Native versus HTML5 – where's mobile programming heading to?

PATRICK MENNIG
University of Applied Sciences Wuerzburg-Schweinfurt

The market for mobile applications has grown from a few pre-installed applications on mobile phones to giant stores for different smartphone platforms. Although for smartphones, Android has a much stronger market share than iOS or Windows, a developer can hardly refrain from offering its applications for all operating systems. For him this represents a significant overhead, since the different platforms are addressed with different programming languages. This means concretely that multiple programming effort, multiple tests and calls for a broad expertise of Java, Objective-C and .NET are required. Web technologies known as HTML5, allow to develop applications that are running on all mobile operating systems, without having to be repeatedly developed. Thanks to modern frameworks web technologies are on a level with native development and the approach needs to be carefully considered. This paper investigates the benefits and risks of different development approaches with the help of certain criteria.

Categories and Subject Descriptors: Mobile and Ubiquitous Computing

Additional Key Words and Phrases: Mobile Applications, Apps, Web Applications, Cross-Platform Application Framework, PhoneGap, native App

1 INTRODUCTION

Mobile applications are part of many areas of daily life. The intensive usage of smartphones and tablet computers increasingly permeates the private and business life. Apps advanced from the beginning, which is marked by the first smartphone with a multitouch screen – the Apple iPhone. These small programs are loaded on the smartphone and tablet to enable a variety of actions a user may perform. The typical mindset for any end-user is as induced by the advertising slogan by Apple. It can be transferred as: "If you want to do something, there's an app for it". It remains an open question where these apps come from. The techniques for app development and their differences shall be illuminated further, comparing the different development approaches based on a certain set of criteria.

2 THE MOBILE APPLICATION MARKET

Mobile applications are - as introduced - an important aspect of mobility. Before any smartphones, applications have already been installed on mobile phones. One prominent example is the game "Snake" which was to be found on the phones from Nokia [Alby, T. 2008]. Other than today, the user was dependent on preinstalled applications. It was only in few circumstances, or with huge efforts possible, to install single applications manually. Furthermore, the operating concept of mobile phones was different from today's smartphones and tablet computers. Users controlled early mobile devices by solid mounted keys or touchscreens reacting

Author's address: Patrick Mennig, University of Applied Sciences Wuerzburg-Schweinfurt, Würzburg, Germany.
www.fhws.de.

to pressure (resistive touch technology). The introduction of the first iPhone by Apple in 2007 changed the situation fundamentally. One aspect has been the preinstalled and manifold applications, which resembled today's apps and covered important activities. Additionally it was possible to control this smartphone with the fingers. The touchscreen is reacting to magnetic fields of human skin. The interaction happened with clearly less pressure and was easier to perform for the users. More than one point of interaction (i.e. finger) can be detected which allows so called multitouch gestures.

Advertised was this behavior as the connection of an iPod (a portable music player with a hard drive), a mobile phone and an internet communicator. The slogan describes the impact of the first "real smartphone" insufficient. The concept of small applications within a comprehensive operating system fascinates both, users and developers. The latter have been excluded in the beginning. A commonly available development environment (IDE) for iPhone applications was not available. In August of 2008, Apple published version 3.1 of its IDE Xcode, including the iPhone SDK. Paired with the app store, introduced in early 2008, the baseline for today's ecosystem of small apps from different app stores was built.

In October of 2008 the first smartphone with the operating system "Android" hit the market. This OS is developed and distributed by the Open Handset Alliance under leadership of Google Inc. Contrary to Apple's iOS, it features a visibly more open architecture. Important to highlight is that different vendors of mobile phones can sell their devices with adjusted versions of Android. Contrary, the operating system iOS can only be found on devices sold by Apple. Since October of 2010, a third important actor in the smartphone market is Windows with its OS "Windows Phone". It shows the same concept of small apps that the user may load from the store. Other vendors like BlackBerry (former ResearchInMotion) focus on less typical smartphone functionality and should be excluded from the examination.

The result of this evolution is a smartphone and tablet market that is contested by more than one vendor. Visible differences are found between phones and tablet computers. According to IDC, the market of tablet operating systems in 2012 is partitioned as follows: Apple leads with its operating system iOS with 53.8 percent market share, followed by the OS Android with 42.7 percent. With large distance follows the mobile versions of Windows, reaching 2.9 percent. It can be said that iOS and Android dominate the market. Microsoft has the potential to reach higher market shares with its new system, Windows 8.

