Native, web or hybrid mobile-app development

IPR2025-01400
CrowdStrike EX1029 Page 1

Contents

2 Introduction

2 Introducing the approaches

2 Native apps

3 The application programming interface (API)

4 Mobile-web apps

6 Hybrid apps

7 Comparing the different approaches

8 Choosing the right approach

8 Scenarios for the native approach

9 Scenarios for the web approach

9 Scenarios for the hybrid approach

10 Summary

Introduction

Many organizations taking their first steps to implement a mobile strategy are facing an important decision that will influence the results of this initiative. The process of choosing a development approach for a mobile application (hereafter referred to as an "app"), namely native, web or hybrid, entails many parameters, such as budget, project timeframe, target audience and app functionality to name a few. Each approach carries inherent benefits and limitations, and finding the one that best addresses the organization's needs could be a challenging task.

The purpose of this white paper is not to identify the best development approach, as none exists, but rather to list the pros and cons each carries and to describe the different scenarios, or enterprise requirements, that best fit one or the other.

Introducing the approaches

Native apps

Native apps have binary executable files that are downloaded directly to the device and stored locally. The installation process can be initiated by the user or, in some cases, by the IT department of the organization. The most popular way to download a native app is by visiting an app store, such as Apple's App Store, Android's Marketplace or BlackBerry's App World, but other methods exist and are sometimes provided by the mobile vendor.

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 and, in many cases, has unique features and functions that are typical of that specific mobile OS.

To create a native app, developers must write the source code (in human-readable form) and create additional resources, such as images, audio segments and various OS-specific declaration files. Using tools provided by the OS vendor, the source code is compiled (and sometimes also linked) 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.

These tools, in addition to other utilities and files, are normally called the software development kit (SDK) of the mobile OS. Although the development process is often similar for different operating systems, the SDK is platform-specific and each mobile OS comes with its own unique tools. The following table presents the different tools, languages, formats and distribution channels associated with the leading mobile operating systems.

These differences across platforms result in one of the most critical disadvantages of the native development approach—code written for one mobile platform cannot be used on another, making the development and maintenance of native apps for multiple OSs a very long and expensive undertaking.

So, why is it that in spite of this costly disadvantage, many companies choose to develop natively? To answer that question, we will need to better understand the role of the APIs.

The application programming interface (API)

Once the native application is installed on the mobile device and launched by the user, it interacts with the mobile operating system through proprietary API calls that the operating system exposes. These can be divided into two groups: low-level APIs and high-level APIs.

	Apple iOS	**Android**	**Blackberry OS**	**Windows Phone**
Languages	Objective-C, C, C++	Java (some C, C++)	Java	C#, VB.NET and more
Tools	Xcode	Android SDK	BB Java Eclipse Plug-in	Visual Studio, Windows Phone development tools
Packaging format	.app	.apk	.cod	.xap
App stores	Apple App Store	Google Play	Blackberry App World	Windows Phone Marketplace

Low-level APIs

It is through these low-level 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. It can access the Global Positioning System (GPS), receive orientation information and, of course, read and write files on the solid-state disk or access any other hardware element available today or in the future.

High-level APIs

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.

Although most mobile OSs include a set of built-in applications that can execute these services, a set of exposed high-level APIs is made accessible for native apps as well, allowing them to access many of the important services mentioned above. Other APIs enable downloadable apps to access various cloud-based services that are provided by the OS vendor, such as push notifications or in-app purchases.

The graphical user interface (GUI) toolkit

Another important set of APIs that the OS provides is the GUI toolkit. Each mobile OS comes with its own set of user interface components, such as buttons, input fields, sliders, menus, tab bars, dialog boxes and so on. Apps that make use of these components inherit the features and functions of that specific mobile OS, which normally results in a very easy and enjoyable user experience.

It's important to note that different mobile platforms carry unique palettes of user interface (UI) components. As a result, apps that are designed to work for multiple operating systems require the designer to be familiar with the different UI components of each OS.

Although APIs are OS-specific and add much complexity and cost to the development of multiple native apps, these elements are the only means of creating rich mobile applications that make full use of all the functionality that modern mobile devices have to offer.

Mobile-web apps

Modern mobile devices consist of powerful browsers that support many new HTML5 capabilities, Cascading Style Sheets 3 (CSS3) and advanced JavaScript. With recent advancements on this front, HTML5 signals the transition of this technology from a "page-definition language" into a powerful development standard for rich, browser-based applications.

A few examples of the potential of HTML5 include advanced UI components, access to rich media types, geolocation services and offline availability. Using these features and many more that are under development, developers are able to create advanced applications, using nothing but web technologies.

It is helpful to distinguish between two extreme web-app approaches. Everyone is familiar with mobile browsing and mobile-optimized websites. 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.

