Native, Hybrid or Mobile Web Application Development

Learn more about the three approaches to mobile application development and the pros and cons of each method.

White Paper

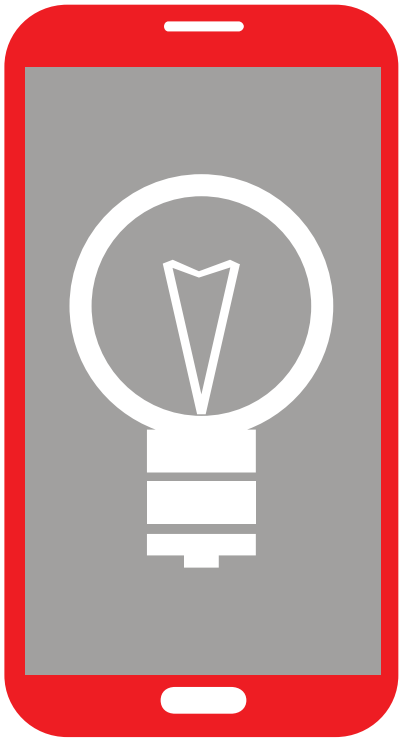
optimus information

Develop a Mobile Application Strategy for Your Business

One of the key components in developing a mobile application strategy is the choice of how and where the mobile applications being developed will run. There are three development approaches: native, web or a hybrid of the first two. The tradeoffs between each can be measured in terms of functionality, performance, the user experience and overall costs.

How these three approaches balance along these four dimensions and potentially fit within an enterprise's mobile application requirements is the topic of this white paper. Once the characteristics of each is understood, and the pros and cons of each as a development and deployment path are described, the choice of one over the other will be clarified.

There is an additional factor that applies if the mobile application's purpose is to create direct revenue, marketing collateral or improve goodwill. This factor is the opportunity and ability to monetize the app either tangibly or intangibly. Some monetization approaches apply equally to each development approach and others do not. This factor can be relevant to customer apps, mobile workforce apps and B2B apps, so a brief discussion of this is offered after examining the pros and cons of the development approaches.



How Each Mobile Development Approach Works

Native Mobile Applications

A native mobile app is intimately tied to the platform on which it is running. It has the ability to fully integrate with the capabilities of both the hardware and the OS on which it resides. It is completely analogous to most PC applications, which are downloaded to a desktop's or laptop's hard drive and completely executed within that machine.

In order to accomplish this tight integration, the mobile app developer utilizes an SDK from the hardware manufacturer directly or via the mobile device's OS vendor. Combined with a comprehensive development tool, the IDE, a developer is able to code the mobile application logic and take advantage of any hardware or OS functionality that is available via the exposed APIs.

Through the APIs, the native app can utilize OS features and services such as the file system, the phone and network drivers. It can interact also with built-in resources such as contacts, multimedia files, calendars, email, browsers and other productivity components. Lower-level APIs provide direct access to specific hardware capabilities such as screen gestures, virtual and hard buttons, sliders, graphics, the audio system, GPS, accelerometers and so on.

Owing to its close relationship with the execution platform, native mobile apps also have full access to the proprietary graphical user interface widgets and functions through a proprietary development environment. Although these IDEs are usually expensive initially, they save time by automating UI creation and providing more complete run-time and compile-time error detection.

Mobile Web Applications

Web-based mobile apps are developed in a very different environment from native apps. These are produced with the same tools used for mobile website development through the use of HTML, CSS style sheets and JavaScript within the HTML5 standard. HTML5 provides the ability to create rich UI experiences with support for rich media, UI components, geolocation, and offline execution.

Third-party suppliers of JavaScript toolkits can supply UI components that allow web-based apps to mimic native look and feel on the mobile device, such as Dojo or jQuery. However, their ability to provide precise native look and feel varies across toolkits. This ability requires

more memory resources for the UI than a native app would consume and also the ability of the browser to render UI components accurately.

Web-based apps execute on the mobile platform but instead of running directly on the OS of the device they are executed within the mobile web browser. Thus, these applications can run on multiple platforms as long as they are compatible with a device's browser rendering engine. In practice, portions of most web apps run on both the device and a backend server. The degree to which app functionality is partitioned between client and server has meaningful impacts on performance, functionality and app maintenance.

