Tendencias en desarrollo móvil bajo las tecnologías Android e IOS

Trends in mobile development under Android and IOS technologies

Ms. Johnny Alexander Salazar Cardona[1]
Ing. David Alberto Angarita[2]
Ing. Juan David Montoya[3]

Recibido: 02/18/2016 - Aceptado: 04/30/2016

Cómo citar este artículo: J. Salazar, D. Angarita y J. Montoya, "Tendencias en desarrollo móvil bajo las tecnologías Android e IOS", *IngEam*, vol. 3, n.° 3, pp. 28 - 35, 2016

Resumen

La computación móvil se ha convertido en un elemento indispensable para los seres humanos, siendo esta una herramienta de trabajo y entretenimiento predominante como acceso a correos electrónicos, control de eventos, geolocalización, redes sociales, chats, videojuegos, etc. Actualmente se pueden encontrar diferentes soluciones móviles para los usuarios finales, las cuales son desarrolladas según un conjunto de parámetros limitados a la tecnología sobre la cual se encuentre controlado el dispositivo en cuestión. Es por esto que el presente artículo busca contextualizar las tendencias de desarrollo móvil para las plataformas más representativas del mercado como Android con su tendencia al lenguaje DART o GO e IOS con el refinamiento de su lenguaje SWIFT, con el fin de entender las posibles mejoras de estos cambios y de diferentes soluciones para el desarrollo nativo e hibrido.

Palabras clave: DART, GO, SWIFT, Android, IOS, computación móvil,

Abstract

Mobile computing has become an indispensable element for human beings, being this a predominant work and entertainment tool like access to emails, event control, geolocation, social networks, chats, video games, etc. Different mobile solutions can now be found for end users, which are developed according to a set of parameters limited to the technology on which the device in question is controlled. This is why this article seeks to contextualize mobile development trends for the most representative platforms in the market such as Android with its tendency to DART language or GO and IOS with the refinement of its SWIFT language, in order to understand the possible improvements of These changes and different solutions for native and hybrid development.

Keywords: DART, GO, SWIFT, Android, IOS, mobile computing

[1] Correo electrónico: alexander9052@gmail.com
[2] Correo electrónico: davidangaritag@gmail.com
[3] Correo electrónico: jdm@eam.edu.co

I. INTRODUCCIÓN

Desde hace algunos años el mercado en dispositivos móviles viene creciendo de una manera estable, debido a las nuevas integraciones desarrolladas con el día a día de una persona promedio como los smartphones, smartwatchs, Smart tv, etc. Estas apoyan procesos cotidianos como mensajería instantánea, correos electrónicos, lectura de signos vitales y fotografía [1]. Facilita procesos laborales como control de actividades, envío de grandes datos, software ofimático y masifica el entrenamiento con las redes sociales, streaming de video y los videojuegos [2, 3]. Con la masificación de este tipo de soluciones se han implementado diferentes tecnologías para administrar y controlar todas estas características, como Android de Google, IOS de Apple, Windows phone de Microsoft, Firefox OS, Ubuntu phone, BlackBerry entre muchos otros, que permiten cada uno a su manera la gestión de todas estas aplicaciones que soportan la cotidianidad una gran cantidad de personas.

La cuota de mercado de estas tecnologías está muy marcada, siendo Android la dominante del mercado debido a su distribución abierta y a que se ha masificado mucho en dispositivos de bajo costo. Seguidamente se encuentra IOS que se encuentra enfocado a equipos de altos costos pero que se encuentra muy bien posicionado respecto a cuotas de mercado. Finalmente las otras tecnologías se disputan el segmento restante, cada uno con una pequeña participación [4, 5].

Con el fin de mejorar la calidad y facilitar el desarrollo de nuevas aplicaciones, han surgido nuevas tecnologías de desarrollo de aplicaciones hibridas, como nuevos lenguajes estándar para las plataformas dominantes del mercado. Debe entenderse como aplicaciones hibridas, desarrollos que son desarrollados en lenguajes no estándar de la tecnología que se quiera abordar, y que puede ser compilada para que se ejecute en múltiples sistemas operativos móviles, con el fin de no desarrollar toda una aplicaciones que ya se encuentra lista en otra plataforma. En el caso de la generación de nuevos lenguajes estándar los casos más representativos que se pueden encontrar DART de Android y SWIFT en IOS, los cuales buscar optimizar los procesos de desarrollo, como también la calidad del producto final.

