Aplicaciones híbridas: estigmas, realidad y futuro

¿Qué son las aplicaciones híbridas? ¿Por qué aún siguen siendo un referente de bajo desempeño y mala usabilidad? Estas son algunas de las preguntas que los nuevos emprendimientos, al igual que corporaciones consolidadas, se hacen al momento de lanzar un nuevo producto o servicio de TI. Pero, ¿son justificadas?

Juan Sebastián Urrego E.

Introducción

En el año 2007, *Apple* lanzó su primera versión de lo que sería la revolución en dispositivos móviles: el *iPhone*. Con su salida al mercado, el concepto de teléfono móvil cambió radicalmente, haciendo entender al mundo entero que de ese momento en adelante hacer llamadas sería lo menos importante en un móvil. Sin embargo, grandes exponentes como Nokia, Blackberry y Microsoft no se quedarían atrás. Mientras el primero le trataba de dar una última oportunidad a *Symbian*, el segundo comenzaba una pequeña revolución en la mensajería virtual, y el tercero le daba un nuevo respiro a su odiado *Windows Mobile*, con el no tan nuevo *Windows Phone*. A este ecosistema se juntaba el gigante en crecimiento, Google, con su reciente compra *Android*, y es ahí donde el mundo de la telefonía móvil sufre un vuelco. Es el año 2011 y ahora Nokia, con un

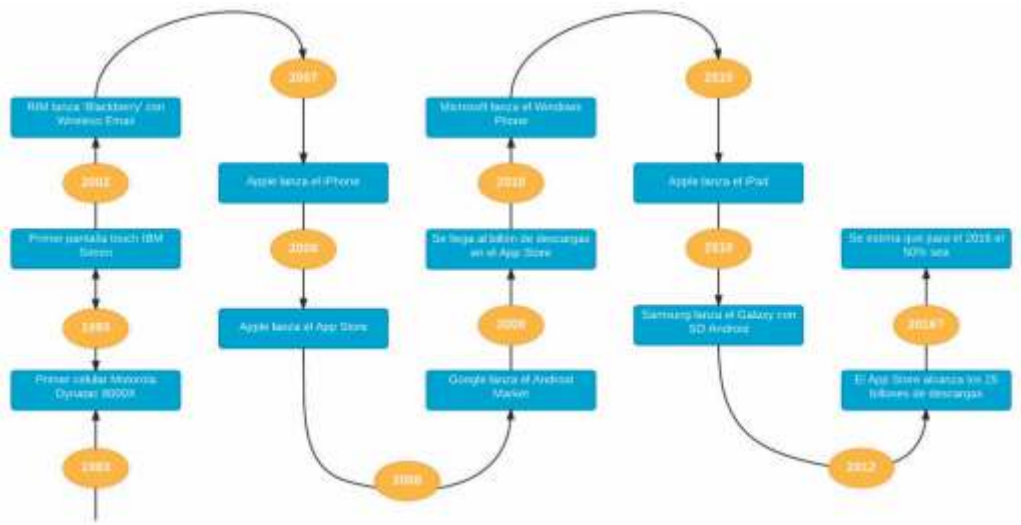

Figura 1. Evolución del desarrollo de aplicaciones móviles. Tomado de: Beginning Hybrid Mobile Application Development 1st ed. 2016 Edition

sistema operativo Windows Phone, quiere entrar al mercado de los *Smartphone*, donde *Android* y *iOS* dominan. Logra entrar, pero el éxito no es el que esperaban: aún falta mucho camino por recorrer (Figura 1).

A cuatro años de cumplir un quinto de siglo, nos encontramos con un ambiente diverso de sistemas operativos, *frameworks* y lenguajes de programación que pareciera tener un crecimiento exponencial. Y es que bajo la creciente demanda de aplicaciones móviles y *web*, el desarrollador actual debe convertirse en un experto en todos los frentes anteriormente nombrados. Sin embargo, la alta demanda exige cortos *time-to-market* y es ahí donde un ecosistema con baja mano de obra y alta demanda de tecnología comienza a colapsar. ¿Cómo desarrollar hasta cuatro aplicaciones en un corto plazo, con recursos humanos y económicos limitados? Hoy en día, una compañía típica puede estar solicitando una aplicación *web*, al igual que aplicaciones móviles para *Android*, *iOS* y *WindowsPhone* como mínimo, para que su proyecto tenga acogida en el público y sea un éxito.

