LUNATECH RESEARCH

Native Cross-platform Mobile Application Development

by
W. de Kraker (0815283)

CMI-Program *Informatics* – Rotterdam University

June 25, 2012

First supervisor	*Dhr. Y. S. Tjang*
Second supervisor	*Dhr. A. Chamani*

Abstract

Nowadays mobile devices are vastly integrated into modern society. They bring us one step closer to satisfy our ever growing need to have information available anytime, anywhere. To help gain access to information on mobile devices we use mobile applications, so called *apps*.

However, the fragmented nature of today's mobile ecosystem poses a challenge for developers to apps which are suitable to run on all mobile devices, since there is no de facto standard in *cross-platform* app development.

Currently there are several solutions available to solve the cross-platform paradigm.

Lunatech, having expressed its interest in mobile app development, would like to know which of these solutions, *if any*, suits Lunatechs needs. A study has been setup in order to resolve this question. The results of which are laid out in this thesis.

Contents

Background

Lunatech Research B.V

Lunatech provides application development services, completely based on open-source web and Java technologies and open standards. They are early adopters of new technology, and use cutting-edge frameworks and tools. To stay up-to-date, their developers have the opportunity to research, try new technologies and contribute to open-source projects. The company consists mainly of software developers. Everyone (except the director) writes code, on top of which some staff have a secondary management role, and the staff who will deliver a project interact with the customer directly.

Rotterdam University of Applied Sciences (Hogeschool Rotterdam)

Rotterdam University is one of the major Universities of Applied Sciences in the Netherlands. Currently almost 30,000 students are working on their professional future at the university. The university is divided into eleven schools, offering more than 80 graduate and undergraduate programmes in seven fields: art, technology, media and information technology, health, behaviour and society, engineering, education, and of course, business.[2]

Student no. 0815283

Student no. 0815283 is a 4th year computer sceince student at the Rotterdam University of Applied Sciences. Student no. 0815283 has a passion for things that are open source or have a sexy Apple logo on it. With over 2 years (professional) experience in mobile development he is the man for the job.

Introduction

Problem statement

Lunatech has demand for the development of cross-platform mobile applications. Currently these applications are bening developed using webtechnologies such as HTML5 and Javascript. A mobile application developed this way is refered to as webapp because it runs in a browserbased environment and is often hosted at a webserver rather than downloaded to the device itself.

The problem with webapps is that they lack in user experience. This is mainly due manner in which user interface components are build in HTML. Every platform has its own set of recognizable elements, but these cannot be accessed from within the browser environment. As a result of this the app will feel dislocated to the user because it's style doesn't match the rest of the platform. It tries to look and feels native, but never gets arround the fact that it's a webapp.

The direct alternative to webapps are native apps, native are writting using technologies proprietary to each platform, hench the term 'native'. What these applications lose in terms of cross-platform support they make up in terms of user experience. A native apps has acces to all the platforms propietary libraries and can rely on the user interface elements provided through these libraries.

Lunatech would like to know how to make use of the look-and-feel from native apps with the cross-platform support of webapps.

Research questions

Main research question:

- *How to develop a cross-platform mobile application while retaining the native look-and-feel?*

Sub research questions:

- *Is it viable to implement a custom solution to cross-platform mobile application development?*
- *Which solutions to cross-platform mobile application development currently exist?*
- *Which of these solutions offer the defined native look-and-feel?*
- *Are these solutions viable for commerical useage?*

Research method

This paragraph will describe the method used during the research.

