Comparison of Popular Cross-Platform Mobile Application Development Tools

Volkan Tunalı
University of the West of Scotland
31 PUBLICATIONS **237** CITATIONS
SEE PROFILE

Senol Zafer Erdogan
Konya Food and Agriculture University
29 PUBLICATIONS **395** CITATIONS
SEE PROFILE

COMPARISON OF POPULAR CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT TOOLS

Assist. Prof. Dr. Volkan TUNALI
Celal Bayar University Faculty of Technology Department of Software Engineering
volkan.tunali@cbu.edu.tr

Assoc. Prof. Dr. Şenol Zafer ERDOĞAN
Maltepe University Faculty of Engineering and Natural Sciences Department of Software Engineering
senolerdogan@maltepe.edu.tr

Abstract

Proliferation of the Internet and mobile devices like smart phones and tablets has dramatically changed the way people do business, access information, communicate with each other, and have leisure time on the go. Mobile application have become more important than ever in daily life in this regard. It has therefore become a challenge for software vendors to create mobile applications that can run on many common mobile operating systems like iOS, Android, and Windows Phone. One approach to this problem is to have separate code bases and separate development teams for each platform for the same application, which is obviously a costly solution. Therefore, applications vendors continuously seek better long-term alternatives. In order to reduce development effort and cost while providing rapid delivery to all application markets at once, a preferred alternative is to use a cross-platform mobile application development tool/framework. In this paper, we present an elaborate comparison of several popular cross-platform mobile application development tools, namely, PhoneGap/Cordova, Xamarin, Appcelerator Titanium, and Smartface App Studio. We provide a pragmatic comparison from different perspectives like ease and cost of development including programming language and tool support, end-product capability and performance, security, and community support. This comparison aims to support mobile application vendors from a large scale, from freelancers to enterprise development teams, when making a choice with respect to their specific requirements and constraints.

Keywords: mobile application development, software development, programming, iOS, Android.

POPÜLER PLATFORMLAR-ARASI MOBİL UYGULAMA GELİŞTİRME ARAÇLARININ KARŞILAŞTIRILMASI

Özet

İnternetin ve akıllı telefon ve tablet gibi mobil cihazların yaygınlaşması, insanların iş yapma, bilgiye erişme, birbirleriyle iletişim kurma, ve boş zaman geçirme şekillerini önemli ölçüde değiştirmiştir. Bu bakımdan mobil uygulamalar, günlük yaşamda her zamankinden daha fazla önemli hale gelmiştir. Dolayısıyla, iOS, Android, ve Windows Phone gibi pek çok yaygın mobil işletim sisteminde çalışabilen mobil uygulamalar geliştirmek, yazılım üreticileri için çok zorlu bir iş haline gelmiştir. Bu probleme yönelik bir yaklaşım, aynı uygulamanın çalışacağı her bir platform için ayrı bir kod tabanı ve ayrı bir geliştirme ekibi oluşturmaktır, ancak bu da açıkça çok maliyetli bir yaklaşımdır. Bundan dolayı, uygulama üreticileri sürekli olarak, uzun süreli kullanılabilecek daha iyi alternatifler arayışındadır. Tüm uygulama marketlerine tek seferde ve hızlı teslimat sağlarken, geliştirme emek ve maliyetini azaltabilmek için oldukça tercih edilen bir yaklaşım ise bir platformlar-arası mobil uygulama geliştirme aracı/çatısı kullanmaktır. Bu bildiride, PhoneGap/Cordova, Xamarin, Appcelerator Titanium, ve Smartface App Studio olmak üzere popüler platformlar-arası mobil uygulama geliştirme araçlarının ayrıntılı bir karşılaştırmasını sunmaktayız. Programlama dili ve araç desteği dahil olmak üzere geliştirme kolaylığı ve maliyeti, son ürün yetenekleri ve performans, güvenlik, ve son olarak topluluk desteği gibi çok farklı bakış açılarından pragmatik bir karşılaştırma yapmaktayız. Bu karşılaştırma, bağımsız çalışan programcılardan kurumsal geliştirme ekiplerine kadar geniş bir yelpazade mobil uygulama üreticilerine, özel gereksinim ve kısıtlarını da hesaba katarak bir seçim yaparlarken destek olmayı amaçlamaktadır.

