es completa o tiene un formato que puede generar errores en la aplicación.