Hybrid Mobile Applications

A hybrid mobile application is developed using both native libraries and web technologies in an attempt to get the best of both worlds. The interface between the separate components is an on-platform, embedded HTML rendering engine, which is either developed in-house or by acquiring one from a 3rd-party.

The native portion of the app can be written as a top to bottom native app, which communicates to a web-based server backend. This has the same porting issues as a purely native app. Alternatively, 3rd-party cross-platform development tools exist that use native library containers to achieve near-native performance. Such tools bring the benefits of cross-platform development in both the native and web-based portions of a hybrid mobile app.

The web side of a hybrid app is often identical in development and execution as a fully web-based app. However, it can also be written to be completely self-contained with all the HTML, CSS, JavaScript, media and UI components stored and executed on the mobile device. Functionally, it behaves as if it were a native app but performance is dependent on the abilities of the rendering engine.

The Pros and Cons of Each Development Approach

Native Mobile Applications

PROS

- **Functionality:** Native apps have direct access to all a mobile device's features including ones that are unique to the platform. These include graphics hardware, motion, location, proximity, visual and audio sensors. Mobile devices may also possess unique security devices, touchscreen functions and peripheral interfaces only available via native libraries.
- **Performance:** Since they run directly on the platform without intermediary containers or abstraction layers, native applications run with the highest performance possible. Native IDEs offer the best runtime optimizations as well.
- **User Experience:** The ability to directly access UI components of specific platforms means that apps can seamlessly blend with the operating environment. Given that native UI layout and functionality also matches the platform, this eliminates disruption to the user experience.

CONS

- **Costs:** Software development environments differ between manufacturers and OS vendors. Thus, each native app must be written in specific programming languages to specific APIs with SDKs and tools for that particular platform. Furthermore, device capabilities may differ between manufacturer models that affect how a native app functions. All of this uniqueness means that to cover every device in a fragmented mobile market requires double or treble the effort in development and maintenance as compared to a one-size-fits-all approach. If the enterprise strategy is to focus solely on one platform, then these additional efforts do not accrue. Alternatively, the purchase of a 3rd-party cross-platform, native development SDK can approximate write-once-run-everywhere capability for native apps but at increased initial cost.

Mobile Web Applications

PROS

- **Costs:** Web-based apps overall have the lowest costs among the three approaches, which helps explain their growing popularity within enterprise mobile strategies. Initial costs are reduced because open standards and open source tools can be used to create the apps. More importantly, once the code is written, the app can run on any platform with the correct HTML rendering engine regardless of the native OS. Since WebKit is a de facto rendering engine standard, this means in a practical sense that write-one-run-everywhere is 99 percent achievable with web-based mobile apps.

CONS

- **Functionality:** Although libraries for web-based apps are able to access many mobile device hardware capabilities, this is done via abstraction layers that must necessarily take an LCD approach to these capabilities. Web-based apps are improving in this regard, however.
- **Performance:** Because web-based mobile apps must code through abstraction layers and execute a step removed from the actual hardware and OS, their performance can never approach that of a native app. Furthermore, many web-based apps are spread between the client and server, so their performance is often impacted by network latency as well.
- **User Experience:** The ability to mimic a platform's native UI/UX features will always have a meaningful gap compared to a native app, but that gap is gradually shrinking thanks to improvements in HTML5 and JavaScript. If the app is designed to access a mobile website specifically designed to render as an app and offline execution abilities are maximized, the UI/UX distinctions blur even further.

Hybrid Mobile Applications

PROS

- **Functionality:** Depending on how a hybrid mobile app is partitioned, there is no reason that it cannot approach the functionality in terms of platform feature access and on-platform device access that native apps enjoy.
- **Performance:** A hybrid app can match the performance of native apps for the portion of the app that runs natively, especially in offline mode.
- **User Experience:** Just as for performance and functionality, the ability to create a fully functional, seamless UI/UX depends on how these characteristics are partitioned between the native and web-based portions of the app. In hybrid apps, getting the UI/UX right is the main motivation for creating the native part of the app in the first place, so usually the UX is nearly identical to native apps.
- **Costs:** Costs are more or less midway between those for native and web-based apps. Where the actual costs lie depends on the balance between native and web-based development efforts and the effort to merge the two. If the balance is heaviest on the native side, then costs increase if the enterprise strategy is to run across multiple hardware/OS platforms. Weighting development toward the web-based portion of the app, reduces the duplication of effort on the native side linearly while taking more advantage of scalability and easier app maintenance.

