ATLOGYS
Technical Consulting

Mobile Application Design and Architecture

Submitted by :
Automated Logical Software Pvt. Ltd.
R8, Nehru Enclave
New Delhi - 110019

MOBILE APPLICATION DESIGN AND ARCHITECTURE

From:
Atlogys Technical Consulting
Your Offshore 'CTO' Consultants
(Chief Technology Officer)
"Helping Innovate design and deployment of large-scale, distributed web and mobile platforms."

ATLOGYS ACADEMY

@atlogys

Latest trends and innovations in web and mobile technology

www.atlogys.com/resources

Guides, Tutorials, advice on best practices, standards, Do's and Don'ts.

blog.atlogys.com

Practical Notes on getting *Good* software development.

Contact Details:

Contact: (+91) 11 26475155	**E-mail:** ritika@atlogys.com	**Address:** R-8, Nehru Enclave,New Delhi – 110019, India

Table of Contents

How To -> MOBILE APPLICATION DESIGN?

A guide from Atlogys on how to go about making Great Mobile Apps.

What are the design considerations and what are the design best practices?

This is a “must follow” guide for any company getting into the space of mobile application development.

1. DESIGN CONSIDERATIONS FOR MOBILE APPLICATIONS

What are the software design considerations and aspects that are specific to the mobile domain of software development? What aspects must be thought through when making a mobile application? Along what all dimensions can mobile application development differ from regular web application development.

Try to answer these questions before getting into the design of your mobile application.

1.1 *USER REQUIREMENTS AND ORGANIZATIONAL CONSTRAINTS*

a. *Application Functionality Requirements* – which user interactions and features need to be supported by the application? Which mobile libraries, SDK's and API's need to be adopted?

b. *Hardware requirements of the application* – E.g. whether it requires access to global positioning system (GPS) or a camera or the user's address book or emails etc. This also impacts the hardware device that can be chosen for deployment of the application.

1.2 *WHAT IS RUNNING IN THE BACKEND*

The backend server technology and DB schema hugely determine the flexibility of the actions and functionality available on the client application. <u>E.g. whether metadata updates can be easily communicated to the client device of not, which data transfer protocol to use?</u>

1.3 *WHETHER TO BUILD A RICH CLIENT, A THIN WEB CLIENT, OR RICH INTERNET APPLICATION (RIA) ?*

a. *Rich Client* – used if application requires local processing and must work in an occasionally connected scenario. These are more complex to install and maintain.

b. *Thin client* – used if application can depend on server processing and will always be fully connected.

c. *RIA Client* – used if application requires a rich user interface (UI), limited access to local resources, and must be portable to other platforms.

1.4 *WHAT ALL DEVICE TYPES TO SUPPORT*

Hardware and software platforms should be chosen on basis of how their speed, power consumption, memory and CPU resources compare with the application's requirements (and not just on basis of the ease of programming).For each device type, look at resource constraints – storage memory, CPU, process restrictions, UI – screen size, resolution (DPI).

1.5 *SDK AND THE DEVELOPMENT TOOL ENVIRONMENTS AVAILABLE:*

a. *Original Company specific* - All major mobile platforms (like Android, iPhone, Windows) release company specific SDK's along with best practices and specifications. These should be followed and used to build the most comprehensive and good looking application for that platform.

b. *Cross Platform / Cross Hardware* - A lot of cross-hardware and cross platform frameworks are also now available and gaining popularity. Whether or not these can be used for the construction of your application depends on the functionality and the native hardware features it wishes to expose.

1.6 *MOBILE CONSTRAINTS* – <u>Security, Battery Life, Network Bandwidth, Limited CPU, Memory Storage capacity, Security (no corporate firewall but public internet), Performance and Authentication.</u> Out of these Battery Life is the most limiting aspect on the mobile

1.7 *PHYSICAL NUANCES IN HARDWARE* – E.g. presence of the back button on the Blackberry and Nokia phones vs. the home key and the back button on the screen on the iPhone.

1.8 *CONNECTIVITY* –fully connected, occasionally connected or limited-bandwidth scenarios need to be considered.