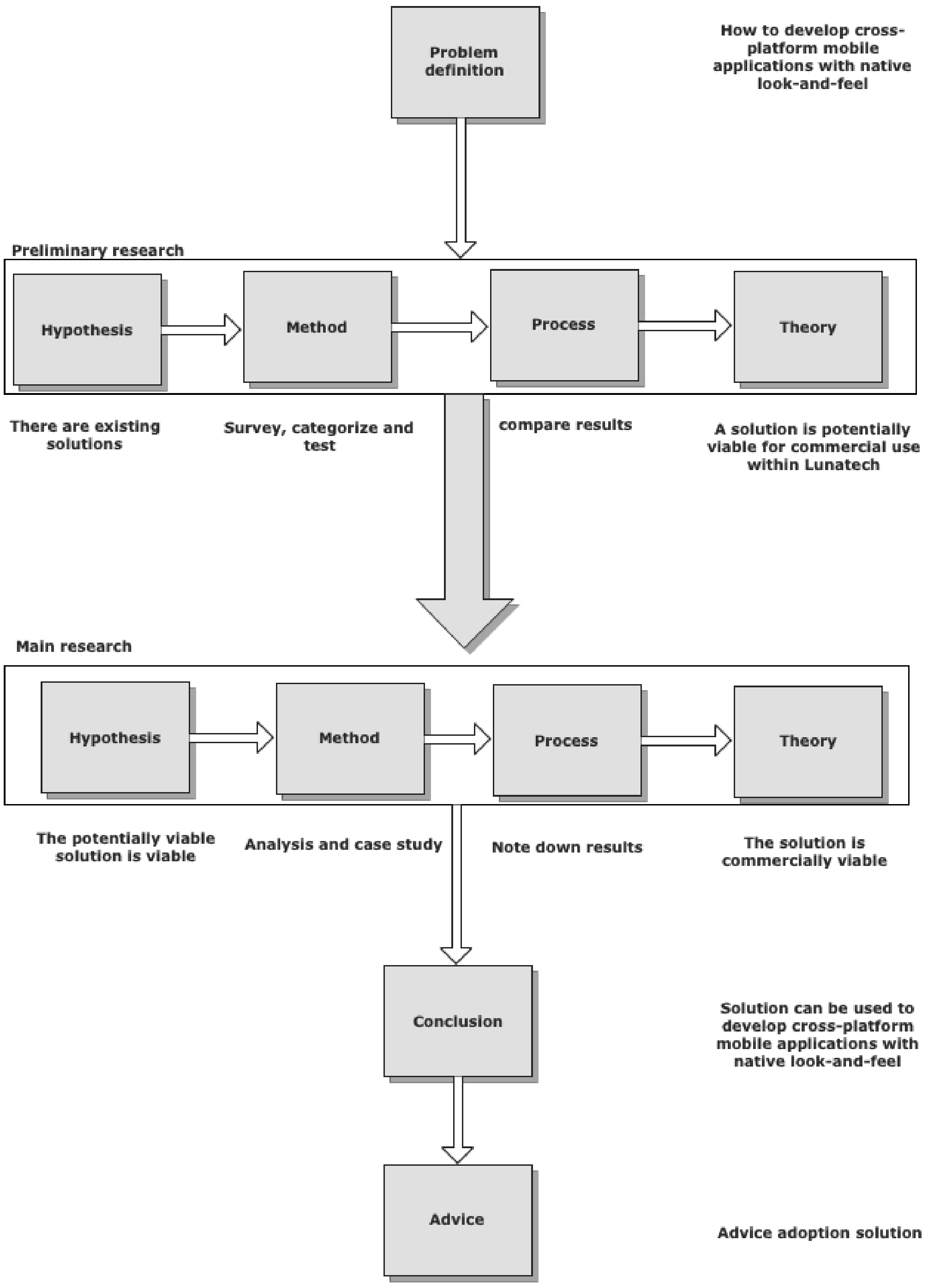
First the context of the main research question needs to be determined by defining the concepts *native look-and-feel* and *cross-platform*. The *native look-and-feel* is defined by the a set of criteria related to the user experience while *cross-platform* is defined by determining which platforms should be targetted.