Según mi experiencia, en promedio, el desarrollo de una aplicación nativa puede tardar de uno a tres meses, de acuerdo con su complejidad y el *framework* que se utilice. Por ejemplo, desarrollar una aplicación nativa para *iOS+Swift,* puede tardar un tiempo promedio de uno a dos meses con las respectivas pruebas, utilizando un

único recurso humano de tiempo completo. Por otra parte, una aplicación nativa para *Android*+ *Java*, puede tardar de dos a cuatro meses, con el mismo recurso humano.

Pero el tiempo de desarrollo no es el único problema. Podría decirse que para las empresas de TI el problema más "pequeño" es la incepción, diseño y construcción de sus productos. Sin embargo, el verdadero problema para dichas corporaciones es, en realidad, la modificabilidad, puesta en producción y futuro mantenimiento del producto.

Cuando tenemos cuatro aplicaciones para el mismo servicio, desarrolladas en cuatro lenguajes y *frameworks* distintos, hasta el cambio más mínimo puede ser demasiado costoso. ¿Cómo hacer para que todos los requerimientos de las aplicaciones -funcionales y no funcionales- se cumplan en las cuatro diferentes versiones? Es acá donde comenzamos a ver la desventaja para nosotros como arquitectos y líderes de tecnología, de la diversidad tecnológica que estamos viviendo (Figura 2).

Lo anterior no es un problema nuevo. En el año 2001 el proyecto Mono fue lanzado para solucionar la reciente tendencia de aplicaciones *desktop* multi-plataforma que se estaba comenzando a dar. Por aquella época, las aplicaciones *Cocoa* de *Apple* ya estaban saliendo a relucir en el panorama y debían ser una competencia a las aplicaciones aún dominantes desarrolladas en *Windows .NET*. Mono fue una plataforma que, usando un único lenguaje de programación (C#) y un único *framework* de desarrollo (.NET), podía compilar aplicaciones para

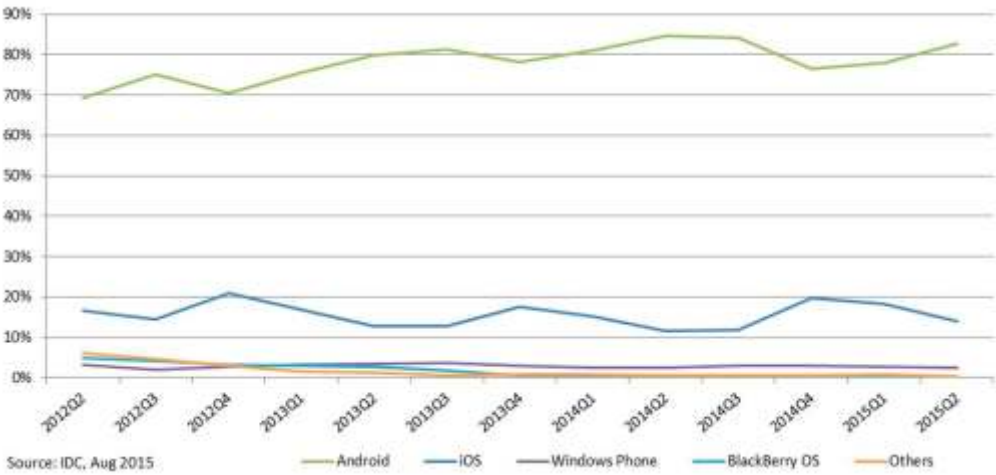

Figura 2. Marketshare mundial de dispositivos móviles. Tomado de: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Windows, Linux y *MacOS*. Esta plataforma abrió muchas puertas y ojos a nivel tecnológico y fue el nacimiento de las llamadas aplicaciones híbridas.

En este documento abordaremos los temas de aplicaciones móviles y *web* alrededor de esta nueva tendencia de aplicaciones híbridas. Entenderemos qué es una aplicación híbrida, los tipos que existen, plataformas de desarrollo y, lo más importante, los estigmas, realidades y futuro que esta tecnología nos brinda.

¿Qué son las aplicaciones híbridas?

Las aplicaciones híbridas son aplicaciones desarrolladas usando un único *stack* y empaquetadas para ser desplegadas en múltiples dispositivos, con diferentes tamaños de pantalla y fabricantes. Estas aplicaciones permiten al desarrollador construir sus productos en tecnologías simples como HTML, CSS y *Javascript* (JS). O, en otros casos, utilizando plataformas *server-side* para su implementación, por medio de lenguajes como C# y *VB.NET*.