1.9 *USABILITY (USER INTERFACE)* -

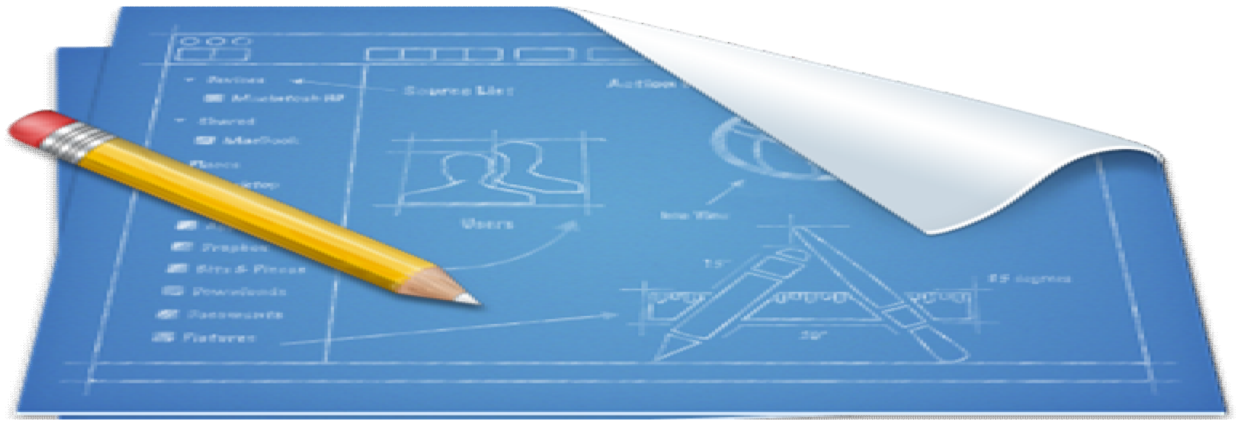
<u>"Mobile optimized experiences produce on average 75% higher rate of engagements per visit for mobile user."</u> - Source – Ed Hewett from Omniture

a. Has to be smaller, simpler and more concise for the mobile platform.

b. Should not take a long time to load, should consume little bandwidth.

c. The navigation, layout, interaction and flow should be appropriate for the small screen.

2. DESIGN PARADIGNMS AND BEST PRACTICES



What are the efficient design best practices, standards and methodologies that should be used to develop good mobile applications?

The following section lists the 'Design Principles/Best Practices' for tacking various aspects in mobile applications design like Authentication, Caching, Power, Security etc.

2.1 MULTI LAYERED ARCHITECTURE

Multi Layers improve reuse and maintainability of application. The aim should be to achieve the smallest footprint on the device by simplifying the design.

Layers typically include User interface, Business and Data Layers. Depending on the application type, multiple layers may be located on the device or the server. Business and Data layers should be pushed to the server until and unless the application requires a rich client interface.

2.2 AUTHENTICATION AND AUTHORIZATION

This is very important for the security and reliability of the application.

- Authenticate at all levels – between client and server, between server and database.
- Account for connectivity interruptions - Consider all possible connectivity scenarios (*over-the-air, cradled synchronization, Bluetooth, local SD card*).

- Account for the difference in security mechanisms available on larger platforms vs. mobile platform e.g. access control lists (ACLs) are not available in Windows Mobile.
- Authentication should be required for access by Bluetooth devices.
- Web Service API calls
  - Make sure while designing the web service layer that logic for authentication, authorization, validation is executed on server side of the call.
  - Perform token based authentication (LDAP) like Oauth or Session based (In RubyOnRails) to identify user and check permissions.

2.3 Performance Considerations

One must optimize the code for performance throughout the phase of mobile application development and not just towards the end.

- Design web services API and data formats to optimize for performance.
- Design configurable options to allow the maximum use of device capabilities – users should be able to turn off features to save CPU and power.
- When on battery power, try not to use a lot of performance enhancing features. Reduce usage of device CPU, wireless communications and screen.
- Use **lazy initialization** where-in possible:
  - Only retrieve data needed by the application.
  - Only retrieve data When the application needs it.
- Limit the number of objects created.
- Use minimum amount of memory – when memory is scarce, release cached intermediate language (IL) code to reduce memory footprint or return to interpreted mode, and thus slow overall execution.
- Follow platform specific best practices - each platform has different code-level best practices for performance optimization depending upon the programming language and frameworks available on the platform.