Once the concepts in main research question are defined it is needed to research which solutions which fit within the context. During a preliminary research a market survey will have to be performed in order to deterime which solutions to cross-platform app development are available on todays market. Each found solution will be given a closer look and categorized based on requirements provided by Lunatech. Once the categorization is complete the solutions will be filtered based on how far the requirements have been met.

A comparisson test will be performed in order to determine which solution meets is potentially viable for commercial use by Lunatech. This test will consist of a short analysis per solution based on by Lunatech criteria, and a practical part where a benchmark application will be built. The results will be compared and the solution which offers the most complete set of features will be deemed potentially viable for commercial use by Lunatech. At this point a decision has to be made whether to continue with the potentially viable existing solution or research a custom solution.

To determine if the chosen solution is viable for commercial use by Lunatech a case study will be performed based on a realistic scenario Lunatech might encounter. During this case study the solution will be analysed and evaluated.

The results of the case study and the analysis will participate to answering how to develop a cross-platform mobile application while retaining the native look-and-feel. In addition to answering the main research question an recommendation will be given to Lunatech regarding the possible adoptation of the solution for commercial usage.


Problem definition
How to develop cross-platform mobile applications with native look-and-feel
Preliminary research
Hypothesis
Method
Process
Theory
There are existing solutions
Survey, categorize and test
compare results
A solution is potentially viable for commercial use within Lunatech
Main research
Hypothesis
Method
Process
Theory
The potentially viable solution is viable
Analysis and case study
Note down results
The solution is commercially viable
Conclusion
Solution can be used to develop cross-platform mobile applications with native look-and-feel
Advice
Advice adoption solution

Research method diagram

Definitions

The goal of this chapter is to define the context of the main research question.

"How to develop a cross-platform mobile application while retaining the native look-and-feel?"

Accordingly this chapter will define:

- the scope of *cross-platform*
- the concept of the *native look-and-feel*

Cross-platform

This paragraph will define the scope of *cross-platform*.

Platforms

In todays market there exists a wide variaty of mobile platforms. In order to determine which platforms should be supported for mobile development the following criteria have been provided by Lunatech:

1. *Platform type*
   Only smartphones will be targetted for development
2. *Platform marketshare*
   The platform should have at least a 10% marketshare in the european continent
3. *Platform marketshare trend*
   The marketshare trend from the past 6 months should not depict a trend directed below the set threshold of 10% within the next 6 months

The platform type criteria is based on Lunatech's requirement to support smartphones. A smartphone can be defined as a smart phone is a next-generation, multifunctional cell phone that provides voice communication and text-messaging capabilities and facilitates data processing as well as enhanced wireless connectivity.[6]

The smartphone only criteria implies that it is not a requirement to support support tablets and other mobile devices. This has significant influence on the marketshare criteria because it eliminates marketshare statistics based on operatingsystems, instead specific version details to be be included.

Apple iOS

iOS is a proprietary mobile operating system, developed by Apple Inc. It was originally released in 2007 for the iPhone and iPod Touch. iOS also became the main operating system of the iPad and Apple TV.

4th generation Apple iPhone running iOS 4

Google Android

Android is a opensource mobile operating system, developed by the Open Handset Alliance, led by Google and other companies.[7]

Samsung Galaxy S2 running Android 2.3

BlackBerry OS

BlackBerry OS is a proprietary mobile operating system, developed by RIM*(Research In Motion)* for its line of BlackBerry mobile devices.

BlackBerry Bold 9900 running BlackBerry OS 7.1

Windows Phone 7

Windows Phone 7 is a mobile operating system developed by Microsoft as a succesor to its Windows Mobile platform.

Windows Phone 7

Other platforms

Java ME

Symbian

Conclusion

Native mobile applications

This paragraph will define the paradigm of *native look-and-feel*. A native application is an application inherent to the platform for which it was built using techniques proprietary to the platform. For example, an iOS application is native when written in Objective-C and an Android application is written in Java. Native applications are typically fast and can access the device's native API's.

