

MOBILE ARCHITECTURE BEST PRACTICES: BEST PRACTICES FOR MOBILE APPLICATION DESIGN AND DEVELOPMENT

By John Sprunger

When developing mobile applications, there are a number of key challenges where architecture and design are fundamentally different from that of a typical enterprise application. Careful consideration should be given to these mobile architecture issues early in the development process in order to mitigate the downstream impact of poor architectural decisions. While some of these best practices also make sense for the development of non-mobile applications, many will become more readily apparent when developing on a mobile platform. The five most important areas for consideration, which are detailed throughout this document, include: performance, usability, data access, security, and connectivity.

PERFORMANCE

While more readily apparent in the previous years of mobile development, the computing power available on mobile devices still lags behind desktop and server counterparts and will continue to do so for the foreseeable future due to smaller device footprints and resource constraints. Even the most recent devices still boast only about one third to one half of the computing resources (CPU, RAM) of a low end desktop computer. Further, the quality of data connections available on a mobile device is often highly variable based on signal strength and is far inferior to broadband Internet access in most cases.

Often during rapid application development, performance considerations are ignored until the end of the project and optimized only when necessary. In mobile development, more consideration to performance constraints of the mobile device may need to be given up front in the design process. Each platform has different code-level best practices for performance optimization depending upon the programming language and frameworks available on the platform. Some best practices, such as judicious usage of memory and limits on the number of unnecessary objects created, however, can be applied across all platforms.

Care should especially be given to architectural decisions that can limit performance and are also difficult to change later in the development cycle, such as the design of web service APIs and data formats. General best practices for the design of web service APIs for use in mobile development could be summed up as:

- ◆ Only retrieve data that the application needs
- ◆ Only retrieve data when the application needs it

These considerations stem mostly from the limited bandwidth available to mobile devices. If possible, APIs used by a mobile application should be designed to retrieve only the most relevant and useful information—excluding any extra data that is not used by the application. When designing APIs to communicate with mobile applications, one recommendation is to use a lightweight data format like JSON instead of more verbose format such as XML in order to make the best use of limited bandwidth available to mobile devices. The use of a lightweight format like JSON will conserve bandwidth, will allow results to be retrieved more quickly, and also will generally enable faster deserialization of the data as it arrives on the mobile client.

Another important performance consideration on a mobile device is battery life. If an application is constantly polling a web service for updates or continually processing data in the background, the battery will be drained much more quickly. If architecturally feasible (and if the push notification capabilities exist on the mobile platform), the use of push notifications for providing data updates is recommended over periodic polling. Push notification capabilities currently exist on the iPhone, Android, and Windows Phone 7 platforms. If an application needs to perform large amounts of data processing or analysis, consider uploading the necessary data to a server-side platform to perform the CPU-intensive processing and then return the results to the device to avoid draining the battery and to provide a more-responsive user experience.



USABILITY

At the end of the day, usability is one of the key factors that will truly make or break user acceptance of an application. Each of the major mobile platform software vendors (Microsoft, Google, Apple) have released user-experience specifications and guidelines specific to their own platforms in an attempt to foster a consistent look and feel across all applications on their platforms—and if the guidelines are enforced by the vendor and followed by developers, then the payoff is absolutely realized. The user experience across applications on most of the major platforms is seamless—for example, on the more stringent iPhone and Windows Phone 7 platforms, the navigation of menus and the look and feel of most applications (down to the fonts and color schemes) are almost identical. This allows users to learn quickly how to use a new application and instead focus on performing the task at hand, rather than “switching gears” between disparate experiences or puzzling over how to interact with a new application. Below are links to the user experience guidelines for each of the major platforms:

- ◆ iPhone
- ◆ Android
- ◆ Windows Phone 7

While each platform may have specific user interface (UI) guidelines, the challenges of mobile application usability are ubiquitous and many best practices can be applied across all platforms. Following are a few of the most important usability considerations.

Consider the limited screen real estate. No longer do users have access to a 23-inch widescreen monitor to display every single piece of information at once. Only display the most relevant information and options on the screen. Menus and UI screens should not be cluttered within rarely used options; rather they should be buried deeper within a settings screen or a submenu. Conversely, if a feature is used on a regular basis, consider assigning it to a hardware button or making it readily available within the UI. For the sake of accessibility, avoid the use of small font sizes in order to cram more information onto the screen. Scrolling in mobile applications can be difficult for the end user, so limit the need to scroll within screens where possible.

