Nalys
Sense & Passion

Mobile Front-end Development

Last update : 10/03/2016

1 Introduction

The mobile world contains plenty of platforms, technologies and technical solutions. As the mobile devices haven't been as broad as now, the presence of a company on several platforms has become a very strong requirement. This paper presents a summary of the existing solutions now, and why you should choose one instead of another, regarding your needs and what workforce your company owns.

This white paper was written by Nalys SPRL employees: Dimitri Dujardin, a senior Java Software Engineer and Maxime Denis, a junior Java Software Engineer who also worked on responsive design solutions in the medical scope and on Android Development.

Some criteria are ranked regarding the following table:

++	Very good
+	Good
+-	Not good and not bad
-	Bad
--	Very bad

2 Mobile development approaches

Three main approaches are considered in this paper. The first one, the native development, consists in writing an app especially for a dedicated platform. The second one, the unified frameworks, will give techniques to build an app on several platforms with high reusable parts or single development. The last part will focus on the web applications adapted to mobile world with the responsive design.

2.1 Native development

Native development consists in producing an app dedicated to a specific platform. The app will use strong platform features and will be optimized for the targeted platform.

Pro	Cons
More efficient	Specific development for each platform (app porting exists)
Better UI Integration	More costs (specific platforms tools)
Complex hardware usage (sensors…)	
Offline possibilities	

Table 1 Pro and cons for native development

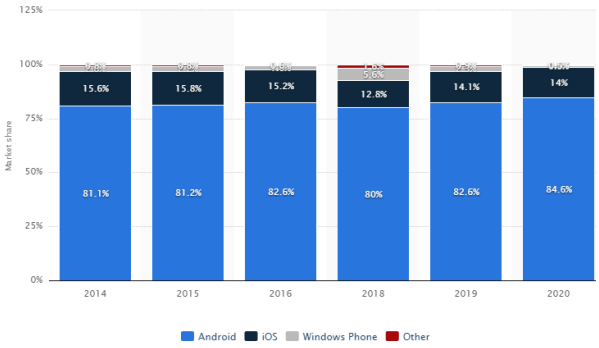

Figure 2 Market shares of the platforms projections, © Statista 2016

2.1.1 Native Android

Programming languages	**Java** (and C/C++ for specific parts using NDK)

Tools	Eclipse, Android Studio, IntelliJ…
Skills availability	++
Investment	+- (Specific testing devices)
To iOS?	J2ObjC, Multi-OS engine… may be complex
To Windows Mobile?	-- (Project Astoria abandoned)
Fragmentation	-- (lots of platforms, OS versions, backward compatibility issues)
Features	++
Look and feel	++

2.1.2 Native iOS

Languages	**Swift, Objective-C**, Object Pascal
Tools	XCode
Skills availability	+-
Investment	-- (Specific development and testing devices)
To Android?	-- (recoding)
To Windows Mobile?	+ (Project Islandwood)
Fragmentation	++
Features?	++
Look and feel	++

2.1.3 Other native platforms

Other native platforms present the same drawbacks and the same benefits, but with less shares (Blackberry, Windows Phone/Mobile).

2.1.4 Why target native platforms?

- When you have strong knowledge in Android or iOS,
- When you only target a single platform and requires strong features
- When strong performance or feature are required (rendering, sensors…)
- When it is required from the customer (stability, security)
- When offline capabilities are required

2.2 Reusable solutions and unified frameworks

This part presents ways to build an app without duplicating the effort for each platform. Techniques exists to produce reusable parts or even to work once and then publish on each platform.

2.2.1 Microsoft Visual Studio Cross Platform

Open source?	No
Base language	**C++, Java (Android), Swift or Objective C (iOS), C# (Windows)**
Features	C++ library for the backend app and specific platform language for the client
Performance	++ (native)
Look and feel	++
Platforms	iOS, Android, Windows Phone
License fee	+- (up to 250$ per month)

Visual Studio contains a cross platform solution to write a C++ core for the mobile application, containing the backend features, and then just a wrapper and a UI written in the platform specifications. This way guarantees performance and look and feel and allows to keep the core code for all the platforms.

2.2.2 Microsoft Xamarin (C#)

Open source?	No
Base language	**C#**
Features	C# applications compiled in native applications.
Performance	+ (games development)
Look and feel	+ (Closer to the OS, animations)
Platforms	iOS, Android, Windows Phone
License fee	- (up 170$ per month)

Xamarin provides an unified way to build an app. The app is written in C# completely. It can be done using Visual Studio, which includes it. The app need no change to be used on iOS or Android.

2.2.3 Qt (C++ / C#)

Open source?	Possible without commercial use
Base language	C++, C#
Features	Create Qt app running on desktop or mobile platform by coding once.
Performance	No data available
Look and feel	No data available
Platforms	iOS, Android, Windows Phone
License fee	+- (up to 350$ per developer per month)

Qt's approach is to provide a single app for multiple platforms. Qt comes with a specific IDE (but can be used with Visual Studio as well), and relies on a specific language for the UI. The backend of the app can be written in C++ or C#.