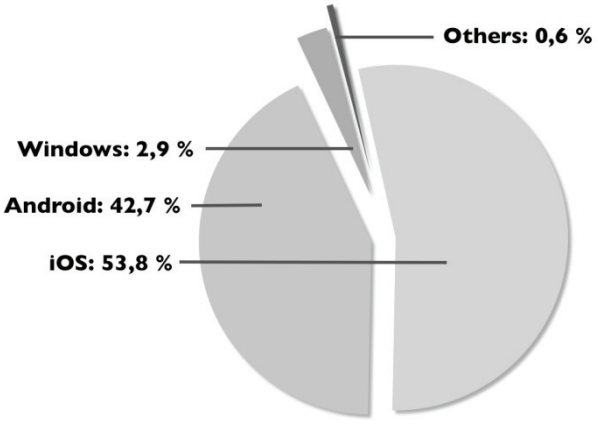

Figure 1: Market share of mobile OS – tablets.

Source: Author's representation, based on IDC, 2013a.

Figure 1 illustrates the tablet computer market shares [IDC, 2013a]. Assuming a developer is only targeting the still dominant iPads with iOS, he is only reaching half of the possible target group. The same applies for Android-only development.

The market of smartphone operating systems shows a different distribution. Android dominates clearly with 70.1 percent. Apple's iOS follows on the second rank with 21.0 percent, behind lies BlackBerryOS with 3.1 percent and Windows with 2.6 percent [IDC, 2013b]. A development for Android may reach circa three quarters of the potential market. The smartphone market is illustrated in Figure 2.

Considering the shown market shares, it follows that when developing an application, more than one target operating system should be chosen. An app that shall reach a market as large as possible cannot be limited to one platform. Rather, it is recommended to create an app that is capable of running on both, smartphones and tablet computers. In case all major operating systems are supported, the greatest part of the potential market is reached. The technological restrictions that appear when choosing a certain development approach for mobile applications, are subject to the following chapters.

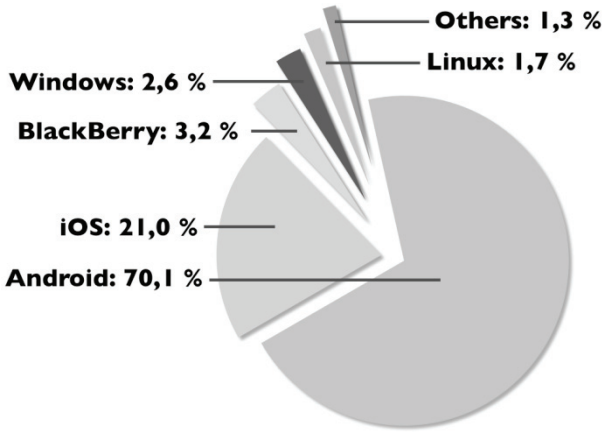

Figure 2: Market share of mobile OS – smartphones

Source: Author's representation, based on IDC, 2013b.

3 MOBILE APPLICATION DEVELOPMENT

Similar to their pendants on normal computers, mobile applications have to be developed with certain programming languages. It applies here, that these differ depending on the targeted platform. Standard iOS apps are developed with the programming language Objective-C. Android apps are to be developed with a certain version of JAVA. For Windows applications, the .NET languages are to be used (C#, C++) [Charland, A. and Leroux, B. 2011]. This includes the creation of the application logic with given tools and methods, as well as the user interface. This is done in different ways. A Mac computer with the development environment Xcode is necessary for iOS development, while Windows apps are created with Visual Studio (which is only running in Windows machines). Android applications can be developed under Mac OS or Windows, for example with the IDE Eclipse. Necessary are only the free available Android Developer, which can be included as a plug in into the development environment. An application created in such a way is called "native". It is written with platform specific code and can only be used on the specific one without adjustments. Porting it to another operating system assumes huge adaptions. Not

only the syntaxes of used languages differ from each other but the semantics, available classes, libraries and the program structure, depending on the targeted operating system.

Apart from native development, another approach called web development exists. Following it, a developer creates so-called web apps. These use the operating systems browser as a runtime environment and are runnable on all mobile operating systems without any installation process. The programming languages to create web applications are HTML, CSS and JavaScript. Those allow platform independent development.[4] The application can be opened by accessing a specific website – nothing else are web applications. A specifically developed and adjusted website mocking a native application in design and functionality [Willnecker, F. et. al. 2012]. With particular techniques, these web apps can also be used without an Internet connection, as long as the user has saved them on the device once and not cleared the browser's cache. In iOS, this is achieved via the function "add to home screen".

