

# SMARTPHONE CROSS-PLATFORM FRAMEWORKS

A case study

Timo Paananen

Bachelor's Thesis  
April 2011

Degree Programme in Media Engineering  
School of Technology



JYVÄSKYLÄN AMMATTIKORKEAKOULU  
JAMK UNIVERSITY OF APPLIED SCIENCES



Author(s) PAANANEN, Timo	Type of publication Bachelor's Thesis	Date 27.04.2011
	Pages 108	Language English
	Confidential <input type="checkbox"/> Until	Permission for web publication <input checked="" type="checkbox"/>
Title SMARTPHONE CROSS-PLATFORM FRAMEWORKS – A CASE STUDY		
Degree Programme Media Engineering		
Tutor(s) MANNINEN, Pasi		
Assigned by AHOLA, Janne, Eagle Mediavision Oy		
Abstract <p>The objective of the Bachelor's thesis was to introduce and compare cross-platform mobile programming frameworks for smartphones, and to find the most optimal framework solution or solutions for the needs of Eagle Mediavision Oy.</p> <p>The thesis reviewed history of mobile communication, defined the major mobile device categories, and introduced the major smartphone platforms of 2011.</p> <p>The research focus was on cross-platform native frameworks. A cross-platform native framework is software that allows a common development approach across platforms but builds to an application indistinguishable by a user from one that is built with native code. The thesis introduced Appcelerator Titanium, PhoneGap, Rhodes Rhomobile, Adobe AIR for Mobile Devices, Adobe Flex SDK "Hero" with Adobe Flash Builder "Burrito" and OpenPlug Elips Studio frameworks. The frameworks were introduced theoretically, and tested practically by programming a demo application for each framework.</p> <p>Features of the frameworks were compared and analyzed. The cross-platform mobile frameworks were compared to native mobile frameworks and evaluated critically. The result of the comparison was that Appcelerator Titanium and PhoneGap were the best frameworks for the mobile development needs of Eagle Mediavision Oy. The result of this thesis was also that the cross-platform native frameworks were at an early stage of their evolution in 2011, and the native mobile programming still had its advantages. However, the future of the cross-platform mobile programming seems to have a great number of possibilities, especially in the area of the mobile web, but also in the cross-platform mobile programming.</p>		
Keywords mobile programming, smartphones, JavaScript, HTML, CSS		
Miscellaneous		



Tekijä(t) PAANANEN, Timo	Julkaisun laji Opinnäytetyö	Päivämäärä 27.04.2011
	Sivumäärä 108	Julkaisun kieli Englanti
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty ( X )
Työn nimi SMARTPHONE CROSS-PLATFORM FRAMEWORKS – A CASE STUDY		
Koulutusohjelma Mediatekniikan koulutusohjelma		
Työn ohjaaja(t) MANNINEN, Pasi		
Toimeksiantaja(t) AHOLA, Janne, Eagle Mediavision Oy		
Tiivistelmä <p>Opinnäytetyön tarkoituksena oli esitellä ja vertailla useiden älypuhelinlaitosten mobiiliohjelmointiympäristöjä ja löytää parhaat mobiiliohjelmointityökalut Eagle Mediavision Oy:n tarpeisiin.</p> <p>Opinnäytetyössä tarkasteltiin matkaviestinnän historiaa, määriteltiin keskeiset matkapuhelimien kategoriat ja esiteltiin pääasialliset vuoden 2011 älypuhelinlaitukset.</p> <p>Tutkimuksen pääpainopiste oli useiden alustojen mobiiliohjelmointiympäristöissä, joilla voidaan tuottaa natiiveja mobiilisovelluksia alustalle. Tällaisen ohjelmointiympäristön voi määrittää ympäristöksi, jolla voidaan tuottaa yhdellä kehitysympäristöllä ja ohjelmointikielellä mobiilisovelluksia, jotka eivät käyttäjän näkökulmasta eroa natiivilla ohjelmointikielellä tuotetuista mobiilisovelluksista. Opinnäytetyö tutki seuraavia ohjelmointiympäristöjä: Appcelerator Titanium, PhoneGap, Rhodes Rhomobile, Adobe AIR for Mobile Devices, Adobe Flex SDK "Hero" ja Adobe Flash Builder "Burrito" ja OpenPlug Elips Studio. Alustoja esiteltiin teoretiedon pohjalta, ja testattiin käytännössä ohjelmoimalla demosovellus jokaisessa esitellyssä ohjelmointiympäristössä.</p> <p>Ohjelmointiympäristöjen ominaisuuksia vertailtiin ja analysoitiin. Usean alustan ohjelmointiympäristöjä verrattiin natiiveihin mobiiliohjelmointialustoihin ja arvioitiin kriittisesti. Appcelerator Titanium ja PhoneGap olivat vertailun perusteella parhaiten Eagle Mediavision Oy:n tarpeisiin soveltuvat ohjelmointiympäristöt. Opinnäytetyön tulos päätelmä oli myös, että useiden alustojen mobiiliohjelmointiympäristöt olivat vuonna 2011 varhaisen kehityksen vaiheessa, ja natiivilla mobiiliohjelmoinnilla on etunsa. Kuitenkin useiden alustojen mobiiliohjelmointiympäristöjen tulevaisuus vaikuttaa vahvalta, erityisesti mobiiliwebin kehityksen myötä, mutta myös ohjelmointiympäristöissä.</p>		
Avainsanat (asiasanat) mobiiliohjelmointi, älypuhelimet, JavaScript, HTML, CSS		
Muut tiedot		

## CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>6</b>
1.1	Thesis Background.....	6
1.2	Eagle Mediavision Oy .....	7
1.3	Research objective .....	7
1.4	Methodology.....	8
<b>2</b>	<b>THE MAJOR SMARTPHONE PLATFORMS .....</b>	<b>9</b>
2.1	A brief history of mobile communication.....	9
2.2	Mobile device categories.....	10
2.3	Definition of a smartphone .....	12
2.4	Smartphone market share.....	13
2.5	Apple iPhone .....	15
2.6	Google Android .....	16
2.7	RIM Blackberry .....	17
2.8	Microsoft Windows Phone 7 .....	17
2.9	Palm webOS .....	18
2.10	Symbian.....	19
2.11	Meego .....	20
2.12	Tablets.....	21

<b>3</b>	<b>CROSS-PLATFORM NATIVE FRAMEWORKS .....</b>	<b>22</b>
<b>3.1</b>	<b>Definition of the cross-platform native frameworks .....</b>	<b>22</b>
<b>3.2</b>	<b>Titanium Mobile .....</b>	<b>22</b>
3.2.1	Getting started.....	24
3.2.2	Example Project: Contacts.....	24
3.2.3	Building for Device .....	29
<b>3.3</b>	<b>PhoneGap.....</b>	<b>30</b>
3.3.1	Getting started.....	31
3.3.2	Example Project: Contacts.....	34
3.3.3	PhoneGap Build .....	37
<b>3.4</b>	<b>Rhodes.....</b>	<b>38</b>
3.4.1	Development Architecture .....	39
3.4.2	Getting started.....	40
3.4.3	Example Project: Contacts.....	40
3.4.4	RhoHub .....	45
<b>3.5</b>	<b>Adobe AIR for Mobile Devices .....</b>	<b>46</b>
3.5.1	Adobe Flex SDK “Hero” framework .....	46
3.5.2	Flash Builder “Burrito” development environment.....	47
3.5.3	Getting started for Android .....	47
3.5.4	Getting started for iOS .....	48
3.5.5	Example Project: Google Maps API with Geolocation .....	48
<b>3.6</b>	<b>OpenPlug Elips Studio .....</b>	<b>53</b>

	3
3.6.1 Getting Started.....	54
3.6.2 Example Project: Contacts.....	55
<b>3.7 Summary .....</b>	<b>61</b>
3.7.1 Corona .....	61
3.7.2 MoSync.....	62
<b>4 COMPARISON OF THE CROSS-PLATFORM NATIVE FRAMEWORKS .....</b>	<b>63</b>
4.1 Smartphone platform support .....	64
4.2 Device APIs comparison .....	65
4.3 Other features .....	68
4.4 Subjective comparison .....	69
4.5 Results of the cross-platform native frameworks comparison.....	72
<b>5 CRITICAL ANALYSIS OF CROSS-PLATFORM MOBILE PROGRAMMING .....</b>	<b>74</b>
5.1 Disadvantages of the cross-platform mobile programming .....	74
5.2 Cross-platform versus native applications – a comparison .....	77
<b>6 RESULTS AND ANALYSIS .....</b>	<b>78</b>
6.1 Choosing the framework .....	78
6.2 Future of the cross-platform mobile programming.....	80
<b>REFERENCES .....</b>	<b>83</b>
<b>APPENDICES .....</b>	<b>90</b>
Appendix 1. Code listing: Contacts application: Titanium .....	90
Appendix 2. Code listing: Contacts application: PhoneGap .....	92

<b>Appendix 3. Code listing: Contacts application, Rhodes .....</b>	<b>94</b>
<b>Appendix 4. Code listing: Google Maps with Geolocation, Adobe Flash Builder “Burrito” .....</b>	<b>99</b>
<b>Appendix 5. Code listing: Contacts application, Elips Studio .....</b>	<b>102</b>

## FIGURES

FIGURE 1. North America smartphone OS market share, Q3 2010.....	14
FIGURE 2. Titanium Platform.....	23
FIGURE 3. New Project window in Titanium Developer .....	25
FIGURE 4. New Project preferences.....	26
FIGURE 5. The Titanium Mobile Project directory structure.....	27
FIGURE 6. Contacts application running on iOS device (left) and Android (right) .....	29
FIGURE 7. New PhoneGap-based application (iOS).....	32
FIGURE 8. New PhoneGap Android application .....	33
FIGURE 9. PhoneGap Contacts application running on iOS device (left) and Android (right).....	37
FIGURE 10. RhoSync and Rhodes application (RhoSync Features).....	39
FIGURE 11. Rhodes project directory structure .....	41
FIGURE 12. Rhodes Contacts, all Contacts list (iOS) .....	44
FIGURE 13. Rhodes Contacts, single contact view (iOS left, Android right) .....	45
FIGURE 14. New Flex Mobile Project 1 / 2 (Android) .....	49
FIGURE 15. New Flex Mobile Project 2 / 2 (Android) .....	50
FIGURE 16. Flex Mobile Project structure .....	50
FIGURE 17. The Google Maps Flex Mobile Project running on Android device.....	53

FIGURE 18. New Elips Project (Adobe Flash Builder 4) .....	56
FIGURE 19. Elips Project Structure (Adobe Flash Builder 4) .....	57
FIGURE 20. Elips filtered contacts list (Android).....	59
FIGURE 21. Elips Contacts, a single contact view (Android left, iPhone simulator right).....	60

## **TABLES**

TABLE 1. Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units) .....	13
TABLE 2. Comparison: Smartphone platform support .....	64
TABLE 3. Comparison: Device APIs support .....	65
TABLE 4. Comparison: Other features .....	68
TABLE 5. Results of the cross-platform native frameworks comparison.....	72
TABLE 6. Cross-platform versus native applications - a comparison.....	77



# 1 INTRODUCTION

## 1.1 Thesis Background

The smartphone is the new personal computer, the most personal device a person owns. Today's mobile phones are being used as computers for multipurpose functions. The number of mobile Internet users is growing rapidly and may overcome the desktop Internet users by 2013-2014 (Devitt, Meeker & Wu 2010, 8).

The mobile market is growing; however, at the same time the market of the mobile platforms is fragmented, as there are many mobile operating systems and programming languages. This has led to situation where mobile developers need to use platform specific tools and APIs in order to write mobile applications in different programming languages on different platforms (Allen, Graupera & Lundrigan 2011, xv). Reprogramming basically the same application into multiple mobile platforms means an increase in development costs and longer development times or decrease in the number of supported mobile platforms and a transfer of focus only on specific devices.

The demand for shorter mobile application development process has driven the need for cross-platform solutions. The idea of cross-platform mobile programming is that with the same codebase with little or no modifications a mobile application could be published into multiple mobile phone platforms.

If the mobile application could be basically programmed once and published on multiple platforms, it would cut down the development costs and time dramatically. And if the mobile applications could be programmed using the well-known Web-technologies, such as HTML, CSS, JavaScript and AJAX-technologies, the developers would not need to master multiple native mobile programming languages, such as Java, Objective C and C++. With some extra studies of the cross-platform frameworks

also web developers would be able to do mobile programming as well as native mobile developers.

On the other hand, the native mobile programming frameworks have their advantages, such as usually better debugging, documentation, supported hardware features and API. This thesis also makes critical research and comparison of different mobile phone programming methods, platforms, frameworks and tools.

## **1.2 Eagle Mediavision Oy**

This thesis was assigned by Eagle Mediavision Oy (later “Client”).

Eagle Mediavision Oy was founded in fall 2010 and is located in Jyväskylä, Finland. The company’s main business areas are in mobile and web development, video productions and printed media productions (book publishing and layout).

Early 2011 Eagle Mediavision Oy was starting its mobile business, and this thesis was assigned to find the optimal solution to the mobile development needs of the company.

## **1.3 Research objective**

The objective of the study was to introduce and compare different cross-platform mobile programming frameworks, and to find the most optimal solution for Cross-Platform mobile application development.

For The Client “the optimal solution” means meeting the following criteria:

- Widest range of mobile platforms supported with at least iPhone and Android -platforms supported

- Possibility to publish native, installable mobile applications that can be distributed via mobile application stores such as Apple's App Store and Android Market
- Phone API, an access to phone features like geolocation, phone, camera etc.
- Web-based programming language as a development language (not critical)
- The quality of framework documentation
- Fast development
- Development status (i.e. is under active development by firm business)
- Active developer community
- Open source or low license fees.

There can be many optimal solutions if founded reasonable.

The solutions were tested theoretically and practically: theoretically by studying all the features of the frameworks and practically by programming a demo application for each platform.

This thesis also made an analysis of cross-platform mobile development versus native mobile application development, i.e. which pros and cons there are when developing with the native frameworks or with the cross-platform frameworks.

## **1.4 Methodology**

The research consists of following parts:

1. Introduction of different smartphone platforms. Defines a smartphone and presents the main smartphone platforms of spring 2011.
2. Presentation of cross-platform native frameworks. Frameworks that are able to produce native, installable applications which can be downloaded from the mobile distribution platforms (i.e. Apple's App Store).

3. Theoretical comparison of different cross-platform mobile frameworks based on client's criteria. The theoretical comparison was made based on source material of different frameworks.
4. Analyzing the results and choosing the best framework for development. Possibility to have multiple framework decisions if found reasonable, for example possibility to develop to multiple platforms or usability to different kind of applications.

## **2 THE MAJOR SMARTPHONE PLATFORMS**

### **2.1 A brief history of mobile communication**

Commercially thinking, the history of mobile communication is brief. First real mobile phones in 1980s were heavy and large as they required enormous batteries to reach the nearest cellular network site. As late as in the 1990s cellular technologies were made financially feasible (Peltomäki 2010a, 16). Mobile phones evolved to be smaller, lighter and cheaper devices that would fit into everyone's budgets and pockets. One of the main reasons that made this evolution possible was the increased density of cellular sites. (Fling 2009, 5.)

These 2<sup>nd</sup> generation (or 2G) GSM mobile phones were mostly used for making voice calls and sending SMS messages. They did not have web browsers, and did not have any software installation possibilities (Firtman 2010, 6). All the software was factory installed.

In the early 2000s the use of the Internet was introduced to the mobile phones, as well as multimedia features such as listening to music and taking photos (Fling 2009, 7). GSM network providers added GPRS, which is a packet oriented data service, and

these two together are often described as 2.5G as it is a technology between the 2<sup>nd</sup> and 3<sup>rd</sup> generations (Peltomäki 2010b, 8).

Web reached the mobile phones; however, using the World Wide Web by mobile browsers was not a success because of the many limitations of the devices of that time and because of high prices. Network operator portals sold installable applications and downloadable ringtones and wallpapers. (Fling 2009, 8.)

When 3<sup>rd</sup> generation (3G) mobile networking became widespread, starting from 2002 and continuing to the end of the decade, the foundation was laid for smartphone's penetration to the market (Peltomäki 2010b, 10). The high speed Internet of the 3G made user experience of web browsing better and this made it possible to offer new services like streaming video or VoIP-calls. Mobile devices learned from desktop computing and became smarter; one could say "personal computers". (Fling 2009, 10.)

## **2.2 Mobile device categories**

Before defining a smartphone other mobile device types that are on the market today need to be defined. According to Firtman (2010), there are following types of mobile devices:

### **Mobile phones**

Mobile phones are phones with call and SMS support, but without web browsers, connectivity or software installation possibilities. One example phone of the category is Nokia 1100, that is, according to Firtman (2010), "the most widely distributed device in the world, with over 200 million sold since its launch on 2003." (Op. cit. p. 6.)

The problem with this type of very low-end mobile phones is that it is not possible for everyone to develop applications or web content for it. However, the situation is changing as even cheaper mobile devices have inbuilt browser support. (Op. cit. p. 6.)

### **Low-end mobile devices**

Low-end mobile devices have web support, basic camera and a music player, but there is no touch support and the memory is limited. Price is an advantage for mobile devices of this class. People who buy this kind of phones are usually not heavy Internet users. (Op. cit. p. 6.)

### **Mid-end mobile devices**

Mid-end devices usually offer a “medium-sized screen, basic HTML-browser support, sometimes 3G, a decent camera, a music player, games, and application support” (Op. cit. p. 7). Firtman (2009) states that one of the key features of mid-end devices is that they do not have a well-known operating system, as High-end and smartphones do. Usually even with the mid-end devices native applications are not available publicly; instead custom applications are run through a runtime, like JAVA ME. (Op. cit. p. 7.)

### **High-end mobile devices**

High-end devices are near the smartphone category, but according to Firtman (2009, 7) the differences are that high-end devices are generally non-multi-touch. Still they have many advanced features, as smartphones do, like accelerometer, camera, Bluetooth and a good web support. The major difference is still the worse user experience. (Op. cit. p. 7).

## 2.3 Definition of a smartphone

It is difficult to define a smartphone, since the definition of “smart” changes all the time as the mobile devices evolve. The mobile device that was considered as “smart” five years ago probably would not be in a smartphone category today. A typical smartphone today seems to be a top high-level phone in terms of price and features.

According to Firtman (2009, 8), a smartphone “has a multitasking operating system, a full desktop browser, Wireless LAN (WLAN, also known as Wi-Fi) and 3G connections, a music player...” Firtman (2009, 8) also mentions that a smartphone has several of the following features:

- GPS or A-GPS
- Digital compass
- Video-capable camera
- TV out
- Bluetooth
- Touch support
- 3D video acceleration
- Accelerometer.

Peltomäki (2010a, 5) adds a multitasking capable operating system to Firtman’s (2009, 8) definition. The PC Magazine Encyclopedia of IT Terms defines a smartphone similarly like mentioned sources, adding a smartphone’s ability to “run myriad applications, turning the once single-minded cell phone into a mobile computer.” (Definition of smartphone.)

In conclusion, in this thesis a smartphone means an advanced mobile phone with a modern day mobile operating system and advanced hardware features. It is possible to install 3<sup>rd</sup> party mobile applications to a smartphone and it has an advanced

browser and an advanced user interface with a touch screen. The thesis emphasizes the Google's Android and Apple's iOS over other smartphone platforms because of the dominant market position of the aforementioned vendors in Europe and the Nordic Countries.

## 2.4 Smartphone market share

There are many smartphone manufacturers on the market, with many mobile operating systems. If the maximum audience for our mobile applications were to be reached, it would be wise to focus on some main Operating Systems to reach the largest audience.

According to Gartner research (TABLE 1), Symbian, Android and Research In Motion are the three most sold operating systems for mobile devices in 2010. Remarkable notice in the Gartner research is that the Symbian market share is dropping while the Android market share is increasing rapidly. Gartner predicts that by the end of the year 2011 "Android will move to become the most popular operating system (OS) worldwide and will build on its strength to account for 49 percent of the smartphone market by 2012." (Pettey & Stevens 2011.)

