www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 4 April, 2013 Page No. 1134-1141

Mobile Application Development Strategies

Ms. Warhekar Snehal P Prof. Gaikwad V. T.*
Sipna C.O.E.T, Amravati , Dept. of IT,
Snehalwarhekar7@gmail.com

Abstract
Mobile application development has entered new stage largely driven by advent of highly influential iOS, android etc. bases smartphones and tablet devices. In sales, smart mobile devices are outpacing conventional computer clients. While mobile app development is still primarily province of customer applications, time has come that enterprise development teams need to prepare their applications to run on intelligent mobile devices. There are few major categories of mobile app delivery available today. These are: Native type, running directly on device, web based types, employing device's web browser; a hybrid of native and web based types. Each approach is suitable to work in different environments and also each has some pros and cons in comparison with others. There are many factors that play a part in deciding mobile development strategy, such as development skills, required device functionality, the importance of security, offline capability, interoperability, multiplatform support, deployment method etc. that must be taken into account.

I. Introduction

The universe of mobile devices comprises the entire range of phones, from low-end feature phones to high-end smartphones and tablets. With the growing rate of adoption of smartphones and tablets across the world, there is a large demand for mobile applications designed for popular mobile devices running operating systems like iOS, Android, Windows Phone/8, and Blackberry operating systems. Moreover these apps need to run on devices with large touch screens, broadband (Wifi, 3G) connectivity, camera and other sensors. OS (operating system) providers are increasing their capabilities to enable developers to be more efficient and functional. Mobile applications are everywhere in categories of games, social networking, productivity tools, infotainment, data management, utilities, and etc. Mobile content, operator and media services are being delivered in the form of mobile applications due to user experience superiority over classical mobile channels like mobile browsers and SMS. Mobile applications are the next step of internet paradigm.

Here we focus on the architecture of mobile applications designed for these smartphones and tablets. Currently there are three types of mobile development approaches available [1]:

- **Native apps** are specific to a given mobile platform (iOS or Android) using the development tools and language that the respective platform supports (e.g., Xcode and Objective-C with iOS, Eclipse and Java with Android). Native apps look and perform the best.
- **HTML5 apps** use standard web technologies—typically HTML5, JavaScript and CSS. This write-once-run-anywhere approach to mobile development creates cross-platform mobile applications that work on multiple devices.
- **Hybrid apps** make it possible to embed HTML5 apps inside a thin native container, combining the best (and worst) elements of native and HTML5 apps.

In the early days of mobile applications development for smartphones, there were only two dominant client architectures, the web app architecture relying on the phone web browser and the native app which involved custom development for the device OS. It was only in

2009 that a third option started to emerge. This was the Hybrid App approach, which attempted to plug the main limitation of the Web App approach: the lack of access to device features such as GPS and Bluetooth. All mobile applications today use one of these three mobile architectures.

II. Native Mobile Applications

Native applications have binary executable files that are downloaded directly to the device and stored locally. To create a native app, developers must write the source code and create additional resources such images, audio segments and various OS-specific declaration files. Using tools provided by the OS vendor, the source code is compiled in order to create an executable in binary form that can be packaged along with the rest of the resources and made ready for distribution.

Native apps are usually developed using an integrated development environment (IDE). IDEs provide tools for building debugging, project management, version control, and other tools professional developers need. While iOS and Android apps are developed using different IDEs and languages, these tools, as well as other utilities and files, are normally called the SDK of the mobile OS.. The following table [2] presents the different tools, languages, formats and distribution channels associated with the leading mobile operating systems.

	iOS	**Android**	**BlackBerry**	**Windows Mobile**
Language	Objective C,C,C++	Java (Some C,C++)	Java	C#,VB. NET etc.
Tools	Xcode	Android SDK	BB Java Eclipse Plug-In	Visual Studio
Packaging Format	.app	.apk	.cod	.xap
Application Stores	Apple App Store	Google Play Store	BlackBerry AppWorld	Windows Phone Market

Table1. OS and development tools

Once the app has been installed on the device, the user launches it like any other service the device offers. Upon initialization, the native app interfaces directly with the mobile operating system, without any intermediary or container. The native app is free to access all of the APIs that are made available by the OS vendor. It is through API calls that the app can interact directly with the touch screen or keyboard, render graphics, connect to networks, process audio received from the microphone, play sounds through the speaker or headphones, or receive images and videos from the camera. In addition to providing the low-level hardware-access services we just mentioned, mobile operating systems also provide higher-level services that are important to the personal mobile experience. Such services include processes like browsing the web, managing the calendar, contacts, photo album, and of course the ability to make phone calls or send and receive text messages.