A third way of developing mobile applications is to combine native and web development. Connecting a container application written in platform specific code with a web app containing the application logic and the user interface leads to a program referred to as "hybrid". In the most extreme case, the native app contains only a single browser window in full screen, running the web app [Franke, F. and Ippen, J. 2012]. The corresponding element is called UIWebView in iOS and WebView in Android and Windows.

As the different development approaches for mobile applications have been shown, the question remains which of these – depending on the applied criterion – offers advantages and disadvantages. The criteria "Content vs. Experience", "Performance", "APIs", "Distribution" and "Development Cost" are to be considered. Those are only a subset of possible criteria (e.g. [Frank, AM 2011], [Charland, A und Leroux, B. 2011], [Corral, L. et. al. 2011]) and represent those with a clear difference in results.

4 ANALYSIS OF CRITERIA

Above-mentioned criteria should be considered hereafter. Interesting for each are the differences between native, hybrid and web applications. The idea behind the examination is a given firm, which is facing the decision which development approach to choose.

4.1 Content vs. Experience

The first step is to decide what one wants to focus on, when developing an application. Two extremes lie in the mere representation of content and the communication of an experience. As long as the goal is to display specific data and information, it is possible speak of a focus on content. This is especially interesting in the area of enterprise applications. From a developers perspective this focus is possible with all three approaches, i.e., native, hybrid and web. There are no particular requirements on the design options.

This is different, however, with a focus on experience. This term is very vague and hardly palpable. It can be described with the feeling that a user has, when using the application. Experience is influenced, inter alia, by the design, the haptic and responsiveness of the application. As the user manipulates the app with his own fingers, he has a very direct interaction. However, this is affected by sluggish or lack of feedback to touch, due to delays in moving content or too functionally designed user interfaces. A very good example of an app that wants to convey

[4] The browser is used as a container. Hence the platform independence is only restricted by the browsers capabilities. As long as the browsers meet the HTML specifications, independence is given. Adjustments and changes to adapt to different browsers and versions are often inevitable.

an experience is the calendar on the iPad under iOS. Although much content is presented, the frame as well as the background are lovingly designed and give the user the feeling that they have a real calendar of leather and paper in hand. Even details such as curved edges, demolition of old calendar pages in the month view or the shadows when turning pages have been minded.

Native applications have a slight advantage in terms of this criterion because they allow a direct access to system resources and therefore can react very quickly to user input. Furthermore, they are using the operating system's control elements, thus enabling a seamless experience across the entire system. The advantage in resource usage allows for more advanced and skeuomorphic user interfaces due to more power. Basically this also possible with web languages, but it requires huge additional effort in creating the user interface, the transitions between screens and feedback on user interaction. Existing frameworks such as JQueryMobile and SenchaTouch help, but can not offset all the disadvantages of web and hybrid applications. When focusing on communicating a strong experience, native development is the better approach, as it allows reaching quality results with less effort.

4.2 Performance

Depending on the use cases, which the application is based on, different requirements towards the performance of an application arise. If only a certain number of data is shown without being calculated intensively, the app has no special requests to the performance. However, once large amounts of data are displayed, emphasis must be placed on the observance of performance limitations.