Anahtar Kelimeler: mobil uygulama geliştirme, yazılım geliştirme, programlama, iOS, Android.

1. INTRODUCTION

Technological developments have always affected the life and work style of both people and organizations. Communication and mobile technologies are no exception. The Internet and mobile and wearable devices like smart phones, tablets, and smart watches are now so widespread that they have considerably changed the way people and organizations do business, access information, communicate with each other, and do other activities. The role of mobile applications (or simply app) in this change, of course, is enormous. According to recent statistics, there are about 1.6 million apps on Google Play, 1.5 million apps on Apple Store, and 1 million apps on other app stores such as Windows Phone Store and Amazon AppStore as of July 2015 (Statistica, 2015). These figures of course are increasing continuosly as a result of increasing user demand and advanced functionalities new mobile devices has to offer.

Today, mobile software vendors face a challenge that they usually have to create mobile apps that run on many common mobile operating systems like iOS, Android, and Windows Phone at the same time because there are business opportunities not to be missed on each app market. One major difficulty here is that vendors need to provide a consistent user experience (UX) across all operating systems, anticipating the platform-/device-specific features. One approach to this problem is to have separate code bases, and, as a result, to have separate development teams for each platform for the same app because each platform requires expert knowledge and experience in the platform. However, this approach is the most costly solution in terms of development, update, and support. In order for software vendors to survive in today’s highly competetive software industry, they have to reduce development effort and cost while providing rapid delivery to all app markets at once. To meet this specific need, cross-platform mobile development tools and frameworks have emerged to help software vendors develop their apps with a single code base once, and run them on different mobile platforms.

Currently, there are many cross-platform mobile app development tools and frameworks, and there are several differences (and similarities, as well) among them with respect to their development approaches. In this paper, we present a detailed comparison of several popular cross-platform mobile development tools, namely, PhoneGap/Cordova, Xamarin, Appcelerator Titanium, and Smartface App Studio. We provide a pragmatic comparison from different perspectives like ease and cost of development including programming language and tool support, end-product capability and performance, security, and community support.

This paper is organized as follows. Section 2 explains the approaches to mobile application development. Cross-platform mobile application development tools examined in this paper are introduced in Section 3. Our comparison criteria are explained in Section 4. A detailed comparison among the tools are given in Section 5. Finally, Section 6 contains some conclusions.

2. MOBILE APPLICATION DEVELOPMENT APPROACHES

For mobile app development, there are several alternative approaches that even software vendors usually have confusion about. Each one of these approaches has its own advantages and disadvantages, depending on their development strategy and user experience their products can offer. In this section, we try to clarify each one, paying careful attention to the terminology often misused in the industry.

2.1. Mobile Responsive Web App Development

In this type of mobile app development, Responsive Web Design (RWD) pattern is usually followed. RWD is an approach to web site design aimed at creating web sites to provide an optimal viewing and interaction experience like easy reading and navigation with a minimum of resizing, panning, and scrolling, across a wide range of devices from desktop computer monitors to tablets and mobile phones (LePage, 2014). Either a new mobile version of an existing traditional web site is created keeping the existing one for desktop access, or the web site is created totally with this approach from scratch. Standard web technologies such as HTML5, JavaScript, and CSS are used in the development of mobile responsive web apps. Although it looks advantageous at first to create mobile apps this way, as with any other website, these type of apps are restricted to the features of the browser used to view the app. In addition, they cannot provide rich user experience that native apps can do by using several device-related APIs (Application Programming Interface) such as Camera, Accelerometer, GPS, and so on (Ramanujam & Natili, 2015).

