Análisis Experimental de desarrollo de Aplicaciones Móviles Multiplataforma

Lisandro Delía[1], Nicolás Galdamez[1], Pablo Thomas[1], Leonardo Corbalan[1]
Patricia Pesado[1]

[1] Instituto de Investigación en Informática LIDI. Facultad de Informática.
Universidad Nacional de La Plata. Argentina
{ldelia, ngaldamez, pthomas, corbalan, ppesado}@lidi.info.unlp.edu.ar

Resumen. Los dispositivos móviles han cambiado la forma de pensar el software. Existe un gran abanico de alternativas de desarrollo. En este trabajo se analizan cuatro enfoques multiplataforma distintos (aplicaciones web móviles, híbridas, interpretadas, compilación cruzada) y se estudian sus características más destacadas, a través de un caso experimental.

Palabras claves: dispositivos móviles, aplicaciones móviles multiplataforma, aplicaciones móviles web, aplicaciones móviles híbridas, aplicaciones móviles interpretadas, compilación cruzada.

1 Introducción

El desarrollo de software para dispositivos móviles plantea nuevos desafíos originados en las características únicas de esta actividad. La necesidad de tratar con diversas plataformas, estándares, protocolos y tecnologías de red; las capacidades limitadas, aunque en continua evolución, de los dispositivos y las exigencias de tiempo del mercado, son sólo algunos de los problemas a tratar. Por ello, el desarrollo de software para dispositivos móviles difiere considerablemente del tradicional [1].

Para maximizar su presencia en el mercado, un producto de software debe correr en la mayor cantidad de dispositivos posible. Una solución consiste en el desarrollo nativo de la aplicación en cada una de las plataformas existentes utilizando el entorno de desarrollo integrado (IDE por sus siglas en inglés), el lenguaje y las herramientas propias de cada plataforma [2].

Las aplicaciones nativas ofrecen la posibilidad de acceder a todas las capacidades del dispositivo (cámara, GPS, acelerómetro y agenda, entre otras), su rendimiento es alto, el acceso a Internet no es estrictamente necesario y pueden ejecutarse en segundo plano notificando al usuario cuando se requiera su atención. Estas aplicaciones pueden distribuirse/comercializarse a través de las tiendas en línea correspondientes. Sin embargo, el precio de todas estas bondades es alto: no es posible reusar el código fuente entre plataformas diferentes, el esfuerzo se multiplica y se elevan los costos de desarrollo, actualización y distribución de nuevas versiones.

El desarrollo multiplataforma se contrapone al nativo y se centra en el reúso de código. La construcción de aplicaciones web móviles constituye un ejemplo

representativo de este enfoque. Sin embargo, las limitaciones derivadas de su ejecución dentro de un navegador, ha motivado a los ingenieros de software a dirigir su atención hacia otro tipo de aplicaciones multiplataforma con el que se obtienen resultados más cercanos a las soluciones nativas. En este contexto, existen diversas subclasificaciones [2] [3] [4] y es de interés analizar las características inherentes a cada una de ellas, a través de la construcción de un prototipo experimental.

Este trabajo, es una evolución de [5] que incorpora un análisis experimental de aplicaciones móviles multiplataforma. En la sección 2 se detallan las características comunes más salientes de aplicaciones móviles multiplataforma. Posteriormente se describe brevemente el caso experimental, y en las secciones siguientes se presenta su desarrollo en Phonegap con Jquery Mobile, Sencha Touch, Appcelerator Titanium 3, Xamarin y en Delphi XE6. Finalmente, se expresan conclusiones y trabajos futuros.

2 Aplicaciones móviles multiplataforma

El desarrollo multiplataforma procura optimizar la relación costo/beneficio compartiendo la misma codificación entre las versiones para las distintas plataformas. Entre otras ventajas sobresalen: menor tiempo y costo de desarrollo; prestaciones cercanas a las nativas con acceso al hardware del dispositivo y disponibilidad de entornos potentes de desarrollo (Delphi, Visual Studio) o, en su lugar, utilización de tecnologías bien conocidas por los desarrolladores web (HTML5, Javascript y CSS) que pueden trasladar sus conocimientos y experiencias al paradigma móvil.

Las aplicaciones multiplataforma pueden clasificarse en: aplicaciones web móviles, híbridas, interpretadas y generadas por compilación cruzada [2].