- Use programming shortcuts to deflate code size and memory consumption. e.g. avoid pure object-oriented practices like abstract base classes and repeated object encapsulation.
- Use CSS Sprites.
- Minimize application startup time:
  - Use Offline Technology like *AppCache – HTML5.*
  - Reduce code to be parsed at startup time – e.g. partition large JavaScript files and load them on-demand.
  - Store snapshot of last application state.
  - Minimize the number of Local Storage Queries before the first view of application can be displayed.
- Minimize perceived latency – use design paradigms like Incremental rendering, Prefetching.
- Support Push notifications (OMA push, native supported by OS) wherever possible.

2.4 COMMUNICATION

Device communication mechanisms include *Wireless communication (over the air), Wired communication with a host PC, Bluetooth, Infrared Data Association (IrDA).*

- Design asynchronous, threaded communication to improve usability in occasionally connected scenarios.
- Suspend and resume or even exit application if there is an incoming phone call.
- Use Web Services for communication – especially when is it required to access data from multiple sources, interoperate with other applications, or work while disconnected.
- Protect communication over untrusted connections, such as web services and other over-the-air methods. e.g. When using web services – use mechanism like WS-Secure.

2.5 Connectivity

Mobile devices at times have intermittent to no connections. The speeds of the connections vary widely along with the quality (**Edge, 3G or Wi-Fi**).

- Design of caching, session state management, and data-access should also depend on the strength and quality of the connection (intermittent or not available).
- Proper fall back mechanisms are needed in case the connectivity suddenly disappears.
- The application should inform user about network access because this drains battery life. It must disclose for how long it needs access, how heavy the access is, and how frequent this access is. (Either during sign-In or installation or by means of a background Icon).
- User should be able to control the network access made by the application - like disable access, adjust polling schedule, and limit activities that can initiate network requests.

2.6 Synchronization& Offline Access

Two basic components must be supported.

- *Queuing mechanism* - The queuing component handles the persistence of data to the back-end services.
  - Places updates in a first-in, first-out queue in memory when there is no connection. Checks queue periodically (in a background thread) for connection availability - sends all data updates to the back-end services in the order in which they were received.
  - Has business logic for reconciliation of data conflicts. <u>e.g. if a data update sent to the server is out of date or invalid, the end user is notified of the error and given a mechanism to correct or discard the update.</u>

    - The queue persists to the local data storage on the device - If the application is closed or interrupted, the queued updates are safe until the next time the application is used.

- *Caching Mechanism*
    - Keeps copies of data retrieved from a server side API call thereby providing offline access. Can also be designed to periodically (in a background thread) retrieve larger data sets needed by the user.
    - Data stored in the cache on the device generally expires after a certain time period.
    - Can use knowledge of the current type of connection to time data retrieval. (e.g. retrieve large sets only on Wi-Fi).
    - Provides for a more responsive user experience.

Other aspects to keep in mind:

- Support for 'Over the Air' Synchronization or Cradled Synchronization or both?
    - Over the air - if your users must synchronize data when away from the office.
    - Cradled - if users will be synchronizing with a host PC.
- Recovery – should be a way to recover if sync is reset.
- Synchronization conflicts must be managed.
- Account for security of data synced over the air?
- Connectivity issues
    - Cancel operations and allow them to resume gracefully when there is a connection interruption.
    - Consider store-and-forward synchronization using WCF rather than e-mail or SMS (text message), as WCF guarantees delivery and works well in a partially connected scenario.
- Bidirectional or Upload Only – Use Merge Replication.

2.7 User Interface

User experience is influenced by a number of factors like latency, interaction method, and data consistency.

Follow the user experience guideline released by each platform as much as possible (e.g. IPhone, Android, Windows 7) – This provides a consistent look and feel across all applications from that platform.