The native look-and-feel

When written in the native framework for a platform a mobile application receives acces to the available public libraries of the platform. These libraries include the UIKit*(on iOS)* which provides the developer with a pre-fabricated set of user interface components. These can be seen as the building blocks for the graphical user interface on that platform. When used, the general style of the mobile application gains consitency to the overal user interface design of the platforms operating system. This gives an application its native look, which in turn participates to the *native feel*.

The *native feel* of a mobile application can be defined as the speed in which the userinterface elements, the responsiveness of user interface elements to touch events, and smoothness of the animation in which the user interface elements are moved. A native mobile application has the advantage to hardware acceleration. This means its code has been precompiled and directly executed by the device CPU, rather than having to be interpreted by the device's browser. As a result of this the user interface elements are rendered faster and it *feels* smooth.

Alternative mobile application types

Web applications

A mobile web application is an application developed with web technologies as JavaScript and HTML5 with CSS3. It is in fact nothing more than a website designed to fit on mobile devices, often they resemble the style of a native application rather than a traditional website. These applications are build with a JavaScript library to add support for scrolling and handling touch events. Touch events are handled via user interface elements provided by the library. Examples of these libraries include jQtouch, SenchaTouch

Hybrid applications

A hybrid application in mobile development refers to an application which use a native shell to wrapped around web app. There are generally two forms of native shells, the first is a *webview* and the second a native framework which exposes a javascript API to provide the web application access to otherwise native API's.

Webview-based hybrid applications

A webview-based hybrid application is a webbased mobile application wrapped in a webview. A webview is a view or element which acts like a browser would, e.g. it is able to render HTML and run javascript. It is readily available in the native libraries. The advantage of a webview-based hybrid application over an normal web application is that it can be published via the devices native application publishing platforms. e.g. a webview-based hybrid application targetted for the iPhone can be placed in the Apple appstore.

Worklight is an example of a framework which can be used to develop webviewbased hybrid applications.

Framework hybrid applications

A Framework based hybrid application is a webviewbased application build upon a framework which provides an API to allow the application access to otherwise native API's. The framework is written in the platforms native programming language making it possible to access the native API, such as reading contact list, composing of SMSes, full access to the location API, etc.

PhoneGap is an example of a framework which can be used to develop mixed hybrid applications.

Preliminary research

The goal of the preliminary research is to determine whether or not an existing solution offers cross-platform mobile application development as defined by Lunatechs' criteria.

Introduction

In todays industry there exist several cross-platform mobile application development frameworks which offer a solution to cross-platform problem. All of these frameworks provide a solution for crossing the bridge between platforms. In order determine which one should be adopted by Lunatech for mobile development the following criteria have been determined for comparisson:

1. *Platform support*
   Which platforms and their versions are supported by the framework.
2. *Native UI support*
   Whether or not native user interface elements are supported for each supported platform.
3. *Programming language*
   Which programming language is used to develop using the framework.
4. *IDE* (Intergrated Development Environment)
   Which IDE can be used to develop using with the framework.
5. *License type*
   Which license types are available.
6. *Application type*
   Which type of mobile application is produced using this framework.

The cross-platform criterium is based on Lunatechs requirement to build mobile applications for the operating systems have at least a 10 percent marketshare in the European continent. Second comes the support for native user interface elements. Together these criteria form the essence of the main research question: *"How to develop a cross-platform mobile application while retaining the native look-and-feel?"* The remaining criteria are of secondary importance, they will provide more detailed means to compare the frameworks which offer native user interface support.

Framework requirements

A cross-platform mobile application development framework has to:

- Support cross-platform mobile application development.
  Build an application which runs on multiple platforms, originating from a single codebase.
- Supported platforms should be at minium iOS and Android.
  As decided in Chapter x. Definitions - *Mobile platforms*
- Be able to offer the native look-and-feel.
  As defined in Chapter x. Definitions - *Defining native*

Method

Selection

There are over 30 frameworks which offer cross-platform development of mobile applications[**?**]. These frameworks have been added to an initial list [1] This intial list contains only frameworks which adhere to the requirement of supporting cross-platform mobile application development.