TABLE 1. Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units) (Pettey & Stevens 2011)

<b>OS</b>	<b>2010</b>	<b>2011</b>	<b>2012</b>	<b>2015</b>
<b>Symbian</b>	<b>111,577</b>	<b>89,930</b>	<b>32,666</b>	<b>661</b>
Market Share (%)	37.6	19.2	5.2	0.1
<b>Android</b>	<b>67,225</b>	<b>179,873</b>	<b>310,088</b>	<b>539,318</b>
Market Share (%)	22.7	38.5	49.2	48.8
<b>Research In Motion</b>	<b>47,452</b>	<b>62,600</b>	<b>79,335</b>	<b>122,864</b>
Market Share (%)	16.0	13.4	12.6	11.1
<b>iOS</b>	<b>46,598</b>	<b>90,560</b>	<b>118,848</b>	<b>189,924</b>
Market Share (%)	15.7	19.4	18.9	17.2
<b>Microsoft</b>	<b>12,378</b>	<b>26,346</b>	<b>68,156</b>	<b>215,998</b>



Market Share (%)	4.2	5.6	10.8	19.5
<b>Other Operating Systems</b>	<b>11,417.4</b>	<b>18,392.3</b>	<b>21,383.7</b>	<b>36,133.9</b>
Market Share (%)	3.8	3.9	3.4	3.3
<b>Total Market</b>	<b>296,647</b>	<b>467,701</b>	<b>630,476</b>	<b>1,104,898</b>

Source: Gartner (April 2011)

A research in North America reinforces the idea of which mobile operating systems are the most successful today and which are gaining more market share and which ones are losing their share. The North American smartphone OS market share of the 3<sup>rd</sup> quarter of 2010 (Butler 2011, 5) shows that in North America Android has become a market leader (FIGURE 1). On the other hand, North America is only one market in the world and it has its own differences from other markets and Gartner's research (Petty et al. 2011) shows that Symbian still had a stronger market position worldwide in 2010. Still it shows the fact that modern smartphone operating systems, Android in particular, are growing fast.

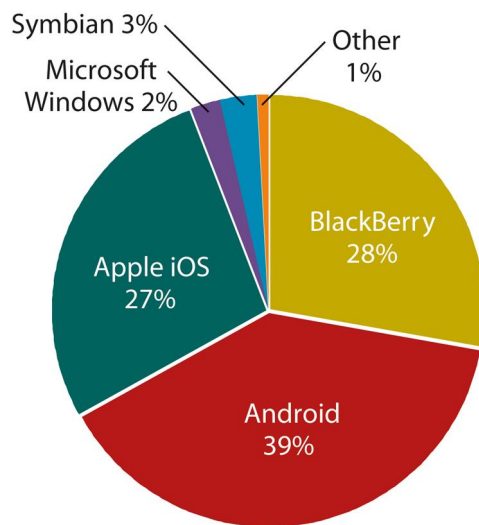


FIGURE 1. North America smartphone OS market share, Q3 2010 (Butler 2011, 5)

Based on the market share Android and iOS platforms are the platforms where the developing focus and effort should be put, according to Gartner's prediction of smartphones market share in the years to come. In particular, Android's market share is growing rapidly, while iOS will hold its positions. Symbian and Research in

Motion are losing their market share, therefore when thinking commercially these platforms may be ignored. Microsoft's mobile market share is growing; therefore the Windows Phone 7 should be taken seriously in the years to come. (Petty et al. 2011.)

## **2.5 Apple iPhone**

As Apple introduced the first iPhone in 2007 it started the new era in the smartphone market (Fling 2009, 10). By the end of the fourth fiscal quarter of 2010, Apple had sold over 70 million iPhones (Kumparak 2010).

In addition to the right business model and successful marketing, Apple had some factors that contributed to the success of the iPhone. According to a study by John Laugesen and Yufei Yuan (Laugesen & Yuan 2010), iPhone had a "rich mobile Internet browsing experience" and a broad range of installable applications, in particular entertainment applications, "clearly meeting the needs of their consumers" (Op. cit. p. 96).

### **Developing**

iPhone's operating system is called iOS (formerly iPhone OS) that is based on Mac OS X. Application development for the iPhone can be done on two platforms: using mobile web technologies, or using the native Cocoa Touch framework built on Objective-C. (Firtman 2010.)

In order to develop for the iPhone, an Intel-based Macintosh computer running a Snow Leopard version of the Mac OS X is required. The iPhone SDK available from the Apple Developers site includes the Xcode IDE, iPhone simulator and additional tools. It is also possible to develop for iPod Touch and iPad with the same tools. The

programs are built using the Apple's Xcode IDE and written in Objective-C language. (Allen, Graupera & Lundrigan 2010, 17.)

## **2.6 Google Android**

In 2005 Google started the development of the Android platform. In 2007 industry leaders such as Google, Motorola, Samsung, Sony Ericsson and Intel formed the Open Handset Alliance and its first key outcome was the Android Platform. Same year the first release of the Android SDK was released. (Hashimi, Komatineni & MacLean 2010, 3.) The first Android phone was released in 2008, and now there are hundreds of Android mobile devices, both phones and tablets, from several vendors. (Allen et al. 2010, 35.)

The Android mobile operating system supports several features, such as 2D and 3D graphics, common media formats, animated transitions and multi-touch input. The Android's web browser is the WebKit based and includes Google Chrome's JavaScript runtime. (Op. cit. p. 35.)

### **Developing**

The native Android applications are written in Java. Android offers its own virtual machine called the Dalvik VM. All the Java classes are recompiled into Dalvik and run on a Dalvik virtual machine. (Op. cit. p. 36.)

To start developing native applications for Android, the Java SE Development Kit (JDK), the Android SDK, and a development environment (IDE) are required. In order to make developing more comfortable, it is recommended to use Eclipse as an IDE and the Android Development Tools (ADT) Eclipse plug-in to compile and to run the Android emulator. (Hashimi et al. 2010, 25.)

## **2.7 RIM Blackberry**

BlackBerry is a product of Research In Motion (RIM). The first BlackBerry smartphone was released in 2002. According to Allen et al., “the BlackBerry has the second largest market share of the smartphones in the US”. (Allen et al. 2010, 51.)

Worldwide, the Research in Motion (RIM) was the third most sold mobile operating system in the world in 2010 (Petty et al. 2011). The BlackBerry is specialized into the enterprise market, and should be taken into account when developing applications for enterprise markets especially in the US (Allen et al. 2010, 51).

### **Developing**

The applications for BlackBerry can be developed using the BlackBerry Web Development or Java Application Development. The Web Development is for developing BlackBerry widgets, which are small web applications, built using HTML, CSS and JavaScript.

The Java Application Development requires installing the Sun JDK, the Eclipse IDE for Java Developers with the BlackBerry plug-in and the BlackBerry JDEs (Op. cit. p. 52-53).

## **2.8 Microsoft Windows Phone 7**

The original Windows Mobile was introduced at the millennium as an operating system for Pocket PCs. The last version of the Windows Mobile was released in 2009. (Udell 2009, 211.) In 2010, the market share of the Windows Mobile was 4.2 percent but in 2011 is growing to 5.6 percent as a result of the release of Windows Phone 7 (Petty et al. 2011).

According to the eWeek.com article (Kolakowski 2011), Microsoft has sold 1.5 million Windows Phone 7 handsets in six weeks after the release. In the first quarter of 2011, some new features are expected to the Windows Phone 7 platform, including “multitasking support, copy-and-paste and increased hardware support for augmented reality applications such as business card scanning.” (Op. cit.)

In February 2011, Nokia and Microsoft announced new partnership plans. The new strategy includes Windows Phone serving as Nokia's primary smartphone platform, and it will likely raise the Windows Phone 7's market share. (Nokia outlines new strategy, introduces new leadership, operational structure, 2011.)

### **Developing**

Microsoft has adapted its existing frameworks into Windows Phone 7 development. The applications are programmed using C# language with .NET framework. There are two major frameworks on top of the .NET framework core: Silverlight and XNA. Silverlight is for business applications and 2D graphics and it uses the Extensible Application Markup Language (XAML) while the XNA is for 3D graphics and games. (Lee & Chuvyrov 2010, 7.)

Developing for the Windows Phone 7 requires installing Windows Phone Developer Tools, which include “Visual Studio 2010 Express, the Windows Phone Emulator, XNA Game Studio 4.0, Microsoft Expression Blend for Windows Phone, Silverlight, and .NET Framework 4” (App Hub 2011).

## **2.9 Palm webOS**

Palm has been a pioneer already in 1995 launching PDA devices, which were successful in the 90s. Palm continued using its Palm OS (later Garnet OS), until in 2005 Palm started to manufacture devices with Windows Mobile. The Palm devices

did not do so well in the market; therefore Palm introduced in 2009 a new, web-orientated platform for its devices, the webOS. In 2010, HP, who said to continue the development of webOS, acquired Palm. It is expected that HP will announce devices with webOS in the following years. (Firtman 2010, 28-29.) On February 2011, HP announced three webOS products: a tablet, and two smartphones running webOS (Gupta 2011).

### **Developing**

The native applications for the webOS are created using web technologies (HTML, CSS and JavaScript), and the operating system and all the device applications are web-based. Plug-ins for the applications can be programmed using C and C++. (Op. cit. p. 418.)

The native webOS application is called a Mojo application, which is a JavaScript UI library. For developing the webOS applications, the SDK, the PDK (Plug-in Development Kit) and the Eclipse plug-in are available from the Palm developer site. (Op. cit. p. 418.)

## **2.10 Symbian**

The historical background of Symbian is in the EPOC operating system for the Psion family of PDAs. From the commercial point of view, Symbian is strongly linked to Nokia, which used the Symbian OS for the majority of its smartphones. (Udell 2009, 245.)

Originally the Symbian Company was formed by a group of manufacturers including Nokia, Ericsson and Motorola. Later Samsung and Sony Ericsson joined the Symbian Company. In 2008 Nokia bought Symbian Ltd. and created the Symbian Foundation to migrate the Symbian operating system to open source. (Firtman 2010, 20.)

In February 2011, Nokia and Microsoft announced new partnership plans. The new strategy includes Windows Phone serving as Nokia's primary smartphone platform. According to the press release (2011), Symbian will be a “franchise platform”. Nokia also expects to sell 150 million Symbian devices in years to come. (Nokia outlines new strategy, introduces new leadership, operational structure, 2011.)

According to Gartner research, Symbian was still the most sold platform of 2010 with its 37.6 percent market share. However, because Symbian was the primary platform of Nokia and will be changed to the Windows Phone 7, the market share of the Symbian is expected to drop to almost zero by 2015 as it is not actively developed anymore. (Petty et al. 2011.)

## **Developing**

Developing for the Symbian Foundation's operating system can be done using the native C++ framework, Java ME, Adobe Flash, web applications, widgets using web technologies, Python, and with Qt, a free C-based framework owned by Nokia. Qt is the recommended framework for creating native applications for Symbian and Meego. (Firtman 2010, 30.)

## **2.11 Meego**

The background of the Meego OS is in Nokia's Maemo and Intel's Moblin OS. Maemo was a Linux-based operating system for small netbooks and devices with full web browsing support. Example of the Maemo-powered device is Nokia N900. (Firtman 2010, 22.)

In February 2010, Nokia and Intel set plans to merge their Maemo and Moblin operating systems in order to create a unified and completely open source Linux-based platform that will “run on multiple hardware platforms across a wide range of

computing devices...and significantly increase opportunities for developers” (Grabham 2010).

However, according to Nokia’s press release (2011), Meego becomes an open-source, mobile operating system project. Nokia has plans to ship a Meego-related product in 2011. (Nokia outlines new strategy, introduces new leadership, operational structure, 2011.)

## **2.12 Tablets**

Although this thesis is focused on smartphones, it is reasoned to mention tablets, since the explosive growth of tablets and as they are expected to change media strategy, consumption and advertising "more than any new technology yet" (Look Ahead 2011).

The tablet can be defined as a mobile device with bigger screen than a smartphone has (at least five inches), a slate-style design, a touchscreen display, a Wi-Fi connectivity and internal components that include a CPU, RAM, and either on-board or MicroSD-based user storage (Strohmeyer & Perenson 2011, 78).

The major tablet operating systems are almost all the same like in the smartphone market; Apple’s iOS and Google’s Android in particular, but also HP’s WebOS tablet Windows-based tablets and RIM’s BlackBerry PlayBook (Op. cit. p. 76). Because the operating system is usually the same with smartphones, it is relatively simple to start developing also for the tablets. The major difference in developing for the smartphones versus tablets is not the code, but the layout that need to be re-designed.



## **3 CROSS-PLATFORM NATIVE FRAMEWORKS**

### **3.1 Definition of the cross-platform native frameworks**

A cross-platform native framework is software that “allows a common development approach across platforms but that build to an application that is indistinguishable by a user from one built with native code” (Allen et al. 2010, xvi). Cross-platform native frameworks let a developer to create a native, installable and mobile application distribution platform capable (i.e. App Store and Android Market) mobile software for multiple platforms using cross-platform APIs and usually HTML/CSS/JavaScript frameworks (Op. cit. p. 10).

However, building applications using native frameworks still requires the vendor SDKs installed and using the vendor-specific techniques for code signing and distributions (Op. cit. p. xvi). The layout and some API functions are also platform specific, so there will be some code modification for different platforms.

### **3.2 Titanium Mobile**

Titanium Mobile is a commercially supported, open source framework for creating native cross-platform applications using web technologies. Appcelerator Inc. introduced the framework in December 2008. (Allen et al. 2010, 153.)

The Titanium Platform is divided into two main products: Titanium Desktop and Titanium Mobile (see FIGURE 2). Titanium Mobile SDK provides the necessary tools, compilers and APIs for building for the target platform, and a visual environment tool called Titanium Developer for creating, running and packaging Titanium applications. (Allen et al. 2010, 153.)

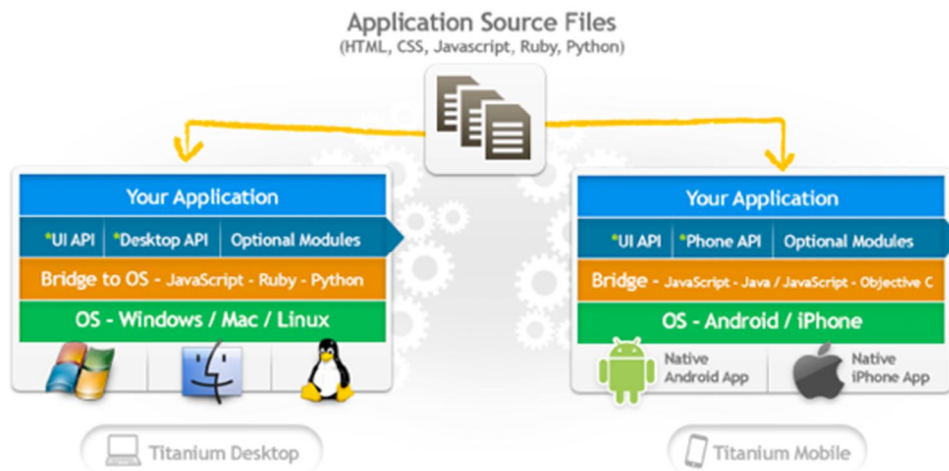


FIGURE 2. Titanium Platform (Getting started with Appcelerator 2011)

However, the Titanium Platform does not contain an IDE or a text editor, but this is subject to change as Appcelerator acquired IDE provider Aptana on January 2011. The new Appcelerator solution will bring features like debugging, code completion, integrated document, and editing tools. (Haynie 2011.)

On April 4, 2011, Appcelerator released the first preview version of Titanium Studio, an integrated IDE with Titanium SDK. It is built on top of Aptana Studio 3.0, and it includes new features such as debugging and code completion among older Titanium Developer features, such as possibility to run and deploy applications. (Muschenetz 2011.)

### Titanium Mobile SDK

The Titanium SDK allows creating, running and packaging native mobile applications for iOS, Android and BlackBerry (beta) devices using cross-platform JavaScript API. The applications are run against a standalone JavaScript engine that invokes native APIs. The Titanium Mobile SDK uses the native platform's SDK to combine the JavaScript source code via a JavaScript interpreter, and static assets of the application into an application binary. (Getting started with Appcelerator 2011.)

Titanium Mobile API uses native UI and platform APIs to access native UI components including navigation bars, menus, dialog boxes, alerts, and native device functionality including the file system, sound, network, and local database. (Allen et al. 2010; Getting started with Appcelerator 2011.)

### **Titanium Developer**

Titanium Developer is a desktop application for creating, running, managing and packaging Titanium Mobile or Desktop application projects. It also keeps Mobile and Desktop SDKs up to date. (Getting started with Appcelerator 2011.)

#### **3.2.1 Getting started**

The Titanium Platform (Titanium Developer and the SDKs) is available for Mac OS X Snow Leopard (iPhone and Android), Windows 7 and XP (Android only) and Ubuntu 9.10 (Android Only). Android and iOS SDKs (if one is running Mac OS X, otherwise only Android SDKs) need to be installed before installing the Titanium Platform. Detailed instructions for installing the Android and iPhone SDKS can be found in the Getting started guide from the Appcelerator website. (Op. cit. 2011.)

If the iPhone and Android SDKs are installed, the Titanium Developer can be downloaded and installed from the Appcelerator website at [www.appcelerator.com](http://www.appcelerator.com). After installing and launching the Titanium Developer, the program will automatically download the most current version of the Mobile and Desktop SDKs for the Titanium. (Op. cit. 2011.) It is also required to sign up for a free account on the Appcelerator Developer Center (Allen et al. 2010, 154).

#### **3.2.2 Example Project: Contacts**

In this chapter an example projects is created to demonstrate functionality of the framework and its API access to smartphone features. All smartphones have a built-

in PIM (Personal Information Management) Contacts application to store phone numbers and addresses. Smartphone platforms usually allow applications to access those contacts through APIs. (Allen et al. 2010, 103.) For this reason a simple example project is a Contacts application to show native PIM contacts using Titanium APIs.

In Titanium Developer, choosing the New Project from the top navigation opens a new project screen (FIGURE 3). The Project type needs to be Mobile for creating iPhone and Android project. In the new project window the application name, application id, project directory, company URL and Titanium SDK version used are also defined.

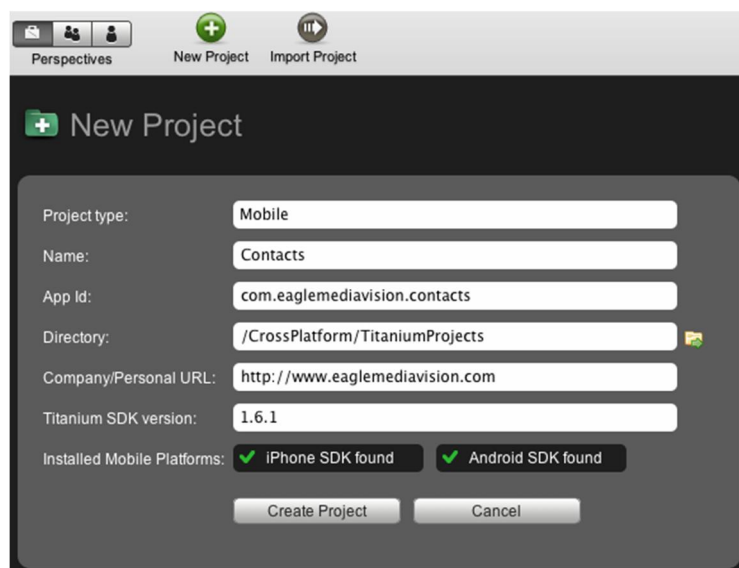


FIGURE 3. New Project window in Titanium Developer

In the next screen more additional information for the project is asked, such as application version control, description of the project, publisher's name and publisher's URL. It is also possible to change the default application icon for the project from the project preferences (FIGURE 4). After saving the changes the new project is ready and all the project files are generated to the directory specified.

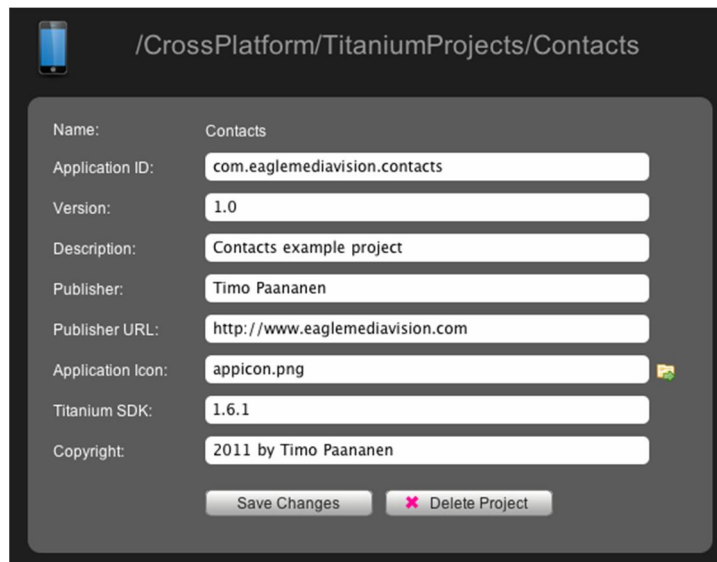


FIGURE 4. New Project preferences

As the Titanium Platform does not provide a text editor in version 1.6.0, the project files can be opened with any text editor that a developer is familiar with. A good choice is Aptana, as Appcelerator acquired it on January 2011, and will be obviously used in future releases of Appcelerator Titanium (Haynie 2011). Aptana is based on Eclipse; therefore the user interface is familiar to many developers.