2.2. Native App (Platform-Based Native) Development

Native apps are the ones specifically designed and developed for a given mobile platform using the development tools and languages that the respective platform natively supports (Smartface, 2013). For example, Xcode IDE (Integrated Development Environment) and Objective-C language (or Apple's new Swift language) are used for iOS, Eclipse IDE and Java language are used for Android, and finally Visual Studio IDE and C# language are used for Windows Phone environment. Native apps can offer the best look-and-feel and user experience to users with smooth and fluid interface. They can also provide rich user experience with a rich set of device-specific features like Camera, Microphone, Accelerometer, GPS, Bluetooth, NFC, Fingerprint Scanner, and so on. Moreover, native apps can be published in app stores and can be discovered by users.

Despite its superior features, on the other hand, software vendors targeting multiple platforms with their apps have to have separate code bases and separate development teams for each platform they target for the same apps if they are to follow the native app development approach. This approach can become very costly depending on the complexity of the apps developed. As a result, several cross-platform solutions have emerged over the years, each with unique features, and most mobile software vendors have been considering these cost-effective alternatives instead of sticking to native app development unless the developed apps strictly require platform-/device-specific capabilities.

2.3. Hybrid App Development

In hybrid development approaches, generally, a web app is wrapped within a native container app that can be installed and run just like any other native app of the respective platform (Smartface, 2014). The development approach and the end products are very similar to web pages/web apps. The web app runs inside a native user interface layer known as WebView (Ramanujam & Natili, 2015). Therefore, hybrid apps usually struggle to establish the smooth and slick usability of native apps. There can also be platform-specific problems like page transitions that do not work smoothly on Android due to lacking CSS/CSS3 implementation. In addition, browsers on different platforms do not uniformly support all the latest HTML features and APIs, which can make development and testing a challenge (Smartface, 2013).

The only advantage of using a hybrid app over a mobile web app is the ability to access to most commonly used device features such as Camera or Contacts, which is almost impossible via a mobile web app. However, access to advanced device features or interaction with other apps requires native development, which eliminates the advantage of hybrid platforms.

2.4. Cross-Platform Native App Development

Cross-platform native app development is a totally different approach than the hybrid one. In this approach, a true native app is generated from the same single codebase for each target platform supported. UI controls are truly native and not visually emulated through CSS. Therefore, cross-platform native apps usually provide better performance and user experience than hybrid apps (Bahrenburg, 2013). While development tools for this type of apps are usually “write once, run anywhere”, some tools allow platform-specific tuning to handle different UI requirements with a “write once, adapt everywhere" philosophy. In general, the user interface code and the code that deals with the device-specific features tend to be written for each platform, while the code for other tasks can potentially be reused such as service client logic, client-side validation, data caching, and client-side data storage, saving a significant amount of time and effort (Reynolds, 2014).

3. CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT TOOLS

In this study, we mainly consider four popular and well-known cross-platform mobile app develeopment tools/frameworks in Turkey as well as in the world. First one is PhoneGap/Cordova, which uses a hybrid approach. The others are Xamarin, Appcelerator Titanium, and Smartface App Studio, all of which are in the category of cross-platform native app development approach.

3.1. PhoneGap/Cordova

PhoneGap, also known as Apache Cordova, is an open source hybrid mobile app development framework. It was originally developed by Nitobi and the company was acquired by Adobe in 2011. After Nitobi was acquired, Adobe donated the PhoneGap code base to the Apache Software Foundation (ASF) under the project name Cordova (Ramanujam & Natili, 2015).

PhoneGap uses standard and well-known web technologies such as HTML, CSS, and JavaScript to create cross-platform mobile apps without using native development languages. It packages app code into an executable program that can run on an array of mobile devices. Developers can write code once and deploy their app across multiple mobile operating systems such as iOS, Android, Windows Phone 8, BlackBerry, and Amazon FireOS. PhoneGap provides a JavaScript programming interface that allows developers to access platform-specific features with plain JavaScript (Fernandez & Alber, 2015; Ramanujam & Natili, 2015). A PhoneGap application should not just be a wrapped static web site that does not take advantage of its platform capabilities via APIs. In fact, for example, Apple is known to refuse apps to the App Store if it feels the app is solely a bundled web site that provides no entertainment value.

