Medición de los problemas comunes de rendimiento en aplicaciones hechas con React Native

Armando Narcizo Rueda Pérez

Propuesta de investigación

Sin asesores aún

Universidad Da Vinci

Tabla de contenido

Medición de los problemas comunes de rendimiento en aplicaciones hechas con React Native

Introducción

Las aplicaciones móviles forman parte importante de muchos servicios ya sea en forma complementaria o totalitaria, y actualmente hay mas de 3 millones de apps en la tienda de aplicaciones de Google (AppBrain, s.f). El sistema operativo de Google, Android, a pesar de tener un gran peso en el mercado de los smartphones, no es el único, y compite con iOS, otro sistema operativo con su propia tienda de aplicaciones.

Las empresas tienen que tomar esto en cuenta y muchas veces eso implica tener dos equipos de desarrollo de software que puedan publicar dos apps idénticas bajo equipos de trabajo diferentes y lenguajes de programación diferentes.

Este escenario ha impulsado la llegada y el mantenimiento de diferentes tecnologías que permiten desarrollar aplicaciones móviles para ambos sistemas operativos con la promesa de utilizar un solo lenguaje de programación para la mayoría de las necesidades. React Native es una de ellas, y actualmente tiene 5 años desde su lanzamiento como framework de código abierto.

Y aunque haya investigaciones que comparen diferentes tecnologías lo cierto es qué React Native ya tiene un espacio en la comunidad de desarrolladores y ha sido escogida por empresas importantes para satisfacer sus necesidades.

Estas necesidades involucran sobre todo dar una experiencia homogénea a sus usuarios. Uno es de los aspectos de esta experiencia tienen que ver cómo el manejo de recursos del dispositivo los cuales bien administrados le dan al usuario una experiencia fluida sobre la interfaz en la que interactúa.

Planteamiento del problema

React Native promete un rendimiento parecido a las aplicaciones nativas debido a que tiene la capacidad de llamar componentes nativos en tiempo de ejecución. A pesar de que las aplicaciones nativas tienen un mejor rendimiento, el propósito de React Native es lograr interfaces de 60 cuadros por segundo para entregar una experiencia nativa.

El problema que se plantea es cómo estos problemas de rendimiento pueden llegar a afectar la experiencia del usuario en cuadros por segundo y cómo estos se acentúan dependiendo de las capacidades del teléfono.

No hay muchas investigaciones al respecto de cómo se cuantifican estos problemas de rendimiento y como son afectadas por las características de los dispositivos por lo que esta investigación pretende aportar mas información al respecto.

Justificación

El uso de tecnologías para desarrollar aplicaciones multiplataforma pueden ser un alivio para las empresas y equipos de desarrollo que pretendan darle soporte a sus aplicaciones en los sistemas operativos Android y iOS pues aseguran una experiencia y diseño homogéneo para usuarios en diferentes sistemas operativos. Pero escoger esta tecnología no significa terminar con los problemas de una experiencia única para todos pues otros factores pueden afectar como los recursos propios de cada dispositivo.

Es incluso posible que los equipos de desarrollo no prueben sus aplicaciones en dispositivos de baja gama al no comprender cuales pueden ser las consecuencias para los usuarios que tengan estos dispositivos.

Esta investigación trata de cuantificar estas consecuencias de tal forma que cualquier interesado pueda sacar a la luz estos problemas y proponer soluciones en sus propios proyectos y organizaciones.

Objetivos generales:

- Medir los efectos de los problemas de rendimiento comunes en aplicaciones de React Native en dispositivos con diferentes capacidades

Objetivos específicos:

- Investigar los escenarios comunes que generan los problemas de rendimiento en las aplicaciones en React Native
- Diseñar y crear una aplicación en React Native
- Investigar literatura relacionada con el tema
- Instalar la aplicación en 4 dispositivos con diferentes capacidades
- Utilizar diferentes herramientas para medir los efectos de los problemas de rendimiento
- Comparar los resultados registrados por cada dispositivo

Antecedentes

Las investigaciones sobre React Native y su rendimiento se han centrado en comparaciones con otras tecnologías para el mismo fin.