2.1. Aplicaciones Web Móviles

Estas aplicaciones, diseñadas para correr dentro de un navegador, se desarrollan con tecnología web (HTML, CSS y JavaScript) y cuentan con una serie de características favorables: no necesitan adecuarse a ningún entorno operativo, son independientes de la plataforma y su puesta en marcha es rápida y sencilla.

Por contrapartida, sus tiempos de respuesta decaen afectados por la interacción cliente-servidor, al mismo tiempo que resultan ser menos atractivas que las aplicaciones nativas ya que no se encuentran instaladas en el dispositivo. Además, las restricciones de seguridad impuestas a la ejecución de código por medio de un navegador, limitan a estas aplicaciones que no pueden acceder a todas las capacidades del dispositivo[6].

2.2. Aplicaciones Híbridas

Las aplicaciones híbridas utilizan tecnologías web (HTML, Javascript y CSS) pero no son ejecutadas por un navegador. En su lugar, corren en un contenedor web del dispositivo con mayor acceso a sus capacidades específicas a través de una API.

Las aplicaciones híbridas ofrecen grandes ventajas permitiendo la reutilización de

código en las distintas plataformas, el acceso al hardware del dispositivo, y la distribución a través de las tiendas de aplicaciones [5].

Se observan dos desventajas de las aplicaciones híbridas respecto del caso nativo: i) la experiencia de usuario se ve perjudicada al no utilizar componentes nativos en la interfaz, y ii) la ejecución se ve ralentizada por la carga asociada al contenedor web.

2.3. Aplicaciones Interpretadas

Las aplicaciones interpretadas consisten en un proyecto base que se traduce en su mayor parte a código nativo mientras el resto es interpretado en ejecución. Se implementan de forma independiente de las plataformas utilizando diversas tecnologías y lenguajes, tales como Java, Ruby y XML, entre otros.

La obtención de interfaces nativas constituye una de las principales ventajas de este tipo de aplicaciones, y la dependencia total con el entorno de desarrollo el obstáculo más notable. Appcelerator Titanium [7] es el entorno de desarrollo más popular.

2.4. Aplicaciones Generadas por Compilación Cruzada

Estas aplicaciones se compilan de manera nativa creando una versión específica de alto rendimiento para cada plataforma destino. Ejemplos de entornos de desarrollo para generar aplicaciones por compilación cruzada son Applause [8], Embarcadero Delphi XE6 [22] y Xamarin[26].

El entorno de desarrollo abierto Applause utiliza como entrada un lenguaje específico del dominio basado en el framework Xtext [9], diseñado explícitamente para aplicaciones móviles orientadas a datos, y genera código fuente en Objective C, Java, C# o Python. Las características de Delphi XE6 y Xamarin se presentan en secciones posteriores.

3 Caso experimental: WebUNLP

WebUNLP es un entorno virtual de enseñanza y aprendizaje que permite a los docentes mediar sus propuestas educativas y crear un espacio de encuentro con los alumnos para comunicarse, compartir material de estudio y generar una experiencia educativa en forma virtual [10].

Actualmente, WebUNLP es una aplicación web para computadoras de escritorio y portátiles no adaptada a las características de los dispositivos móviles. Sin embargo, en [5] se extendió cierta funcionalidad de WebUNLP a estos dispositivos como parte de un estudio experimental sobre tipos de aplicaciones para dispositivos móviles. En el presente artículo se enriquece dicho análisis a partir de la experimentación con los enfoques multiplataforma: web móvil, híbrido, interpretado y compilación cruzada

La utilidad elegida para la incursión de WebUNLP en las plataformas móviles ha sido la cartelera electrónica, una herramienta de comunicación para publicar novedades de los cursos, como por ejemplo, el cambio de horario de una cursada o el recordatorio de las fechas de entrega de un trabajo práctico.

Algunos requerimientos a cumplir por la aplicación móvil son:

- El usuario debe poder ingresar a la aplicación con las mismas credenciales que las utilizadas para acceder a la versión web.
- El usuario debe poder acceder a la cartelera de todos los cursos en los que participa, ya sea como docente o alumno.
- El usuario debe recibir una notificación en su dispositivo cuando una novedad es publicada en la cartelera. Este requerimiento no se puede cumplir en la versión web accesible desde computadoras de escritorio y/o portables.
- El usuario debe tener la misma experiencia de uso en todas las plataformas operativas
- La aplicación web existente debe estar sincronizada con la aplicación móvil a desarrollar, esto significa que cualquier cambio realizado desde la aplicación móvil debe verse reflejado en la versión web y viceversa.