There is a wide range of possibilities between these two extremes, with most websites implementing their own mix of features. Mobile web apps are a very promising trend. To capitalize on this trend and help developers build the client-side UI, a growing number of JavaScript toolkits have been created, such as dojox.mobile, Sencha Touch and jQuery Mobile, which generate user interfaces that are comparable in appearance to native apps. Both execute 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.

One of the most prominent advantages of a web app is its multiplatform support and low cost of development. Most mobile vendors utilize the same rendering engine in their browsers, WebKit—an open-source project led mainly by Google and Apple that provides the most comprehensive HTML5

Feature	Pure mobile web apps	Pure mobile websites
Tools and knowledge	Written entirely in HTML, CSS and JavaScript	Written entirely in HTML, CSS and JavaScript
Execution	"Installed" shortcut, launched like a native app	Reached by navigating to a website by way of a Uniform Resource Locator (URL)
User experience	Touch-friendly, interactive UI	Navigational UI between pages displaying static data
Performance	UI logic resides locally, making the app responsive and accessible offline	All code executed from a server, resulting in network-dependent performance

implementation available today. Because the application code is written in standard web languages that are compatible with WebKit, a single app delivers a uniform experience for different devices and operating systems, making it multiplatform by default. However, these advantages are not without a price.

Despite the potential and promise of web technologies in the mobile space, they still carry significant limitations. To understand these limitations we need to explain how web applications operate.

Unlike native apps, which are independent executables that interface directly with the OS, web apps run within the browser. The browser is in itself a native app that has direct access to the OS APIs, but only a limited number of these APIs are exposed to the web apps that run inside it. While native apps have full access to the device, many features are only partially available to web apps or not available at all. Although this is expected to change in the future with advancements in HTML, these capabilities are not available for today's mobile users.

Hybrid apps

The hybrid approach combines native development with web technology. Using this approach, developers write significant portions of their application in cross-platform web technologies, while maintaining direct access to native APIs when required.

The native portion of the application uses the operating system APIs to create an embedded HTML rendering engine that serves as a bridge between the browser and the device APIs. This bridge enables the hybrid app to take full advantage of all the features that modern devices have to offer.

App developers can choose between coding their own bridge or taking advantage of ready-made solutions such as PhoneGap—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. In some cases, a solution will allow the developer to use any native knowledge he or she might have to customize the native container in accordance with the unique needs of the organization.

The web portion of the app can be either a web page 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. Both approaches carry advantages and limitations. HTML code that is hosted on a server enables developers to

introduce minor updates to the app without going through the process of submission and approval that some app stores require. Unfortunately, this approach eliminates any offline availability, as the content is not accessible when the device is not connected to the network. On the other hand, packaging the web code into the application itself can enhance performance and accessibility, but does not accept remote updates. The best of both worlds can be achieved by combining the two approaches. Such a system is designed to host the HTML resources on a web server for flexibility, yet cache them locally on the mobile device for performance.

Comparing the different approaches

To summarize, a comparison of all three development approaches follows.

The native approach excels in performance and device access, but suffers in cost and updates. The web approach is much simpler, less expensive and easier to update, but is currently limited in functionality and cannot achieve the exceptional level of user experience that can be obtained using native API calls. The hybrid approach provides a middle ground which, in many situations, is the best of both worlds, especially if the developer is targeting multiple operating systems.

As can be inferred from the table above, no single approach delivers all the benefits all the time. Choosing the right approach depends on the specific needs of the organization and can be driven by many parameters, such as budget, timeframe, internal resources, target market, required application functionality, IT infrastructure and many others.

One thing is clear: most companies today face an obvious tradeoff between user experience and application functionality on the one hand, and development costs and time to market on the other. The challenge becomes choosing the right development approach that will balance the organization's requirements with its budget and time-to-market constraints.

Feature	Native app	Hybrid app	Web app
Development language	Native only	Native and web or web only	Web only
Code portability and optimization	None	High	High
Access device-specific features	High	Medium	Low
Leverage existing knowledge	Low	High	High
Advanced graphics	High	Medium	Medium
Upgrade flexibility	Low (Always by way of app stores)	Medium (Usually by way of app stores)	High
Installation experience	High (From app store)	High (From app store)	Medium (By way of mobile browser)

Choosing the right approach

The following is a list of scenarios to help guide organizations in the process of choosing an approach.

Scenarios for the native approach

Existing native skills. One of the main arguments against the native approach is its lack of multiplatform support. Organizations asking to develop an application for multiple mobile platforms need to hire new employees or train in-house developers in a variety of native languages. Organizations that have such native skills in-house are able to take advantage of them, without significant new investments.