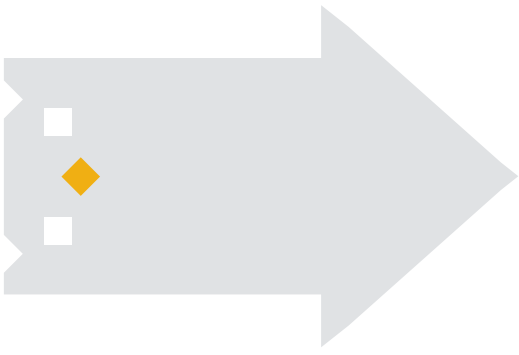
When displaying information, make use of the Summary / Detail / Edit UI paradigm. A ‘Summary’ view displays on the most important and relevant information and actions to the user. To access less important or less commonly used information, the user can then drill down to a more complete ‘Detail’ view. If the user needs to edit the information, switch to an ‘Edit’ view that will provide richer functionality around editing, validating, and persisting the data.

Factor in the lighting conditions. For both enterprise and consumer applications, consider whether the application will be primarily used indoors or outdoors. If the application will be used in low light or sunlight conditions, make use of high contrast, sufficiently bright colors. If the application is to be used in specific locations, such as a warehouse or factory floor, ensure that the application has undergone appropriate testing in the target setting.

Ensure that UI elements are sized appropriately. Each platform will have separate specifications for minimum button sizes relative to the screen resolution of the devices. For example, Microsoft’s Windows Phone 7 guideline for minimum size for any interactive UI element is 7mm or 26 pixels. Take into consideration the conditions in which the mobile application will be used – for example, if an application will be used on a warehouse floor, the users of the application may be wearing gloves and would require larger buttons and increased spacing between UI elements.

DATA ACCESS

One commonality between the most modern mobile platforms (iPhone, Android, Windows Phone 7) is that none of them offer any capability to connect directly to a database – for good reason. The current mobile architecture paradigm simply doesn’t support this scenario for modern database platforms in their current state. Given that most mobile applications communicate over the public Internet, access to a database would require exposing that database publicly – and in this age, no sane IT or database administrator would publicly expose an instance of Oracle, SQL Server, or MySQL outside the firewall without measures like a VPN or IP restrictions in place. While VPNs are becoming more available on modern mobile platforms, the complexities around cost, bandwidth, and end-user configuration simply don’t make business sense when compared with fronting a database with a more secure web service front-end.

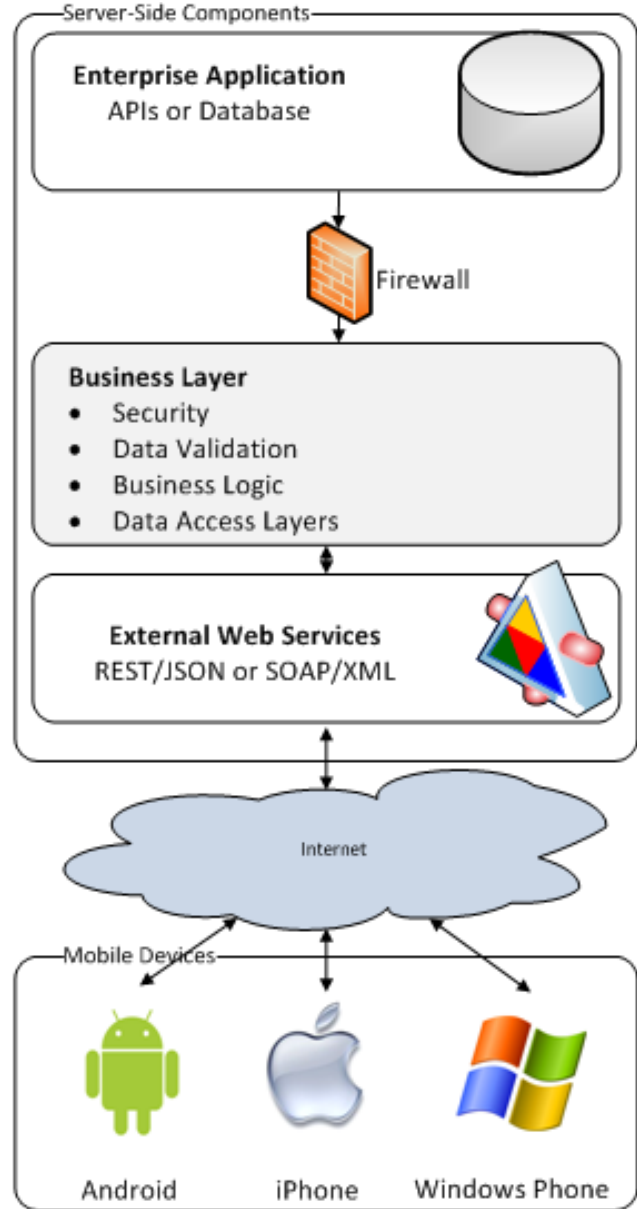


Rather than attempting to provide support for database client connectivity, the current paradigm for data access from mobile applications is based around web services. For the example scenario of extending a common two-tier enterprise application onto a mobile platform, usually a web services layer would first have to be created that would exist in front of the database or APIs of the enterprise application. In the design of a web services layer for a mobile application, logic around authentication, authorization, validation, and business rules should all be executed on the server-side web services of the extended application. As the web services are now exposed publicly for use by any properly authenticated user of your application, the validity of the data and the user's right to call the web service cannot be trusted without first performing additional server-side checks. Logic for validation and authorization can be duplicated on the mobile client side of the application to provide a more responsive user experience, but the user's actions should be checked again on the server side after the data is passed to the web service.