II. MARCO TEÓRICO

Actualmente para el desarrollo de una aplicación móvil se cuenta con diferentes enfoque: Nativa, Hibrida y Web [6]. Casa uno de estos enfoques posee unas características específicas, que según las necesidades puntuales y el personal disponible para el desarrollo del software, se puede optar por alguna de estas [7, 8].

A. Desarrollo nativo

Un desarrollo nativo es aquel que se implementa de forma específica para un sistema operativo, bajo el lenguaje estándar disponible para lograrlo. Para lograrlo Android, IOS o Windows phone ponen a disponibilidad un Software Development Kit o SDK, por lo que se debe desarrollar en cada una según la tecnología y restricciones si se desea una

aplicación nativa para cada una de las plataformas [9]. Como explica IBM [10] Cada sistema operativo posee su propio lenguaje: En Android las aplicaciones nativas de desarrollan sobre Java, en IOS se desarrollan sobre Objective-C o como es tendencia últimamente sobre el lenguaje Swift. Windows phone es desarrollado bajo el framework .NET.

El desarrollo nativo tiene una serie de ventajas como el acceso completo al dispositivo, mejor experiencia de usuario, visibilidad en la tienda de aplicaciones, acceso a notificaciones. Entre las desventajas se encuentra la codificación independiente para cada plataforma destino y que por lo general tiende a ser más costoso su desarrollo.

B. Desarrollo hibrido

Un desarrollo hibrido es la implementación de una aplicación que combine las características de una aplicación nativa y una web es decir, que sea desarrollada bajo lenguajes y tecnologías web, con acceso a ciertas características de hardware pero siendo el producto final una aplicación instalable en el dispositivo [10].

Entre las ventajas se encuentran el soporte multiplataforma, publicación directa en tiendas de aplicaciones, con conocimientos de desarrollo web puede ser construida una aplicación con este enfoque y se tiene acceso a algunas características del hardware del dispositivo. Entre las desventajas se encuentra el rendimiento y que la experiencia de usuario es muy web para ser una aplicación instalada.

C. Desarrollo web

Un desarrollo web móvil implica tecnologías como HTML, JavaScript y CSS. Es básicamente una página web clásica, pero versátil con la capacidad de adaptarse como una aplicación móvil cuando sea accedido por este. Para lograrlo existen diferentes frameworks, que aplican un CSS pre establecido para adaptar la página utilizando Tags específicos sobre el HTML [10, 11].

Entre las ventajas de este enfoque de desarrollo se encuentra la reutilización de código y acceso multiplataforma, desarrollo sencillo y económico, no necesita instalación, publicación a través de cualquier navegador web, actualizaciones inmediatas. Entre sus desventajas se encuentra que solo funcionan con conexión a internet, acceso limitado a hardware de dispositivo, tiempos de respuesta lentos a comparación de una aplicación nativa y para su difusión requiere un mayor esfuerzo al no estar publicada en una tienda de aplicaciones.

III. TENDENCIAS

Sin importar el número de enfoques disponibles para el desarrollo de aplicaciones móviles, lo ideal es implementar de manera nativa con el fin de obtener el mejor rendimiento, acceso a recursos, fluidez y versatilidad requerida. Es por esto que empresas como Google e Apple han buscado mejorar sus respectivos lenguajes nativos para así agilizar los procesos de desarrollo, atraer a un mayor grupo de desarrolladores y optimizar más las aplicaciones que se

ejecuten sobre sus respectivos sistemas operativos.

A. Desarrollo nativo DART

Es un lenguaje diseñado por google inicialmente con el propósito de para reemplazar a JavaScript en la web, pero debido a sus beneficios empezó a ser aplicado para desarrollo en Android con el fin de no depender de tecnologías de terceros, como es el lenguaje Java de Oracle utilizado para desarrollos nativos sobre la tecnología móvil de Google [12].

Aunque DART para Android esta se encuentra en una fase prototipo bajo el nombre de "Sky", ya ofrece resultados prometedores. Como resalta Palazuelos [13], el principal objetivo del proyecto es ofrecer mayor fluidez y rapidez de las aplicaciones que corran sobre el sistema operativo. Actualmente se llegan a tasas de 60 FPS, pero con la implementación de este nuevo lenguaje se espera llega a los 120 FPS. Para lograrlo, el hilo principal encargado de la interfaz gráfica nunca se bloquea al llamar a las APIs del sistema operativo, por lo que así hayan retrasos el usuario final nunca se percatara de ello.