En cuanto a la interfaz gráfica, se planteó un diseño de navegación en serie, en el orden en que se presentan en el mockup [11] de la figura 1.

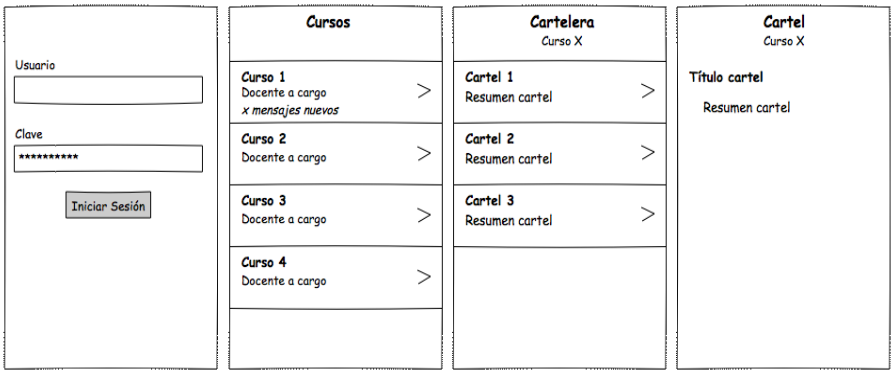

Fig. 1. Mockup del prototipo de cartelera WebUNLP

4 Aplicación Web Móvil de WebUNLP

Se desarrolló una aplicación web capaz de acceder a la cartelera de WebUNLP, disponible para cualquier dispositivo móvil que cuente con un navegador que soporte las características utilizadas para el desarrollo: HTML5, CSS3 y Javascript.

Como la velocidad de transmisión/recepción de datos de un dispositivo móvil a través de WiFi, y en particular 3G, es inferior a la velocidad de una computadora de escritorio, la versión web de la cartelera de WebUNLP es liviana y gran parte de los requerimientos son implementados a través de Ajax [12] para evitar, ante algún cambio, la recarga de la página completa.

Debido a las limitaciones que tienen las aplicaciones que corren sobre un navegador, no es posible implementar la recepción de una notificación en el dispositivo cuando una novedad es publicada en la cartelera.

5 Aplicación Híbrida de WebUNLP

5.1. Aplicación Híbrida desarrollada con PhoneGap y Jquery Mobile

Se utilizó el framework gratuito y open source PhoneGap [13] que permite desarrollar aplicaciones móviles que utilizan tecnologías comunes a todos los dispositivos: HTML5, CSS y Javascript. Asimismo se utilizó el framework Javascript denominado Jquery Mobile [14] para lograr interfaces con apariencia y comportamiento consistente en las diferentes plataformas móviles.

Uno de los requerimientos de la aplicación experimental WebUNLP consiste en recibir una notificación en el dispositivo, en el momento en que se publica una nueva novedad en la cartelera. Para satisfacer este requerimiento se utilizó el plugin oficial de Phonegap conocido como PushPlugin [15], el cual implementa la recepción de notificaciones en el dispositivo. Este plugin está implementado para las plataformas: Android, iOS, Windows Phone 8 y Amazon Fire OS.

En la figura 2, se presentan las interfaces de la aplicación desarrollada.