To determine which frameworks should be included in the comparison we'll take a closer look at each and filter out those who don't adhere to criteria of supporting native user elements. This should leave us with less than x frameworks to compare. which is the goal because the timeframe of the internship doesn't allow for more.

Evaluation process

Estimated is that it takes 4 days to do experiments with a framework, this is based on N+1 in which N is the number of days it took to develop the benchmark app + 1 to familiarize with the framework. If in those 4 days I'm unable to complete the benchmark app, it is still a result. The experiment consumes a timespan of 32 hours per framework and consist of the following activities:

- Install framework
- Familiarize with how it works
- Rewrite the benchmark app in it
- Noting down results and documenting the experience

The remaining framework will be Evaluated on available of documentation, licensing, community, and flexibilityy. The research itself is included as an appendix.

[1] see appendix: *Existing solutions*

Results

From 30 existing solutions a total of 4 frameworks have been selected to get evaluated:

- Titanium
- RhoMobile
- MoSync
- Worklight

todo.

Titanium analysis

Introduction

This chapter will analyse how Titanium works and why it provides the desired native *look-and-feel*.

Inner workings

At runtime a mobile application developed with Titanium consists of three major components:

- The JavaScript sourcecode
- A platform-specific implementation of the Titanium API
- A JavaScript interpreter

During runtime the JavaScript sourcecode will be intergrated in a native class where it is encoded as a string and compiled. The implementation of the Titanium API done in a platform specific native programming language, Java for Android and Objective-C for iOS. The JavaScript interpreter evaluates the JavaScript code at runtime. Each platform has its own specific JavaScript Interpreter.

V8 is the default for Android but Rhino is also supported. V8 is has a better performance dealing as Rhino because it is directly intergrated to the NDK[2]. This means it does the code does not have to run trough the JVM[3]. Performance gain can exceed over 200% processing time when parsing a JSON object.[4]

For iOS JavaScriptCore is the choosen interpreter.

Runtime

At runtime a JavaScript execution environment set up in the native evironment this is where the application sourcecode is evaluated. Injected into JavaScript execution environment are so called *proxy* objects.

[2]Native Development Kit
[3]Java Virtual Machine

Proxy objects

A proxy object is an JavaScript object with a paired object in native code.[8] This means the object exists in both JavaScript and native code. Proxy objects gap the bridge between the native and the JavaScript environment. A global Titanium object in JavaScript exposes access to the proxy objects.

So, for example var label = Titanium.UI.createTabel(text: "label"); will invoke a native method which creates a native UILabel object.

var b = Ti.UI.createButton(title:'Title');, that will invoke a native method that will create a native UI object, and create a “proxy” object (b) which exposes properties and methods on the underlying native UI object to JavaScript. UI components (view proxies) can be arranged hierarchically to create complex user interfaces. Proxy objects which represent an interface to non-visual APIs (like filesystem I/O or database access) execute in native code, and synchronously (or asynchronously for APIs like network access) return a result to JavaScript.

Par example: In the JavaScript code, when a function is called on the global Titanium object to create a native UILabel a proxy object is created. #TODO: voorbeeld afmaken

JavaScript-object
```
var label = Titanium.UI.createTabel({
   text: "Lorem impsum",
   top: 10,
   left: 10,
   width: 100,
   height: 20
});
```

In iOS the proxy button object:

Native-object
```
-(UILabel*)label
{
    if (label==nil)
    {
        label = [[UILabel alloc] initWithFrame:CGRectZero];
        label.backgroundColor = [UIColor clearColor];
        label.numberOfLines = 0;
        [self addSubview:label];
    }
    return label;
}
```

JavaScript

CommonJS

Modules

Eclipse

Buildsystem

XCode CLI and the iOS SDK

Android SDK

Performance versus flexibility

tableview.