Another important set of APIs that the OS provides is the GUI toolkit [3]. Each mobile OS comes with its own set of user interface components such as buttons, input fields, sliders, menus, tab bars, dialog boxes, and more. Apps that make use of these components inherit the look and feel of that specific mobile OS which normally results in a very smooth user experience.

Some of the examples of native apps are Angry Bird – a popular game , Instagram [4] – a photo editing and sharing tool , Shazam, Amazon's Kindle reader etc.

2.1 Advantages of Native Applications:

- **Device Integration** - Native development takes full advantage of mobile device capabilities such as the camera, barometer, gyroscope, accelerometer, and network communications.
- **Offline and Synchronization Capability [5]** - Native development allows access to local device storage for offline storage capability, and provides developer's greater flexibility in developing customized database/storage synchronization.

- **Push Notification Capability [6]-** Push notification reduces network usage, saves bandwidth and consumes less battery power than the traditional method of background process and continuous polling. Each mobile platform vendor offers a unique push notification service that can only be integrated and employed when developing a solution natively.
- **Application Market Integration** - Mobile application market integration is a necessary element in any mobile strategy. It provides for the distribution and monetization of the mobile application. By developing natively, a developer can just submit the binary distribution file to the application market, and participate in the market ecosystem.
- **Improved User Experience** - With native development, developers are able to take advantage of the hardware acceleration feature on the mobile device. There is also less of a layer between the code and its kernel. As a result, native mobile applications are the fastest in terms of load times and execution speed. These factors improve the users' experience of the mobile application tremendously.
- **Multi touch support [7]** - double taps, pinch-spread, and other compound UI gestures are supported in native applications.
- **Ease of use** - The native platform is what people are accustomed to, and so when you add that familiarity with all of the native features they expect, you have an app that’s just plain easier to use.

2.2 Challenges for Native Mobile Applications

Along with various advantages native apps also comes with various challenges. Some of these are as follows:

- **Distribution:** There may be stringent requirements for admission into public app stores. Apple Inc., for instance, requires that developers submit iPhone mobile digital device applications for testing within Apple to facilitate such compatibility
- **Deep Platform Knowledge** - Native development involves understanding the platform operating system, and learning new programming languages developers may not be familiar with, such as Objective-C for Apple mobile applications, C# for Windows Phone, and Java for Android.
- **Limited Portability** - Also with native development, existing code developed for one mobile platform may not necessarily be easily ported to another platform, limiting common features between device versions.
- **Update :**As native app is installed directly on users device, when any change or update is to be made in application ,then user have to install the app again on his/her device to have the updated app.

III. Mobile Web Applications

Unlike native apps, which are independent executables that interface directly with the OS, web apps run within the browser.. Web apps entirely within the browser of the mobile device and make use of the newest JavaScript, CSS and HTML5 features that are available in modern mobile browsers. Most mobile vendors utilize the same rendering engine in their browsers, WebKit[8] – an open source project led mostly by Google and Apple that provides the most comprehensive HTML5 implementation available today. Since the application code is written in standard web languages that are compatible with WebKit, a single app delivers a uniform experience across different devices and operating systems,.

HTML 5 is the first HTML version to support multimedia without plugins. The HTML5 standard was created so web apps can be accessible and used on any device via a browser. HTML5 apps also have the ability for offline access and usage via the application cache, which means working without a network connection is now possible. A few examples of the potential of HTML5 include advanced UI components, access to rich media types, geo-location services .Using these features and many more that are under development, developers are able to create advanced applications using nothing but web technologies [10].

There are two extreme web-app approaches used in mobile web app. They are mobile browsing and mobile-optimized web sites. These sites recognize when they are accessed by a smartphone and serve up HTML pages that have been designed to provide a comfortable “touch experience” on a small screen size. But some companies go even further and enhance the user experience by creating a mobile website that looks like a native app and can be launched from a shortcut that is indistinguishable from that used to launch native apps. Some of the examples of mobile web apps are: Mercedes-Benz International site, http://m.usa.gov/, the *Guardian* and *Financial Times* newspapers both make compelling use of HTML5 in their mobile applications

3.1 Tools for developing mobile web applications

A growing number of JavaScript toolkits have been created, such as Sencha Touch and jQuery Mobile, which generate user interfaces that are comparable in look and feel to native apps.