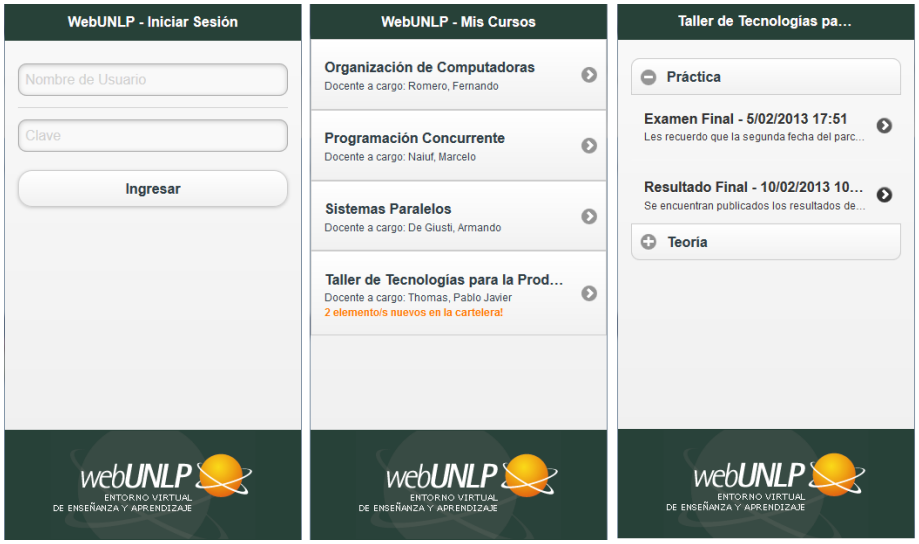

Fig. 2. Aplicación desarrollada con PhoneGap y Jquery Mobile

5.2. Aplicación Híbrida desarrollada con Sencha Touch

Sencha Touch [16] es un framework gratuito MVC JavaScript construido sobre el sistema de clases de Ext JS, diseñado especialmente para el desarrollo de aplicaciones web móviles para dispositivos táctiles [17].

Al igual que PhoneGap, Sencha Touch, permite a los desarrolladores crear aplicaciones móviles a partir de un desarrollo web HTML5, CSS y Javascript.

El desarrollo de WebUNLP utilizando este framework se vio favorecido por el uso

del patrón de diseño MVC. Ésto permitió mayor flexibilidad y legibilidad en el código [17][18].

Sencha ofrece Sencha Command, una herramienta multiplataforma de línea de comandos que provee tareas automatizadas para todo el ciclo completo de una aplicación [19]. En el caso de WebUNLP se utilizó tanto para la generación del proyecto como para el empaquetado de la aplicación.

En la figura 3, se presentan las interfaces de la aplicación desarrollada.

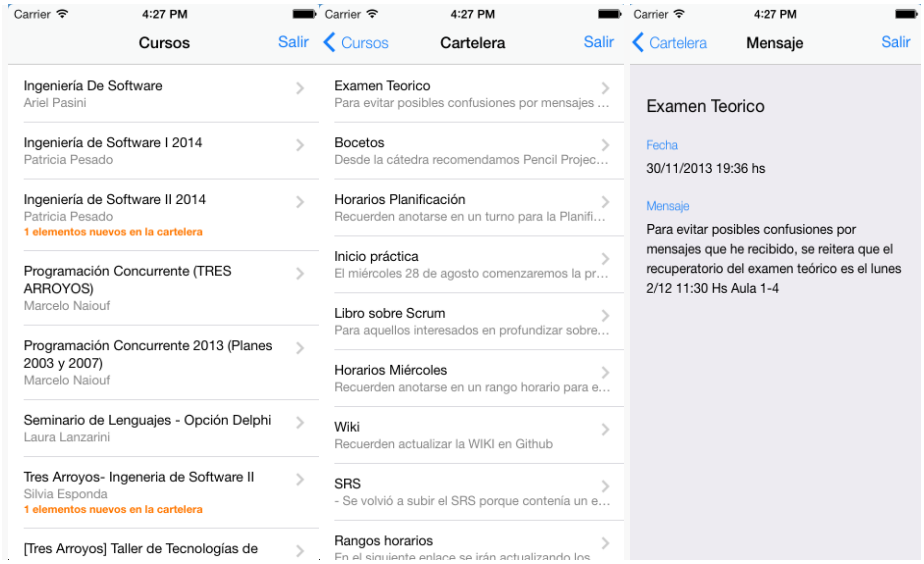

Fig. 3. Aplicación desarrollada con Sencha Touch

6 Aplicación Interpretada de WebUNLP con Appcelerator Titanium 3

Para desarrollar una aplicación móvil experimental según el enfoque interpretado se utilizó el entorno de desarrollo gratuito y open source Appcelerator Titanium 3 [7].

Este entorno de desarrollo utiliza el framework Alloy diseñado para el desarrollo ágil de aplicaciones móviles. Alloy está basado en la arquitectura MVC y soporta el uso de tecnologías populares como Backbone.js [20] y Underscore.js [21].

Se destacan la simplicidad y legibilidad de los controladores y modelos de la aplicación. Para la construcción de las vistas se puede optar por la programación mediante Javascript y la API de Titanium, o bien por una especificación XML con estilos TSS (Titanium Style Sheets). Esta última opción simplifica el proceso de creación de las vistas aunque falta todavía un buen editor visual de interfaces que lo potencie.

Para cubrir requerimientos de notificaciones en el dispositivo, Titanium provee el módulo PushNotifications para plataformas Android e iOS.

La figura 5a muestra el experimento de desarrollo con Titanium.