Conclusion

Case study

Introduction

Mobile application has been developed with Titanium to study its flexibility and features.

Stager

In 2011, live music venue WORM hired Lunatech to build *Stager*, a modern web-based resource planning and ticketing application to help manage live music events. Lunatech took the opportunity to use the relatively new Play framework to build a web application with an HTML5 and Java architecture. Stager has broad requirements ranging from high performance and security for the public ticket sales component to high usability for the internal resource planning component that will be used for hours a day by employees and being open to enhancements in the future for new customers. [**?**]

WORM

WORM is an institute for avantgardistic recreation Rotterdam, consisting of an artistscollective, a podium with a bar and Parallel University (DIY workshops for film, music and media). Born under the stars of punk, Dada, Fluxus, Situationism and futurism WORM is grown into a headstrong organization that the 'Do-It-Yourself' mentality of their ancestors, combined with ultra-pragmatism, love of technique (s) and proper accounting. Worm outputs film, radio, concerts, courses, partys, publications, performances, web projects, installations, workshops and an accumulation of tactile media and internet.WORM focuses (cheerful yet serious) in avantgarde, resource scarcity and opensource. [**?**]

Stager app

As described in the chapter *Background* Stager is an planning and ticketing application to help manage live music events. In addition to planning and ticketing Stager features an *atomfeed* to publish events. A Stager app would make use of this atomfeed to list any published events on a mobile device.

Stager application requirements

List of current and upcoming events events are downloaded in JSON format from the Stager event atomfeed at /web/feeds/events events are displayed in a row-based layout events are linked to their corresponding event detailview events in the list are sorted by date (asc) events in the list contain labels with event title, subtitle, date

Detailview of an event Shows detailed event information of an selected event: event title, subtitle, date, times (doors open, start, end), location details (venue name, street, number, city), event content (html rendered text)

Add event to agenda Prompts the user: "Add event 'x' on date 'y' in agenda?" Adds a selected event to the mobile devices agenda. Prompts the user of succes of add action.

Start gps-based navigation towards physical location of event Prompts the user "Navigate to y?" (y in the format of: streetname, housenumber, cityname, postalcode) Opens map application with address as argument.

View media attached to an event Media defined as: URL's to event images, videos, websites.

Display in a grid or list, categorize media types. Each displayed media item is resembled by a tumbnail or icon. When selected a media item opens to its content in: images an included webview, websites the device browser, for videos the youtube app or the browser(depending on video type & location).

Share event details to social media An event detail view will contain a 'share/deel' button. Prompts the user for platform to share. (twitter/facebook/email) Default value (editable): "I am attending event x on date y in location x !"

(Un)Register device to receive push notifications on new events of interest Register the device to receive pushed notifications about upcoming events which might be of interest to the user. Based on Relation.interest model in Stager.

Events

Notifications

Tickets

i18n

Mobile payment

Used techniques and methodologies

Javascript

For Titanium Javascript is the only option. Everything that can be written in JavaScript will eventually be written in JavaScript.

CommonJS

Playframework

Java

JSON

Conclusion

Conclusion and Recommendations

Project goals

Stager case study

Cross-platform Mobile Application Development using Titanium

Evaluation of Titanium

Limitations of Titanium

Future work

appendix I - Preliminary research

Appcelerator Titanium

Appceleator Titanium is an commericially supported opensource platform for developing cross-platform mobile applications. It was introduced by Appcelerator Inc in December 2008. Built upon the Eclipse IDE Titanium offers a Javascript API to native proxy classes which allow the developer to generate truely native cross-platform mobile applications.