A single mobile OS. In some cases, an organization will aim to release a mobile application to a limited target audience—one that is known to use a single mobile OS. For example, consider a scenario in which an internal application is distributed within an organization that issues a BlackBerry device to its employees. In this case, achieving multiplatform coverage might not be a priority and, as developing a single native application requires a limited set of skills and tools, this approach can make much sense.

Native functionality. Some applications are built around a single functionality. Take Skype, for example: Voice over Internet Protocol (VoIP) and access to the user's contacts are key elements of the app and, given available technologies today, can only be developed natively. For such applications, web languages are simply not yet sufficiently evolved and are far from capable of achieving the desired functionality.

Rich UI requirements. For game-like applications that require a rich UI that provides real-time responsiveness, web technologies do not yet provide an adequate solution. For applications with such requirements, developers are still better off taking the native approach.

Scenarios for the web approach

Direct distribution. Some organizations prefer distributing their apps in a manner that is controlled internally and is not subjected to what can sometimes turn into a long and uncertain approval process. In such cases, using purely web languages can completely circumvent the app-store process and allow the organization to fully control the distribution of the app and its periodical updates.

Pilot app. When comparing the costs and time to market involved in the development of a native as opposed to a web app, using the web approach to create a pilot version of the app can be a compelling and cost-effective tactic. Once the concept has been proved, the organization can choose to create a new application from the beginning or use portions of the existing code in a hybrid application.

Visibility. In addition to the distribution we already mentioned, another benefit of creating a web application is its visibility in search engine results which, in many cases, expose the application to a larger audience than that available through the app store alone.

Scenarios for the hybrid approach

Balancing the tradeoff. Using the hybrid approach, companies can enjoy the best of both worlds. On the one hand, the native bridge enables developers to take full advantage of all the different features and capabilities that modern mobile devices have to offer. On the other, all the portions of the code that are written using web languages can be shared among different mobile platforms, making the development and ongoing maintenance process centralized, shorter and cost-effective.

In-house skills. Web development skills are very common and can easily be found in many organizations. By choosing the hybrid approach, supported by the right solution, web developers are able to build applications with nothing but web skills, such as HTML, CSS and JavaScript, while delivering a native-like user experience.

Future considerations. HTML5 is rapidly growing in both availability and capabilities. Many analysts predict that it is likely to become the default technology for client-side application development. By writing most of the app in HTML, and using native code only where needed, companies can make sure that the investments they make today do not become obsolete tomorrow, as HTML functionality becomes richer and addresses a wider range of the mobile requirements of modern organizations.

IPR2025-01400
CrowdStrike EX1029 Page 9

Summary

As mobile apps continue to take a central role in the business landscape, organizations around the world are mobilizing a growing number of mission-critical services. Many companies are striving to find the optimal development approach to achieve their goals, but what many are quickly realizing is that each approach carries inherent limitations and no single approach can address all the growing needs and complexity of the modern mobile enterprise.

As this paper attempts to show, the answer lies not in one development approach, but rather in a flexible solution—one that can harness the benefits that each provides and support not only the development of a first mobile app, but of all future apps, regardless of their development approach.

The choice between hybrid, native and web development approaches, although certainly a major one, is not the only one. Companies forming their mobile strategy must also consider the future of this market, illustrated by the following trends and developments:

- Further fragmentation of mobile devices and technologies, which in turn will continue to increase the overall costs and complexities that are associated with mobile-app development, integration and ongoing management
- Accelerated mobile adoption by consumers and within the enterprise, increasing the requirements around security, scalability and ongoing control
- New device features and complementing technologies, such as near-field communication, geolocation, augmented reality, social networks and others, which will undoubtedly give rise to new types and new use cases of mobile apps
- New distribution channels for the apps, both public and private, enabling organizations to easily place the apps in the hands of the user, quickly deploy updates and manage its entire portfolio of apps, without going through a long submission and approval process

Taking all these parameters into consideration, companies must choose a solution that is not only flexible enough to support all types of apps, but would also support the secure and scalable integration of the apps into the IT infrastructure and enable them to monitor and control their entire portfolio of applications from one centralized interface.

For further information, and to download a free trial version, please visit the following website:
ibm.com/software/mobile-solutions/worklight/

IPR2025-01400
CrowdStrike EX1029 Page 10

For more information

To learn more about IBM Worklight assets for mobile-app development, please contact your IBM marketing representative or IBM Business Partner, or visit the following website: **ibm.com**/software/solutions/mobile-enterprise

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We'll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit: **ibm.com**/financing

© Copyright IBM Corporation 2012

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
April 2012

IBM, the IBM logo, ibm.com, and WebSphere are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at **ibm.com**/legal/copytrade.shtml

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

It is the user's responsibility to evaluate and verify the operation of any other products or programs with IBM products and programs. THE INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM products are warranted according to the terms and conditions of the agreements under which they are provided.

Please Recycle

WSW14182-USEN-01