❖ **Sencha Touch**

Sencha Touch is a user interface (UI) JavaScript library, or framework, specifically built for the Mobile Web[11]. It is fully based on web standards such as HTML5, CSS3 and JavaScript. Sencha Touch aims to enable developers to quickly and easily create HTML5 based mobile apps that work on Android, iOS and Blackberry devices, and produce a native-app-like experience inside a browser.

Sencha Touch includes a set of graphical user interface GUI-based controls (or components) for use within mobile web applications. All the components can be themed according to the target device. This is done using Sass, a style sheet language built over CSS.Sencha Touch has four in-built transition effects: slide over or under the current element, pop, flip, and cube. It supports common touch gestures built from touch events, which are Web standards but supported only by Android, iOS, and some touch enabled devices. These are tap, double tap, swipe, scroll, and pinch.

Example apps using sencha touch: The Watch List, Touch Tweets, Getographer, RSS Reader

❖ **jQuery Mobile**

jQuery Mobile is a touch-optimized web framework currently being developed by the jQuery project team. The development focuses on creating a framework compatible with a wide variety of smartphones and tablet computers, [12] jQuery Mobile provides a powerful theming framework that allows developers to customize color schemes and certain CSS aspects of UI features

Examples of apps using jQuery mobile framework: Slideshare , Stanford, Disney World.

3.2 Advantages of Mobile Web Applications

- **Multiplatform Support:** Since the application code is written in standard web languages that are compatible with WebKit, a single app delivers a uniform experience across different devices and operating systems, making it multiplatform by default.
- **Ease of development:** If developer has experience developing Web apps, HTML5 app development is easy. If developer is new to Web development, the technological bar is lower; it's easier to get started here than in native or hybrid development.
- **Immediate updates and distribution control:** One of the biggest benefits to IT organizations developing mobile applications in HTML5 is the ability to deploy those apps and updates directly to the user community via the browser. No third party or extra step is needed for distribution.

3.3 Challenges for Mobile Web App Development

- **User Experience**: the HTML5 standard has delivered more native-like capabilities such as access to the GPS location or accelerometer for mobile web applications and more as of late. However, these still fail to deliver the same user experience on different devices and perform slower when compared to a native implementation on the iPhone or Android..
- **Security**: HTML5 presents unique security risks when compared with native apps.HTML5 now offers the ability to

cache data within the browser. None of the vendors can affect data on the browser's cache so therefore; they cannot secure or manage that data.

- **Performance:** Mobile web apps are slower since their code is interpreted by the JavaScript engine running within the browser. Thus, if the user interfaces are graphic heavy or require excessive data processing, the Web App approach struggles to deliver the goods.
- **Native Look and Feel:** There are several web frameworks that provide libraries that can be used by mobile web apps and hybrid apps to re-create and imitate native mobile interfaces and behavior. However, the effort required to build these interfaces using native code is a fraction of the effort required to mimic the native look and feel.
- **Offline Capability: W**eb apps stop functioning when the user experiences unexpected loss of connectivity due to network (Radio Frequency) issues or the device deliberately goes off the grid (like when on an airplane). HTML5 has some support for offline functionality, but not all mobile browsers support this in a standard way.

IV. Hybrid Mobile Applications

Hybrid development combines the best of both the native and HTML5 worlds. We define hybrid as a web app, primarily built using HTML5 and JavaScript, which is then wrapped inside a thin native container that provides access to native platform features [3]. The native portion of the application uses the operating system API's to create an embedded HTML rendering engine that serves as a bridge between the browser and the device API's. This bridge allows the hybrid app to leverage all the features that modern devices have to offer. App developers can choose between coding their own bridge and taking advantage of ready-made solutions such as PhoneGap – an open source library that provides a uniform JavaScript interface to selected device capabilities that is consistent across operating systems.

The native portion of the app can be developed independently, but some solutions in the market provide this type of a native container as part of their product, thus empowering the developer with the means to create an advanced application that utilizes all the device features using nothing but web languages

The web portion of the app can be either a webpage that resides on a server or a set of HTML, JavaScript, CSS and media files, packaged into the application code and stored locally on the device. The best of both worlds can be achieved by combining the two approaches. Such a system is architected to host the HTML resources on a web server for flexibility, yet cache them locally on the mobile device for performance.

Examples of apps using hybrid approach: TripCase, Panasonic, World Heritage Calendar - using PhoneGap, other examples of apps are Facebook, TuneIn Radio, and LinkedIn etc.

Some organizations provide framework for developing Hybrid applications. One of the most popular frameworks is PhoneGap.