3.2. Xamarin

Xamarin is a development platform that allows developers to code native, cross-platform iOS, Android, and Windows Phone apps in C# language, which is the flagship and the most popular programming language for the .NET platform of Microsoft (Hermes, 2015). The Xamarin platform has been descended from the open source Mono Project that has brought .NET to Linux, and it is now a port of .NET to iOS and Android operating systems with support for Windows Phone. Xamarin uses C# bindings, called Xamarin.Android and Xamarin.iOS, to the native Android and iOS APIs for development on mobile and tablet devices. Xamarin.Android is actually Mono for Android, and Xamarin.iOS is actually MonoTouch.

Xamarin provides development environments and designers to build mobile apps on Windows or Mac operating systems. The two common choices for Xamarin development environments are Xamarin Studio on Mac or Windows, or Visual Studio on Windows with the Xamarin for Windows plug-in. A Mac is always required for compiling iOS apps, even if Visual Studio is used as the development environment.

Until recently, Xamarin developers were required to code separately for each platform to achieve native UI look-and-feel. Cross-platform code is usually referred to as shared code, or core code, as it is shared among projects for different mobile platforms. For standard Xamarin projects, this is still the case, but there is a new offering called Xamarin.Forms, which allows single code based apps for a specified set of UI controls. It produces high quality native output but the components are currently limited (Hermes, 2015; Smartface, 2015a).

3.3. Appcelerator Titanium

Appcelerator Titanium is a platform for building cross-platform native mobile apps using well-known web technologies, such as JavaScript, CSS, and HTML. Titanium is an open source project developed by Appcelerator Inc and licensed under the Apache Public License (Bahrenburg, 2013). Titanium has the architectural goal of providing a cross-platform JavaScript runtime and API for mobile development, which differs from the hybrid frameworks. Besides Android and iOS, Titanium can produce native output for Windows Phone 8. Titanium is not a “write once, run anywhere” application framework, but more of a “write once, modify for each platform” one, especially for the user interface of the apps developed. Appcelerator reports 60-90% code reuse across device platforms in general (Appcelerator, 2015).

Titanium uses a JavaScript interpreter to create a bridge between the app's JavaScript code and the underlying native platform. This approach allows Titanium to expose a large number of APIs, and native UI components without sacrificing performance.

In development with Titanium, the app code is first compiled by the Appcelerator compiler and then with the native platform compilers to produce native output. Appcelerator provides an IDE called Appcelerator Studio, which provides the tools needed to configure the development environment, write, test, debug, package, and eventually publish the apps to the app stores. Appcelerator Studio is based on Eclipse, a well-known open source coding editor (Alcocer, 2015). This IDE, however, does not provide any WYSIWYG (What You See Is What You Get) design editor for user interface design.

3.4. Smartface App Studio

Smartface is a relatively new, native cross-platform JavaScript interpreter framework for iOS and Android. Headquarters being located in the US, research and development center of Smartface is located in Istanbul, Turkey. It has a growing community in Turkey with apps developed for many sectors like banking, e-commerce, telecom, and so on (Smartface, 2015b).

Smartface is an interpreter framework. The objects developed in design time are optimized and converted into native components during runtime. As an interpreter framework, Smartface processes the code during runtime; therefore, it is not necessary to include all JavaScript codes in the project. Even the full native app can be downloaded from a server, which is something that cannot be done with the platform-based native frameworks. Furthermore, Smartface core is developed in C++ for iOS and Android, and the architecture is designed to minimize the performance loss. Smartface reports that there might be a performance loss up to 3% (Smartface, 2015b). Smartface claims to be a “write once, run anywhere” framework, however, it is also possible to add platform-specific customizations with simple conditional statements in JavaScript code.