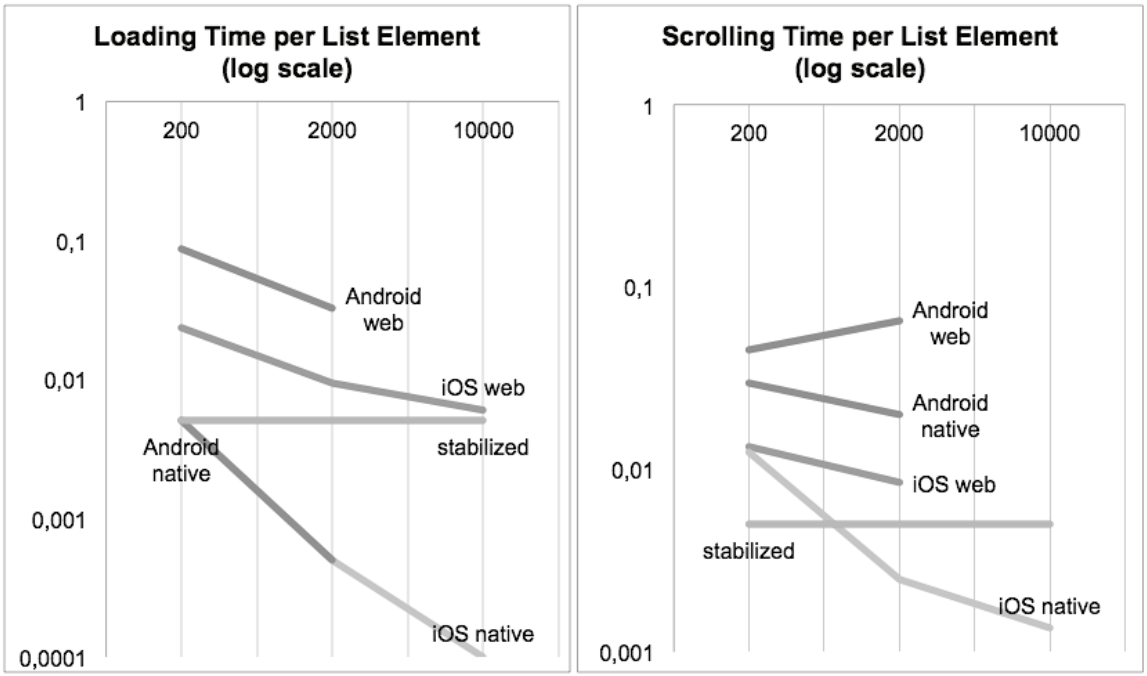

Figure 3: Loading and scrolling time in seconds per list item on mobile devices.

Source: Author's representation, based on [web & mobile developer 2013].

Performance issues have been proven in an experiment published in the journal “web & mobile developer”. It based on the assumption that users expect rapid applications that react quickly to touch input. The set-up is formed by two native applications and a web-application. These are displaying 200, 2,000 and 10,000 list items either natively on iOS and Android or in the web browser on iOS and Android. Goal of this test was to measure the time necessary to load and display the list as well as manual scrolling time from top to the bottom. It was expected that both. Loading and scrolling time will increase with more list items. The experiment came to the conclusion that as early as 2000 list items are shown, significant performance losses are felt. It shows, among other things, that the mobile version of Safari, takes 23 times the time to load the list of 2,000 items, compared to the native application. Also in terms of manual scrolling it shows that in the web application it takes three times the time needed for scrolling than in the native application (from the beginning of the list to the end). The consequence is that the app feels slower and less responsive. At 10,000 list items, a display in the web was no longer possible for most browsers [web & mobile developer 2013].

The illustration Figure 3 shows two diagrams, indicating loading and scrolling time divided by the number of list items for iOS and Android, both web and native (different browsers on one platform have been averaged). The time is shown on a logarithmic scale (in seconds to $\log_{10}$). Both diagrams show a stabilized series, assuming a linear growth of time. In case the display of data or scrolling was impossible, no value is marked for the specific series.

This shows a clear advantage for native development in terms of performance. Considering it as an important criterion for a good user experience, one should be careful in the case of hybrid or web-based apps, estimating early the amount of power necessary. In case very large numbers of records are displayed, complex calculations are carried out or complex 2D and 3D graphics are desired, it is often advisable to develop a native app. In a different application, without these requirements, a hybrid or web approach makes sense too.

4.3 APIs

Smartphones are interesting mainly because of their extensive technical capabilities. These include access to special equipment features such as the camera, the module for determining the position or acceleration sensors. This is generally addressed by so-called application programming interfaces (APIs). In principle, the access to these APIs from the native code is no problem. The functions can be used in accordance with the provided hardware of the device.[5]

Web Code, however suffers from limitations in terms of access to device-specific APIs. The specifications for access to them are not yet available for all functions and therefore not implemented. The module to determine the position can be accessed from the web browser, as well as the camera and the photo browser [Firtman, M. 2013]. One possible way to circumvent the limitations is building hybrid applications that can access the native features with certain technical ways from the web container. A framework that supports the developer with them is Apache Cordova (formerly PhoneGap). It provides a JavaScript library with allows addressing the various operating systems uniformly [Apache Cordova 2013]. The following Table 1 shows a sample of mobile browsers capabilities in terms of API access.