Una parte de las investigaciones se centran en comparar React Native con aplicaciones nativas, es decir, aplicaciones hechas con los medios oficiales de los sistemas operativos iOS y Android. Según Eskola (2018): “Los hallazgos indican que React Native incurre en penalizaciones de rendimiento significativas en comparación con el código nativo”. Por otra parte Gill (2018) indica que “React-Native ha demostrado ser capaz de superar a Swift en lo que respecta al uso de memoria con procesos exigentes (…)”. Así mismo Kuitunen (2019) menciona que “No se observaron caídas notables en el rendimiento durante la prueba, pero la aplicación de prueba no es compleja y una aplicación más grande podría revelar más problemas con el rendimiento”. Por otra parte Danielsson (2016) dice: “La aplicación React Native no tiene el mismo rendimiento que la aplicación de Android. Sin embargo, la diferencia es menor de lo que cabría esperar”.

Otra parte de las investigaciones se enfocan en comparar diferentes tecnologías multiplatforma entre sí. Drgoň (2017) considera que React Native es una tecnología con alto rendimiento en comparación con Xamarin, Ionic 2 y NativeScript. El autor Ruano Valenzuela (2018) considera que “en este aspecto Ionic reacciona mucho más rápido”. Por otra parte Eshetu Fentaw (2020) considera que “se puede concluir que tanto Flutter como React Native logran usar la CPU con la misma eficiencia dependiendo de las pruebas”.

Poco se menciona sobre React Native y su rendimiento en dispositivos con recursos de procesamiento diferentes. Eskola (2018) menciona “En muchos casos, el impacto en el rendimiento no es lo suficientemente significativo como para causar frustración al usuario, pero especialmente en dispositivos más antiguos, las operaciones comunes, como el inicio de aplicaciones y la representación de componentes, son notablemente más lentas y pueden tener una latencia hasta 10 veces mayor que el equivalente nativo”, esto podría indicar que los dispositivos mas antiguos, que pueden tener recursos mas limitados para procesar, pueden amplificar los problemas de rendimiento en React Native.

Marco contextual y teórico

Latencia en interfaces de usuario

Según Eskola (2018): La latencia se puede medir como el tiempo que tarda un sistema en responder a una entrada determinada. Hay muchos componentes que contribuyen a la latencia general de un dispositivo. Considere un ejemplo simplificado de los muchos pasos involucrados cuando un usuario navega a otra vista en una aplicación de teléfono inteligente

1. El usuario toca un botón de navegación

2. El toque está registrado por la pantalla táctil

3. El evento atraviesa los controladores del sistema operativo

4. El evento atraviesa conjuntos de herramientas de interfaz de usuario y ventanas

5. El evento llega a la aplicación que actúa sobre el evento, carga otra vista 6. La vista finalmente se carga, el subsistema de gráficos dibuja un nuevo marco

7. El marco terminado se voltea a la pantalla.

8. El controlador de pantalla envía una nueva señal a los píxeles, los píxeles comienzan a actualizarse

9. El usuario ve una imagen actualizada

Cada paso de este ejemplo presenta latencia. Por ejemplo, el sistema operativo puede tener otras tareas que realizar y, por lo tanto, es posible que no procese los eventos táctiles al instante.

Aplicaciones Nativas

Según Eskola (2018): “Se espera que la escritura de aplicaciones utilizando el entorno nativo de cada plataforma tenga el mejor rendimiento y muestre los tiempos de respuesta más bajos. Apple y Google se han concentrado en optimizar sus herramientas y entornos para desarrolladores para que las aplicaciones funcionen lo mejor posible en su plataforma”.

Popularidad de tecnologias multiplataforma

En el sitio web de la comunidad de desarrolladores de Stack Overflow realizó una encuesta en 2020 a aproximadamente 65,000 de sus desarrolladores. La encuesta preguntó qué tecnologías están usando los participantes, así como qué tecnologías les gustan más o les gustaría usar (2020 Developer Survey, s.f.).