Las aplicaciones híbridas tratan de mezclar lo mejor de ambos mundos: *Server-side power* e interacción gráfica con componentes simples, para beneficio de la experiencia de usuario (UX). Las aplicaciones híbridas son más que interacción gráfica como mucha gente cree, con éstas podemos utilizar componentes nativos de cada dispositivo, tratando de sacar el máximo provecho a los recursos físicos y lógicos de cada uno.

Las aplicaciones híbridas nos proveen múltiples ventajas entre las que se encuentran:

- Desarrollo independiente de plataforma.
- Desarrollo sencillo (corto plazo).
- Bajos costos.
- Mantener el mismo *look-and-feel.*

Hay que considerar que este estilo de aplicaciones también cuenta con ciertas desventajas, tales como:

- API límitado, con respecto al uso de recursos propios por plataforma.
- En ciertos casos, desempeño degradado (se debe hacer una buena optimización para ello).
- Mantener el mismo *look-and-feel* (en ciertos casos puede ser perjudicial para la UX).

Como se puede observar, mantener el mismo *look-and-feel* es bueno y malo. Esto puede sonar extraño, pero se debe a que si no aprendemos a usar las buenas prácticas para cada sistema operativo (SO), podemos crear

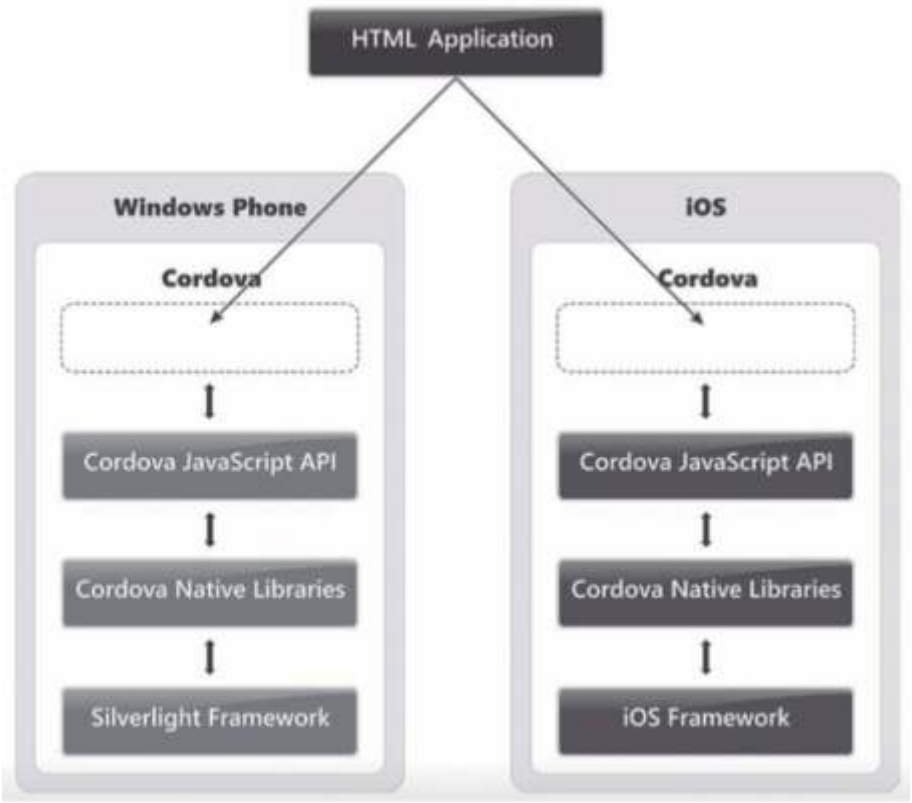

Figura 3. Arquitectura de Cordova. Tomado de: Beginning Hybrid Mobile Application Development 1st ed. 2016 Edition

problemas de interacción entre usuarios de diferentes plataformas.

A continuación se presentan los diferentes tipos de aplicaciones híbridas y sus características principales.