All projects have similar directory structure: (FIGURE 5)

- A *Build* directory. Phone-specific native project files and resources. Titanium SDK build scripts dynamically generate this directory.
- A *Resources* directory. Contains all application source code in JavaScript and other files such as HTML and image files. In the *Resources* directory, *app.js* serves as the application's entry point, and the root execution context. All the coding starts from the *app.js*.
- A *Tiapp.xml* file for static configuration of the application.
- Other files include a change log file to keep a track of version changes, a License file to specify the application's license and optional *i18n* directory for creating localization resources.

(The Application Project Structure 2011.)

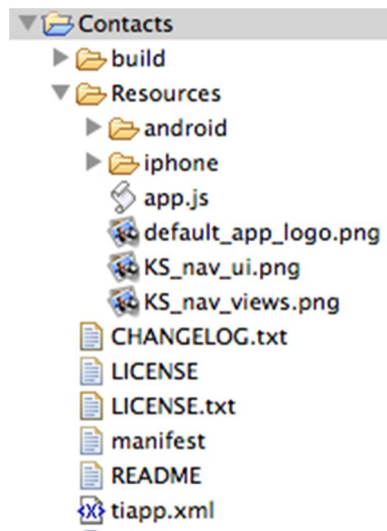


FIGURE 5. The Titanium Mobile Project directory structure

Programming the Contacts application begins with editing *app.js* as in all applications made with Titanium. Since the application is meant to be a simple contact picker application that has access to smartphone's contacts, *app.js* contains only basic UI settings and a window creation that points to external JavaScript file, *contacts.js*, where the actual application functionality is.

```
Titanium.UI.setBackgroundColor('#fff');

var win = Titanium.UI.createWindow({
    title:'Contacts',
    exitOnClose: true,
    url:'contacts.js'
});
win.open();
```

It is reasonable to put views, windows or application logic into separate JavaScript files for easier project control. For this reason the *contacts.js* file is created to hold the contacts picker functionality. If the application should be extended later, it will be easier to create application navigation to *app.js* and have the contacts picker functionality as one of the application's views.

Full code listing of the contacts application is in APPENDIX 1. The application code explained:

1. Create a button that opens a native contacts picker application (PIM).
2. Attach a click event listener function to listen when user taps the button and show the native contacts picker to the user.
3. When user selects a contact, return to the application and show some basic information of the selected contact such as picture, name, phone number and address information.

The Titanium function to show the contacts picker is

*Titanium.Contacts.showContacts()* that takes a *selectedPerson* as an argument and runs a function defined after user selects a person. Returned user data is an object of *Titanium.Contacts.Person* that can be accessed in this case with *event*.

The example sets to the contact info label text to show the selected person's full name. Some information in contacts can be multi-value, for example a person can have more than one phone number. If all phone numbers of the person would need to be retrieved, looping through all the data could do it:

```
for (var label in e.person.phone) {
    var phones = e.person.phone[label];
    for (var i = 0; i < phones.length; i++) {
        contactinfoLabel.text += label+' : '+ phones[i] + '\n';
    }
}
```

This example adds phone label text, for example “work” or “home”, and the actual phone number to the contact info label to the application. After all data from the contact is fetched, the application shows the contact info (see FIGURE 6).

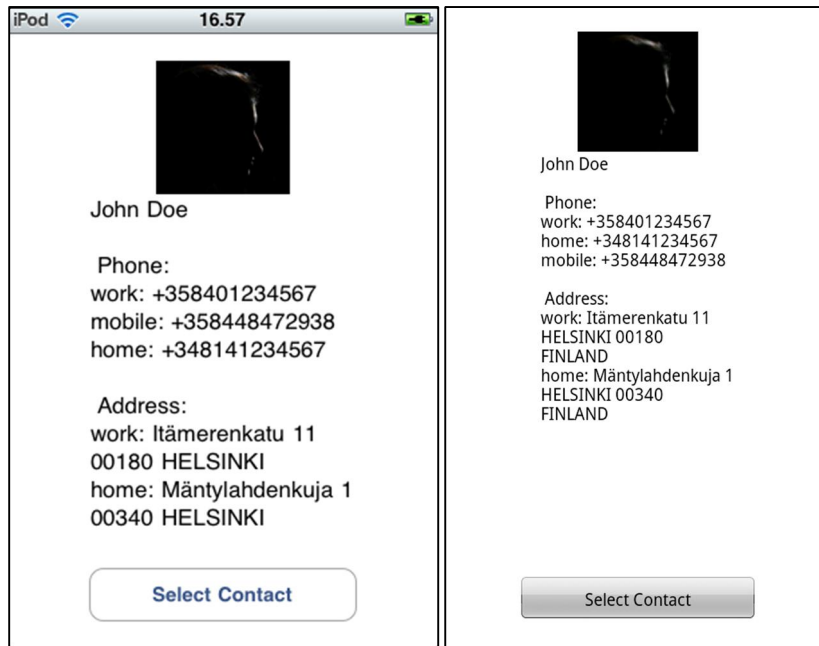


FIGURE 6. Contacts application running on iOS device (left) and Android (right)

### 3.2.3 Building for Device

Testing a Titanium application on an Android device is a straightforward process, if the Android SDK is already installed: turn on USB debugging on the Android Device, connect the device to the computer with an USB cable and from the Test&Package section of the Titanium Developer, choose Run on Device and then Android. The application will be installed to the device. It is required to install device drivers for the USB connection of the Android phone if using Windows or Linux (Using Hardware Devices 2011).

To test an application on an iOS device it is needed to complete series of steps required by Apple, for example to sign-up for iOS Developer Program, Register an iPhone with Apple, obtain developer certificate etc. (Testing your Applications 2011). The iOS Developer Program fee is 99 US dollars per year (iOS Developer 2011).



### 3.3 PhoneGap

PhoneGap is an open source framework for building native mobile applications using HTML, CSS and JavaScript. Nitobi Software introduced PhoneGap in 2008, and is free to use under an MIT license.

With PhoneGap it is possible to develop applications for iPhone, Android, BlackBerry, webOS and Symbian WRT (Allen et al. 2010, 131). Support for Windows Phone 7 is planned (Get Started 2011). PhoneGap is at its best for transforming a mobile web application into native application. It advantages particularly web developers since all the application code can be HTML, CSS and JavaScript. (Allen et al. 2010, 131.)

The principle of PhoneGap is that it provides a client-side JavaScript APIs with a method for hosting a web application within a native mobile application that an end user may install. Basically a PhoneGap application is native application with a full-screen browser (Firtman 2010, 417). As a native application, there are some capabilities that are not available from a web application, such as access to contacts data, geolocation, camera, and accelerometer using PhoneGap's JavaScript API (Op. cit. p. 131).

Developing with PhoneGap starts by writing a mobile web application using HTML, CSS and JavaScript. The content of the application does not have to be in any particular structure, developer has much choice how to form the mobile web application layout and structure. PhoneGap is as its best on platforms that include the WebKit browser with the advanced JavaScript and CSS of HTML5, such as iPhone and Android. (Op. cit. p. 132.)

PhoneGap's goal is to use advanced features of HTML 5 and to implement standards such as W3C Device API Group (<http://www.w3.org/2009/dap/>) that defines standards for JavaScript APIs for mobile phone features, such as contacts and

camera. Since the W3C Device API standards are not fully developed, PhoneGap contains APIs that diverge from the standard in order to build real, native applications. (Op. cit. p. 132.)

Like any other cross-platform frameworks that use the browser for UI, PhoneGap is not well suited for applications that require intense math calculations, 3D animations or data-driven applications, like most enterprise applications that must work offline and synchronize local data. There is no support for database on PhoneGap, instead it relies on HTML5 database APIs that is unavailable for some devices. (Op. cit. p. 132.)

### **3.3.1 Getting started**

PhoneGap supports iOS, Android, Blackberry, webOS and Symbian WRT. Support for Windows Phone 7 is planned. Since the Client's mobile application strategy will be focused on iOS and Android devices, the Getting Started section is focused on them.

The first step to get started with PhoneGap is to download the latest copy of PhoneGap from the PhoneGap website and extract its contents (Get Started 2011). The zipped file includes all necessary files for all supported platforms.

#### **iOS**

The iOS development tools are available only for Mac OS X computers. The prerequisite for iOS developing is to install the iOS SDK and Xcode, which includes the Xcode IDE, iOS Simulator, and a suite of additional tools for developing apps for iPhone, iPad, and iPod touch (iOS Dev Center 2011). Downloading the tools requires registering as an Apple Developer, which is free of charge. The free developer account allows testing applications in iOS simulator, but submitting apps to the App Store or testing apps on iOS devices requires enrollment in an iPhone developer program. (Stark 2010.)

Under the iOS directory of the PhoneGap there is an installer file to install the PhoneGap template for Xcode. After installing, there is a new option under Xcode's New Project window to create a new PhoneGap-based application (FIGURE 7). (Get Started 2011.) Choosing a new PhoneGap-based application creates a new, empty PhoneGap project. After choosing the base SDK version of the iOS and setting the application to run in Debug mode in Simulator, the PhoneGap application is ready to run.

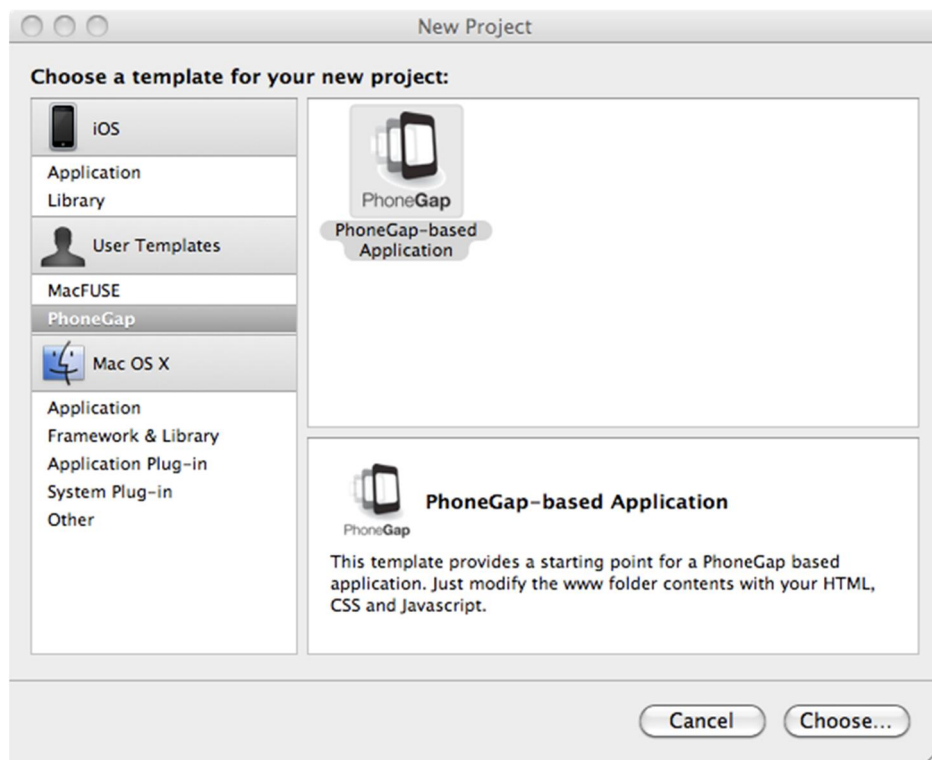


FIGURE 7. New PhoneGap-based application (iOS)

## Android

Prerequisites for Android developing are downloading and installing Android SDK, Eclipse, and ADT Plug-in for Eclipse. The tools are available for Windows, Mac and Linux. After installing the tools, a new PhoneGap project can be created in Eclipse by creating a new Android project with information as in the FIGURE 8.

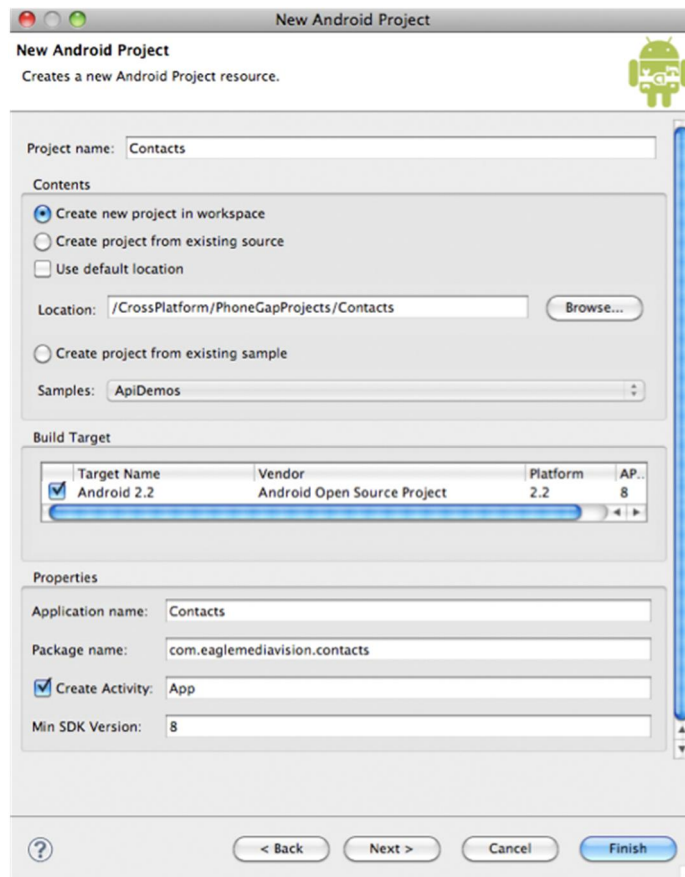


FIGURE 8. New PhoneGap Android application

The created project is a default Android project and therefore needs some customization for the PhoneGap. Details of the customization steps can be found at PhoneGap's Get Started guide for Android (Get Started 2011).

## Blackberry

Blackberry development tools are available for Windows XP or Windows 7. For developing PhoneGap applications with Blackberry, it is required to install Sun's JAVA JDK, Apache ANT, Blackberry Widget SDK and the latest copy of PhoneGap. More detailed instructions can be found from Get Started guide for Blackberry. (Get Started 2011.)

## webOS

HP's webOS developing can be done from Mac OS X, Windows or Linux since webOS SDK runs on top of an open source virtual machine, Virtual Box. On Mac and Linux webOS developing does not require other components than Virtual Box, webOS SDK and PhoneGap installed. Windows requires an additional component *cygwin*. When all required components are installed, a Palm Emulator can be run and an example project installed to the emulator by typing "make" in PhoneGap's webOS directory in Terminal / Cygwin. (Get Started 2011.)

## Symbian WRT

Developing for Symbian WRT can be done from Windows, OS X or Linux. PhoneGap installation files include an example project for Symbian WRT that can be generated to a Nokia Web Runtime Widget by typing "make" in Symbian subdirectory under PhoneGap installation directory. (Get Started 2011.)

Any text editor can be used for developing WRT Widgets, but there are free tools by Nokia for Nokia Web Runtime Widget developing and deploying to a simulator. With Nokia WRT Plug-in for Aptana Studio it is possible to create Nokia WRT Widget projects and to run them on browser-based Nokia simulator. If running Windows, there is the Nokia S60 SDK available that includes the S60 Emulator. (Op. cit.)

### 3.3.2 Example Project: Contacts

This example project uses W3C (World Wide Web Consortium) Contacts API specification to provide access to the device contacts database. The latest W3C Contacts API can be seen at <http://www.w3.org/TR/contacts-api/>. The Contacts application is only for Android and iPhone, since the Client will aim at these platforms.

The example differs from Titanium contacts example described in chapter 3.2.2 since in PhoneGap API it does not seem to be possible to show the native smartphone's contacts picker application to pick a contact data to the application. Instead, a text search field is used to filter content to the application. If there are multiple results with a contact search string, the first contact result will be viewed to the user.

Programming begins with creating a new PhoneGap project. Project creation is introduced in previous chapter. The first thing is to form the layout of the project. It is basically normal HTML and CSS layout markup, the only difference in this step is define a viewport for different mobile devices and resolutions for the layout to be almost the same in all devices:

```
<meta name="viewport" content="initial-scale=1.0, user-scalable=no,
width=device-width" />
```

The mobile application layout contains HTML and CSS formatted elements, such as title bar, content area for contact data, text input field for searching content and buttons for starting the contact search and clearing previous results. These elements are defined in *index.html* file that is located in the *www* folder of the PhoneGap project:

```
<div id="title_bar"><strong>Contacts</strong> app</div>
<div id="contactPicked" style="display:none"></div>
<div id="searchdiv" class="view">
  <div id="searchboxdiv">
    <input type="text" value="" id="contact_search"/>
  </div>
  <div class="app_button" id="contacts_button">Search Contact
  </div>
  <div class="app_button" id="clear_button">Clear</div>
</div>
```

The application logic begins with adding the mandatory reference to PhoneGap's JavaScript file that works as a bridge to the PhoneGap's API access to native features,

in this case the Contacts API. The actual application logic is in a separate JavaScript file, *app.js*. In addition, a XUI JavaScript library, a tiny DOM library for mobile web applications similar to jQuery, is added to make the JavaScript programming more straightforward. These libraries are added by adding two `<script>` tags to the *index.html*:

```
<script src="phonegap.0.9.4.min.js" type="text/javascript"
charset="utf-8"></script>

<script src="js/app.js" type="text/javascript" charset="utf-
8"></script>

<script src="js/xui-2.0.0.min.js" type="text/javascript"
charset="utf-8"></script>
```

In *app.js*, the “search contact” button waits for user tap and after that event happens, the contents of the input field will be sent to the function that accesses the Contacts API and receives the input field text as a filter to the contact search.

```
function getContacts(selectedstring) {
    navigator.service.contacts.find(['photos', 'name',
    'addresses.streetAddress', 'addresses.postalCode',
    'addresses.locality', 'phoneNumbers', 'emails'],
    successContactFindCallback,
    generalErrorCB,
    {filter: selectedstring}
    );
}
```

The *navigator.service.contacts.find* function is the W3C Contacts API function to access the device contacts. A successful contact search triggers the callback function where the returned contact data can be formatted to the PhoneGap application with HTML DOM. For example, to output a contact’s photo and full name to the application view:

```
for (var j in contacts[0].photos) {
    x$('#contactPicked').html('before', '<p></p>');
}
```

```

var namevar = contacts[0].name;
x$('#contactPicked').html('before', '<p>'+namevar.givenName + ' ' +
namevar.familyName+'<br/>');

```

When all eligible data is printed, the application is ready to run. The application's final state can be seen in FIGURE 9. The full source code of the application is in APPENDIX 2.

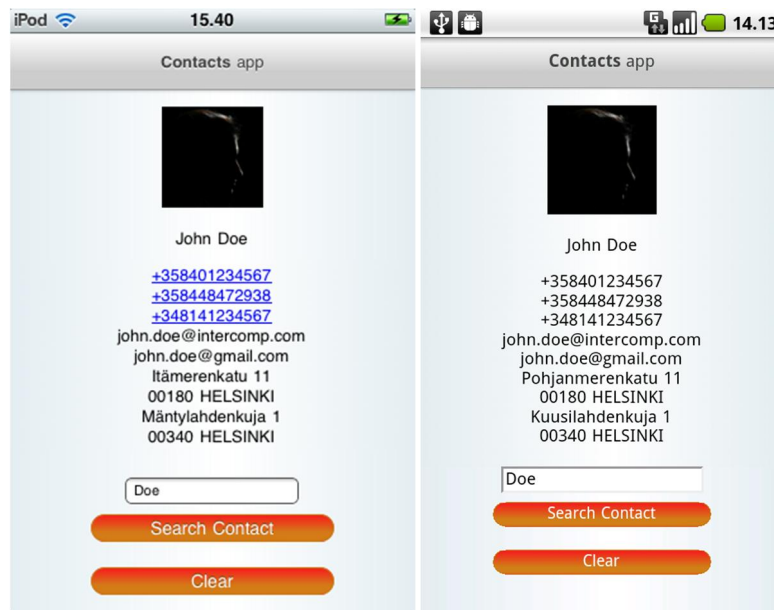


FIGURE 9. PhoneGap Contacts application running on iOS device (left) and Android (right)

### 3.3.3 PhoneGap Build

PhoneGap Build is a service that lets developers compile or build their PhoneGap based apps in the cloud service. The service's goal is to eliminate the need to download and configure mobile platform SDKs required to build native/installable applications. (Charland 2010.)