2.2.4 Adobe PhoneGap (Web)

Open source?	Yes
Base language	Web (HTML/CSS/JS)
Features	Possibility to access phone's features (camera, contacts, GPS…) through Apache Cordova.
Performance	-
Look and feel	- (Close to web)
Platforms	iOS, Android, Windows Phone
License fee	++ (free)

PhoneGap allows to write an app using web technologies, but allows to access specific platform features (camera, sensors…). Hence the UI will be less integrated and less fluid.

2.2.5 Why using frameworks?

- When you own knowledge of the language of one specific framework,
- When your mobile application need specific features / look that the web cannot offer but still has to be on several platforms
- When the application need performance but it is not the top priority
- When the app may need offline content

2.3 WEB DEVELOPMENT

Web development consists in the development or the adaption of a web based client, using mainly responsive design principles and JavaScript. A standard web site can be adapted using responsive design techniques to be mobile compliant. The features will stay the same on all the platforms.

2.3.1 Responsive Design

Responsive design is a good practice that ensures a website to stay usable and fully functional even on a small screen (tablet or smartphone). The website must always suit the graphical chart of the desktop version. This means that only a single (more careful) development can produce an app mobile and desktop compliant.

Figure 3 Responsive Design concept illustrated with water

Figure 4 Responsive design (Mashable)

Programming languages	JavaScript, CSS3, HTML5
Tools	WebStorm, IntelliJ, Visual Studio, Eclipse…
Workforce	++
License fee	++
Compatibility	Mobile/Desktop compatible (except old versions: IE8-)

Fragmentation	+- (browsers)
Features	+- to – (regarding the mobile device used)
Performance	-
Look and feel	-

2.3.2 Encapsulate web applications into app containers

The web client can be encapsulated into a specific mobile container (an app) that can be installed on the various mobile platforms through their store. This allows to create an app which is in fact a shortcut to a web site.

2.3.3 Why using web mobile development?

- When you already own a strong web application
- When you own web knowledge,
- When your application does not have to face strong requirements (simple application, no specific need)
- When your applications require constant connectivity

3 Conclusion

Choosing a mobile development technique can be considered following three point of views.

Features and look and feel goals are mainly achieved by the native development or the cross platforms solutions, while easily porting apps and less investments can be achieved by using unified frameworks or web responsive design development.

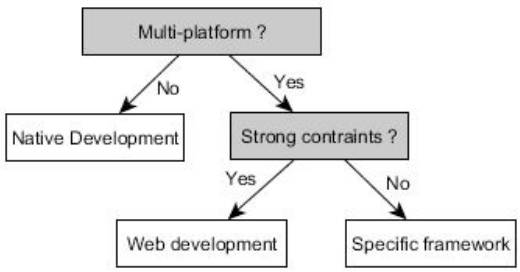

Figure 5 Decision process

4 References

1. Mobile application development Wikipedia - https://en.wikipedia.org/wiki/Mobile_application_development (3/03/16)
2. Android development VS iOS - http://techcrunch.com/2013/11/16/the-state-of-the-art/ (3/03/16)
3. Porting iOS to Android - http://www.mobilephonedevelopment.com/porting-ios-to-android (3/03/16)
4. J2objc - http://google-opensource.blogspot.be/2012/09/j2objc-java-to-ios-objective-c.html (3/03/16)
5. Multi-OS Engine, Intel Blog - https://software.intel.com/en-us/blogs/2015/07/30/power-your-mobile-app-development-with-the-multi-os-engine-of-intel-inde (3/03/16)
6. Multi-OS Engine - https://software.intel.com/en-us/multi-os-engine (3/03/16)
7. Astoria Project - http://www.windowscentral.com/microsoft-shows-how-easy-it-will-be-port-android-apps-windows-10-new-video (3/03/16)
8. Phonegap - http://blogs.interknowlogy.com/2012/01/26/how-to-use-phonegap-to-port-quickly-your-web-app-to-a-native-ios-and-android-device/ (3/03/16)
9. Qt - http://www.qt.io/ (10/03/16)
10. Microsoft Visual Studio Cross Platform Mobile development - https://www.visualstudio.com/en-us/features/mobile-app-development-vs.aspx (10/03/16)
11. Xamarin - https://xamarin.com/ (10/03/16)
12. Responsive Design - http://alistapart.com/article/responsive-web-design (10/03/16)
13. iOS SDK - https://developer.apple.com/ios/download/ (10/03/16)

14. Android SDK embedded in Android Studio - http://developer.android.com/sdk/index.html (10/03/16)
15. Android NDK - http://developer.android.com/ndk/index.html (10/03/16)
16. Universal Windows Platform (UWP) - https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide (10/03/16)

5 AUTHORS

- Maxime DENIS – mdenis@nalys-group.com
- Dimitri DUJARDIN – mdujardin@nalys-group.com
- Mojez PUNJWANI - mpunjwani@nalys-group.com