[5] For Android devices, the hardware is very different depending on manufacturer and model. It is therefore possible that, although the same operating system is installed, not all APIs are available on all devices. This restriction does not apply to iOS yet, as iPhones and iPads have homogeneous functions.

Platform	**Browser**	**Safari on iOS**	**Android Browser**	**Internet Explorer**	
	Platform	**iPhone, iPad**	**Phones & Tablet**	**Windows Phone 7.5**	**Windows 8**
	Versions tested	**3.2 to 6.1**	**1.5 to 4.2**	**9**	**10**
Feature	**Geolocation**	yes	yes (2.0+)	yes	yes
	Motion Sensors	yes (4.2+)	yes (3.0+)	no	no
	Touchevents	yes	yes (2.1+)	no	yes
	File API	yes (6.0+)	yes (3.0+)	no	yes
	Media Capture	partial (6.0+)	yes (3.0+)	no	no
	Notifications API	no	no	no	no
	Network Information API	no	yes (2.2+)	no	no

Table 1: API coverage of mobile browsers.

Source: Author's representation, based on [Firtman, M. 2013].

4.4 Distribution

Not only the development of an application in itself should be considered. Also, one should keep in focus that an app has to come to the customer. In the case of enterprise apps, it is easier due to separate stores. But if a developer goes for a "typical" app for the end-user, he needs ways and means to expel the product. The manufacturers of the various mobile operating systems provide this in form of their own app stores. On these central platforms, developers can publish and sell their applications without having to worry about hosting and billing. In return, the suppliers keep a given percentage (usually 30%) of the revenue generated [iOS Developer Program 2013] [Android Developer 2012] [Windows Dev Center 2013].

To mention is that only native and hybrid applications can be set in the stores. A web application is running as pure web site and offers no standard way to distribute them commercially. Although some stores for web apps are created already, they are still little frequented. A separate billing system can certainly be created, but this means extra effort in development.

4.5 Development Cost

In addition to the criteria already considered the cost plays a strong role in the development. It is important to keep in mind that native development only targets a single platform. This means that the same application has to be developed several times in case it should be available on multiple platforms (for example iOS and Android). It follows that this also multiplies development costs. In contrast, web development allows that the app, once created, can be used on different operating systems. Hybrid development is a middle way. Since a large part of the application logic is platform-independent, only slightly higher costs for the preparation of native container arise.

The costs of development for the various operating systems are far apart. An examination of the estimates of sample applications for various platforms shows that creating a native iOS application can be almost twice as expensive as developing a web application with similar functionality. The exact figures are not important at this point, however, the trend is interesting. It

clearly shows that native development is more expensive [Frank, AM 2011]. Figure **4** shows this relationship again.

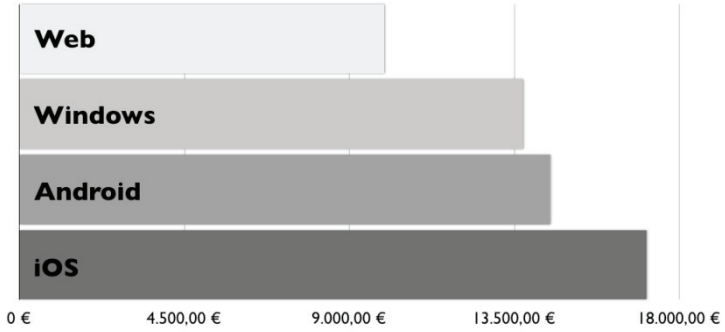

Figure 4: Development cost of a catalogue app.

Source: Author's representation, based on Frank, A. M. 2011.

Furthermore, the majority of developers is developing for iOS and Android (80 percent). These two platforms form the baseline, as the expected revenue on them is the highest. When asked, which system they prefer in terms of development cost, 20 percent of developers answered iOS, compared to 32 percent answering Android. On the other hand, they score iOS the better platform in terms of app discovery and revenue potential (50 & 66 percent iOS vs. 23 & 12 percent Android). Developers, that do not develop for one of the two platforms, earn, on average, only half the revenue of those creating apps for at least one of both. HTML as a platform for app development and deployment is used by 50 percent of developers. 75 percent of all develop for up to three mobile platforms [Developer Economics 2013]. As a conclusion, the sole development cost for web based applications are lower than for Android or iOS but the potential revenue expected is highest on these two native platforms.