❖ PhoneGap

PhoneGap is a mobile development framework produced by Nitobi, purchased by Adobe Systems [14]. It enables software programmers to build applications for mobile devices using JavaScript, HTML5 and CSS3, instead of device-specific languages such as Objective-C. The resulting applications are hybrid, meaning that they are neither truly native nor purely web-based.

The core of PhoneGap applications uses HTML5 and CSS3 for their rendering and JavaScript for their logic. The PhoneGap framework embeds HTML5 code inside a native WebView on the device, using a Foreign Function Interface to access the native resources of the device.

PhoneGap currently supports development for the operating systems Apple iOS, Google Android, LG webOS, Microsoft Windows Phone, Nokia Symbian OS, RIM BlackBerry and Tizen.

4.1 Advantages of Hybrid Applications

- **Native Development Advantages** - With hybrid, all the advantages of native development, such as device integration, offline access and synchronization, push notification, application market integration, and a better user experience can be realized.

- **Unifies Web and Mobile Development -** HTML, CSS and JavaScript, with their wide adoption and easy portability, have allowed mobile application development to be truly cross-platform
- **Hybrid Development Leverages HTML5 -** The majority of HTML language that is used in mobile web and hybrid development is HTML5. It brings with it a richer user interaction and capabilities with the web browser.

Following features of hybrid apps make them beneficial over mobile web apps.

- **Database storage:** In mobile web, data is stored as plain text. There is a room for security threat (a) If data is lost (b) if the storage location is fixed and other apps can access it. With hybrid applications, data can be stored securely with encryption.
- **Media:** With hybrid applications, data can not only be played back, but recorded using the native bridge.
- **Network connections:** HTML5 supports web sockets. The Hybrid approach can handle full socket communications. Native mobile components can open a socket and can communicate with the server/ other devices, just like in traditional socket communication.
- **Push notification:** Real-time push notifications are possible with the Hybrid approach via the use of native components.

4.2 Challenges of Hybrid Mobile Applications:

- **Security vulnerabilities:** Security vulnerabilities are generally the same for hybrid and web applications. Most modern web browsers prevent certain vulnerabilities such as malicious scripts or cross-domain requests; however, typically hybrid applications use uiWebview (iOS), WebView (Android); these are native controls that use WebKit engines and do not offer the same level of support as a browser does
- **Performance:** Performance is one of the paramount concerns for the application developers. The key reasons for performance degradation are:
  - Data transfer across multiple layers of native, JavaScript libraries and WebView.
  - Rendering of web pages from the server.
  - Loading of larger images.
- **Content adaptation/switching:** Hybrid applications have the challenge of dynamically adjusting the content to the form factor of the web view window.

The content adaptation can be done on the server, client or both.

- **Application upgrades:** Upgrades of hybrid apps can be tricky since the content can exist on the web and the native parts of the application. Upgrades become mandatory with a change in the native content or in the native wrapper libraries. Changes in the web content may also necessitate an upgrade if the web content is embedded as part of the application.
- **Data sharing aspects:** As mentioned earlier, along with the navigation, data flow can also happen across components. Application developers should design a channel for sharing the transient data across the native -> web -> native transitions

V. Comparison of three approaches

Following figure illustrates the architectural difference among three approaches:

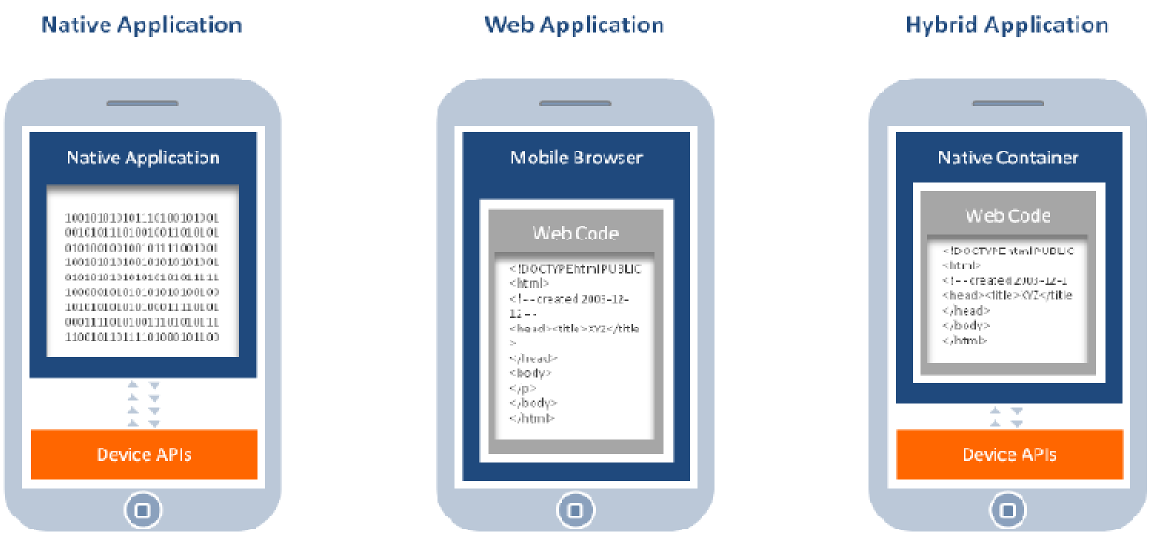

Figure 1: Comparison of three mobile application development Strategies [3]

As shown in figure, the native application directly interacts with device OS to access device APIs and executes directly on device. Web application executes in device web browser and consists of web code developed using HTML5, CSS3 and JavaScript. The Hybrid application is combination of native as well as web approach. The web portion of the app can be either a webpage that resides on a server or a set of HTML, JavaScript, CSS and media files, packaged into the application code and stored locally on the device. The native portion of the application uses the operating system API's to create an embedded HTML rendering engine that serves as a bridge between the browser and the device API's. This bridge allows the hybrid app to leverage all the features that modern devices have to offer.

Following table illustrates comparison of three development approaches according to various criteria:

	Native	**HTML5**	**Hybrid**
App Features			
Graphics	Native APIs	HTML, Canvas, SVG	HTML, Canvas, SVG
Performance	Fast	Slow	Slow
Native look	Native	Emulated	Emulated
Distribution	Appstore	Web	Appstore
Device Access			
Camera	Yes	No	Yes
Notifications	Yes	No	Yes
Contacts, calendar	Yes	No	Yes
Offline storage	Secure file storage	Shared SQL	Secure file system, shared SQL
Geolocation	Yes	Yes	Yes
Gestures			
Swipe	Yes	Yes	Yes
Pinch, spread	Yes	No	Yes
Connectivity	Online and offline	Mostly online	Online and offline
Development skills	Objective C, Java	HTML5, CSS, JavaScript	HTML5, CSS, JavaScript
Upgrade Flexibility	Low	Medium	High

VI. Conclusion

Today, the development strategies used for mobile applications can be broadly classified into the mobile Web App approach, Native App approach and Hybrid App approach. Each of these approaches has its strengths and drawbacks. 'One size fits all' is not applicable. There is a distinct difference between the Web App approach and the Native approach. The two key drawbacks of using the Web App approach over the Native approach are:

- Inability of web apps to access the device sensors/other hardware

- Difficulty in building a unique game-like interface with native look and feel.

The key benefit of the Web App approach is cross-platform support. The Hybrid App approach emerged precisely to bridge the gap between the Web App and the Native App approaches. With the Hybrid App approach, one gets cross-platform support without having to forgo access to device capabilities.

Thus choice of development approach is that the ideal development approach for any mobile application depends on the enterprise and the application needs. Yet, with a solid hybrid application development framework, for enterprise applications we lean towards the Hybrid App approach more often than not since it provides multi-platform support cost-effectively, has lower TCO and does not limit access to the device hardware.

References

[1] Native, HTML5, or Hybrid Understanding Your Mobile Application Development Options - developer.force.com.htm

[2] HTML5, Hybrid or Native Mobile App Development, -Worklight

[3] Jack Vaughan, The path to mobile application development-TechTarget

[4] http://instagram.com/#

[5] Chris McGuirk, Tony Pekala, Jason Petrin, and Eka Renardi, Choosing the Right Mobile Development Method, RDA Corporation Whitepaper

[6] Mac Developer Library, http://developer.apple.com/library/mac/#documentation/

[7] Eve Hoggan and Other Authors -Touch Rotation Gestures: Performance and Ergonomics, ACM, April 27–May 2, 2013

[8] www.webkit.org/

[9] Native Vs. Html5 Mobile App Development: Which Option Is Best? Appcelerator Whitepaper, 2012 Appcelerator

[10] Eric Freeman, Elisabeth Robson, Head First HTML5 Programming, O'Reilly Media

[11]http://www.sencha.com/products/touch

[12] http://en.wikipedia.org/wiki/JQuery_Mobile

[13] Sankar Srini, Hybrid Mobile Application Development Approaches, TCS Whitepaper

[14] Rohit Ghatol, Yogesh Patel, Beginning PhoneGap -Mobile Web Framework for JavaScript and HTML5, Appress