7 Aplicación WebUNLP Generada por Compilación Cruzada

7.1. Desarrollo con Xamarin/Visual Studio

Xamarin es una plataforma de desarrollo propietaria y no gratuita que permite escribir y compilar aplicaciones 100% nativas para iOS, Android y Mac compartiendo el mismo código base escrito completamente en el lenguaje C#. Si bien cuenta con su propio IDE denominado Xamarin Studio, es posible integrarlo con Microsoft Visual Studio, y de esta manera generar también aplicaciones para Windows, incluido Windows RT para tablets y Windows Phone para celulares.

Xamarin propone un enfoque de desarrollo multiplataforma en el que se comparte la codificación completa de la lógica del negocio. Sin embargo, las interfaces deben ser programadas de manera independiente para cada una de las plataformas destino (ver figura 4). Así, la reutilización de código, según estudios estadísticos de la compañía Xamarin, es cercana al 85%.

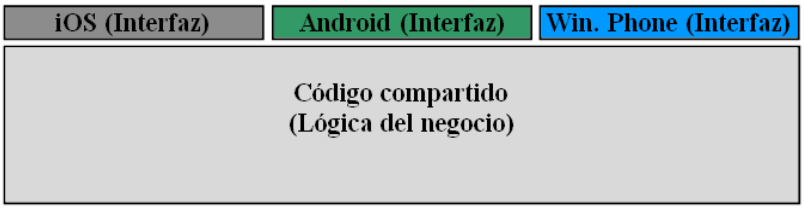

Fig. 4. Enfoque único de desarrollo Xamarin

Para el desarrollo de WebUNLP se ha utilizado Xamarin integrado con Microsoft Visual Studio. Siguiendo la estrategia más eficiente para trabajar en este entorno se ha creado una solución única conteniendo tres proyectos distintos. Uno de ellos se utilizó para generar la aplicación Android, otro para la aplicación Windwos Phone 8 y el tercero para la implementación de una biblioteca de clases portable (PCL por sus siglas en inglés) con todo el código compartido. Los tres proyectos fueron desarrollándose de manera conjunta y concurrente.

Aunque una de las ventajas más importantes de Xamarin es la reutilización de código, ésta se ve fuertemente afectada por el tipo de aplicación que se desarrolla. La relación existente entre la complejidad de la interfaz y la lógica del negocio impacta directamente sobre la posibilidad de mantener la mayor cantidad de código compartido en la PCL. En el desarrollo de WebUNLP, la reutilización de código ha sido cercana al 50%. Sin embargo, no debe despreciarse la ventaja de utilizar el mismo lenguaje, entorno y conjunto de herramientas comunes en el desarrollo de aplicaciones para distintas plataformas móviles.

La figura 5b presenta la interfaz desarrollada para Windows Phone 8.

7.2. Desarrollo con Embarcadero Delphi XE6

Embarcadero Delphi XE6 [22] es una plataforma de desarrollo propietaria y no

gratuita, que permite a los desarrolladores crear aplicaciones rápidamente a través de un entorno visual cómodo e intuitivo. La aplicación desarrollada puede ser compilada para múltiples plataformas, incluyendo Windows, Mac, Android e iOS.

Delphi XE6, al igual que Xamarin, aventajan a otros tipos de aplicaciones multiplataformas con resultados 100% nativos, con acceso total a los sensores y capacidades del dispositivo (cámara, notificaciones, GPS, acelerómetro, etc).

En particular, para el desarrollo de WebUNLP fueron de gran utilidad tanto los componentes provistos por la herramienta para acceder a servicios RESTful como el entorno visual de desarrollo.

Para la recepción de notificaciones en el dispositivo de las novedades de WebUNLP se utilizó el componente TPushEvent conectado al servicio Kinvey [23].

La figura 5c presenta la interface de la aplicación desarrollada.

Fig. 5. Aplicación desarrollada con: a) Titanium; b) Xamarin/Visual Studio y c) Delphi XE6

8 Conclusiones

Dado el aumento en la demanda de software específico para dispositivos móviles y el número creciente de plataformas, el desarrollo móvil se ha visto nutrido por la aparición de nuevas herramientas y tecnologías. Vista la necesidad de las empresas de cubrir la mayor parte del mercado, la implementación de aplicaciones multiplataforma es considerada una opción atractiva tomando como objetivo la reducción de tiempos y costos.

Considerando la dificultad de desarrollar aplicaciones nativas para las múltiples plataformas móviles se realizó un análisis experimental de desarrollo de aplicaciones móviles multiplataforma.