Bajo la categoría de "Otros frameworks, librerías, y herramientas" se encuentran las tecnologías móviles multiplataforma y se muestra en los resúmenes de los datos que React Native es usado por 11.5% de los participantes seguido de Flutter con 7.2%, Cordova 6.0% y Xamarin 5.8% (2020 Developer Survey, s.f.). En conclusión React Native es una tecnología relevante.

React

React, según la pagina oficial de React, es una biblioteca de JavaScript para construir interfaces de usuario. Esta es una tecnología bastante popular para construir aplicaciones de una sola pagina (React. Una biblioteca de JavaScript para construir interfaces de usuario, s.f.).

React utiliza conceptos como Virtual DOM, componentes y flujo de datos unidireccional para crear aplicaciones web. El objetivo es facilitar la creación de interfaces con un buen rendimiento.

React Native

React Native es, según la página oficial de React Native, un marco de código abierto para crear aplicaciones de Android e iOS utilizando React y las capacidades nativas de la plataforma de aplicaciones (React Native. Learn once, write anywhere, s.f.).

Según Eskola (2018): “El resultado final es que los desarrolladores pueden crear aplicaciones con los componentes nativos de la plataforma mientras escriben la aplicación en un lenguaje de nivel superior, utilizando los conceptos e ideas de React".

Este entorno se logra con la integración de múltiples hilos que logran procesar los eventos dentro de la aplicación para realizar efectos en la interfaz, utilizar recursos del teléfono y comunicarse con otros sistemas.

Hipótesis

La hipótesis con la que esta investigación empieza es que las aplicaciones en React Native tienden a presentar problemas de rendimiento en escenarios no optimizados comunes de una manera lineal en la medida de los recursos de procesamiento de los dispositivos.

Marco Metodológico

La investigación se centrara en medir el impacto de los problemas comunes de rendimiento en diferentes dispositivos y se plantea comparar ese impacto en diferentes modelos de smartphones con el sistema operativo Android y que cuentan con recursos de procesamiento diferentes.

Se escogerán 4 modelos de smartphone Android con capacidades de 512 MB de RAM, 1 GB de RAM, 2 GB de RAM y 4 GB de RAM.

Se encontraran las situaciones comunes en las que los problemas de rendimiento aparecen en las aplicaciones móviles desarrolladas con React Native.