The principle is to write the application in HTML, CSS and JavaScript and upload it to the PhoneGap build service and get back app-store ready application binaries for



Apple iOS, Google Android, Palm, Symbian, BlackBerry, and later for Windows Phone 7, Meego and Bada (Introducing PhoneGap Build).

In March 2011, the service was in closed beta, and according to Nitobi, it will remain free for open source projects. When the service will get closer to public launch, the pricing will be announced for commercial applications. (PhoneGap Build FAQ.)

### **3.4 Rhodes**

Rhodes is a commercially supported open source cross-platform smartphone application framework by Rhomobile. It was released in December 2008. (Allen et al. 2010, 83.) Rhodes is available for BlackBerry, Windows Mobile (up to 6.1), Android and iOS platforms (Rhodes Introduction). The Rhodes applications are created using HTML, CSS, JavaScript and Ruby programming languages. The Rhomobile tools and frameworks can be used across Mac, Windows and Linux, and require the target platforms SDKs to be installed. (Allen et al. 2010, 83.)

Rhodes is targeted at enterprise applications and is the most suitable for applications that present a series of screens that include standard UI widgets, including common phone UIs. Rhodes leverages Model-View-Controller (MVC) approach in application framework. (Op. cit. p. 83.)

Rhodes includes an ORM (Object Relational Mapper) called Rhom (RhoSync Features). It provides database-abstraction functionality to make the local database easier to program to (Allen et al. 2010, 86). Rhodes also includes an optional sync server called RhoSync, which is a standalone mobile sync server to keep enterprise application data current and available on users' smartphones (FIGURE 10). The information is stored locally on a user's device and is available when disconnected. Using RhoSync to commercial projects requires a commercial license that can be purchased from Rhomobile. (RhoSync App Integration Server.)

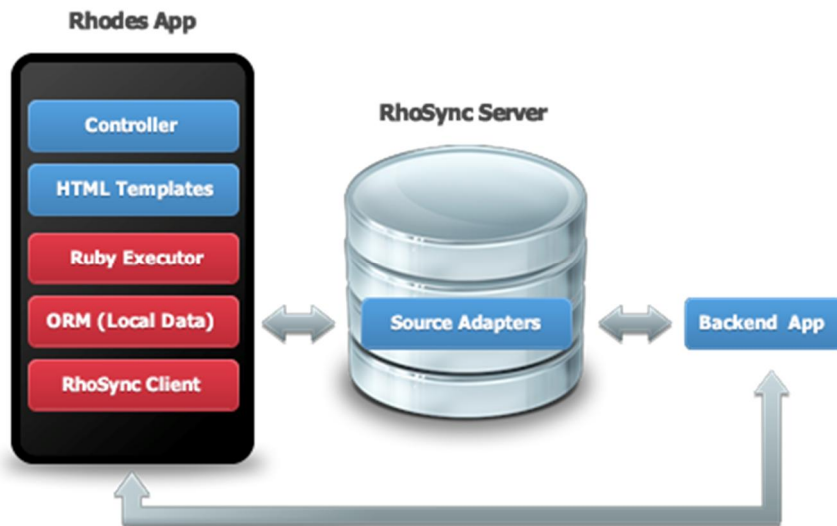


FIGURE 10. RhoSync and Rhodes application (RhoSync Features)

### 3.4.1 Development Architecture

Rhodes follows the Model-View-Controller (MVC) pattern. In controller, methods that define actions that map to HTTP requests are implemented. Controller action will fetch data from model that is implemented in the Rhodes ORM Layer, Rhom, and will render a view, which is implemented in HTML ERB. (Allen et al. 2010, 84.)

In view the user interface of the application is defined in HTML and CSS. At runtime the HTML and CSS is rendered in a native browser UI control that is embedded in the application by the Rhodes framework. JavaScript may be used for some interaction control, but using embedded Ruby (ERB) application logic can be added to views. ERB is similar to PHP, in that sense that in the both code can be mixed with markup to create dynamic HTML. (Op. cit. p. 84.)

Rhodes files are compiled into a native executable that can be run on the device or in a desktop simulator using command line tools or the web interface on rho-hub.com. Rhodes applications are compiled into Ruby byte code, except on BlackBerry where the applications are cross-compiled into Java byte code. Rhodes includes a Ruby executor that runs the byte code on the device. The native mobile applications can

be then submitted and distributed through the application stores such as iTunes App Store. (Op. cit. p. 85.)

### 3.4.2 Getting started

Rhodes requires Ruby, Ruby's library packaging system, *RubyGems*, and GNU *make* installed. The device SDKs for the desired target platforms also need to be installed. If running Windows, all the required tools and Rhodes can be installed with Rhodes installer executable. On other platforms, if the required components are installed, Rhodes can be installed by typing:

```
gem install rhodes
```

After the installation script is finished, Rhodes can be configured by typing "*rhodes-setup*" in command line. The Rhodes setup script is for configuring paths for mobile platform SDKs. (Op. cit. p. 88, 89.)

### 3.4.3 Example Project: Contacts

The contacts example project in Rhodes allows showing, editing and deleting native PIM contacts using Rhodes APIs on both the iPhone and Android.

Using the *rhogen app* command creates the initial skeleton for the application: a starting directory with support files. Rhodes applications are organized in a fixed directory structure. (Allen et al. 2010, 89.) In this case the command for creating Contacts application named *rhodescontacts* is:

```
rhogen app rhodescontacts
```

The created directory structure can be seen in FIGURE 11:

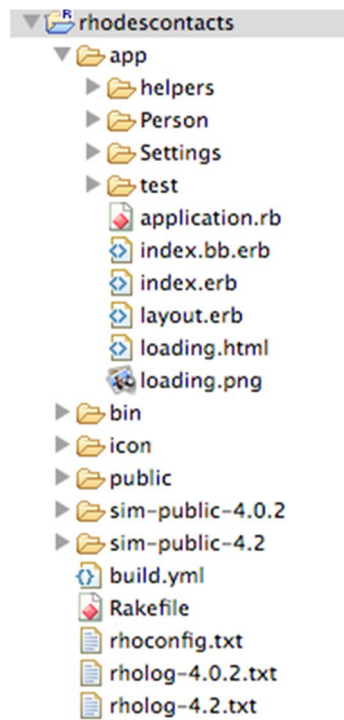


FIGURE 11. Rhodes project directory structure

In the created project, the *app* folder contains models, device settings, default landing page, and an application layout page. Inside the *app* folder, the *application.rb* contains application specific setup and configuration, the *index.erb* is a default-landing page for the application that has links to the controllers for data models. The *public* folder includes static files accessible by the application, such as CSS, images and JavaScript libraries. The *build.yml* file is for application specific build information such as the name of the application and the SDK information for different platforms. The *rhoconfig.txt* file contains application specific options and configurations such as the *start\_path*, logging, and the optional URL for sync server.

The actual application logic is implemented to the Model-View-Controller (MVC) pattern, similar to the Ruby on Rails. Rhodes includes a script to generate a MVC pattern that will implement common actions to display a list of items, show an individual item's details, create, update and delete. (Allen et al. 2010, 95.)

Because the example application implements the native Rhodes Contacts API, a simple *person* model is created with just a *name* attribute.

```
rhogen model person name
```

This command creates a subdirectory named *Person* under the */app* folder. The created model files include the views for the default controller actions, the configuration file for the model, and the controller. (Op. cit. p. 96.)

The *person\_controller.rb* implements the controller for the model. In the model there are also many *.erb* files for all the views associated with the model. Finally, there is a *person.rb* file that sets the properties on the model. Each Rhodes controller implements actions to perform basic CRUD (create, read, update, and delete) operations on the generated objects by default. (Op. cit. p. 96.)

To set the application default-landing page directly to the generated *Person* model, the start path definition in *rhoconfig.txt* is changed to:

```
start_path = '/app/Person'
```

Programming the application logic starts with editing the *person\_controller.rb* file. The first step is to require Rhodes PIM contacts API by adding the line:

```
require 'rho/rhocontact'
```

The device's full contact list is rendered to the default landing-page, *index.erb*, of the application. It is programmed to the controller by adding some Ruby code to the corresponding part:

```
#GET /Person  
def index
```

```

    @people = Rho::RhoContact.find(:all)
    @people = @people.to_a.sort! { |x,y| x[1]['first_name'] <=>
y[1]['first_name'] } if @people
end

```

The *Rho::RhoContact.find(:all)* method returns all the contacts on the device. The next line sorts the contacts array by person's first name. The array needs to be rendered to the view. For that some Ruby code is added to the *index.erb* view:

```

<div class="content">
  <ul>
    <% @people.each do |person| %>
      <li>
        <a href="<%= url_for :action => :show, :id =>
person[1]['id'] %>">
          <span class="title"><%= person[1]['first_name'] %>
          <%= person[1]['last_name'] %></span>
          <span class="disclosure_indicator"></span>
        </a>
      </li>
    <% end %>
  </ul>
</div>

```

The array of contacts is iterated through and each one is output in a list. The default generated HTML and CSS classed define the output to look like a native table in each platform (FIGURE 12).

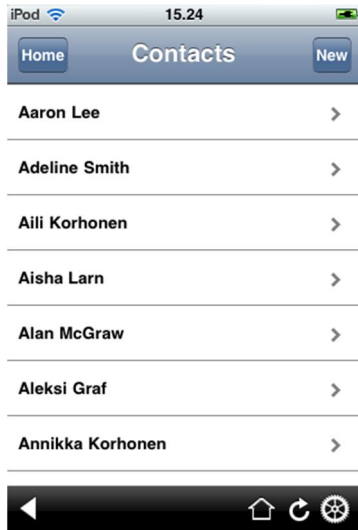


FIGURE 12. Rhodes Contacts, all Contacts list (iOS)

When user selects a contact from previously generated contact list, a single contact is viewed. To implement that feature, the *RhoContact.find* needs to be implemented to the controller:

```
# GET /Person/{1}
def show
  @person = Rho::RhoContact.find(@params['id'])
  if @person
    render :action => :show
  else
    redirect :action => :index
  end
end
```

In the *show.erb* view, the contact data is printed to the view. According to Rhodes API, the PIM Contacts support following fields on all devices:

```
"id", "first_name", "last_name", "mobile_number", "home_number",
"business_number", "email_address", "company_name"
(Device Capabilities.)
```

Some contact fields are available only on iOS, such as street address field. For this reason some contact fields are empty on Android, but are in use with iOS device (FIGURE 13).

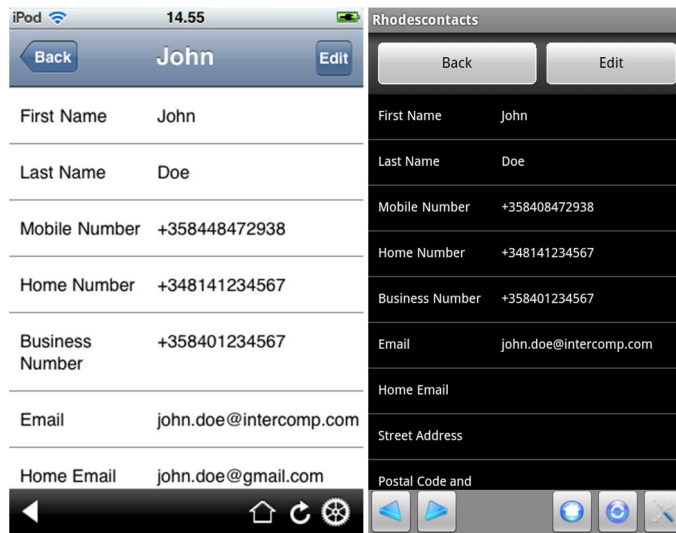


FIGURE 13. Rhodes Contacts, single contact view running on iOS (left) and Android (right)

The same functionality as in other frameworks' contacts examples is achieved with the code described in this chapter. In addition, it is easy to implement more functionality to the Contacts application with Rhodes: possibilities to add a new contact, edit a contact, update a contact and delete a contact. Code listings for these functions can be seen in the APPENDIX 3.

#### 3.4.4 RhoHub

RhoHub is a cloud service hosted by Rhomobile that offers a build service for all supported smartphones, hosted data synchronization and automatic source control (RhoHub: Hosted Development for Mobile Apps 2011). It is free for open source applications and has pricing for private applications (Allen et al. 2010, 122).

The idea of RhoHub is to simplify the development and deployment experience. The idea could be achieved with the build service, which eliminates the need to install smartphone SDKs, by integrated hosted data synchronization server, RhoSync, and by GitHub based source code control. (RhoHub: Hosted Development for Mobile Apps 2011.)



### 3.5 Adobe AIR for Mobile Devices

Adobe Integrated Runtime (AIR) is a cross-operating system runtime that enables to use ActionScript or HTML/JavaScript development skills and tools to build web applications that run as standalone client applications. In AIR version 2.6, support for Android (2.2), BlackBerry Tablet OS, and iOS mobile operating systems is available. (Adobe AIR 2011.) The Adobe AIR client runtime is a combination of Flash Player, an embedded SQLite database engine, and the WebKit browser engine (Introducing the Flash Platform 2011).

Adobe AIR runs on Windows, Mac OS X and Linux. AIR software can be developed with multiple tools, such as Adobe Dreamweaver CS5, Flash Builder 4, Flash Catalyst CS5, Flash Professional CS5, Aptana Studio with AIR plug-in installed or any text editor with AIR SDK. Development can be done using alternatively ActionScript 3 with Adobe Flash, ActionScript 3 and MXML markup for presentation logic with Flash Builder, or HTML, JavaScript and AJAX with any text editor. (Adobe AIR 2011.) For the mobile AIR applications only ActionScript or Flex development is possible, while for desktop it is also possible to use HTML, JavaScript and CSS (Corlan 2011).

Adobe's development tools are commercial products, but Adobe AIR SDK is free and includes the tools necessary to build and deploy Adobe AIR applications (Op. cit.). Adobe Flex SDK is also open source, but Flash Builder is not. Adobe Flex SDK can also be used with an open source IDE, such as Eclipse, instead of Flash Builder. (Introducing the Flash Platform 2011.)

#### 3.5.1 Adobe Flex SDK "Hero" framework

The Flex SDK "Hero" builds on top of the core runtime APIs provided by Flash Player and AIR. It provides a framework for building application UI and connecting to server-side data. Flex includes a set of built-in UI components, data access components and

data binding, declarative UI creation through MXML, dynamic layout and an extensible component architecture. (Jaramillo 2011.)

The Flex SDK “Hero” is a preview release, and it introduces support for mobile Flex application development, including support for Android with BlackBerry and iOS support planned. Furthermore, the framework expands and refines Flex Spark components. (Adobe Flex SDK “Hero” 2011.)

### **3.5.2 Flash Builder “Burrito” development environment**

The “Burrito” codenamed preview release of the Flash Builder IDE brings a design/build/debug workflow to mobile development. It extends the existing ActionScript and Flex development workflows to mobile development, from project creation through packaging the final application. (Jaramillo 2011.)

“Burrito” provides two mobile project types: ActionScript Mobile Project and Flex Mobile Project. ActionScript Mobile Project gives a clean slate on which to write an application in ActionScript. Creating a Flex Mobile Project gives an access to the Flex framework, as well as the Flex-related features of Flash Builder, such as Design mode and the data connectivity features introduced in Flash Builder 4. (Op. cit.)

The Flash Builder “Burrito” and the Flex SDK “Hero” are preview releases, the final releases of Flash Builder 4.5 and Flex 4.5 SDK will be available for download in May 2011 (Subramaniam 2011).

### **3.5.3 Getting started for Android**

Adobe AIR for Android devices is available for Android 2.2 (“Froyo”) devices or higher. Developing Android applications begins with downloading Flash Builder “Burrito” and the Flex “Hero” framework. (Corlan 2011.)

Once Flash Builder “Burrito” is installed, it is possible to create Android applications either using ActionScript or Flex. In addition, development can be done using Flash Professional CS5. Applications can be tested on a computer using an emulator, deployed to an Android device using the USB cable, and be submitted to Android Market using Flash Builder Export Release feature to build the APK file. (Op. cit.)

AIR applications use the AIR runtime installed on Android device. If the end user of the application does not have the AIR runtime installed, the application automatically prompts the user to install the runtime. (Op. cit.)

#### **3.5.4 Getting started for iOS**

Adobe released the Packager for iPhone in 2010 as part of the Adobe Creative Suite 5 launch in 2009 (Corlan 2011). Later on, Adobe stopped development on AIR for iOS due to a change in Apple's developer program license and focused on Android instead. The result of the work was AIR 2.5. When Apple removed restrictions, the iOS and Android versions of AIR were out of sync. (Cantrell 2011.)

Adobe released AIR 2.6 in March 23, 2011. The new version brings the Android and iOS releases into alignment by bringing new features and APIs for iOS devices, such as Microphone, Camera and Multi-tasking support. (Op. cit.)

At the time of writing, Adobe has not yet released an updated version of Flash Builder in order to create iOS applications with Flash Builder. Until then, the development for iOS is done with the Packager for iPhone and Flash Professional CS5 (Corlan 2011).

#### **3.5.5 Example Project: Google Maps API with Geolocation**

Since there is currently no API support for device Contacts API in Adobe AIR or Flex mobile, an example project in this case is Google Maps API with geolocation feature.

The application shows the user's current location on the Google Maps. The application is programmed only for Android since there is no Flex support for iOS yet.

From Flash Builder "Burrrito", choosing a New Flex Mobile Project creates a template for new Android mobile project. In the example application, the project features are following: (FIGURES 14 and 15).

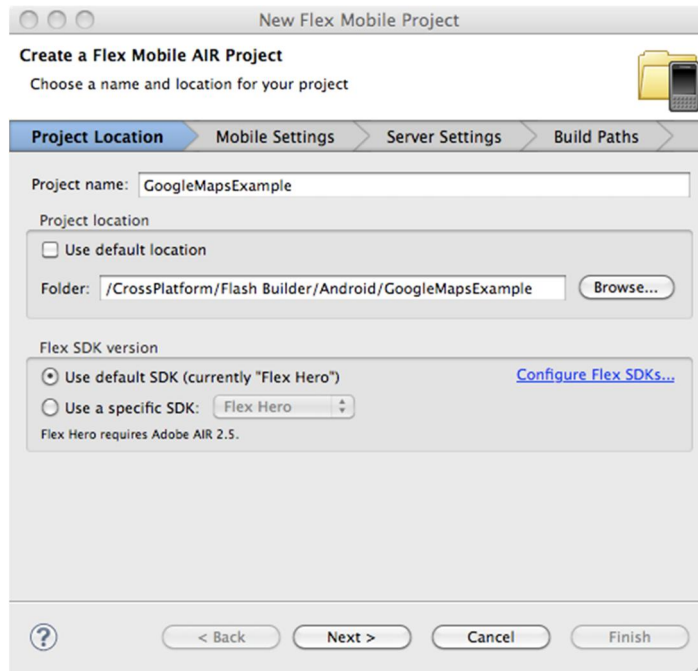


FIGURE 14. New Flex Mobile Project 1 / 2 (Android)

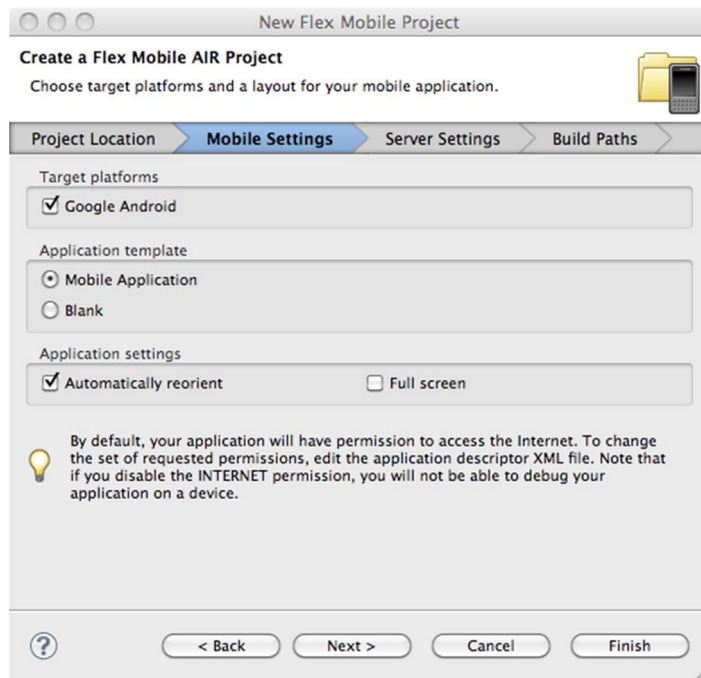


FIGURE 15. New Flex Mobile Project 2 / 2 (Android)

The project is ready to be created after these two steps, since in this project no server side data is needed and the default build paths are fine. The created project structure is following: (FIGURE 16).