Se optó como caso de estudio WebUNLP, un entorno virtual de enseñanza y aprendizaje utilizado por diversos cursos de grado y postgrado de la Universidad Nacional de La Plata. El desarrollo fue replicado en cuatro clases de aplicaciones:

web, híbrido, interpretado y compilación cruzada.

De la aplicación web móvil se destaca como principal ventaja la simpleza del desarrollo y la facilidad de distribución (sólo es necesario un navegador). Como contrapartida no es posible recibir notificaciones en el dispositivo cuando una novedad es publicada en la cartelera.

Por otra parte, la aplicación híbrida desarrollada con Phonegap logró conjugar la simpleza del desarrollo web con el uso de todas las capacidades del dispositivo. Su rendimiento es superior a una aplicación web móvil pero inferior a una aplicación nativa.

Para el caso de las aplicaciones interpretadas se estudió Titanium y entre sus ventajas sobresale la generación de código nativo para la interfaz logrando un alto rendimiento. Como crítica, se señala la falta de una herramienta visual que asista en el diseño de las interfaces.

Finalmente para el caso de aplicaciones generadas por compilación cruzada se exploraron dos alternativas: Xamarin/Visual Studio y Delphi XE6. En ambos casos se obtuvieron aplicaciones totalmente nativas aunque con distintos niveles de reutilización de código: 100% .con Delphi XE6 y 50% con Xamarin/Visual Studio. La causa de esta diferencia radica en la necesidad en el último caso de codificar las interfaces individualmente para cada plataforma.

Como conclusión de las experiencias realizadas se señala que las mejores alternativas que conjugan buen rendimiento y experiencia del usuario nativa son las desarrolladas con Titanium 3, Xamarin y Delphi XE6, destacando la plataforma Titanium por ser gratuita y de código abierto.

8 Trabajo futuro

Profundizando el estudio sobre el desarrollo de aplicaciones multiplataforma se plantea como trabajo futuro el análisis experimental sobre el alcance de otras herramientas disponibles en el mercado, como AppMethod [24] y Applause [25] entre otras.

Para llevar a cabo este estudio se medirán experimentalmente valores cuantitativos y cualitativos como cantidad de código reutilizado, rendimiento, consumo de batería, "*look and feel*" aceptación del usuario etc., que permitan caracterizar con precisión cada enfoque/herramienta analizada.

Se plantea además extender la aplicación móvil WebUNLP incorporando funcionalidades de la plataforma web, como la mensajería y el foro, añadiendo también un nuevo servicio de chat.

Referencias

1. Hayes, I. S. *Just Enough Wireless Computing*. Prentice Hall Professional Technical Reference . 2002. ISBN:0130994618
2. Spyros Xanthopoulos, Stelios Xinogalos, *A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications*, BCI' 2013, Greece

3. Yonathan Aklilu Redda, *Cross platform Mobile Applications Development*, Norwegian University of Science and Technology, Norwegian University of Science and Technology, Norwegian University of Science and Technology, Master in Information Systems, June 2012.
4. Dalmasso I., Datta S.K., Bonnet C. Nikaein N., *Survey, comparison and evaluation of cross platform mobile application development tools,* Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International.
5. Delia L., Galdamez N. Thomas P., Pesado P., Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles, XVIII Congreso Argentino de Ciencias de la Computación, CACIC 2013.
6. Tracy, K.W., *Mobile Application Development Experiences on Apple's iOS and Android OS*, Potentials, IEEE, 2012
7. http://www.appcelerator.com/
8. https://github.com/applause/applause
9. http://www.eclipse.org/Xtext/
10. http://webunlp.unlp.edu.ar
11. http://es.wikipedia.org/wiki/Mockup
12. https://developer.mozilla.org/en/docs/AJAX
13. http://phonegap.com/
14. http://jquerymobile.com/
15. https://github.com/phonegap-build/PushPlugin
16. http://www.sencha.com/
17. Kosmaczewski, A. (2013). Sencha Touch 2 Up and Running. O'Reilly.
18. Clark, J. (2013). Creating Mobile Apps with Sencha Touch 2. Packt Publishing.
19. Lee Bonstra. (2014). Hands-On Sencha Touch 2.
20. http://backbonejs.org/
21. http://underscorejs.org/
22. https://www.embarcadero.com/es/products/delphi
23. http://www.kinvey.com/
24. http://www.appmethod.com/
25. https://github.com/applause/applause
26. http://xamarin.com/