Según Zamora [12], DART como lenguaje orientado a objetos, tiene características muy interesantes respecto a Java como actual solución, entre las que se pueden destacar : No hay datos primitivos (todo es un objeto), sintácticamente simple, ninguna limitación en constructores (como restricciones de nombre), paso de parámetros por nombre – posición y/o con valores por defecto, etc. Aunque ofrece hasta el momento elementos muy prometedores, no todo es bueno para esta tecnología. Al estar enfocado a la web, no depende de ninguna plataforma en concreto y solo y solo requiere máquina virtual que la soporte, pero debido a su naturaleza todas las llamadas son ejecutadas a través de HTTP, por lo que no podrá ejecutarse cuando se encuentre offline [13].

B. Desarrollo nativo GO

En un leguaje compilado, concurrente, imperativo, estructurado, no orientado a objetos y con auto recolector de basura lanzado por google en 2009 aunque llevaba 2 años de desarrollo. Este lenguaje es muy parecido a C tanto en sintaxis como en rendimiento, y aunque no es orientado a objetos al no poseer una jerarquía, permite implementar interfaces [14]. Nació a partir de la limitación actual de encontrar un lenguaje que permita compilación ágil, ejecución eficiente, facilidad de programación o moderno (soporte a redes y procesamiento en múltiples núcleos), debido a que no se encuentra nada en el mercado que brinde estas tres características como explica Michelone [15].

El punto fuerte del lenguaje GO en Android son los juegos, ya que como explica Rodrígez [16], se pueden dibujar objetos en pantalla, reproducir sonidos y controlar eventos táctiles, permitiendo construir binarios para procesadores ARM y luego son cargados como librerías .so en las aplicaciones.

Aunque GO en sus inicios se perfilaba para ser el nuevo lenguaje C y posible lenguaje insignia de google para Android, este último ha perdido mucha fuerza sin importar su rendimiento y sencillez de implementación, debido a que DART

hasta el día de hoy está dando muy buenos resultados para desarrollos multiplataforma, elementos indispensable que no brinda GO.

C. Desarrollo nativo SWIFT

Tim Cook presento el nuevo lenguaje llamado SWIFT 2.0 en el World Wide Developers Conference en 2015 en el Moscone Center, San Francisco. Con el nuevo lenguaje de programación creado por la empresa Apple se buscó una nueva manera para el desarrollo de aplicaciones en sistemas operativos MAC OSX (orientado a estaciones de trabajo) e IOS (orientado para dispositivos móviles como IPad, IPhone y los nuevos relojes inteligentes). Este lenguaje presenta una serie de características interesantes como la generación de código rápido, acceso fácil a documentación y sintaxis dinámica que se puede combinar con Object-C para mayor velocidad y fácil integración [17].

Con Swift 2.0 se pretende incrementar el número de desarrolladores que puedan crear aplicaciones para las plataformas de Apple, sin embargo se percibe una división en la comunidad un primer grupo denominado los programadores nativos con Object-C sienten que pudieron haber perdido el tiempo, debido a que la línea de aprendizaje de este tipo de lenguaje es alta y el costo de esfuerzo es largo. El otro grupo manifiesta muy acertada la posición en la creación de un lenguaje más limpio.

Apple publica en el sitio oficial de Swift la información para desarrolladores y en la tienda IBook libros de descarga gratuita y toda la documentación necesaria para una implementación en XCode 7 que va permitir ampliar el número adeptos para este tipo plataformas. Adicionalmente Swift cuenta con la licencia apache 2.0, abriendo el lenguaje a la comunidad libre en Swift y pronto los usuarios Linux será parte del ecosistema.

D. Desarrollo hibrido con JQUERY: JQUERY Mobile

Este framework nacido en 2010 aunque este lejos de ser el mejor, es el que tiene una conexión directa con JQUERY, debido a que fue creado por la misma fundación. Este permite realizar implementaciones basadas en HTML5 con soporte multiplataforma como Android, IOS, Windows phone, BlackBerry, o cualquier sistema operativo con un navegador que soporte dicha tecnología. Este no ofrece una arquitectura bajo MVC o MVVM, pero puede ser implementado por cuenta del desarrollador, o puede ser integrado con AngularJS o Backbone para brindar un soporte estándar. Este lenguaje al ser multiplataforma (Desarrollos con Phonegap) y multinavegador hace que su rendimiento no sea el más óptimo, aunque se han enfocado esfuerzos en aumentar dicho aspecto [18, 19].