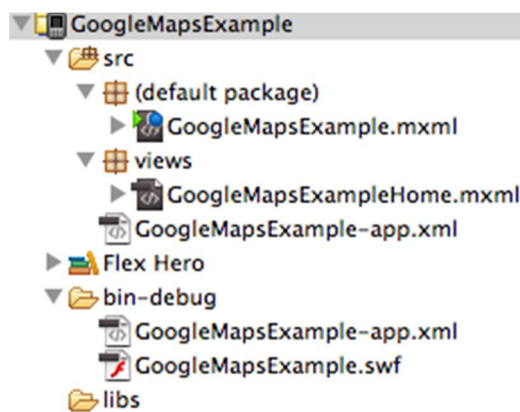


FIGURE 16. Flex Mobile Project structure

The project consists of *src* folder, which has the main application *mxml* file in the default package and all the views in the *views* folder. In this case there is only one view, but a Flex mobile application could have multiple views. One way to build a

navigation between views would be by using a `<s:TabbedMobileApplication>`, which creates a tabbed navigation to the application, in main application file instead of `<s:MobileApplication>`.

In the `src` folder there is also an xml file that specifies parameters for identifying, installing, and launching AIR applications. It also specifies the Android application permissions. It has Internet usage permission as default, but in order to use location based services, the following permissions need to be added to the Android manifest section of the xml file.

```
<uses-permission
  android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
  android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Since there is no navigation in the application, the application MXML file can point directly to the view file by setting the `firstView` attribute in `<s:MobileApplication>` to the following:

```
firstView="views.GoogleMapsExampleHome"
```

Using the Google Maps API requires signing up for a Google Maps API key, and downloading and installing the Google Maps API for Flash SDK. Instructions for obtaining a Google Maps API key and downloading the Google Maps API for Flash SDK can be found at Google Maps API for Flash website. The downloaded file contains a `.swc` file for Flex. Copying it to the project's `libs` folder adds it to the build path and adds support for Google Maps APIs in Flash Builder.

The application logic can be programmed directly to the `GoogleMapsExampleHome.mxml`. The application uses `flash.sensors.Geolocation` object to access device's location data, and Google Maps API to render the map on

screen based on latitude and longitude coordinates received from the Geolocation object. To hold the latitude and longitude data temporarily in the application, a custom class *GPSData* is created and used in the application as object.

The core of the Geolocation function in the application is made with following code:

```
g = new Geolocation();
g.setRequestedUpdateInterval(100);
g.addEventListener(GeolocationEvent.UPDATE, geoLocation_UpdateHandler,
false, 0, true);

protected function
geoLocation_UpdateHandler(event:GeolocationEvent):void {
    gpsdata.setLatitude(event.latitude);
    gpsdata.setLongitude(event.longitude);
}
```

It sets the Geolocation object, adds an event listener for Geolocation data to be updated (in interval of 100 milliseconds) and in the update the handler function sets the latitude and longitude data of the custom *gpsdata* object.

The core to set the Google Maps is following code:

```
gMap = new Map();
gMap.key = "(google maps key here)";
gMap.url = "http://www.eaglemediavision.com"
gMap.sensor = "true"
gMap.width = 480;
gMap.height = 682;
gMap.addEventListener(MapEvent.MAP_READY, mapReadyHandler);
mapContainer.addChild(gMap);

private function mapReadyHandler(e:MapEvent):void {
    gMap.setCenter(new
LatLng(gpsdata.getLatitude(),gpsdata.getLongitude()), 15,
MapType.NORMAL_MAP_TYPE);
    gMap.setSize(new Point(mapContainer.width,
mapContainer.height));

    var latlng:LatLng = new
LatLng(gpsdata.getLatitude(),gpsdata.getLongitude());
    var marker:Marker= new Marker(latlng);
    gMap.addOverlay(marker);
}
```

And in the MXML, a Flex component to render the Google Maps to:

```
<mx:UIComponent id="mapContainer" width="100%" height="100%"/>
```

The application can be run in a simulator, where it is possible to simulate multiple Android devices. The application can also be installed to Android device for debugging (FIGURE 17). With Flash Builder it is also possible to submit an application to the Android Market with custom icons and startup screen images.

The full source code of the example is in the APPENDIX 4.

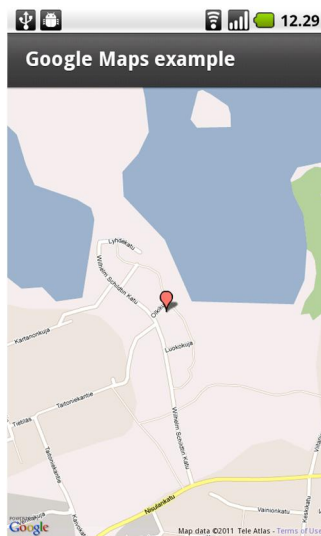


FIGURE 17. The Google Maps Flex Mobile Project running on Android device

### 3.6 OpenPlug Elips Studio

Elips Studio is a cross-platform mobile application development toolset by Alcatel-Lucent (former OpenPlug). The framework was originally introduced in 2009. It allows developing native mobile applications and deploying them on Android, iOS, Windows Mobile and Symbian platforms. (ELIPS Studio: cross-platform native mobile application development.)



Elips Studio is available as a plug-in for Adobe Flash Builder versions 3 or 4 or as a stand-alone Eclipse based IDE. Mobile applications are developed using ActionScript, the application UI and visual elements using MXML. In addition, the SDK supports web services, network APIs, and device APIs such as GPS, Photos and SMS. (Op. cit.)

The product offers native UI controls mapped directly from each device's OS into code, including lists and screen transitions. It is also possible to manage graphical assets and UI styles through CSS for each variant of an app on each class of device platform and form-factor. The Elips compiler cross-compiles the application code into C++ code, which allows direct access to native APIs. Finally, an application is packaged into native, installable mobile application. (Op. cit.)

Elips Studio is not open source. It is free for evaluation only for corporations, for individual developers the framework is free, but ad-supported, which means that an advertisement banner is automatically added to all applications. A developer receives 60% of the banner ad revenue. (ELIPS Studio FAQ.)

### **3.6.1 Getting Started**

There is an installation package available on the Elips Studio developer website. The installation package is available for Windows and Mac OS X. The tools can be installed as a plug-in for Adobe Flex Builder 3 or Flash Builder 4, or as a stand-alone IDE. In this case the framework is installed as a plug-in for Adobe Flash Builder 4 on Mac OS X.

Installation package creates an executable, in this case Elips FB4, since Adobe Flash Builder 4 is chosen as the IDE. The executable launches the Adobe Flash Builder 4 with Elips plug-in. At the first run Elips Studio asks to set the workspace (base directory on hard drive) for Elips Studio projects. Elips developer login details are also asked. The developer account is free, but mandatory. After setting the account and restarting the Adobe Flash Builder, the Elips Studio is ready for development.

### 3.6.2 Example Project: Contacts

Elips Studio offers a Contacts API access to the device's native address book. The contact list can be retrieved either from SIM or Phone memory. In this example project the contacts are retrieved from Phone's memory, listed and when a contact is tapped, single contact is viewed on different screen view. Additionally, there is a search filter functionality to filter the contact list to find the contact by name. The source code of this example application extends some features to an Elips sample application called Elips Contacts Manager Sample, made by Nicolas Sauvage, OpenPlug developer. The original sample project can be downloaded from <http://developer.openplug.com/forum/download/file.php?id=95>.

A new Elips project can be created in Adobe Flash Builder 4 either by selecting File→New→Other→Elips Project or by clicking the "Create a new Elips Project" button in Elips toolbar. A new project window appears (FIGURE 18). Noteworthy is that the Application type is set to *Screen Based Application*, which creates a `<mob:ScreenStackApplication>` that supports multiple screens, or views. Other option is a "Standard Application", which creates template based on `<mx:WindowedApplication>` container that is to be used for single screen applications or custom application such as games. (Tutorial: [Hello World] Using new project templates.)

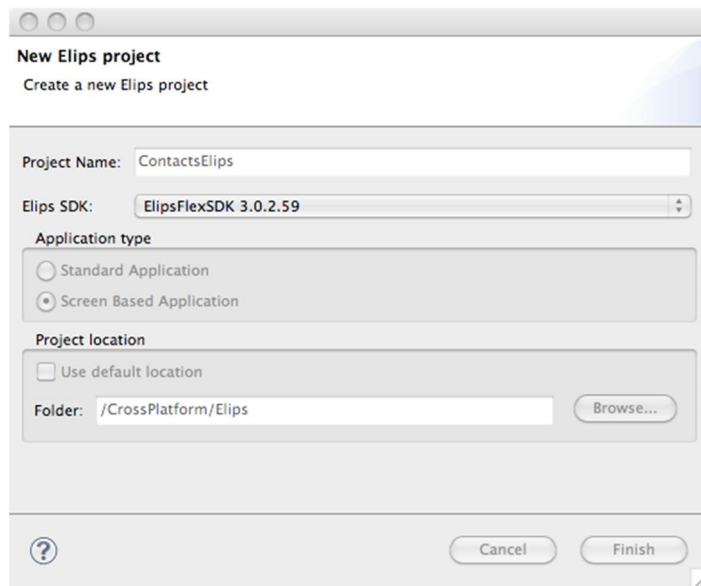


FIGURE 18. New Elips Project (Adobe Flash Builder 4)

The created project structure is quite similar to Adobe Flash Builder (FIGURE 19). However, there are some differences; Elips has *elips\_styles* folder for setting the CSS styles for elements (although CSS styles can be set in the MXML files also), *capabilities.xml* file for targeted platform specific settings, *extensions* folder for compiler settings file, *locales* for localization files, *output* for generated executable for targeted platforms, and *resources* for platform specific resource files, for example images.

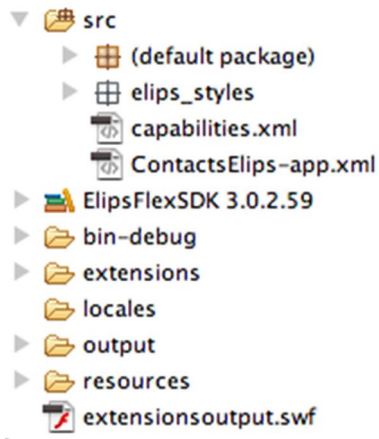


FIGURE 19. Elips Project Structure (Adobe Flash Builder 4)

After the project has been created, it is needed to choose the devices wanted to target the application with. The Elips targeted devices can be chosen from Project→Properties→Elips Targeted devices. Multiple devices can be selected to the project, such as iPhone, Generic Android devices with multiple screen resolutions, S60 Devices of different editions, Windows Mobile devices, and Symbian^3.

Programming the Contact application begins by editing the basic UI and screens of the *ContactsElips.mxml* main application file. By default, a *ScreenStackApplication* template has one screen with two tab based navigation views. In this case, two different screens are wanted, and the navigation between them occurs when the user taps a contact in the contact list screen (the main view), and the single contact details are rendered to another screen. Two screens are defined:

```
<mob:ScreenView id="allContactsScreen">
</mob:ScreenView>
<mob:ScreenView id="singleContactScreen">
</mob:ScreenView>
```

The UI and the layout of the screens are defined with Flex UI components and containers, such as *VBox*, *HBox*, *Label*, *Button*, and *Text*. However, Elips Studio does not implement the full namespace of Flex. Generic UI component classes contained in the *<mx>* namespace are common Flex UI component redesigned to fit mobile

phones. It includes basic controls like buttons, images or containers that have the same rendering whatever the underlying platform. Moreover, native UI components classes contained in the `<openplug.elips>` namespace are UI components that may have a native implementation specific to each platform (or a generic one if not available). That is the case of the HTML class that will be rendered as the native `UIWebView` on iPhone and as a native `Android WebView` on Android devices. (Nicolot 2010.)

In the business logic of the application, the first step is to get the Address book. There are two Address books available: internal (in Phone's memory) and a SIM address book. In this case internal one is used. The command used is:

```
_addressBook = _contactManager.getAddressBooks() [0] as AddressBook;
```

To list the contacts to the application, the sample application uses the `FindContactsResult` class that implements the native `ContactArraySuccessCallback` class. The class filters the Address Book data if an optional search filter is set, iterates the `Contacts (openplug.elips.services.Contact)` list, saves the data to an `ArrayCollection` and returns the data to the main program.

Successfully returned contacts provide the contacts `ArrayCollection` data as a `dataProvider` to the `<mob:list>` component that lists the contacts to the first screen of the application.

```
<mob:List id="listContacts"
  width="100%"
  height="90%"
  rowHeight="80"
  backgroundColor="white"
  itemRenderer="itemRenderers.ListItemRenderer"
  itemTouchTap="openContact(event) "
/>
```

When a single contact is tapped, a new screen is opened and a contact details are viewed. It is done by adding a tap listener function to the `<mob:List>` component that contains all the contacts, or if the user has filtered the contact list with search function, a filtered list (FIGURE 20).

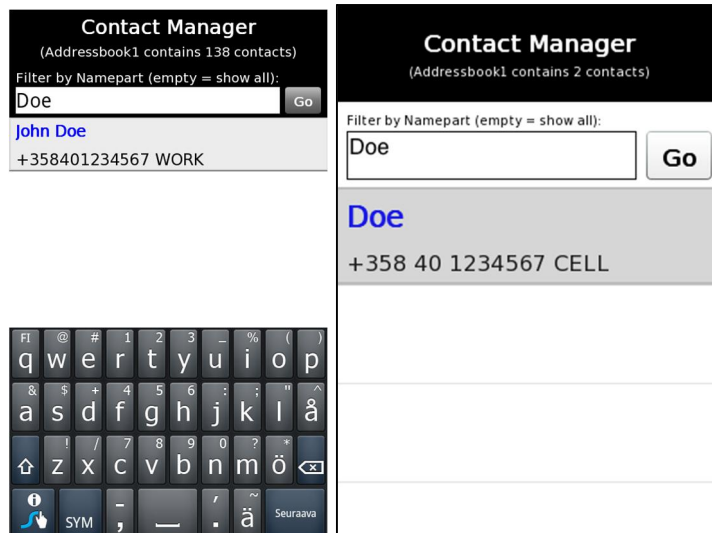


FIGURE 20. Elips filtered contacts list (Android)

The `openContact` function receives a clicked item as `ItemTouchTapEvent`. From the event item a new `ContactExt` (extended Contact class in the sample) object is created and then a contact data can be printed to the single contact screen. Some data is returned as Array, because there can be multiple values, for example a contact can have multiple phone numbers. The arrays are iterated when details of a single contact are printed.

```
public function openContact(evt:ItemTouchTapEvent):void {
    var contact:ContactExt = evt.item;
    var contacttext:String;
    var contactAddresses:Array = contact.addresses;
    var contactPhones:Array = contact.phoneNumbers.number;
    var contactEmails:Array = contact.emails;
    contacttext = "";

    contacttext += contact.name + "\n";

    if (contact.phoneNumbers != null) {
        var i:int;
        for (i=0; i<contact.phoneNumbers.length; i++) {
            var phoneNum:ContactPhoneNumber=
```

```

        contact.phoneNumbers[i];
        contacttext+=phoneNum.type+": "+phoneNum.number+
        "\n";
    }
}

if (contact.emails != null) {
    for (i=0; i<contact.emails.length; i++) {
        var email:ContactEmail = contact.emails[i];
        contacttext+=email.type + ": " + email.email + "\n";
    }
}

if (contact.addresses != null) {
    for (i=0; i<contact.addresses.length; i++) {
        var address:ContactAddress = contact.addresses[i];
        contacttext+=address.street+" "+address.postalCode+"
        "+ address.city + "\n";
    }
}