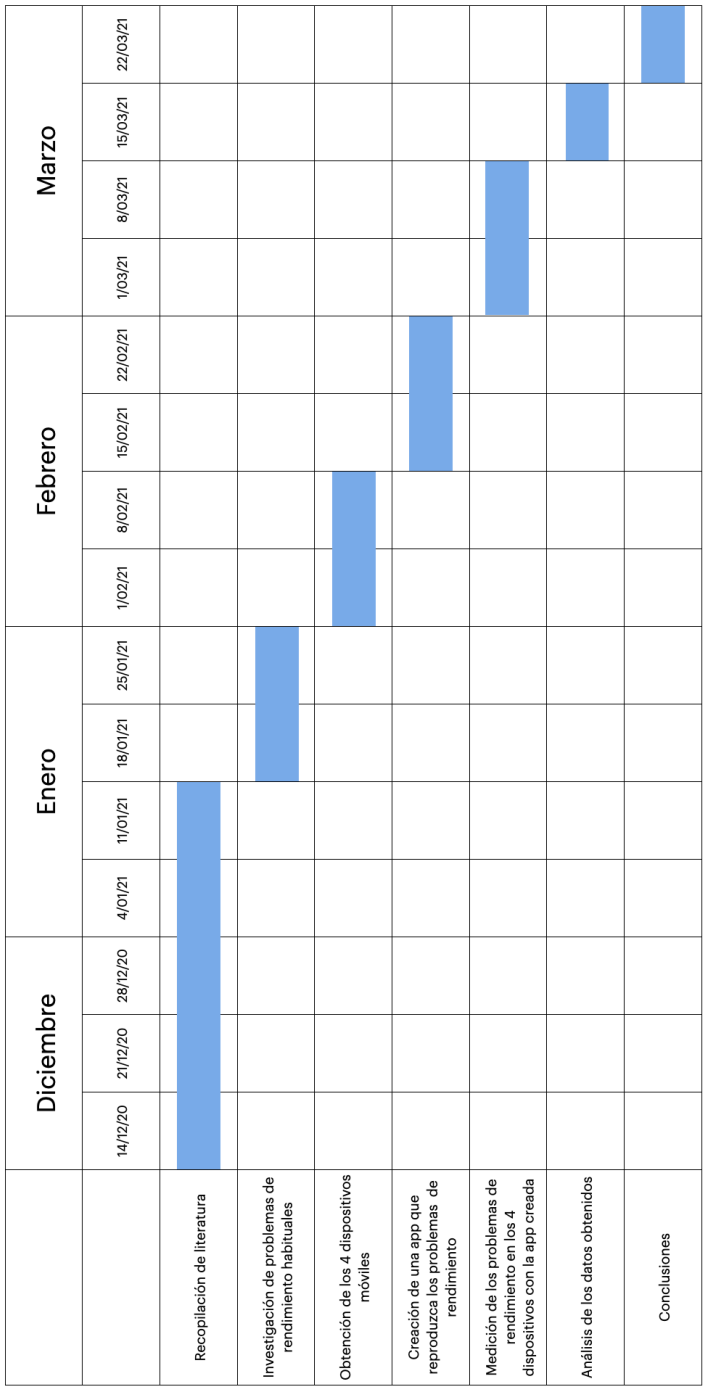
Una vez creados los escenarios que presentan los problemas de rendimiento, con la ayuda de una aplicación hecha específicamente para ello, se cuantificara el impacto de dichos problemas en FPS (frames per second) en los 4 diferentes dispositivos conseguidos. Estos dispositivos estarán conectados directamente a una computadora portátil durante las pruebas.

Se decidió por las especificaciones 512 MB, 1 GB, 2 GB y 4 GB de RAM debido a que son las especificaciones de los teléfonos mas comunes en países latinoamericanos como Colombia (Piejko, 2017), y por lo tanto ignorar estas especificaciones puede resultar poco útil para los interesados en esta investigación que quieran darle soporte a teléfonos de baja gama.

Las pruebas se harán en versiones de lanzamiento pues este modo es el recomendado para este tipo de benchmark pues elimina tareas que se ejecutan en modo de desarrollo para ayudar a los programadores y ocupan recursos del dispositivo.

Los datos serán recolectados con el Monitor de Rendimiento de React Native y la herramienta *systrace* de Android. Estos datos serán usados para un análisis posterior.

Cronograma


Diciembre
Enero
Febrero
Marzo
14/12/20
21/12/20
28/12/20
4/01/21
11/01/21
18/01/21
25/01/21
1/02/21
8/02/21
15/02/21
22/02/21
1/03/21
8/03/21
15/03/21
22/03/21
Recopilación de literatura
Investigación de problemas de rendimiento habituales
Obtención de los 4 dispositivos móviles
Creación de una app que reproduzca los problemas de rendimiento
Medición de los problemas de rendimiento en los 4 dispositivos con la app creada
Análisis de los datos obtenidos
Conclusiones

Resultados

Tiempos de carga

	ZTE Blade L8/1 GB RAM	AllCall S1/ 2 GB RAM	Redmi Note 8/4 GB RAM
Carga de 100 elementos con ScrollView	2s 56ms	2s 468ms	1s 280ms
Carga de 100 elementos con FlatList	679ms	370ms	439ms
Carga al modificar un elemento con Scrollview	1s 545ms	1s 510ms	651ms
Carga al modificar un elemento con FlatList	676ms	927ms	498ms
Carga al modificar un elemento memorizado con Scrollview	482ms	988ms	467ms
Carga al modificar un elemento memorizado con FlatList	424ms	927ms	469ms

Al final se opto por evaluar el desempeño de 6 escenarios comunes.

El primer escenario es el tiempo que le toma al dispositivo para mostrar una lista de 100 elementos con el componente Scrollview.