Other tips are as follows:

- Reduce application startup time to provide a better user experience (See performance section for tips and how-to's).
- There is limited screen size – only display what is really necessary, avoid scrolling.
- If application will run across a lot of devices, then design for all UI interaction methods like Focus, Touch and Pointer.
- Design for a single-window, full-screen UI.
- Use Wizard-like and/or Tabbed interface.
- Design for less typing – provide automatic data fills, scroll lists for dates.
- Allow input from various sources like stylus (pen based inputs), keypad, and touch.
- Size UI elements appropriately – use Large buttons and layout. (Recommended button minimum size is 44 pixels - high or wide).
- Positioning of controls like menu bars/buttons is important – e.g. A person's hand can block a touch-screen UI during input with a stylus or finger, place menu bars at the bottom of the screen and place expanding options upwards.
- Give the user visual indication of blocking operations; for example, an hourglass cursor.
- Enable deep links for bookmarking and browser history (as back buttons are cumbersome).

- **<u>TIPS:</u>** Phone-numbers should be (click to call), Enable automatic sign-in.
- Text orientation adjustment - use percentage and relative measures (instead of absolute or pixel measures) for containers so that text can re-flow automatically.
- Pages that are specifically designed for the target screen size should not be automatically scaled up and down by the mobile browsers – prevent this by using the meta viewport element.
- Use CSS Media Types for different media like print, screen, mobile.
- Use CSS media queries to apply specific style rules - based on screen width, orientation, or resolution.
- Design Non JS and non-Ajax versions using synchronous FORM posts in place of XHR requests.
- Account for Graphics
    - Design for the highest resolution available first – Then scale down because pixel density is a real issue.
    - Use vector graphics and shapes (groups) for easy scaling.

2.8 SECURITY & PRIVACY

Mobile devices contain a lot of personal data which must be protected.

- Always use web service API layer to access and connect to backend systems (vs. direct connections).
- Use SSL-secured connection for all communication between the mobile client and the web server. This prevents "man-in-the-middle" attacks.
- Do not store confidential data on the mobile device (but on the back-end server).
- Encrypt the data with a key that is not stored on the device. Mobile platform vendors now provide support for automatic encryption of disk storage.
- Do not execute unescaped or untrusted JSON data
- Ensure user is informed about the use of personal and device information when they first access the application on the mobile.

- Provide user the capability to control application behavior like access to network and device data (address book, call history, calendar etc.).

The Complete Detailed Report..
(Want The Full Exhaustive List?)

The Full Version of this report continues to discuss the rest of the following 'design considerations' in much more detail. Find out more about:

- **Data Transfer**
- **Data Access and Bandwidth**
- **Supporting Multiple Hardware (Device Hardware Differences)**
- **Exception Management and Logging**
- **Testing**
- **Porting applications from Desktop to Mobile**
- **Power and Battery Life**
- **Application Installation on Device, Deployment and Upgrades**
- **Configuration Management**
- **Caching**

Request it by writing to:

Ritika Garga at: ritika@atlogys.com

**Source: Atlogys Insights and Experiences, W3C.org, Mobile Design Case Studies*

ABOUT: ATLOGYS TECHNICAL CONSULTING

Who are we?

A group of Ex-Google, Ex-Carnegie Mellon and Ex-IIT computer scientists who are super passionate about technology.

What We Do?

We act as your CTO (Chief Technology Officer) to take full ownership of your technology.

Expert IT Consulting
+
Managed Offshore Development

Atlogys is an IT consulting company comprising of extremely passionate and technically sound computer scientists and software engineers from Google, Carnegie Mellon and IIT.

We believe that the process of offshore software development is inefficient and expensive for the end customer. ***Our goal is to enhance this process by making it TURNKEY, RELIABLE AND HASSLE-FREE.***

We do this by working for you as your CTO.

We provide detailed and personalized techno-business strategy on your product → We do detailed architecture design for your application → We manage the complete offshore lifecycle and delivery of your software project with your development vendor →We Offer an extra Engineering Eye for Detail to ensure your product is 360 degree complete with respect to performance, security, scalability.

Engage us on your next software masterpiece … so you can focus on your business while we handle all of your software!