singleContactText.text = contacttext;
goToScreen("singleContactScreen");
}

```

The full code listing is in APPENDIX 5. Screenshots of a single contact view running on Android and iPhone (simulator) are illustrated in FIGURE 21.

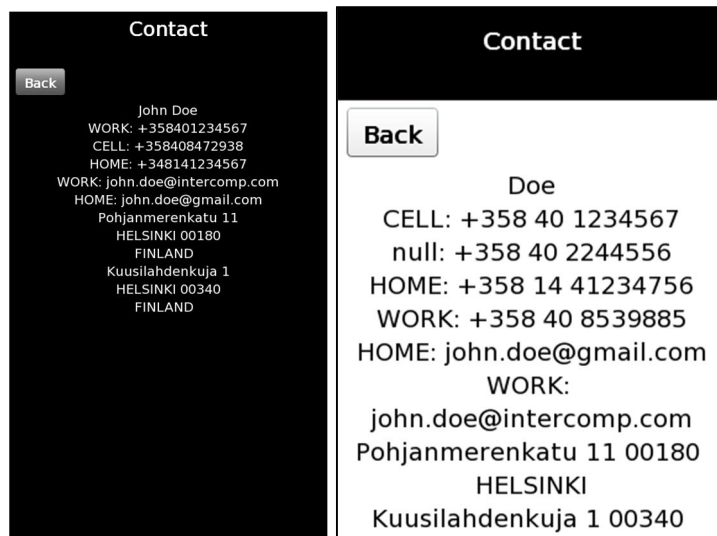


FIGURE 21. Elips Contacts, a single contact view (Android left, iPhone simulator right)

## 3.7 Summary

In this chapter the most common cross-platform native frameworks today are introduced. The chapter creates the basis for comparison of the cross-platform frameworks in the next chapter. The unifying factor for all introduced cross-platform frameworks is that they let a developer create a native, installable and mobile application distribution platform capable mobile software for more than one platform with the same codebase. Nevertheless, there are differences in the areas of API capabilities supported, in the (native) UI capabilities, and in the overall ease of use and user experience. It is clear that some of the introduced frameworks fit to the Client's needs better than others. The next chapter compares the features of the cross-platform native frameworks in order to find the most appropriate frameworks for the Client.

However, some cross-platform native frameworks were ignored from a more detailed presentation for several reasons. Some frameworks were ignored since they were not actual cross-platform tools by supporting only one platform. With some others it was not possible to build installable applications.

In the next subchapters, there are short introductions of frameworks that would be suitable in the future needs of the Client's mobile application strategy.

### 3.7.1 Corona

Corona SDK is a mobile development framework for creating applications and games for the iPhone, iPad, and Android. The developing language is a scripting language called Lua. With the framework it is possible to create native, installable and distributable applications on the iTunes App Store and Android Market. (Corona FAQ.)



The Corona SDK is graphics and game oriented framework, and it offers an integrated game engine and support for the graphics acceleration hardware of the device (OpenGL/ES) (Op. cit.).

The framework is not open source: the subscription for one year is 199 \$ / platform or 349 \$ / year for iOS and Android (Op. cit.).

Since the framework is game oriented and a closed source, it is not applicable to the Client's needs. However, the framework is introduced here, since the mobile application needs may be changing in the future, and an advanced mobile graphics performance may be needed even in some enterprise mobile applications.

### **3.7.2 MoSync**

MoSync is a SDK that allows development of applications for all major mobile platforms using a single environment and C/C++ code base on Windows or OS X. MoSync produces real native applications, packaged and ready for distribution in each platform's native installation format. In MoSync API there is support for most of the device features such as location and contacts. The framework supports also OpenGL. (What is MoSync.)

The developing is done using standard C/C++ in an Eclipse-based environment preconfigured with a set of MoSync-specific plugins. Since MoSync uses standard C/C++, many libraries are readily available to developers. Examples of libraries used with MoSync are SDL, yajl, STLPort and SQLite. (Op. cit.)

MoSync supports the iOS, Android, Windows Mobile, Symbian, JavaME and Moblin platforms. Blackberry and Windows Phone 7 support is coming later. MoSync is dual licensed: open source GPL v2 for open source projects and a commercial license for closed source applications. (Op. cit.)

MoSync is a product of the MoSync AB, Sweden. One of the financiers for the MoSync is Michael “Monty” Widenius, better known as a “father of the MySQL”. He hopes that there would be one day a “write once, run anywhere” environment, which is a target for MoSync. (Ahokas 2011.)

MoSync is an ideal cross-platform framework for enterprise and customer applications. However, the framework was excluded from a more detailed study, since the development language is C++ and the client does not have enough know-how for the C++ at the moment. If the situation changes in the future, by recruitment or by other means, MoSync is a seriously considered option for the cross-platform development.

## **4 COMPARISON OF THE CROSS-PLATFORM NATIVE FRAMEWORKS**

In this chapter the cross-platform native frameworks introduced in Chapter 3 are compared against each other. The comparison is based on the Research Objective in Chapter 1.3. The objective is to find for the Client “the optimal solution”, or solutions, that meets the criteria described in Chapter 1.3.

The comparison method is done by tables with simple true / false comparison of each property. By counting the number of “true” properties, an indicative score can be calculated. Since there are many properties to compare, the chapter is organized into subchapters, which contains tables of each subject, for example a comparison table of supported API features. The indicative total score can be calculated from the subchapter tables and used for guidance when selecting the cross-platform frameworks for the Client.

## 4.1 Smartphone platform support











Many frameworks include support for Windows Mobile 6.x versions, but since Microsoft released a Windows Phone 7, it is no longer worthwhile to put developing efforts on old Windows Mobile versions. For this reason only Windows Phone 7 is taken into account in this comparison.

Adobe Flash Builder “Burrito” includes support for BlackBerry Playbook tablet OS. That was excluded from the table, since the thesis focuses on smartphones.

When counting the supported devices at the moment of writing, PhoneGap has the widest support for smartphone platforms with its five supported platforms (see TABLE 2).

TABLE 2. Comparison: Smartphone platform support (Lukasavage 2011a; Getting Started with Appcelerator 2011; Get Started 2011; Rhodes Introduction; Jaramillo 2011; Supported Platforms and App Stores)

	Appcelerator Titanium	PhoneGap	Rhodes	AIR for Mobile Devices	OpenPlug Elips Studio	Notes !
Android	✓	✓	✓	✓	✓	
iOS (iPhone, iPod, iPad)	✓	✓	✓	!	✓	Adobe: Not yet implemented in Flash Builder “Burrito”
BlackBerry Phone	!	✓	✓	✗	✗	Titanium: support in closed beta
Symbian	✗	✓	!	✗	✓	Rhodes: Symbian not actively maintained

Palm						
Windows Phone 7						PhoneGap: WP7 is coming
<b>TOTAL</b> (platforms supported)	<b>2</b>	<b>5</b>	<b>3</b>	<b>1</b>	<b>3</b>	

## 4.2 Device APIs comparison

The property was set “true”, if a feature was supported on Android and iOS, since these platforms are in the Client’s priority.

On the Adobe AIR for Mobile Devices the source of information was Adobe Flash Builder “Burrito” and Flex SDK “Hero” preview versions. Since the product is not officially published, the device API is not complete and therefore not such comparable.

The main reference was API documentations of each framework. However, the documentation of API was outdated in some cases or was lacking information. In some cases, the information is based on framework’s developer discussion forums or sample code found from the framework’s developer site. If the information of the specific feature was missing from all known sources of information, the assumption was that the feature is not supported and the feature was set to “false”.

In addition, some of the features are not built-in to the frameworks and instead are extensible modules. These cases were ignored in order to maintain the comparability and only built-in features were approved. Because of some assumptions made, and the incompleteness of the compared frameworks, the result is indicative. However, by the result, Rhodes has the best device API support and Titanium is at the same

level of the API support, if the Android-only Calendar feature would be counted (see TABLE 3).

TABLE 3. Comparison: Device APIs support (Lukasavage 2011a; Titanium Mobile API Reference; Appcelerator Community Questions & Answers; Rhodes Device Capabilities; Jaramillo 2011; Tour de Mobile Flex; Beta ActionScript 3.0 Reference; ELIPS Studio API Reference; OpenPlug Support Forum)

	Appcelerator Titanium	PhoneGap	Rhodes	AIR for Mobile Devices	OpenPlug Elips Studio	Notes !
Accelerometer	✓	✓	✓	✓	✗	
Barcode	!	✗	✓	✗	✗	Titanium: available as an extension module (commercial)
Bluetooth	✗	✗	✓	✗	✗	
Calendar	!	✗	✓	✗	!	Titanium: Android only OpenPlug: not implemented on Android / iPhone
Camera	✓	✓	✓	✓	✓	
Contacts	✓	✓	✓	✗	✓	
Compass	✓	✓	✗	✗	✗	
Copy / Paste	✓	✓	✗	✓	✓	
Email	✓	✗	✓	✓	✗	
File System	✓	✓	✓	✓	✓	











Gestures / Multitouch	✓	✓	✗	✓	!	Rhodes: no information found Elips: No MultiTouch
Geolocation	✓	✓	✓	✓	✓	
Local and Remote Data	✓	✓	✓	✓	✓	
Native Maps	✓	✗	✓	✗	✗	
Orientation	✓	✓	✓	✓	✓	
Photo Gallery	✓	✗	✗	✓	✓	
Proximity Detector	✓	✗	✓	✗	✗	
Ringtones	✗	✗	✓	✗	✗	
SMS	!	✗	✓	✓	✓	Titanium: iPhone only, commercial extension module
Sounds	✓	✓	✓	✓	✓	
Sound recording	✓	✗	✓	✓	✗	
Vibration	✓	✓	✓	✗	✗	
Video Capture	✓	✗	✓	✓	✗	
<b>TOTAL</b>	<b>18</b>	<b>12</b>	<b>19</b>	<b>14</b>	<b>10</b>	

### 4.3 Other features

Native UI support, Native Code support, Development language (if JavaScript) and License were counted to the Total score. The reason for this is that the Native UI makes UI developing easier, Native Code support allows extending the capabilities of the framework with native code, the Client has already the most expertise in JavaScript (Development language), and the Open Source license helps cut down the development costs.

When comparing the frameworks by these features, Appcelerator Titanium and Rhodes have the best score. However, there are some advantages with non-native UI support, such as Adobe AIR for Mobile Devices has, for example the UI is exactly the same across platforms (see TABLE 4).

TABLE 4. Comparison: Other features (Lukasavage 2011a; Getting Started with Appcelerator 2011; Rhodes Introduction; Titanium Mobile Programming Guides; Extending the Rhodes Framework; ELIPS Studio: cross-platform native mobile application development; Tutorial: Implementing a native extension for iPhone & Android)

	Appcelerator Titanium	PhoneGap	Rhodes	AIR for Mobile Devices	OpenPlug Elips Studio	Notes !
Native UI support						PhoneGap, AIR: 3 <sup>rd</sup> party libraries required
Native code support [1]						
Development language	JavaScript (HTML, CSS)	JavaScript, HTML, CSS	Ruby (HTML, CSS, JavaScript)	ActionScript3 (MXML)	ActionScript3 (MXML)	

Interpreting	JavaScript mapped to native code	Rendered in WebView control	Runs a Ruby bytecode through bundled RubyVM interpreter	Adobe Air Runtime	AS3 compiled to C++, which compiled to standalone native application installer	
License	Open Source: Apache 2.0	Open Source: MIT	Dual License: Open Source: MIT; Closed source for commercial projects	Closed source, Flex SDK mostly Open Source	Closed source, free for individuals (ad-supported)	
<b>TOTAL</b>	<b>4</b>	<b>3</b>	<b>3</b>	<b>0</b>	<b>1</b>	

[1] Native code support is a feature that allows extending the built-in functionality by writing extension modules/libraries in the platform's native programming language and using the module with the framework's non-native code.

#### 4.4 Subjective comparison

All the features described in Chapter 1.3 (Research objective) are not objectively measurable. The subjective features in the Research objective are:

- The quality of framework documentation
- Fast development
- Development status (i.e. is under active development by firm business).



## **The quality of framework documentation**

Titanium Developer and Rhodes have reasonably good documentation. In addition to API documentations, they both offer code samples and tutorials. However, both suffer from some out-of-date documentation or non-documented features.

PhoneGap API documentation seems to be insufficient and partly out of date, since some of the existing API features were not found in the official documentation. One reason for that might be that PhoneGap is a Nitobi sponsored project, not a product by Nitobi, and mostly developed by an open source developers' community.

Adobe Flash Builder "Burrito" and the Flex SDK "Hero" have a considerable amount of documentation. The problem is that the documentation is fragmented around numerous different Adobe websites and is difficult to find, which may be caused by the fact that the product is still in beta-phase. The API documentation is centralized to one website, and it is comprehensive with possibilities to filter the content by runtimes and products.

OpenPlug Elips Studio API documentation is organized. However, it is not always easy to locate specific classes and methods, since the package name may not hint of the content classes of the package. Overall the API is the least extensive of the compared frameworks.

## **Fast development**

The definition of "fast development" is relative, since the experience of the frameworks always accelerates the development. Rather the question is: how do the framework and its tools help to construct code quickly? For example: how much code is needed to build a Tab Bar navigation system with multiple views to a mobile application?

With the compared frameworks, there is much variation how the views are created and handled. Rhodes has its strengths in this area, since it implements the Model-View-Controller architecture, which is a practical way to separate the business logic from views. Titanium has a large set of built-in native UI classes; therefore it is relatively easy to build navigation, views and native UI quickly by setting some variables.

Adobe Flex “Hero” and OpenPlug Elips Studio both are missing some core components of the native UI, which makes a standard mobile application development more difficult. On the other hand, Adobe’s systems are at their best when the native look and feel with the UI is not even the objective. The mobile applications built with Adobe Flex or OpenPlug Elips Studio should be concerned as applications that have a custom UI and the functionality that differs from other frameworks and the native mobile applications.

### **Development status**

All the compared cross-platform native frameworks are comparatively young: the frameworks are introduced between 2008 and 2011. Therefore the development of the cross-platform frameworks is still at an early stage, and the pace of development is fast in all frameworks.

The Titanium Mobile is developed by Appcelerator Inc., which acquired Aptana and will release an updated version with an integrated IDE. It seems that the company will continue the active development of the framework, and focus on selling more support services and commercial extension modules to the framework. The development may lead to commercial proprietary software framework, as the framework grows with an IDE and other extensions.

According to PhoneGap the PhoneGap framework “will always remain free and open source under an MIT license” (PhoneGap FAQ). Since the PhoneGap is developed by an Open Source developer company, and sponsored by Nitobi Company, the development will surely continue. However, the changes may happen slowly, since the PhoneGap’s goal is to implement standards of developing W3C Device API Group and HTML5.

Rhodes, Adobe Flash Builder / Flex SDK, and OpenPlug Elips Studio are commercial products of companies behind them, and the development status seems to be firm. In particular, Adobe has massive resources for its framework development. OpenPlug competes in the same market with Adobe, since the OpenPlug’s Elips Studio is a plug-in for Adobe Flash Builder, and Adobe has also the same goal to extend the mobile application development functionality to its products. Because of the resources Adobe has, and the fact that Adobe Flash Builder is a commercial product by Adobe, OpenPlug Elips Studio may not have so bright future as the other compared frameworks.

#### **4.5 Results of the cross-platform native frameworks comparison**

In TABLE 5, the result scores of the different comparison areas are summarized. The underlined numbers present the best framework on each category.

TABLE 5. Results of the cross-platform native frameworks comparison

	Appcelerator Titanium	PhoneGap	Rhodes	AIR for Mobile Devices	OpenPlug Elips Studio
Smartphone platform support	2	<u>5</u>	3	1	3
Device APIs	18	12	<u>19</u>	14	10
Other features	<u>4</u>	3	3	0	1
<b>TOTAL</b>	<b>24</b>	<b>20</b>	<b>25</b>	<b>15</b>	<b>14</b>

When comparing the number of supported smartphone platforms, PhoneGap has the widest support. PhoneGap seems to be the best framework choice if it is planned to develop a lightweight web based mobile application, and to release it to as many platforms as possible in order to reach the maximum audience for the application. The situation will be even better for PhoneGap if the Windows Phone 7 support will come to the framework.

If the application needs an advanced device API support, Rhodes and Titanium are frameworks that most likely have support for the required features, since their Device API support is widest.

In other areas, such as the expandability of the framework and the native UI support, Titanium has the best support. But if a native look and feel is not desired, other frameworks also have their advantages.

By the comparison numbers, Rhodes and Titanium seem to have the best support for the Client's needs. And if those two are compared, Titanium has some advantages more, such as JavaScript as a development language, since the Client does not have Ruby knowledge at the moment.

However, the final decision and discussion of the appropriate cross-platform native frameworks is done in Chapter 6, after some comparison to the native mobile application frameworks and some critical point of view to the cross-platform frameworks.

## **5 CRITICAL ANALYSIS OF CROSS-PLATFORM MOBILE PROGRAMMING**

The cross-platform mobile programming is such a new phenomenon that it has achieved almost a *hype*-status. Therefore there has not been much criticism in the used sources. An objective discussion has been difficult to maintain, since the main sources of information have been official websites of the frameworks, which have been mostly written with extravagant advertising language.

Because the idea of cross-platform mobile programming is new, there is not either much literature sources available. Those few sources have been written between 2008 and 2010, and they do not provide sufficient in-depth analysis of the cross-platform frameworks in order to be critical. The lack of criticism in the literature sources may also be due to the novelty of cross-platform mobile programming; the frameworks were even less developed at the time of writing.

### **5.1 Disadvantages of the cross-platform mobile programming**

The cross-platform native frameworks are marketing their frameworks with the idea of “code once, run everywhere”. Only practical testing of each framework on multiple platforms with some demo applications, as in the example projects of Chapter 3, raise the omissions, failures and problems of the framework to be seen. Some evidence of the frameworks’ problems can also be seen in the developers’

discussion forums of the framework. If there are many bug reports and questions about the basic functionalities of the framework, the quality of the product can be questioned.

The advertised “code once, run everywhere” idea is not the entire case, as some customizations are always needed for each targeted platform. In some cases only some minor changes to the UI are enough, in some cases some API features are not implemented for all platforms, or are functioning differently.

PhoneGap has one of the best “code once, run everywhere” implementations, since only minor UI changes were needed across platforms in the demo application programming in Chapter 3.3.2. The advantage for PhoneGap UI is that it does not try to implement native UI elements and instead relies on HTML/CSS user interfaces that are similar on each platform. Titanium has much of platform-specific classes and functions, even with the UI creation, that leads to using of conditional clauses in programming (i.e. *if platform is Android, then do something*). However, Titanium relies on native UI on each platform, while PhoneGap offers a uniform UI experience across platforms. Eventually the decision needs to be made: is the native or uniform UI wanted? If the answer is native, platform-specific customizations are more easily done than rewriting an entire application for each platform in its native programming language.

A larger problem is that all the cross-platform frameworks are still in their infancy. They still have lot of bugs, undocumented features, and constant changes to the framework, which may drop the backward compatibility of the developed application and cause rewriting applications to the new versions of the SDKs. They are unstable, and usually have very limited memory management and debugging possibilities. They usually do not have multitasking or threading capabilities.

Many of the listed problems will probably be fixed as the frameworks evolve, but unfortunately it seems that under hard competition cross-platform framework developers are more interested in adding new features to their frameworks than developing frameworks' stability (Lukasavage 2011b).

An unstable framework with limited debugging is a dangerous combination, since it can lead to programming by trial and error. As a developer cannot be sure if a bug is in the application code or in the framework, the bug is difficult to debug and can lead to delayed timelines of projects and worsened quality.

Since the cross-platform frameworks are an "extra layer" in a programming stack, it naturally affects the application performance, which is weaker with cross-platform frameworks than with native frameworks. Usually the performance is not a problem in lightweight applications that do not use large amounts of data, memory or processor, and are graphically simple. Therefore the cross-platform frameworks are usually not intended for games or other heavyweight mobile applications, but are at their best with applications that are web-based and use web-based data to present simple text and image based data to the user.

A risk particularly for the cross-platform iOS based developing is that Apple has changed their policies for applications generated with third party tools several times. Some tools, for example iOS applications made with Adobe Flash, have been banned from Apple's App Store. Today Apple allows third party tools for their applications, but is not guaranteed that this would not change in the future. If Apple restricts third party tools, it is possible that some cross-platform frameworks will become useless or less useful.

All the cross-platform frameworks introduced in this thesis have weaknesses. Many of them are result of incompleteness of the frameworks. In the years to come, the cross-platform frameworks will evolve, but at the moment using cross-platform

frameworks for mobile programming should be seriously considered. The frameworks may save time and costs, but may weaken the quality of the end product.

## 5.2 Cross-platform versus native applications – a comparison

In TABLE 6, Cross-platform frameworks are compared against native frameworks by analyzing advantages and disadvantages of each mobile programming method. With the comparison it is possible to analyze which choice is the best choice for different programming needs.

TABLE 6. Cross-platform versus native applications – a comparison (Douangboupha 2009, 5)

	<b>Native</b>	<b>Cross-Platform</b>
<b>Advantages</b>	<ul style="list-style-type: none"> <li>• Library updates</li> <li>• Stable</li> <li>• Performance</li> <li>• Code size</li> <li>• App store compatibility</li> <li>• Native UI</li> <li>• Full native API</li> </ul>	<ul style="list-style-type: none"> <li>• One programming language for all platforms</li> <li>• Common programming language (e.g. JavaScript)</li> <li>• Common UI design across platforms</li> <li>• Fast development</li> <li>• Multi-platform development</li> </ul>
<b>Disadvantages</b>	<ul style="list-style-type: none"> <li>• Different programming languages, SDKs and tools for each platform</li> <li>• Different UI design</li> <li>• Slow development</li> <li>• Not all are Open Source</li> </ul>	<ul style="list-style-type: none"> <li>• Unstable</li> <li>• Code size</li> <li>• Performance</li> <li>• Lack of multitasking</li> <li>• Debugging</li> <li>• Limited API features</li> <li>• Bugs</li> <li>• Quality</li> <li>• Documentation</li> <li>• UI design in some frameworks</li> </ul>



Summarizing TABLE 6, native application programming is the best choice for applications that require stability (enterprise applications, branded applications), performance (games), full API features (applications that use device APIs not found from the cross-platform frameworks), and native look and feel for the application. It may also be concluded that native mobile application development is worthwhile if the developer has enough financial and time resources available, and the programming skills for the targeted platforms.

Cross-platform development is a good choice, if a developer has no native programming skills, the schedule is tight and there is not a great amount of financial resources available for the development. Cross-platform is also a good choice if the application uses much web-based data or shares resources with a website, or it is a quite lightweight application that does not require much of the smartphone's hardware resources.

## 6 RESULTS AND ANALYSIS

### 6.1 Choosing the framework

The thesis introduced multiple cross-platform native frameworks that would fit as a part of Client's mobile application strategy. By the results of the comparison in Chapter 4, the "optimal solution" for the cross-platform mobile programming for the Client is **Appcelerator Titanium** in the Client's current situation.

The reasons for the choice are Titanium's advanced API support, Native UI, JavaScript as development language, and the possibility to publish native applications. In particular, the development status of Titanium seems promising, and as they acquired Aptana and released a first preview version of Titanium Studio, an integrated IDE with Titanium SDK, the development of the framework is going to the

right direction as the code completion makes programming easier and the debugging possibilities have significantly increased.

However, there are also risks that must be taken into account when using Titanium in productive use for the Client's mobile projects. The most significant risk is the incompleteness of the framework. The framework is under constant changes, since there are frequently major updates to the framework. The result of the framework's current development situation is instability and bugs in the framework. The risks can be managed by sufficiently long delivery times for mobile projects made with Titanium, and by using native frameworks for time-critical or large scale mobile projects. In addition, buying Titanium official support services is an option to be considered.

The supporting framework in the Client's current situation is **PhoneGap**. In particular, PhoneGap has the widest platform support and will have a support for the Windows Phone 7 in the future. To have a possibility to develop applications for iOS, Android and Windows Phone 7 would be an advantage to the Client. PhoneGap would also fit to applications that do not require native look and feel, such as applications that have more "branded" UI and therefore custom UI.

PhoneGap has almost similar disadvantages and risks in productive use as Titanium. The development pace of the framework does not seem to be as fast as it is with Titanium and instead, PhoneGap seems to be more stable. However, since PhoneGap does not have such comprehensive API coverage as Titanium, PhoneGap fits better to more lightweight web-based applications with custom UI. The risks can be managed by using PhoneGap only for lightweight applications, and in large-scale applications, buying PhoneGap official support services.

In the future, when the Client's mobile business evolves, there can be multiple framework choices. If the mobile businesses of the Client evolved quickly, and there

would be enough resources for the large-scale mobile development, the native development would be the most recommended choice, since the cross-platform frameworks still have their limitations, as analyzed in Chapter 5.

However, if the cross-platform frameworks evolve, as they likely will, good choices for the cross-platform developing from today's perspective would also be Rhodes, and Adobe AIR/Flex frameworks for mobile devices.

Rhodes has an advanced Model-View-Controller architecture and a support for synchronized offline data. If the Client had Ruby knowledge or developers with experience from Ruby-on-rails, the Rhodes framework would be a good solution particularly for enterprise mobile application development.

The Adobe's AIR and Flex framework for mobile is probably the newest player on the market. The framework still has much to be developed, but Adobe's mobile solutions would be a good solution for the Client's mobile application strategy in the future, in particular for the multimedia-driven applications such as games and applications that contain rich media (video, audio, pictures).

## **6.2 Future of the cross-platform mobile programming**

It is said: "if you want to understand today, you have to search yesterday" (Quote by Pearl Buck). The history has seen a cross-platform evolution before, in the 1990s in desktop computing. Microsoft Windows and Apple Macintosh provided GUI platforms for desktop computers, and software vendors needed to create applications for both platforms. Cross-platform desktop frameworks abstracted the differences, making it easier to develop applications that ran across platforms. (Allen et al. 2010, 9.)

However, the Cross-platform desktop applications, such as popular Java applications, were never a breakthrough. In the 2000s more applications moved to the Web and cross-platform libraries and web based cross-platform libraries and frameworks were created, such as jQuery (Op. cit. p. 9). The Web became a winner with the rise of cloud computing and RIAs (Rich Internet Applications) and other web applications.

If history repeats itself, the cross-platform native frameworks will be just a staging post in the evolution of cross-platform mobile programming, as the Web based applications will eventually conquer the mobile landscape as well. From that perspective the evolution of the Web and HTML5 in particular should be taken seriously.

With the ongoing evolution of the Web, smartphones and tablets, the traditional desktop computing is moving towards mobility. Software development will shift towards mobile development and the software industry will be more mobilized (Op. cit. p. 1). With mobile devices, such as smartphones and tablets, it is easier and faster to do most of the tasks that are done with desktop computers today such as social media, emails, communication, web browsing, graphics and even word processing.

Totally new kind of applications will evolve, as even people who rarely use computer and Internet today will use a variety of services through mobile devices. Only imagination is the limit, since technological possibilities are enormous. Mobile devices would also have very advanced useful applications alongside entertainment applications. For example, in elderly care there would be applications for telemedicine and monitoring. People could have video calls with doctors, and send them wirelessly medical data, such as blood pressure data from a blood pressure monitor attached to the mobile device.

The evolution of mobile devices is so convincing that there will certainly be a huge demand for mobile solutions and companies that offer them. Therefore, this thesis

raises a need for further research: how will the mobile web landscape and the cross-platform mobile programming evolve in the future? How to be a part of the rapidly evolving mobile landscape? What kind of innovations would there be in the mobile programming? It seems to be clear that this thesis was written in the beginning of major changes of what we have used to consider as mobile and desktop computing.

## REFERENCES

Adobe AIR. 2011. Product page on Adobe website. Accessed on 22 March 2011.  
<http://www.adobe.com/products/air/>.

Adobe AIR Developer Center. 2011. Adobe website. Accessed on 23 March 2011.  
<http://www.adobe.com/devnet/air.html>.

Adobe Flex SDK "Hero". 2011. Adobe Labs / Technologies website. Accessed on 23 March 2011. [http://labs.adobe.com/technologies/flexsdk\\_hero/](http://labs.adobe.com/technologies/flexsdk_hero/).

Ahokas, K. 2011. Yritä jotain uutta. Article on Tietoviikko magazine, number 5, March 18, 2011.

Allen, S., Graupera, V. & Lundrigan, L. 2010. Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution. 1<sup>st</sup> Ed. New York: Apress.

Appcelerator Community Questions&Answers. n.d. A community forum on Appcelerator Developer Center website. Accessed on 30 March 2011.  
<http://developer.appcelerator.com/questions/>.

App Hub. 2011. How it works. The Windows Phone developer site. Accessed on 9 February 2011. [http://create.msdn.com/en-us/home/about/how\\_it\\_works](http://create.msdn.com/en-us/home/about/how_it_works).

Beta ActionScript 3.0 Reference. 2011. API documentation on Adobe Help website. Accessed on 31 March 2011.  
[http://help.adobe.com/en\\_US/FlashPlatform/beta/reference/actionscript/3/class-summary.html](http://help.adobe.com/en_US/FlashPlatform/beta/reference/actionscript/3/class-summary.html).

Butler, M. 2011. "Android: Changing the Mobile Landscape," Pervasive Computing, IEEE , vol.10, no.1, pp.4-7, Jan.-March 2011. doi: 10.1109/MPRV.2011.1. Accessed on 21 January 2011. <http://www.jamk.fi/library>, Nelli-portal, IEEE Xplore - IEEE/IEE Electronic Library.

Cantrell, C. 2011. iOS features in Adobe AIR 2.6. Article on Adobe Developer Connection website. Accessed on 23 March 2011.  
[http://www.adobe.com/devnet/air/articles/ios\\_features\\_in\\_air26.html](http://www.adobe.com/devnet/air/articles/ios_features_in_air26.html).

- Charland, A. 2010. Blog on Nitobi Website. Accessed on 5 March 2011.  
[Http://blogs.nitobi.com/andre/index.php/2010/11/09/build-phonegap-com-private-beta-opening/](http://blogs.nitobi.com/andre/index.php/2010/11/09/build-phonegap-com-private-beta-opening/).
- Corlan, M. 2011. Developing for mobile devices with the Adobe Flash Platform. Article on Adobe Developer Connection website. Accessed on 23 March 2011.  
[Http://www.adobe.com/devnet/devices/articles/mobile-development-flash-platform.html](http://www.adobe.com/devnet/devices/articles/mobile-development-flash-platform.html).
- Corona FAQ. n. d. Frequently Asked Questions on Anscamobile website. Accessed on 28 March 2011. [Http://www.anscamobile.com/corona/faq/](http://www.anscamobile.com/corona/faq/).
- Definition of smartphone. n.d. Encyclopedia. PC Magazine website. Accessed on 21 January.  
[Http://www.pcmag.com/encyclopedia\\_term/0,2542,t=Smartphone&i=51537,00.asp](http://www.pcmag.com/encyclopedia_term/0,2542,t=Smartphone&i=51537,00.asp).
- Device Capabilities. n.d. Rhomobile API Documentation. Accessed on 20 March 2011.  
[Http://docs.rhomobile.com/rhodes/device-caps](http://docs.rhomobile.com/rhodes/device-caps).
- Devitt, S., Meeker, M. & Wu, L. 2010. Internet Trends. Morgan Stanley Research. Accessed on 9 January 2011.  
[Http://www.morganstanley.com/institutional/techresearch/pdfs/Internet\\_Trends\\_041210.pdf](http://www.morganstanley.com/institutional/techresearch/pdfs/Internet_Trends_041210.pdf).
- Douangboupha, P. 2009. Smart Phone Platform Comparison. A Comparative Analysis. Presentation. Accessed on 4 April 2011.  
[Http://www.r2integrated.com/Portals/21/PDFs/SmartPhoneComparison.pdf](http://www.r2integrated.com/Portals/21/PDFs/SmartPhoneComparison.pdf).
- ELIPS Studio API Reference. n.d. Documentation of ELIPS Studio mobile Flex framework. OpenPlug Developer Zone website. Accessed on 30 March 2011.  
[Http://developer.openplug.com/code/api](http://developer.openplug.com/code/api).
- ELIPS Studio: cross-platform native mobile application development. n. d. OpenPlug Developer Website. Accessed on 21 March 2011.  
[Http://developer.openplug.com/product/overview](http://developer.openplug.com/product/overview).
- ELIPS Studio FAQ. n. d. Frequently Asked Questions about ELIPS Studio on OpenPlug Developer website. Accessed on 24 March 2011.  
[Http://developer.openplug.com/product/faq](http://developer.openplug.com/product/faq).
- Extending the Rhodes Framework. n.d. Article on Rhomobile Documentation. Accessed on 30 March 2011. [Http://docs.rhomobile.com/rhodes/extensions](http://docs.rhomobile.com/rhodes/extensions).

Firtman, M. 2010. Programming the Mobile Web. O'Reilly Media, Inc. Sebastopol, CA.

Fling, B. 2009. Mobile Design and Development. O'Reilly Media, Inc. Sebastopol, CA.

Getting Started with Appcelerator. 2011. Documentation on Appcelerator website. Accessed on 20 February 2011.

[Http://wiki.appcelerator.org/display/guides/Getting+Started+with+Titanium](http://wiki.appcelerator.org/display/guides/Getting+Started+with+Titanium).

Get Started. 2011. PhoneGap Getting Started tutorial. Accessed on 27 February 2011.

[Http://www.phonegap.com/start](http://www.phonegap.com/start).

Grabham, D. 2010. Intel and Nokia merge Moblin and Maemo to form MeeGo. Article on TechRadar. Accessed on 16 February 2011.

[Http://www.techradar.com/news/phone-and-communications/mobile-phones/intel-and-nokia-merge-moblin-and-maemo-to-form-meego-670302](http://www.techradar.com/news/phone-and-communications/mobile-phones/intel-and-nokia-merge-moblin-and-maemo-to-form-meego-670302).

Gupta, P. 2011. HP announced three webOS products: a tablet PC TouchPad [first webOS tablet PC of HP], and two smartphones Pre 3, and Veer. Article on Gadget Reviews. Accessed on 20 February 2011. [Http://www.gadgetsreviews.com/hp-announced-three-webos-products-touchpad-tablet-pc-first-webos-running-tablet-pc-and-pre-3-and-veer-smartphones/7321.html](http://www.gadgetsreviews.com/hp-announced-three-webos-products-touchpad-tablet-pc-first-webos-running-tablet-pc-and-pre-3-and-veer-smartphones/7321.html).

Hashimi, S., Komatineni, S. & MacLean, D. 2010. Pro Android 2. New York: Apress.

Haynie, J. 2011. Appcelerator Acquires Leading IDE Aptana. Appcelerator blog.

Accessed on 20 February 2011. [Http://www.appcelerator.com/appcelerator-acquires-ide-aptana.html](http://www.appcelerator.com/appcelerator-acquires-ide-aptana.html).

Introducing PhoneGap Build. n.d. PhoneGap Build official website. Accessed on 5 March 2011. [Https://build.phonegap.com/](https://build.phonegap.com/).

Introducing the Adobe Flash Platform. 2011. Article on Adobe Developer Connection website. Accessed on 23 March 2011.

[Http://www.adobe.com/devnet/flashplatform/articles/flashplatform\\_overview.html](http://www.adobe.com/devnet/flashplatform/articles/flashplatform_overview.html).

iOS Developer Program. 2011. Apple Developer Website. Accessed on 27 February 2011. [Http://developer.apple.com/programs/ios/](http://developer.apple.com/programs/ios/).

iOS Dev Center. 2011. Apple Developer Website. Accessed on 3 March 2011.

[Http://developer.apple.com/devcenter/ios/index.action](http://developer.apple.com/devcenter/ios/index.action).



Jaramillo, N. 2011. Mobile development using Adobe Flex SDK "Hero" and Flash Builder "Burrito". Article on Adobe Developer Connection website. Accessed on 23 March 2011.

[Http://www.adobe.com/devnet/flex/articles/mobile\\_development\\_hero\\_burrito.html](http://www.adobe.com/devnet/flex/articles/mobile_development_hero_burrito.html).

KitchenSink. 2011. Demo application for Titanium Mobile. Source code. Accessed on 27 February 2011. <https://github.com/appcelerator/KitchenSink>.

Kolakowski, N. 2011. Windows Phone 7 Has Chance of 2011 Success: Analyst. Article on eWeek.com. Accessed on 9 February 2011. <http://www.eweek.com/c/a/Mobile-and-Wireless/Windows-Phone-7-Has-Chance-of-2011-Success-Analyst-166874/>.

Kumarak, G. 2010. Apple sold 14.1 million iPhones last quarter, over 70 million since launch. Article on MobileCrunch. Accessed on 4 February 2011.

<http://www.mobilecrunch.com/2010/10/18/apple-sold-14-1-million-iphones-last-quarter-over-70-million-since-launch/>.

Laugesen, J. & Yuan, Y. 2010. "What Factors Contributed to the Success of Apple's iPhone?," Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on, vol., no., pp.91-99, 13-15 June. Accessed on 4 February 2011. <http://www.jamk.fi/library>, Nelli-portal, IEEE Xplore - IEEE/IEE Electronic Library.

Lee, H. & Chuvyrov, E. 2010. Beginning Windows Phone 7 Development. 1st Ed. New York: Apress.

Look Ahead 2011. 2011. Advertising Age, 82, 2, pp. 4-21, Business Source Elite, EBSCOhost. Accessed on 20 February 2011. <http://www.jamk.fi/library>, Nelli-portal, EBSCO Business Source Elite.

Lukasavage, T. 2011. Appcelerator vs. PhoneGap vs. Adobe Air. Comparison article on Savagelook.com. Accessed on 30 March 2011.

<http://savagelook.com/blog/portfolio/appcelerator-vs-phonegap-vs-adobe-air>.

Lukasavage, T. 2011. Are You Actually Saving Time With Mobile Frameworks? Blog article. Accessed on 1 April 2011. <http://savagelook.com/blog/portfolio/are-you-actually-saving-time-with-mobile-frameworks>.

Muschenetz, I. 2011. Titanium Studio 1.0 Preview with Titanium Mobile Debugging. Article on Appcelerator Developer Blog. Accessed on 8 March 2011.

<http://developer.appcelerator.com/blog/2011/04/titanium-studio-1-0-preview-with-titanium-mobile-debugging.html>.

Niclot, E. 2010. About UICatalog sample and platform variant management. OpenPlug developer zone blog. Accessed on 27 March 2011.

<http://developer.openplug.com/about/blog/193-about-uicatalog-sample-and-platform-variant-management->.

Nokia outlines new strategy, introduces new leadership, operational structure. 2011. Nokia Press Releases. Accessed on 16 February 2011.

<http://www.nokia.com/press/press-releases/showpressrelease?newsid=1488004>.

Nourie, D. 2005. Getting Started with an Integrated Development Environment (IDE). Article on Sun Developer Network (SDN). Accessed on 4 February 2011.

<http://java.sun.com/developer/technicalArticles/tools/intro.html>.

OpenPlug Support Forum. n.d. Developer forum on OpenPlug Developer Zone website. Accessed on 30 March 2011. <http://developer.openplug.com/forum/>.

Peltomäki, J. 2010. Introduction to mobile computing. Material for the cross-platform mobile programming course on Fall 2010.

Peltomäki, J. 2010. Mobile networking and location based systems. Material for the cross-platform mobile programming course on Fall 2010.

Pettey, C. & Stevens, H. 2011. Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012. Gartner Press Releases website. Accessed on 10 April 2011.

<http://www.gartner.com/it/page.jsp?id=1622614>.

PhoneGap API Reference. n.d. PhoneGap Documentation website. Accessed on 5 March 2011. <http://docs.phonegap.com/>.

PhoneGap Build FAQ. n.d. PhoneGap Build Frequently Asked Questions article on PhoneGap Build website. Accessed on 5 March 2011.

<https://build.phonegap.com/docs/faq>.

PhoneGap FAQ. n.d. PhoneGap Frequently Asked Questions on PhoneGap website. Accessed on 31 March 2011. <http://www.phonegap.com/faq>.

Rhodes Device Capabilities. n.d. Rhodes Developer Reference. Accessed on 30 March 2011. <http://docs.rhobile.com/rhodes/device-caps>.

Rhodes Introduction. n.d. Rhodes Developer Reference. Accessed on 10 March 2011. [Http://docs.rhobile.com/rhodes/introduction](http://docs.rhobile.com/rhodes/introduction).

RhoHub: Hosted Development for Mobile Apps. 2011. RhoHub official website. Accessed on 11 March 2011. [Http://rhohub.com/](http://rhohub.com/).

RhoSync App Integration Server. n.d. Rhomobile website. Accessed on 10 March 2011. [Http://rhobile.com/products/rhosync/](http://rhobile.com/products/rhosync/).

RhoSync Features. n.d. Rhomobile website. Accessed on 10 March 2011. [Http://rhobile.com/mobile-solutions/](http://rhobile.com/mobile-solutions/).

Stark, J. 2010. Building iPhone Apps with HTML, CSS and JavaScript. 1<sup>st</sup> Edition. Sebastopol: O'Reilly Media, Inc.

Strohmeier, R. & Perenson, M. 2011. 'WHY YOUR NEXT PC WILL BE A TABLET'. PC World, 29, 1, pp. 73-80, Business Source Elite, EBSCOhost. Accessed on 20 February 2011. [Http://www.jamk.fi/library](http://www.jamk.fi/library), Nelli-portal, EBSCO Business Source Elite.

Subramaniam, D. 2011. Introducing Adobe Flex 4.5 SDK. Article on Adobe Developer Connection website. Accessed on April 11, 2011. [Http://www.adobe.com/devnet/flex/articles/introducing-flex45sdk.html](http://www.adobe.com/devnet/flex/articles/introducing-flex45sdk.html).

Supported Platforms and App Stores. n.d. List of platforms that ELIPS Studio supports. Article on OpenPlug Developer Zone website. Accessed on 30 March 2011. [Http://developer.openplug.com/learn/platforms](http://developer.openplug.com/learn/platforms).

Testing your Applications. 2011. Documentation on Appcelerator website. Accessed on 27 February 2011. [Http://guides.appcelerator.com/en/testing.html](http://guides.appcelerator.com/en/testing.html).

The Application Project Structure. 2011. Documentation on Appcelerator website. Accessed on 27 February 2011. [Http://guides.appcelerator.com/en/app\\_structure.html](http://guides.appcelerator.com/en/app_structure.html).

Titanium Mobile API Reference. n. d. Titanium Mobile API reference for Titanium Mobile 1.6.1. Appcelerator Developer Center website. Accessed on 30 March 2011. [Https://developer.appcelerator.com/apidoc/mobile/latest](https://developer.appcelerator.com/apidoc/mobile/latest).

Titanium Mobile Programming Guides. n.d. Article on Appcelerator Developer Center website. Accessed on 30 March 2011. [Http://developer.appcelerator.com/doc/mobile/guides](http://developer.appcelerator.com/doc/mobile/guides), Module Development.

Tour de Mobile Flex. n.d. An Android application for exploring the new Flex mobile capabilities. Adobe Flex website. Accessed on 30 March 2011.  
[Http://flex.org/tourmobile](http://flex.org/tourmobile).

Tutorial: [Hello World] Using new project templates. n.d. Article on OpenPlug developer zone website. Accessed on 27 March 2011.  
[Http://developer.openplug.com/index.php/learn/tutorials/?videoid=35](http://developer.openplug.com/index.php/learn/tutorials/?videoid=35).

Tutorial: Implementing a native extension for iPhone & Android. n.d. Article on OpenPlug Developer zone website. Accessed on 31 March 2011.  
[Http://developer.openplug.com/learn/tutorials/?videoid=27](http://developer.openplug.com/learn/tutorials/?videoid=27).

Udell, S. 2009. Pro Web Gadgets Across iPhone, Android, Windows, Mac, iGoogle and More. New York: Apress.

Using Hardware Devices. 2011. Android developers guide. Accessed on 27 February 2011. [Http://developer.android.com/guide/developing/device.html](http://developer.android.com/guide/developing/device.html).

What is MoSync. n. d. Article on MoSync website. Accessed on 28 March 2011.  
[Http://www.mosync.com/what-is-mosync](http://www.mosync.com/what-is-mosync).

Whinnery, K. 2011a. Titanium For New Developers. Presentation on Appcelerator website. Accessed on 27 February 2011.  
[Http://developer.appcelerator.com/blog/2011/01/titanium-for-new-developers.html](http://developer.appcelerator.com/blog/2011/01/titanium-for-new-developers.html).

Whinnery, K. 2011b. What's New In Titanium Mobile 1.5. Presentation on Appcelerator website. Accessed on 27 February 2011.  
[Http://developer.appcelerator.com/blog/2011/01/whats-new-in-titanium-1-5.html](http://developer.appcelerator.com/blog/2011/01/whats-new-in-titanium-1-5.html).

## APPENDICES

### Appendix 1. Code listing: Contacts application: Titanium