El segundo escenario es medir el tiempo de carga para mostrar esos 100 mismos elementos pero con FlatList.

El tercer escenario es cuando se oprime un botón que cambia la información del elemento en la lista usando Scrollview.

El cuarto escenario es cuando se oprime un botón que cambia la información del elemento en la lista usando FlatList.

El quinto escenario es cuando se oprime un botón que cambiar la información de un elemento usando ScrollView pero memorizando sus props.

El sexto escenario es cuando se oprime un botón que cambiar la información de un elemento usando FlatList pero memorizando sus props.

Conclusiones y sugerencias

Analizando los datos obtenidos se observa que en definitiva los tiempos de carga son mejores en el dispositivo con mayor capacidad de RAM. Sin embargo no parece que haya una relación lineal al disminuir la capacidad de RAM del dispositivo. Aunque si se observa un deterioro en los tiempos de carga también se observa que en algunos casos el dispositivo con 1 GB de RAM es superior al que tiene una capacidad de 2 GB.

Quizá la capacidad de RAM sea un elemento dentro de muchos para determinar prever sus tiempos de carga pero no debería ser el único. Faltaría también investigar si la version de android y él procesador también pueden ser tomados en cuenta para predecir los tiempos de carga del dispositivo.

La investigación se encontró con el problema de que no era buena idea medir el rendimiento de la app mediante FPS pues en muchos casos los eventos tomaban mas de 1 segundo para completarse y se determino que era mejor medir el rendimiento por tiempos de carga.

Aun así la investigación considera que no es posible probar o no la hipótesis planteada debido a que hay elementos que pueden sugerir qué es verdad y otros que no. Por lo que se esperaría mejorar la metodología en otra entrega.

Para mejorar la investigación en el futuro se deberían incluir mas dispositivos con la misma version de android y también debido a que quizá los datos del systrace son difíciles de interpretar y cambian dependiendo del dispositivo a lo mejor sea necesario usar una herramienta que involucre personas y preguntas acerca del rendimiento de la app.

Bibliografía

Eshetu Fentaw, A. (2020). Cross platform mobile application development: a comparison study of React Native Vs Flutter (Tesis de pregrado). UNIVERSITY OF JYVÄSKYLÄ, Finlandia.

Drgoň, F. (2017). Developing a mobile application using multiplatform frameworks with a native output (Tesis de pregrado). Masaryk University, Republica Checa.

Gill, O. (2018). Using React Native for mobile software development (Tesis de pregrado). Metropolia University of Applied Sciences, Finlandia.

Kuitunen, M. (2019). CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT WITH REACT NATIVE (Tesis de pregrado). Tampere University of Technology, Finlandia.

Danielsson, W. (2016). React Native application development – A comparison between native Android and React Native (Tesis de pregrado). Linköpings universitet, Suecia.

Eskola, R. (2018). React Native Performance Evaluation (Tesis de pregrado). Aalto University, Finlandia.

Eshetu Fentaw, A. (2020). Cross platform mobile application development: a comparison study of React Native Vs Flutter (Tesis de pregrado). UNIVERSITY OF JYVÄSKYLÄ, Finlandia.

Ruano Valenzuela, R. V. (2018). “ESTUDIO COMPARATIVO DE LOS FRAMEWORKS IONIC Y REACT NATIVE” APLICACIÓN MÓVIL DE PEDIDOS A DOMICILIO BASADA EN LA NORMA ISO 9126 (Tesis de pregrado). Universidad Técnica del Norte, Ecuador.

Piejko, P. (2017). What are the most common RAM specs for smartphones? Recuperado de https://mobiforge.com/news-comment/what-are-the-most-common-ram-specs-for-smartphones

2020 Developer Survey. (s.f.). Recuperado de https://insights.stackoverflow.com/survey/2020#most-popular-technologies

React Native. Learn once, write anywhere. (s.f.). Recuperado de https://reactnative.dev/

React. Una biblioteca de JavaScript para construir interfaces de usuario. (s.f.). Recuperado de https://es.reactjs.org/