Platform support

As of may 2012 Titanium supports iOS and Android. Next to building a native application for these platforms Titanium offers the option to generate a web application. Support for Research in Motion (BlackBerry) is in active (however closed from public) development. May first 2012 Appcelerator announced that it is extending its core value of cross-platform native application development beyond iOS and Android, on to RIM's BlackBerry devices.[1]

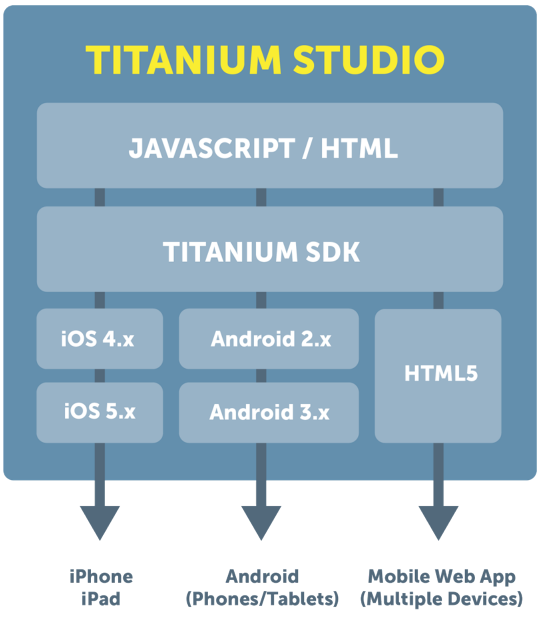
Techniques and tools

TitaniumStudio is an Eclipse based IDE with integration the propriatory mobile SDKs and simulators. For iOS this means Titanium requires Xcode with the iOS SDK to be installed, for Android the Android SDK and the Android AVD[4] are required.

Titanium applications are written in JavaScript but can be augmented with HTML & CSS. During runtime the JavaScript is evaluated and executed via so-called *proxy objects*.Proxy Objects are objects which are paired to a native object and can resemble native user interface elements.[5].

[4]AVD: Android Virtual Device (device simulator)

[5]Proxy objects are discussed in more detail in chapter x:Proxy objects #TODO



Titaniums bridge architecture[**?**]

Application type

As mentioned above, Titanium applications are written in JavaScript but can be augmented with HTML & CSS. The latter is in case a web application is required rather than a native application. This devices applications built with Titanium in two types:

- Webapplications
- Native applications

Philosophy

The goal of Titanium is to provide a high level, cross-platform JavaScript runtime and API for mobile development.[8] Titanium aims to help developers levarage their JavaScript knowledge to build native mobile apps that run across multiple platforms.

Results

Rhodes

Rhodes is an open source Ruby-based framework to build native applications for all major smartphone operating systems (iPhone, Android, RIM, Windows Mobile and Windows Phone 7). These are true native device applications (not mobile web applications) which work with synchronized local data and take advantage of device capabilities such as GPS, PIM contacts and calendar and the camera.

Platform support

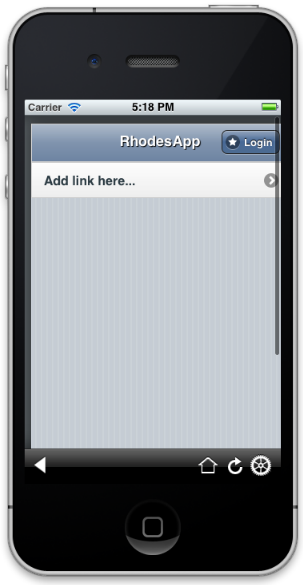
iOS, Android, BlackBerry, Symbian, Windows Mobile

Techniques and tools

Eclipse based studio, Ruby & HTML

Application type

Even though Rhodes advertises producing fully native apps[**?**], they do not. In fact all they do is provide the user with a framework, access to some navigational user interface controls, but the rest is a webview.