Smartface framework consists of two main elements: Run-Time Engine and Desktop IDE. The IDE is a Windows-based one and has no dependency on Mac for iOS testing and debugging, meaning that whole iOS development can be done on Windows alone. Smartface is also known to be the only cross-platform native framework with a WYSIWYG design editor for user interface design. One of the unique features of Smartface framework is that it is possible to run apps on real devices instantly with so called on-device emulators for iOS and Android. Any iOS device can be used for emulation, and both virtual and real devices are supported for Android. In addition, on-device debugging is possible with standard debugging facilities as well as advanced debugging features such as code injection during runtime.

4. COMPARISON CRITERIA

Mobile app vendors need to develop high quality apps with least development and maintenance effort and cost in minimal time, to publish their apps to app stores, and to reach as many users as possible. In order to fulfill this requirement, a cross-platform mobile app development tool should support as many as popular target platforms, with single code base for all of them. Platform-specific development should not be needed much in general, but it should be possible when needed. In addition, familiarity of the development team with the development environment and the programming language is always a big advantage. In addition to programming language familiarity, possibility of reuse of existing code base even at least partly is a huge asset. Therefore, programming language and tool support are very important decision criteria when selecting a mobile app development tool/framework. Moreover, a Mac computer is required for developing, testing, and debugging apps for iOS platform, even though a cross-platform development tool is used, which creates an extra cost item if the general development environment is based mostly on Windows or Linux platforms. Presence of this requirement can also be an important criterion.

There are several product related factors that affect the acceptance and diffusion of a mobile app among users. Besides the functionality an app provides, there are non-functional criteria that are very important to users such as look-and-feel, responsiveness, performance, stability, security, and in general, total user experience of the app. In addition, modern mobile devices are equipped with features and sensors that an app can offer added value with such as Camera, Accelerometer, GPS, NFC, and so on. One criterion that can affect the decision can be the accessibility of device features and sensors from code.

Software developers often encounter problems and difficulties during app development and they usually search the Web for quick solutions and code snippets. When a development tool or language has a large developer community all over the world, it becomes much easier to find a solution because it is highly probable that someone else has encountered the same problem before, and a solution has been shared already. Therefore, community size and the support that can be obtained from either the community or the tool vendor are considerable decision criteria.

5. COMPARISON OF TOOLS

In this section, we provide a pragmatic comparison among the cross-platform mobile app development tools/frameworks mentioned in Section 3 with respect to the criteria given in Section 4. A side by side comparison among the four development tools is given in Table 5.1. The table not only presents comparison data related to these tools but also allows the comparison of platform based native development with them as well.

When we consider the diversity of mobile platforms the tools support, PhoneGap/Cordova has a clear advantage over the others because its hybrid approach makes it possible to wrap the same web app into a native container that can be delivered to platforms other than iOS and Android. Mobile operating system market is obviously dominated by iOS and Android, thus, advantage of supporting other platforms might be limited to mobile app vendors. With respect to mobile platform support, both Xamarin and Appcelerator Titanium can produce native output for Windows Phone whereas Smartface is focused only on iOS and Android, and Smartface has no plans for Windows Phone development.

According to Table 5.1., all tools support single code base development with some exceptions. For example, Xamarin has a new approach under the name of Xamarin.Forms but it is currently provided at extra pricing. Titanium and Smartface provide a similar approach to single code base, however, Smartface unifies platform-specific components as much as possible, but Titanium opts for some level of separation, therefore, there may be a bit more platform-specific coding in Titanium than Smartface.