Tipos de aplicaciones

Las aplicaciones híbridas se pueden dividir en dos: *Web Embedded* y *Native Embedded.* El primer tipo de aplicación se caracteriza por un desarrollo con el *stack web* de HTML5. Por medio de este *stack*, somos capaces de emplear un lenguaje estándar, utilizado ampliamente a nivel global. Así mismo, podemos usar librerías de JS, tales como *JQuery*, que nos permite agregar interacción a nuestras aplicaciones. Su exponente principal en la actualidad es *Apache Cordova*, el cual es un *framework* multiplataforma que nos permite hacer un desarrollo usando HTML5, CSS3 y JS. Así mismo, *Cordova* nos crea un puente entre nuestra aplicación y los recursos físicos de los dispositivos, interactuando con los compo-

nentes nativos de cada SO. La Figura 3 nos muestra un poco la arquitectura de *Cordova*. Actualmente, la mayoría de *Frameworks Web-embedded* usan *Cordova* como su plataforma de compilación. Entre ellos encontramos: *Adobe Phonegap*, *Ionic*, *Monaca*, *OnSen UI*, *Appcelerator* y *Meteor*, entre otros.

Por otra parte, contamos con las aplicaciones *Native-Embedded*, las cuales, a diferencia de las *Web*, podemos desarrollar en un ambiente de programación *server-side* con la posibilidad de compilar nuestro código en nativo para los diferentes dispositivos. Actualmente, el exponente principal es *Xamarin*, una extensión de *Mono* que nos permite desarrollar nuestro código usando C# y compilando en componentes nativos de las diferentes plataformas. A diferencia de las aplicaciones *web*, *Xamarin* obliga al programador a tener un código compartido y un desarrollo dedicado (también en C#) para cada plataforma a nivel de capa de presentación. La Figura 4 nos presenta un ejemplo de la arquitectura de *Xamarin*.

Como se ha podido observar, las aplicaciones híbridas nos permiten

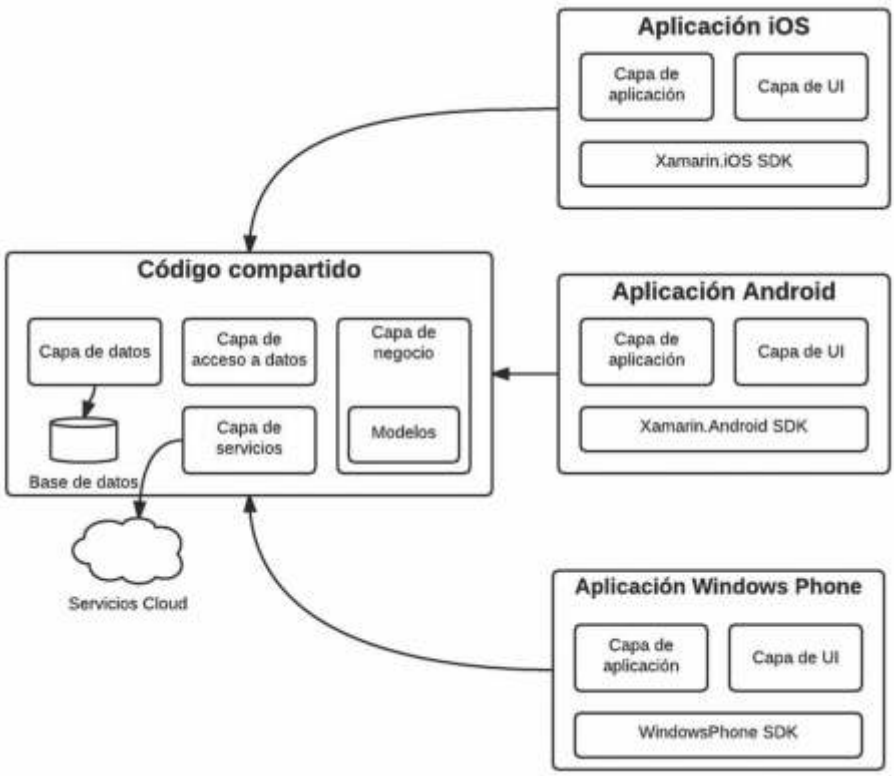

Figura 4. Arquitectura de Xamarin.
Tomado de: https://developer.xamarin.com

hacer un desarrollo una única vez y replicarlo a las diferentes plataformas disponibles en el mercado. Lo anterior permite que nuestro código sea más reusable, limpio y modificable en el tiempo. Sin embargo, ¿cómo afecta esto el presente y futuro de las *Apps*? En la siguiente sección daremos una introducción a este tema.