5 CONCLUSION

After considering the pros and cons of native, hybrid and web development using various criteria, it's not possible to draw a definite conclusion. On the one hand, native applications offer advantages over web and hybrid applications in factors such as "Content vs. Experience", "Distribution", "APIs" and "Performance". On the other hand, the obvious cost advantage of HTML5 Apps is observed. The set of criteria evaluated in this paper however, does not claim completeness. Certain company-specific conditions can affect the selection. Especially "bring your own device" (BYOD) as a strategy has to solve the problem of integrating various mobile operating systems to support the majority of devices that are used by the employees. The comparison of web and native applications is subject to other studies too, for example [Charland, A und Leroux, B. 2011] or [Corral, L. et. al. 2011]. An outstanding example of web application performance can be found in the app "fastbook" (mimicking the Facebook-app functionality), created by developers from "Sencha" to prove their framework's performance. The web app's performance and user experience exceeds the native app by far [Sencha Fastbook 2012].[6]

[6] The reservation must be made, however, that the native Facebook-app's performance has increased significantly since fastbook was first developed.

In the future, the support of device-specific APIs and the performance of web languages will increase significantly. Especially the emerging HTML5 standard offers many opportunities here. Therefore, the pros and cons, shown in this paper only present a snapshot of the current situation. Nevertheless, currently, it is still imperative to perform a thorough examination of criteria before deciding for a development approach. A pure cost consideration should never be the only basis for a decision.

REFERENCES

ALBY, T. 2008. Das mobile Web. Hamburg, 2008.

ANDROID DEVELOPER 2012. Android Developer Help – Transaction Fees. https://support.google.com/googleplay/android-developer/answer/112622?hl=en. Accessed on 01.06.2013.

APACHE CORDOVA 2013. Apache Cordova is a platform for building native mobile applications using HTML, CSS and JavaScript. http://cordova.apache.org. Accessed on 28.03.2013.

CHARLAND, A UND LEROUX, B. 2011. Mobile Application Development: Web vs. Native. acmqueue Volume 9 Issue 4. New York. 2011.

CORRAL, L. ET. AL. 2011. Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm. Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software. New York. 2011. 181-183.

DEVELOPER ECONOMICS 2013. Developer Economics 2013: The tools report. http://www.visionmobile.com/product/developer-economics-2013-the-tools-report/?image3=1&utm_expid=1534519-18. Accessed on 02.06.2013.

FIRTMAN, M. 2013. Trying to understand HTML5 compatibility on mobile and tablet browsers. http://mobilehtml5.org. Accessed on 28.03.2013.

FRANK, A. M. 2011. Was kostet die Entwicklung von Apps? Ellwangen. 2011.

FRANKE, F./ UND IPPEN, J. 2012. Apps mit HTML5 und CSS3 für iPad, iPhone und Android. Bonn. 2012.

IDC 2013a. IDC Raises Tablet Forecast for 2012 and Beyond As iOS Picks Up Steam, Android Gains Traction, and Windows Finally Enters the Market. http://www.idc.com/getdoc.jsp?containerId=prUS23833612#.UV77q78pG3h. Accessed on 28.03.2013.

IDC 2013b. Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year. http://www.idc.com/getdoc.jsp?containerId=prUS23946013#.UV77J78pG3h. Accessed on 28.03.2013.

iOS DEVELOPER PROGRAM 2013. iOS Developer Program – 3. Distribute. https://developer.apple.com/programs/ios/distribute.html. Accessed on 01.06.2013.

SENCHA FASTBOOK 2012. The Making of Fastbook: An HTML5 Love Story. http://www.sencha.com/blog/the-making-of-fastbook-an-html5-love-story. Accessed on 07.06.2013.

WEB & MOBILE DEVELOPER 2013. Web-Apps oder native Apps. Issue 2/2013.

WILLNECKER, F., ISMAILOVIC, D. UND MAISON, W. 2012. Architekturen Mobiler Multiplattform-Apps. Verclas, S. und Linnhoff-Popie, C. (ed.): Smart Mobile Apps - Mit Business Apps ins Zeitalter mobiler Geschäftsprozesse. Heidelberg, London, New York et. al. 2012. 403-418.

WINDOWS DEV CENTER 2013. Windows Dev Center – Windows Store Apps – App Developper Agreement. http://msdn.microsoft.com/en-us/library/windows/apps/hh694058.aspx. Accessed on 01.06.2013.