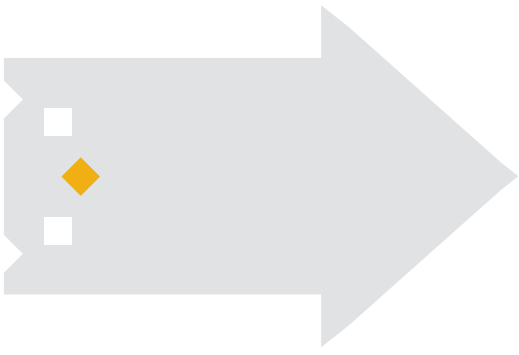
The architecture diagram at right below depicts how an enterprise application could be extended onto a mobile platform by wrapping either the application's APIs or database with a business layer that performs additional processing for validation and security. Note that if validation or authorization is built into the enterprise application's APIs or data access mechanism, then it is not necessary to re-implement this functionality within the web services layer.

SECURITY.

As previously mentioned, data access on mobile platforms generally requires some form of Internet-facing service or data access point that can be communicated with via a mobile device. Database servers and platforms in their current state are not good candidates for public exposure without additional layers of security that are generally not feasible or cost effective on mobile devices. Web servers are generally more hardened to attack and, thus, web services are an excellent candidate for exposure outside the firewall to mobile devices over the Internet. But what about securing these web services?



ENTERPRISE APPLICATION EXTENDED TO MOBILE PLATFORM



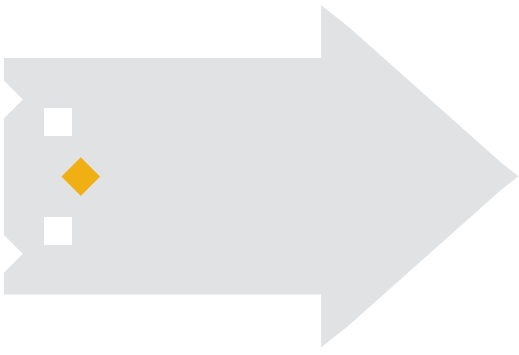
In most cases, the use of a web service API first requires authentication to ensure that the caller of the web services is who they say they are. Usually, web service API security will use a form of token-based authentication – this could be something like OAuth or as simple as sessions built into any modern server-side framework, such as ASP.NET or Ruby on Rails. In the general workflow of token based authentication, the web service caller sends a username and password and then receives a unique token back after his/her identity has been verified by the authentication service (e.g. LDAP). The token is then passed back to the web service on all subsequent requests and can be used on the server side to determine the identity of the user. Depending upon the security constraints of the application, the token generally expires after a certain period of inactivity. Regardless of the technology used to accomplish the token based authentication, all communication between the mobile client and the web server should be performed over an SSL-secured connection in order to prevent the token from being captured via packet sniffing on a wireless connection or any other “man-in-the-middle” attack. If the token were to be compromised by a third party, the third party would then be able to imitate the identity of the actual application user and would be able to make malicious requests, if inclined.

Another security issue inherent to mobile platforms is the security of data that exists locally on the device itself. Obviously, any mobile device can be compromised much easier than a server residing within a secure data center. If possible, confidential data should not be stored on the mobile device itself and should be stored instead on a back-end server and downloaded to the device when necessary. If for architectural reasons confidential data must be stored on the device, then measures should be taken to encrypt the data with a key that is not stored on the device, if possible. Fortunately, mobile platform vendors are providing more and more support for automatically encrypted disk storage, which makes implementation of secure data storage on the device much easier. One further consideration for mobile data storage security is that highly confidential data, such as private health information (PHI), should not be stored on a mobile device under any circumstances, encrypted or otherwise.

CONNECTIVITY

The final major architecture consideration for mobile applications is connectivity. It can no longer be assumed that the application being built will have access to an “always-on” high-speed Internet connection. In the wild, mobile devices will frequently switch between different types of connections (e.g. Edge, 3G, or WiFi) with wildly varying speeds and will often have no data connection. Often, the implementation of offline access for a mobile application simply doesn’t make sense business-wise, architecturally – perhaps the application must have access to only the most relevant and up-to-date data (e.g., traffic conditions), or when data is persisted it must be immediately validated and processed (e.g., stock trades). For most business applications, however, there are use cases for which offline access is absolutely necessary in order to maintain the end user’s productivity. One simple way to design offline access and data synchronization involves the creation of two basic components within the mobile application – a caching mechanism and a queuing mechanism.