Table 5.1. Comparison of Mobile App Development Tools/Frameworks

	Platform Based Native	**PhoneGap/Cordova**	**Xamarin**	**Appcelerator Titanium**	**Smartface App Studio**
Output	Native	Hybrid	Native	Native	Native
Supported Platform	iOS, Android	Cross-platform	Cross-platform: iOS, Android, Windows Phone	Cross-platform: iOS, Android, Windows Phone	Cross-platform: iOS, Android
Single Code Base	No	Yes	Only for Xamarin.Forms Projects (Not free)	Yes (Platform-specific code required)	Yes
Development Environment	Xcode, Eclipse or Android Studio	Jetbrain Webstorm, Sublime Text, Intel XDK, Eclipse	Xamarin Studio or Visual Studio	Titanium Studio	Smartface IDE
Development Language	Objective-C, Swift, Java	HTML5/JavaScript/CSS	C#	JavaScript	JavaScript
WYSIWYG Design Editor	iOS: Yes, Android: Partially	Depends on tech	Partially with Designers	No	Yes
Code Based Design Editor	Yes	Depends on tech	Yes	Yes	Yes
iOS Development on Windows	No	No	No (Workaround available with a Mac on network)	No	Yes
Mac Required for iOS Test and Debug	Yes	Yes	Yes	Yes	No
Developer Adaptation	Mobile know-how	Web know-how	C# know-how	Web know-how	Web know-how
Look and Feel, Sense, UX	Native	Native-like	Native	Native	Native
UI Responsiveness	Smooth	Not smooth	Smooth	Smooth	Smooth
Performance	Fastest	Fast Enough	Faster	Faster	Faster
Device Specific Features	Yes	Depends on tech	Yes	Yes	Yes
Device Sensors	Yes	Needs plug-in	Yes	Yes	Yes
Offline Storage	Device storage	Limited device storage and shared web storage	Device storage	Device storage	Device storage
Stability	Highest	Higher	Higher	High	Higher
Security	Most secure	Depends on tech	Secure	Secure	Most secure
Community	Largest	Large	Large	Large	Medium
Technical Support	Community	Community & Inexpensive license packages sufficient for direct support	Community & Inexpensive license packages sufficient for direct support	Community & Expensive enterprise package required for direct support	Community & Inexpensive license packages sufficient for direct support
Risk	Learning Cost, Operational Cost	Browser Compatibility Issues	Dependency	Dependency, Expensive licenses after free version	Dependency

Development environment and tools are very critical for developer productivity and end product quality. For software vendors with experience and expertise in web development using HTML, CSS, and JavaScript, PhoneGap/Cordova can be an easy transition to mobile platforms with less effort and less adaptation problems. Xamarin can be a very good alternative for vendors with expertise in Microsoft .NET environment and C# programming language as it allows developers to use Visual Studio for development as well as its Xamarin Studio IDE. One advantage of C# projects over JavaScript projects is that due to the structure of the languages, code checking before build and AutoComplete feature works more accurately along with XAML support. Titanium and Smartface are similar in that they both use JavaScript as the development language, which is one of the most popular programming languages today. Therefore, developers with web development knowledge can adapt easily to both environments. Among all tools, Smartface is the only one that allows full iOS development on Windows, and only Smartface provides on-device running, testing, and debugging capabilities for both iOS and Android.

Non-functional qualities such as look-and-feel, responsiveness, performance, stability, security, and total user experience of an app are very important to its users. Unfortunately, apps developed with PhoneGap/Cordova cannot offer native user experience because the user interface responsiveness is not smooth enough and the look-and-feel is usually far from native. On the other hand, all other three tools/frameworks provide native user experience with smooth and fast user interface as they all produce real native products. Furthermore, PhoneGap/Cordova has limited or no support for advanced device-specific features and interaction with other apps. Unlike PhoneGap/Cordova, Xamarin, Titanium, and Smarface all can provide these facilities natively.

With regard to community size, Smartface falls behind the others as Smartface is a rather new mobile development platform. Relatively larger developer communities of other tools make them justifiably appealing because the more tried and tested a development environment, the less the number of problems you encounter and the more the solutions you can find online via the community forums, blogs, and other social platforms. However, Smartface has been gaining popularity especially in Turkey.

6. CONCLUSION

In this paper, we provide a detailed comparison of popular cross-platform mobile development tools, namely, PhoneGap/Cordova, Xamarin, Appcelerator Titanium, and Smartface App Studio. All of these tools help app vendors develop apps that can run on different mobile platforms, with less effort and cost in less time, thanks to their "write once, run anywhere" or similar philosophy. Despite their common objective, they all have distinctive features that make each one superior and preferable to the others. Therefore, software vendors need to understand the advantages and disadvantages of them, evaluate each one considering their own specific development requirements and constraints, and then make their choices wisely.