```

var win = Ti.UI.currentWindow;

var imageDisplay = null;

var contactinfoLabel = Ti.UI.createLabel({
    top:120,
    width:'auto',
    height:'auto',
    color: '#000'
});

var btn_contactSelect = Ti.UI.createButton({
    title:'Select Contact',
    bottom:20,
    width:200,
    height:40
});

btn_contactSelect.addEventListener('click', function() {
    Titanium.Contacts.showContacts({
        selectedPerson: function(e) {

            var image = e.person.image;
            if (imageDisplay != null) {
                imageDisplay.image = null;
            }

            contactinfoLabel.text = e.person.fullName + '\n';

            contactinfoLabel.text += '\n Phone:\n'

            for (var label in e.person.phone) {
                var phones = e.person.phone[label];
                for (var i = 0; i < phones.length; i++) {
                    contactinfoLabel.text += label+' : '+ phones[i] +
                        '\n';
                }
            }

            contactinfoLabel.text += '\n Address:\n'

            for (var label in e.person.address) {
                var addrs = e.person.address[label];
                for (var i = 0; i < addrs.length; i++) {
                    if (Titanium.Platform.name == 'android') {
                        contactinfoLabel.text += label + ': ' +
                            addrs[i].Street + '\n';
                    }
                    else {
                        contactinfoLabel.text += label + ': ' +
                            addrs[i].Street + '\n' + addrs[i].ZIP + ' ' +
                            addrs[i].City + '\n';
                    }
                }
            }
        }
    });
}

```