The caching component handles offline access for data that would normally be retrieved as needed from a server-side API. The caching component can be designed to periodically (in a background thread) retrieve larger data sets that are potentially relevant to the needs of the end user, or it can be designed to only keep copies of any data previously retrieved from server. Data stored in the cache on the device should generally expire after a certain time period has passed in which the data is no longer useful or relevant. Another feature that can be designed into the caching feature is some level of intelligence related to the current type of connection on the device. For instance, if the cache is designed to periodically download large data sets, then perhaps it will only do so when the device is connected to WiFi in order to conserve bandwidth when connected to slower connection types. The implementation of a caching component can also provide the benefit of a more responsive user experience, as data can then be retrieved from the local cache rather than round-tripping to a server over a slow connection.



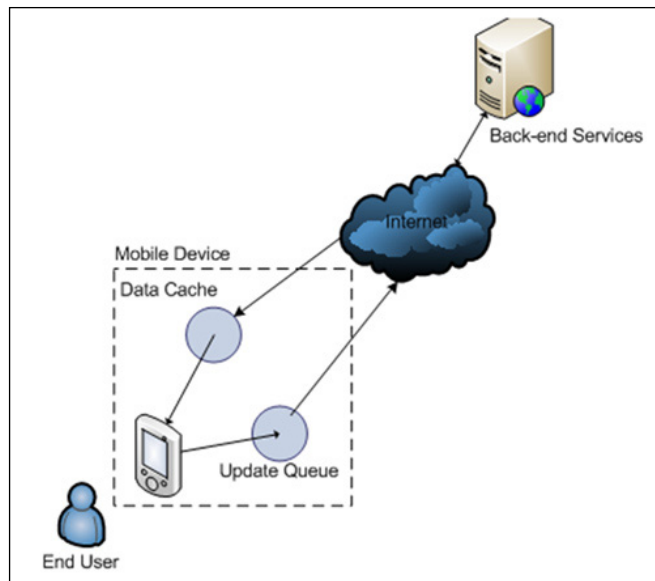
The queuing component handles the persistence of data to the back-end services. The queuing component can be designed to sit in front of the web service API client within the mobile application and check to see whether or not a connection is available when attempting to call the web services. If a data connection is unavailable, then the update is placed into a first-in, first-out queue in memory. The queue should then periodically check (in a background thread) to see if a connection is available and then send all data updates to the back-end services in the order in which they were received. The queue should also be designed with business logic around the reconciliation of data conflicts. For example, if a data update is sent to the server and is determined to be out of date or invalid, then the end user should be notified of the error and given a mechanism to correct or discard the update. Another feature that should be designed into the queue is the persistence of the queue to local data storage on the device; if the application is closed or interrupted, then the queued updates will be kept safe until the next time the application is used.

At right is a depiction of a mobile architecture using local caching and queuing services to provide offline data access and data synchronization.

As you can see, when designing an application that will be living “in the wild”, outside the corporate firewall, there are numerous challenges that simply don’t exist when building enterprise applications that run in well-known conditions, safe and secure within a corporate datacenter or colocation facility. Performance and usability will make or break the usage and acceptance of any mobile application. Now that users are used to the snappy and responsive interfaces of their modern iPhone, Android, and Windows Phone platforms, they will loathe using any application with a sluggish, unusable interface. Accessing data on a mobile device can be a whole new ball game for enterprise developers who haven’t worked with web services or have spent years writing and maintaining classic two-tier or mainframe-based applications. Security is rarely a concern for developers writing

applications that are safely tucked away behind a corporate firewall and intrusion-protection systems, but when exposing APIs with access to business-critical information to the public Internet, there is no way that security-through-obscurety will suffice any longer. Connectivity is especially challenging to design around on a mobile device that will commonly have a very slow connection or no connection at all – for an enterprise application running in a data center, on the other hand, it can usually be assumed there is a redundant, high-availability, high-bandwidth Internet connection available.

In summary, while it can be challenging, there are well known solutions for each of the previously mentioned issues. And though each mobile platform will have its own specific best practices for each area, many of the best practices are standard across all mobile platforms, regardless of the technology used.



West Monroe Partners is an international, full-service business and technology consulting firm focused on guiding organizations through projects that fundamentally transform their business. With the experience to create the most ambitious visions as well as the skills to implement the smallest details of our clients’ most critical projects, West Monroe Partners is a proven provider of growth and efficiency to large enterprises, as well as more nimble middle-market organizations. Our more than 300 consulting professionals drive better business results by harnessing our collective experience across a range of industries, serving clients out of offices across the United States and Canada.