PhoneGap/Cordova, which has a hybrid approach, looks very appealing because it supports many mobile platforms with very little effort. However, apps developed with it are not fully native, and as a result, especially user experience might cause user dissatisfaction due to non-smooth and slow user interface. We know that users compare the apps with the ones like Facebook and Twitter, and they expect the same user experience from all apps. We also know from our experience with PhoneGap/Cordova and from our contacts with the industry that apps developed with PhoneGap/Cordova fall below a certain standard in terms of smoothness and performance even on very powerful modern smart phones and tablets.

Xamarin is a very good alternative for software vendors with Microsoft .NET and C# background as they can exploit their platform and language expertise for mobile app development. However, unless the new Xamarin.Forms feature is used, developers still need to code separately for each platform. Besides, additional Xamarin licenses must be purchased separately for each native platform, which can be restrictive in terms of cost.

Appcelerator Titanium is a very popular cross-platform mobile app development tool, which uses JavaScript as the underlying coding language. In many ways, Smartface and Titanium are similar. They are both JavaScript based and they both produce native output for iOS and Android. Titanium can also produce Windows Phone 8 output. Both platforms have their own IDEs. However, Titanium does not support iOS development on Windows nor iOS emulator or debugger. If you want to develop iOS apps on Titanium, there is no other alternative than using a Mac. Again, Smartface is the only cross-platform native framework that supports whole iOS development cycle on Windows. The most notable advantage of Titanium over Smartface seems to be the size of the community as Titanium has a larger developer community. There are also license pricing differences between Titanium and Smartface, which can be an important factor for especially getting support from the tool vendors, either by e-mail or by direct contact.

It is not an easy task to choose a mobile app development tool because there are a lot of alternatives with a lot of different capabilities and offerings. Therefore, software vendors need to evaluate each one carefully, and we believe that it is best to try the ones that seem reasonable on small pilot projects to see if they are satisfactory in terms of the criteria we described.

REFERENCES

Alcocer, R. (2015). *Build Native Cross-Platform Apps with Appcelerator: A Beginner's Guide for Web Developers*: J.B. Orion.

Appcelerator. (2015). Mobile App Development Platform – Appcelerator Retrieved August, 18, 2015, from http://www.appcelerator.com/

Bahrenburg, B. (2013). *Appcelerator Titanium Business Application Development Cookbook*. Birmingham, UK: Packt Publishing.

Fernandez, W., & Alber, S. (2015). *Beginning App Development with Parse and PhoneGap* New York, NY: Apress.

Hermes, D. (2015). *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals*. New York, NY: Apress.

LePage, P. (2014). Responsive Web Design Basics - Web Fundamentals Retrieved August, 18, 2015, from https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/

Ramanujam, P., & Natili, G. (2015). *PhoneGap Beginner's Guide* (3rd ed.). Birmingham, UK: Packt Publishing.

Reynolds, M. (2014). *Xamarin Mobile Application Development for Android*. Birmingham, UK: Packt Publishing.

Smartface. (2013). Native vs. Hybrid Retrieved August, 18, 2015, from http://www.smartface.io/native-vs-hybrid/

Smartface. (2014). Native Smartface vs. PhoneGap/Sencha/Cordova Hybrids Retrieved August, 18, 2015, from http://www.smartface.io/smartface-app-studio-vs-phonegapcordova-cross-platform-native-vs-hybrid-environments/

Smartface. (2015a). Smartface vs. Xamarin (Cross-Platform Native Frameworks) Retrieved August, 18, 2015, from http://www.smartface.io/smartface-app-studio-vs-xamarin-c-sharp-javascript-based-cross-platform-native-frameworks/

Smartface. (2015b). Smartface: Develop Cross-platform Native iOS & Android Apps Retrieved August, 18, 2015, from http://www.smartface.io/

Statistica. (2015). Number of apps available in leading app stores 2015, Statistica Retrieved August, 18, 2015, from http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/