Actualmente es el framework más utilizado para desarrollos híbridos y debido a la facilidad con la que se pueden generar desarrollos multiplataforma es una excelente opción para prototipado rápido. Adicionalmente cuenta con una muy buena documentación y desarrollos de terceros. Otras opciones disponibles con JQUERY son Kendo UI (Adicionalmente integra AngularJS) y DEVXTREME Mobile (Antiguamente conocido como EX PHONEJS).

E. Desarrollo hibrido con ANGULAR JS y HTML: Ionic Framework

Como explica Es un framework que nació en 2013, es open Source y ofrece una biblioteca HTML, CSS y JS optimizado para dispositivos móviles. Toda su arquitectura está basado sobre SASS, permite integrarlo con Cordova JS (Phonegap) y finalmente integrado con Angular JS. Este aunque es hibrido no cuenta con soporte total a todas las plataformas. Oficialmente cuenta con soporte sobre Android e IOS y algunos pinceles sobre Windows Mobile pero no oficialmente. Además de no soportar todas las plataformas móviles, aun la optimización no se encuentra igualada para navegadores web, aunque hasta cierto punto trabajara sin ningún tropiezo.

Para publicar aplicaciones implementadas bajo esta metodología, no es necesario subirlo directamente a una tienda de aplicaciones oficial de la plataforma en cuestión, basta con subirla al sistema de aplicaciones de Ionic y desde ahí distribuirla. Para desarrollo bajo Angular existen otras soluciones robustas como Kendo UI, Mobile Angular UI, Onsen UI y Supersonic. Finalmente para desarrollos enfocados bajo HTML5 se pueden encontrar Famo.us, OnsenUI, Sencha Touch y Titanum Appcelerator.

IV. CONCLUSIONES

Aunque existen diferentes enfoques de desarrollo móvil, seleccionar cual es la mejor opción para trabajar no es una decisión que deba ser tomada a la ligera. Se deben evaluar diferentes factores según cada desarrollo, con el fin de determinar cuál es la mejor opción. Entre estas características a valuar se debe tener en cuenta los costos, acceso necesario de hardware (GPS, cámara, etc.), publico objetivo, diseños a la medida, acceso total a APIs nativas, escalabilidad y curvas de aprendizaje de los responsables del proyecto.

Los desarrollos híbridos han evolucionado mucho, pero se deben focalizar esfuerzos en el rendimiento para no sentir esos retardos significativos que hacen que la experiencia de usuario sea intratable. Además, los desarrollos web para páginas o sistemas de información pueden ser una excelente opción, pero no para una aplicación que necesite ejecuciones puntuales con el hardware del dispositivo o para ejecuciones simples que no necesites peticiones a un servidor.

Respecto a los nuevos lenguajes que surgen con el fin de reemplazar a Java como estándar de Android, se debe considerar que introducir un nuevo lenguaje por parte de google implica mucho trabajo, especialmente en rendimiento, compatibilidad o interoperabilidad. Se debe proyectar que pasara con todos los proyectos actuales de android que están sobre java y sus APIs, ¿Estos deben ser reescritos sobre DART o GO según el ganador? . Además las Apis de Java no están escritas de la mejor forma, por lo que pasarlas a otro lenguaje seria complejo, alejándose del cambio de paradigma, ¿Esta google preparado para algo así?

Si google decidiera establecer DART como lenguaje estándar de Android se debe considerar que depender de la web

para ejecutar aplicaciones móviles es impensable, debido a que no todos cuentan con conexión a internet, y según la ideología de Android este puede ser ejecutado sobre dispositivos de gama baja. Este segmento del mercado el cual es muy alto no garantiza una conexión permanente a la red, aunque en el futuro puede ser una posibilidad.

Otras posibles alternativas basadas en JVM para no sufrir un cambio tan drástico son soluciones como Scala, o Kotlin (Lenguaje de los creadores de Android Studio).

Referencias bibliográficas

[1] B. Markelj and I. Bernik, "Safe use of mobile devices arises from knowing the threats," *journal of information security and applications*, vol. 20, pp. 84 - 89, 2015.