Estigmas

Para los conocedores de esta tecnología, el temor más grande es la falta de desempeño que suelen sufrir estas aplicaciones por aquellas capas adicionales que se deben implementar en el proceso. No obstante, en los últimos tres años el avance en los *frameworks* de aplicaciones híbridas ha mejorado de manera notoria, haciendo creer a los usuarios que lo que está usando es una aplicación nativa, cuando en verdad es una aplicación *web* embebida. A continuación se presentan algunos de los estigmas clásicos.

Bajo desempeño. Durante la concepción de esta tecnología esto era una completa verdad. Sin embargo, con el pasar de los años, los *frameworks* como *Apache Cordova* han mejorado considerablemente el desempeño, por lo que *frameworks* basados en esta plataforma como *Ionic* han optimizado el manejo de archivos CSS y JS, para que la UX sea similiar a la de una aplicación nativa. Por ejemplo, muchos *frameworks* han eliminado la llamada latencia de los 300 milisegundos que los móviles generan en aplicativos *web*.

Mala usabilidad e interfaz genérica. Hasta hace unos años las interfases de aplicaciones híbridas no se diferenciaban de las *web*, lo que hacía que se tuviesen aplicaciones móviles que no parecían móviles y un poco erróneas a nivel de interacción. Esto ha cambiado totalmente, toda vez que ya existen librerías y *frameworks* que acomodan la interfaz para que se usen los elementos gráficos propios de cada sistema operativo.

Dificil de desarrollar y configurar. Al igual que los anteriores, en un principio desarrollar en estas plataformas era limitante e irritante, para ser sincero. No obstante, con la llegada de *Cordova* y el uso de *NodeJS* para la compilación de los archivos, el desarrollo se ha vuelto extremadamente sencillo y simple tanto así que, en la mayoría de los casos, ha resultado ser más fácil que realizar un desarrollo nativo. *Frameworks* como *Ionic* y *Phonegap* nos permiten probar nuestra aplicación en diferentes dispositivos sin tener que instalar la aplicación en los dispositivos de prueba.

Conclusiones: presente y futuro

Según Gartner, para el año 2016, el 50% de las aplicaciones en el mercado serán híbridas. Lo

anterior nos indica que en la actualidad hay un fuerte mercado, el cual va a tender a crecer con el tiempo. Esto no significa que el desarrollo de aplicaciones nativas vaya a desaparecer, puesto que los lenguajes y *frameworks* nativos nos siguen brindando ciertas ventajas sobre el desarrollo híbrido. En grandes compañías donde el tiempo y recurso económico no son un problema, se prefiere un desarrollo nativo que beneficie la UX y la optimización de recursos computacionales.

El desarrollo híbrido se nos presenta como una oportunidad para hacer aplicaciones *web* y móviles multiplataforma, que sean reusables, en un corto tiempo y modificables a bajos costos. El futuro todavía es incierto con respecto a qué nuevos *frameworks* y tecnologías nos va a traer el tiempo, pero algo claro es que las aplicaciones híbridas seguirán dando de qué hablar y se enfrentarán a nuevos retos como la optimización de recursos y tendencias como *Internet of Things (*IoT).

Referencias

[1] Panhale, M. (2016) *BeginningHybrid MobileApplicationDevelopment*.

[2] Gartner. (2013) Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid. Febrero. Recuperado de:http://www.gartner.com/newsroom/id/2324917

[3] Statista (2015) Number of apps in leading stores. Julio. Recuperado de: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[4] IDC. (2015) Smartphone OS Market Share, 2015 Q2. Recuperado de: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[5] Mendoza, A. (2015) Mobile User Experience: Patterns to Make Sense of it All.

[6] Esposito, D. (2012) *Architecting Mobile Solutions for the Enterprise*.

[7] Camden, R. (2016) Apache Cordova in Action

[8] Hermes, D. (2015) Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals.

[9] Unger, R. Chandler, C (2012) *A Project Guide to UX Design: For user experience designers in the field or in the making*.

[10] McWhester, J. (2012) *Professional Mobile Application Development.*

Juan Sebastián Urrego, MSc .*Profesor instructor de la Universidad de los Andes. CEO y co-fundador de Novcat. Arquitecto de software y solución con experiencia en análisis, consultoría, diseño, modelado y construcción de software empresarial. Conferencista en eventos internacionales como ICSE, XP y Saturn.*