Rhodes sample iOS application: not fully native

Philosophy

Results

Worklight

Worklight Studio is an eclipse based IDE for the cross-platform development of mobile applications. Worklight Studio was introduced in 200x by Worklight Inc. In early 2012 Worklight Inc. became an IBM company. Worklight Studio offers mobile development trough the use of webtechnologies such as HTML5, and Javascript.

Worklight advertises the capability for developers to develop crossplatform HTML5, hybrid and native mobile applications.

Platform support

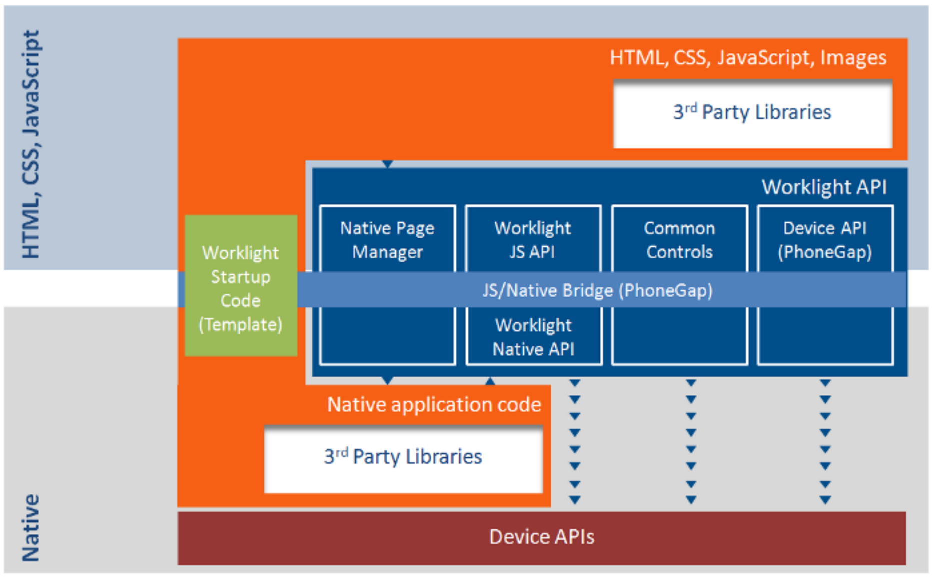
Worklight supports the following platforms: iOS, Android, BlackBerry and Windows Mobile 7.

Techniques and tools

As mentioned above, Worklight Studio is an eclipse based IDE. Allows the developer access to device APIs using native code or standard HTML5, CSS3 and JavaScript over a uniform PhoneGap bridge.

Application type

Applications developed with Worklight can be classied as *framework built hybrid applications*. As defined in the previous chapter this means the applications are based fitted inside webview and build on a framework which provides an API to allow the application access to otherwise native API's. The latter is achieved trough Worklights SDK while the former is made possible by an implementation of PhoneGap. [**?**]



Worklights bridge architecture[**?**]

Philosophy

Worklight aims to grant developers access trough HTML5 to the capabilities that mobile devices provide.

Results

MoSync

The MoSync mobile SDK offers cross-platform development trough the use of webtechnologie or C/C++.

Platform support

Techniques and tools

Application type

Philosophy

Results

Comparisson

Conclusion

Bibliography

[1] Jill Asher. Appcelerator Expand Native Mobile Application Support on RIM's BlackBerry Devices, 2012.

[2] Hogeschool Rotterdam. Rotterdam University, 2012.

[3] IBM-Worklight. Mobile application types, 2012.

[4] Tony Lukasavage. V8 Performance in 1.8.0.1, 2011.

[5] Netmarketshare. Mobile Top Operating System Share Trend, 2012.

[6] L.M. Ni. Spotlight: The Rise of the Smart Phone. *IEEE Distributed Systems Online*, 7(3):3–3, March 2006.

[7] Open Handset Alliance. Android Philosophy and Goals, 2012.

[8] Kevin Whinnery. Comparing Titanium and PhoneGap. 2012.

Evaluation