[2] H. Jamaluddin, Z. Ahmad, M. Alias, and M. Simun, "Personal Internet use: The use of personal mobile devices at the workplace," *Procedia - Social and Behavioral Sciences, Global Conference on Business & Social Science-2014*, vol. 172, pp. 495 – 502, 2015.

[3] A. Sattineni and T. Schmidt, "Implementation of mobile devices on jobsites in the construction industry," *Procedia Engineering - Creative Construction Conference 2015*, vol. 123, pp. 488 – 495 2015.

[4] F.-M. Tseng, Y.-L. Liu, and H.-H. Wu, "Market penetration among competitive innovation products: The case of the Smartphone Operating System," *Journal of Engineering and Technology Management*, vol. 32, pp. 40-59, 2014.

[5] comScore. (2015). *comScore Reports March 2015 U.S. Smartphone Subscriber Market Share*. Available: http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-March-2015-US-Smartphone-Subscriber-Market-Share

[6] R. Appel, "Mobile Web sites vs. native apps vs. hybrid apps," *MSDN Magazine.*, vol. 29, pp. 76-79, Appel, Rachel.

[7] R. Newhook, D. Jaramillo, J. G. Temple, and K. J. Duke, "Evolution of the mobile enterprise app: A design perspective," *6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015*, vol. 3, 2015.

[8] J. Perchat, M. Desertot, and S. Lecomte, "Component Based Framework to Create Mobile Cross-platform Applications," *The 3rd International Symposium on Frontiers in Ambient and Mobile Systems*

(FAMS) - Procedia Computer Science, vol. 19, 2013.

[9] S. Church, "Develop hybrid native and mobile web apps," *MSDN Magazine.*, vol. 27, pp. 40-47, 2012.

[10] IBM. (2012). *El desarrollo de aplicaciones móviles nativas, Web o híbridas*. Available: ftp://ftp.software.ibm.com/la/documents/gb/commons/27754_IBM_WP_Native_Web_or_hybrid_2846853.pdf

[11] A. Charland and B. LeRoux, "Mobile Application Development:Web vs. Native," *acmqueue*, vol. 54, pp. 49-54, 2011.

[12] J. A. Zamora. (2014, Diciembre). *Dart, ¿el lenguaje de programación perfecto para Android?* Available: http://www.elandroidelibre.com/2014/06/dart-el-lenguaje-de-programacion-perfecto-para-android.html

[13] F. Palazuelos. (2015, Diciembre). *¿Podrá ser Dart y no Java el futuro de Android?* Available: http://hipertextual.com/2015/05/dart-android

[14] O. Campos. (2011, Diciembre). *Introducción al lenguaje de programación Go*. Available: http://www.genbetadev.com/herramientas/introduccion-al-lenguaje-de-programacion-go

[15] M. L. Michelone. (2012, Diciembre). *El lenguaje Go cumple tres años*. Available: https://www.unocero.com/2012/11/15/el-lenguaje-go-cumple-tres-anos/

[16] T. Rodrígez. (2014, Diciembre). *Con Go 1.4 ya es posible desarrollar aplicaciones Android*. Available: http://www.genbetadev.com/desarrollo-aplicaciones-moviles/con-go-1-4-ya-es-posible-desarrollar-aplicaciones-android

[17] Apple Inc, *The Swift Programming Language (Swift 2.1)* vol. 1: 1 Infinite Looop, 2015.

[18] B. Broulik. (2011). *Pro jQuery Mobile*. *11*. Available: http://eds.b.ebscohost.com.ezproxy.unal.edu.co/eds/detail/detail?sid=d1b5f8fb-20b0-4fdc-b69c-58b8bc641f9c@sessionmgr198&vid=1&hid=127&bdata=Jmxhbmc9ZXMmc2l0ZT1lZHMtbGl2ZQ==#db=cat02704a&AN=unc.000780008

[19] M. Gifford. (2012). *PhoneGap Mobile Application Development Cookbook*. Available: http://eds.b.ebscohost.com.ezproxy.unal.edu.co/eds/detail/detail?sid=26222538-351a-432e-9907-bf5ed209210f@sessionmgr115&vid=1&hid=127&bdata=Jmxhbmc9ZXMmc2l0ZT1lZHMtbGl2ZQ==#AN=499136&db=edsebk