CONS

- **Costs:** Though the native portion enjoys high performance, the web-based portion of a hybrid mobile app suffers from the same deficiencies as any web-based application. The degree to which this reduced performance affects the app overall mainly depends on how much of the app's web-based logic executes on the device, how much is on a server component and the network latency between the two.

Other Important Mobile Development Decisions

Monetization Considerations

Enterprises may target their mobile app strategy toward various kinds of uses:

- Business-oriented, productivity apps with specific functions for a mobile workforce
- B2B applications intended for customers and partners
- End user apps for customers or the general public

Depending on the intended target market for a mobile app and the approach to its development, any of these app usages can present marketing and monetization opportunities.

Monetization Methods

Native apps achieve monetization chiefly from download or licensing fees although they may generate receive revenue from advertising via third-party portals. These are usually heavily restricted by manufacturers. The payment infrastructure for downloads is already set up in app stores.

Web-based apps, on the other hand, must set up their own payment process in order to charge for download or access. A common approach to this is to charge for subscriptions rather than a one-time fee. Web-based apps more easily take advantage of advertising revenue by embedding ads on off-platform portions of the app. On the other hand, embedding ads in apps is not considered best practice as users often find them annoying.

An emerging approach to mobile app monetization, especially for B2B apps, is selling access to transactional data either per use or on a subscription basis. These data could be used by third-parties to analyze pricing models or customer behavior for example. Data can be supplied from repositories within the enterprise IT structure or real-time data is provided directly from within an app.

Naturally, the data cannot include sensitive, private information about customers or the enterprise. With regard to this monetization approach, the superior performance and UX of native apps are not typically required.

If marketing or monetization opportunities are a possibility within the organization's mobile strategy, care must be taken to build it into the initial requirements and not as an afterthought. Additionally, if a cross-platform approach is envisioned, how monetization is presented should be optimized for each platform's UX.

Application Adoption

If a mobile application is being developed in partnership with a mobile device manufacturer or OS provider, a native app is least likely to negatively affect user opinion about the platform itself. If widespread adoption of the app is also a requirement, then adequate performance is critical, which also argues for native development.

Web-based application adoption can suffer also because these apps do not appear in the most popular app repositories such as Google Play and App Store, so additional marketing effort is usually required for them to gain optimum exposure.

Summary

The key metrics for deciding which mobile app development strategy is the best for an organization are functionality, performance, user experience and the costs of development and maintenance. There are some clear winners in each of these, especially when comparing native and web-based app development. These measurements blur somewhat for hybrid apps, however. Keep in mind also that embedded into the cost picture is the current skillset of your development teams. Choosing one approach over another may incur re-training costs.

All of the metrics and approaches must be taken into account in order to arrive at a decision that encompasses business goals, the target market, time to completion and application requirements for usability and functionality. If monetization opportunities exist and these are important to the success of the product, these must be included in the final decision as well.

There is, of course, nothing to preclude the adoption of two or more of these approaches simultaneously within an organization. Comparative baseline measures for costs and time could be developed by producing an identical app using more than one method, say, native versus hybrid. If an enterprise is anticipating multiple app projects that target different purposes or audiences, then a decision framework should be constructed to evaluate the appropriateness of each approach per app. It is probable that a single approach will not be adequate to cover all products.

If the final approach is to support multiple development methodologies, be sure the organization is flexible enough to support that. In particular, since the management and maintenance of the portfolio of apps is likely to fall to the IT department, ensure the app management interface is as uniform as possible regardless of the original app development methodology that was used.

About Optimus Information

Headquartered in Vancouver, Canada with delivery centers in Canada and India, we work as a trusted partner to medium and large businesses to solve their software and technology challenges. With a team of 150+ people Optimus Information provides global organizations with scalable, flexible and cost efficient solutions. **Optimus Information provides global reach with a local presence.**

604-736-4600 | **info@optimusinfo.com** | **www.optimusinfo.com**