```
        }  
    }  
  
    if (imageDisplay == null) {  
        imageDisplay = Ti.UI.createImageView({  
            image:image,  
            width:100,  
            height:100,  
            top:20  
        });  
        win.add(imageDisplay);  
    }  
    else {  
        imageDisplay.image = image;  
    }  
}  
});  
});  
  
win.add(contactinfoLabel);  
win.add(btn_contactSelect);
```

## Appendix 2. Code listing: Contacts application: PhoneGap

### Index.html

```

<!doctype html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no,
width=device-width" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8">

<title>Contacts</title>
<link rel="stylesheet" href="css/master.css" type="text/css" media="screen"
/>

<script src="phonegap.0.9.4.js" type="text/javascript" charset="utf-
8"></script>

<script src="js/xui-2.0.0.min.js" type="text/javascript" charset="utf-
8"></script>

<script src="js/app.js" type="text/javascript" charset="utf-8"></script>

</head>
<body>
    <div id="title_bar"><strong>Contacts</strong> app</div>

    <div id="contactPicked" style="display:none"></div>

    <div id="searchdiv" class="view">
        <div id="searchboxdiv"><input type="text" value=""
id="contact_search"/> </div>
        <div class="app_button" id="contacts_button">Search Contact</div>
        <div class="app_button" id="clear_button">Clear</div>
    </div>

</body>
</html>

```

### app.js

```

x$(window).load(function(e) {
    x$('#contacts_button').on('touchstart', function () {
        var selectedstring =
            document.getElementById("contact_search").value;
        getContacts(selectedstring);
    });

    x$('#clear_button').on('touchstart', function () {
        window.location.reload();
    });

    function successContactFindCallback(contacts) {

```

```

x$('#contactPicked').setStyle('display', 'block');

for (var j in contacts[0].photos) {
    x$('#contactPicked').html('before', '<p></p>');
}

var namevar = contacts[0].name;
x$('#contactPicked').html('before', '<p>'+namevar.givenName
+ ' ' + namevar.familyName+'<br/>');

for (var j in contacts[0].phoneNumbers) {
    x$('#contactPicked').html('before',
    contacts[0].phoneNumbers[j].value+'<br/>');
}

for (var j in contacts[0].emails) {
    x$('#contactPicked').html('before',
    contacts[0].emails[j].value+'<br/>');
}

for (var j in contacts[0].addresses) {
    x$('#contactPicked').html('before',
    contacts[0].addresses[j].streetAddress + '<br/>' +
    contacts[0].addresses[j].postalCode + ' ' +
    contacts[0].addresses[j].locality+'<br/>');
}

x$('#contactPicked').html('before', '</p>');
}

function getContacts(selectedstring) {
    navigator.service.contacts.find(['photos', 'name',
    'addresses.streetAddress', 'addresses.postalCode',
    'addresses.locality', 'phoneNumbers', 'emails'],
    successContactFindCallback,
    generalErrorCB,
    {filter: selectedstring}
    );
}

function generalErrorCB(error) {
    alert(error.code);
}

});

```



## Appendix 3. Code listing: Contacts application, Rhodes

### person\_controller.rb:

```

require 'rho/rhocontroller'
require 'rho/rhocontact'
require 'helpers/browser_helper'

class PersonController < Rho::RhoController
  include BrowserHelper

  #GET /Person
  def index
    @people = Rho::RhoContact.find(:all)
    @people = @people.to_a.sort! { |x,y| x[1]['first_name'] <=>
      y[1]['first_name'] } if @people
  end

  # GET /Person/{1}
  def show
    @person = Rho::RhoContact.find(@params['id'])
    if @person
      render :action => :show
    else
      redirect :action => :index
    end
  end

  # GET /Person/new
  def new
    render :action => :new
  end

  # GET /Person/{1}/edit
  def edit
    @person = Rho::RhoContact.find(@params['id'])
    if @person
      render :action => :edit
    else
      redirect :action => :index
    end
  end

  # POST /Person/create
  def create
    @person = Rho::RhoContact.create!(@params['person'])
    redirect :action => :index
  end

  # POST /Person/{1}/update
  def update
    Rho::RhoContact.update_attributes(@params['person'])
    redirect :action => :index
  end

  # POST /Person/{1}/delete
  def delete
    Rho::RhoContact.destroy(@params['id'])
    redirect :action => :index
  end
end

```

```

    end
end

```

### index.erb

```

<div class="pageTitle">
  <h1>Contacts</h1>
</div>

<div class="toolbar">
  <div class="leftItem regularButton">
    <a href="<%= Rho::RhoConfig.start_path %>">Home</a>
  </div>
  <div class="rightItem regularButton">
    <a class="button" href="<%= url_for :action => :new %>">New</a>
  </div>
</div>

<div class="content">
  <ul>
    <% @people.each do |person| %>
      <li>
        <a href="<%= url_for :action => :show, :id => person[1]['id'] %>">
          <span class="title"><%= person[1]['first_name'] %> <%=
person[1]['last_name'] %></span><span class="disclosure_indicator"></span>
          </a>
        </li>
      <% end %>
    </ul>
  </div>

```

### show.erb

```

<div class="pageTitle">
  <h1><%= @person['first_name'] %></h1>
</div>

<div class="toolbar">
  <div class="leftItem backButton">
    <a href="<%= url_for :action => :index %>">Back</a>
  </div>
  <div class="rightItem regularButton">
    <a href="<%= url_for :action => :edit, :id => @person['id'] %>">Edit</a>
  </div>
</div>

<div class="content">
  <ul>
    <li>
      <div class="itemLabel">First Name</div>
      <div class="itemValue"><%= @person['first_name'] %></div>
    </li>
  </ul>

```

```

<li>
  <div class="itemLabel">Last Name</div>
  <div class="itemValue"><%= @person['last_name'] %></div>
</li>

<li>
  <div class="itemLabel">Mobile Number</div>
  <div class="itemValue"><%= @person['mobile_number'] %></div>
</li>

<li>
  <div class="itemLabel">Home Number</div>
  <div class="itemValue"><%= @person['home_number'] %></div>
</li>

<li>
  <div class="itemLabel">Business Number</div>
  <div class="itemValue"><%= @person['business_number'] %></div>
</li>

<li>
  <div class="itemLabel">Email</div>
  <div class="itemValue"><%= @person['email_address'] %></div>
</li>

<li>
  <div class="itemLabel">Home Email</div>
  <div class="itemValue"><%= @person['home_email_address'] %></div>
</li>

<li>
  <div class="itemLabel">Street Address</div>
  <div class="itemValue"><%= @person['street_address_1'] %></div>
</li>

<li>
  <div class="itemLabel">Postal Code and City</div>
  <div class="itemValue"><%= @person['zip_1'] %> <%=
    @person['city_1'] %></div>
</li>

</ul>
</div>

```

### edit.erb

```

<div class="pageTitle">
  <h1>Edit <%= @person['first_name'] %></h1>
</div>

<div class="toolbar">
  <div class="leftItem backButton">
    <a href="<%= url_for :action => :show, :id => @person['id']
      %>">Cancel</a>
  </div>
  <div class="rightItem regularButton">
    <a class="button" href="<%= url_for :action => :delete, :id =>
      @person['id'] %>">Delete</a>
  </div>
</div>

```

```

    </div>
</div>

<div class="content">
  <form method="POST" action="<%= url_for :action => :update %>">
    <input type="hidden" name="person[id]" value="<%= @person['id'] %>" />
    <ul>
      <li>
        <label for="person[first_name]" class="fieldLabel">First
          Name</label>
        <input type="text" name="person[first_name]" value="<%=
          @person['first_name'] %>" <%= placeholder( "First Name" ) %> />
      </li>

      <li>
        <label for="person[last_name]" class="fieldLabel">Last
          Name</label>
        <input type="text" name="person[last_name]" value="<%=
          @person['last_name'] %>" <%= placeholder( "Last Name" ) %> />
      </li>

      <li>
        <label for="person[mobile_number]"
          class="fieldLabel">Mobile</label>
        <input type="text" name="person[mobile_number]" value="<%=
          @person['mobile_number'] %>" <%= placeholder( "Mobile" ) %> />
      </li>

    </ul>
    <input type="submit" class="standardButton" value="Update" />
  </form>
</div>

```

## new.erb

```

<div class="pageTitle">
  <h1>New Contact</h1>
</div>

<div class="toolbar">
  <div class="leftItem backButton">
    <a class="cancel" href="<%= url_for :action => :index %>">Cancel</a>
  </div>
</div>

<div class="content">
  <form method="POST" action="<%= url_for :action => :create %>">
    <ul>
      <li>
        <label for="person[first_name]" class="fieldLabel">First
          Name</label>
        <input type="text" name="person[first_name]" <%=
          placeholder("First Name") %> />
      </li>

      <li>
        <label for="person[last_name]" class="fieldLabel">Last

```

```
        Name</label>
        <input type="text" name="person[last_name]" <%=
placeholder("Last Name") %> />
    </li>

    <li>
        <label for="person[mobile_number]"
class="fieldLabel">Mobile</label>
        <input type="text" name="person[mobile_number]" <%=
placeholder("Mobile") %> />
    </li>

</ul>
    <input type="submit" class="standardButton" value="Create"/>
</form>
</div>
```

## Appendix 4. Code listing: Google Maps with Geolocation, Adobe Flash Builder “Burrito”

### GoogleMapsExampleHome.mxml

```

<?xml version="1.0" encoding="utf-8"?>
<s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:s="library://ns.adobe.com/flex/spark" title="Google Maps example"
creationComplete="view1_creationCompleteHandler(event)"
xmlns:mx="library://ns.adobe.com/flex/mx">

<fx:Script>

<![CDATA[
    import com.eaglemediavision.gpsdata.GPSData;

    import com.google.maps.LatLng;
    import com.google.maps.Map;
    import com.google.maps.MapEvent;
    import com.google.maps.MapType;
    import com.google.maps.overlays.Marker;
    import flash.sensors.Geolocation;

    import mx.events.FlexEvent;

    private var gMap:Map;
    private var g:Geolocation;
    private var gpsdata:GPSData;

    protected function
    view1_creationCompleteHandler(event:FlexEvent):void {

    if (Geolocation.isSupported) {

        gpsdata = new GPSData();

        gMap = new Map();
        gMap.key = "";
        gMap.url = "http://www.eaglemediavision.com"
        gMap.sensor = "true"
        gMap.width = 480;
        gMap.height = 682;
        gMap.addEventListener(MapEvent.MAP_READY, mapReadyHandler);

        mapContainer.addChild(gMap);

        g = new Geolocation();
        g.setRequestedUpdateInterval(100);

        g.addEventListener(GeolocationEvent.UPDATE, geoLocation_UpdateHan
        dler, false,0,true);
    }
    }

    protected function

```

```

        geoLocation_UpdateHandler(event:GeolocationEvent):void {
            gpsdata.setLatitude(event.latitude);
            gpsdata.setLongitude(event.longitude);
        }

        private function mapReadyHandler(e:MapEvent):void {
            gMap.setCenter(new
                LatLng(gpsdata.getLatitude(),gpsdata.getLongitude()), 15,
                MapType.NORMAL_MAP_TYPE);
            gMap.setSize(new Point(mapContainer.width,
                mapContainer.height));

            var latlng:LatLng = new
                LatLng(gpsdata.getLatitude(),gpsdata.getLongitude());
            var marker:Marker= new Marker(latlng);
            gMap.addOverlay(marker);
        }

    ]]>
</fx:Script>

<mx:UIComponent id="mapContainer" width="100%" height="100%"/>
</s:View>

```

### GPSData.as

```

package com.eaglemediavision.gpsdata
{
    public class GPSData
    {
        private var latitude:Number;
        private var longitude:Number;
        private var altitude:Number;

        public function GPSData() { }

        public function setLatitude(latitude:Number):void {
            this.latitude = latitude;
        }

        public function setLongitude(longitude:Number):void {
            this.longitude = longitude;
        }

        public function setAltitude(altitude:Number):void {
            this.altitude = altitude;
        }

        public function getLatitude():Number {
            return this.latitude;
        }

        public function getLongitude():Number {
            return this.longitude;
        }
    }
}

```

```
    }  
    public function getAltitude():Number {  
        return this.altitude;  
    }  
}  
}
```



## Appendix 5. Code listing: Contacts application, Elips Studio

### ContactsElips.mxml

```

<?xml version="1.0" encoding="utf-8"?>
<mob:ScreenStackApplication xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute"
xmlns:mob="openplug.elips.controls.*"
xmlns:screen="openplug.elips.controls.screenClasses.*"
xmlns:components="components.*"
creationComplete="onCreationComplete()">

<mx:Script> <![CDATA[
    import mx.renderers.ListItemRenderer;

    import models.ContactExt;
    import models.FindContactsResult;

    import openplug.elips.events.ItemTouchTapEvent;
    import openplug.elips.services.AddressBook;
    import openplug.elips.services.Contact;
    import openplug.elips.services.ContactAddress;
    import openplug.elips.services.ContactEmail;
    import openplug.elips.services.ContactManager;
    import openplug.elips.services.ContactPhoneNumber;

    private var _contactManager:ContactManager;
    private var _addressBook:AddressBook;
    private var _loadContactsResult:FindContactsResult;

    //*****
    // INIT
    //*****

    private function onCreationComplete():void {
        var delayedLoadContacts:Timer = new Timer(5000, 1);
        delayedLoadContacts.addEventListener(TimerEvent.TIMER,
        loadContacts);
        delayedLoadContacts.start();
        goSearch.enabled = false;
        titleBar.titleTextSub = "Please wait while loading contacts...";
    }

    public function loadContacts(e:TimerEvent):void {
        _contactManager = new ContactManager();
        _addressBook = _contactManager.getAddressBooks()[0] as
        AddressBook; //get the first one (=internal one, not sim)
        _loadContactsResult = new FindContactsResult();
        _loadContactsResult.onSuccessFunction = loadContactsSuccess;
        _addressBook.findContacts(_loadContactsResult,
        loadContactsError, null);
    }

    public function loadContactsSuccess():void {
        titleBar.titleTextSub = "(Addressbook1 contains " +
        _loadContactsResult.arrcolContacts.length + " contacts)";
        listContacts.dataProvider = _loadContactsResult.arrcolContacts;
    }
    ]]>

```

```

        loadCont.stop();
        loadCont.visible = false;
        goSearch.enabled = true;
    }

    public function loadContactsError():void {
        titleBar.titleTextSub = "Error occured when searching for
        contacts!";
        loadCont.stop();
        loadCont.visible = false;
    }

    //*****
    // FILTER CONTACTS
    //*****

    private function filterContacts():void {
        _loadContactsResult.filterByName(textName.text);
        listContacts.dataProvider = _loadContactsResult.arrcolContacts;
    }

    public function openContact(evt:ItemTouchEvent):void {
        var contact:ContactExt = evt.item;
        var contacttext:String;

        var contactAddresses:Array = contact.addresses;
        var contactPhones:Array = contact.phoneNumbers.number;
        var contactEmails:Array = contact.emails;

        contacttext = "";

        contacttext += contact.name + "\n";

        if (contact.phoneNumbers != null) {
            var i:int;
            for (i=0; i<contact.phoneNumbers.length; i++) {
                var phoneNum:ContactPhoneNumber =
                contact.phoneNumbers[i];
                contacttext += phoneNum.type + ": " + phoneNum.number
                + "\n";
            }
        }

        if (contact.emails != null) {
            for (i=0; i<contact.emails.length; i++) {
                var email:ContactEmail = contact.emails[i];
                contacttext += email.type + ": " + email.email + "\n";
            }
        }

        if (contact.addresses != null) {
            for (i=0; i<contact.addresses.length; i++) {
                var address:ContactAddress = contact.addresses[i];
                contacttext += address.street + " " +
                address.postalCode + " " + address.city + "\n";
            }
        }

        singleContactText.text = contacttext;
        goToScreen("singleContactScreen");
    }

```

```

    }

  ]]>
</mx:Script>

<!-- Screen definition, you can have several screens in a screen stack -->
<mob:ScreenView id="allContactsScreen">

  <mx:VBox width="100%" height="100%">

    <components:TitleBar id="titleBar" titleTextMain="Contact
Manager" />

    <mx:VBox
      width="100%"
      paddingLeft="10"
      paddingRight="10"
      verticalGap="0"
    >

      <mx:Label
        width="100%"
        fontSize="10"
        text="Filter by Namepart (empty = show all):"
      />

      <mx:HBox
        width="100%"
        horizontalAlign="center"
      >
        <mob:TextInput id="textName"
          width="100%"
          height="40"
          fontSize="18"
        />

        <mx:Button
          height="40"
          label="Go"
          id="goSearch"
          fontSize="20"
          touchTap="filterContacts()"
        />
      </mx:HBox>
    </mx:VBox>

    <mob:List id="listContacts"
      width="100%"
      height="90%"
      rowHeight="80"
      backgroundColor="white"
      itemRenderer="itemRenderers.ListItemRenderer"
      itemTouchTap="openContact(event)"
    />

    <mob:ActivityIndicator x="{width/2}" y="{height/2}"
      width="25" height="25" id="loadCont" autoStart="false"/>

  </mx:VBox>

</mob:ScreenView>

  <mob:ScreenView id="singleContactScreen">

```

```

<mx:VBox
    width="100%"
    height="100%"
    >

    <components:TitleBar id="singleContactBar"
        titleTextMain="Contact"
    />

    <mx:VBox
        width="100%"
        paddingLeft="10"
        paddingRight="10"
        verticalGap="0"
        >

        <mx:Button
            height="40"
            label="Back"
            id="goback"
            fontSize="20"
            touchTap="goBack()"
            />

    </mx:VBox>

    <mx:Text
        width="100%"
        id="singleContactText"
        text=""
        textAlign="center"
        fontSize="20"
        color="white"
        />

</mx:VBox>

</mob:ScreenView>

</mob:ScreenStackApplication>

```

### FindContactResult.as

```

package models
{
    import mx.collections.ArrayCollection;

    import openplug.elips.services.Contact;
    import openplug.elips.services.ContactArraySuccessCallback;
    import openplug.elips.services.ContactPhoneNumber;

    public class FindContactsResult implements
    ContactArraySuccessCallback
    {
        private var _contacts:Array;
        private var _contactsFiltered:Array;
    }
}

```

```

public function FindContactsResult()
{
}

//*****
// GET CONTACTS FROM THE ADDRESSBOOK
//*****
public function onSuccess(contacts:Array):void
{
    var isAlternateColor:Boolean = true;
    _contacts = new Array();
    for each (var contact:Contact in contacts)
    {
        // Ensure the contact has a name
        if (contact.name != null)
        {
            var c:ContactExt = new ContactExt();
            c.id = contact.id;
            c.addresses = contact.addresses;
            c.emails = contact.emails;
            c.name = contact.name;
            c.nickNames = contact.nickNames;
            c.photo = contact.photo;
            c.phoneNumbers = contact.phoneNumbers;

            // If the contact has at least one phone number,
            // init firstPhoneNumber with the 1st number in
            // the list + its type
            if (c.phoneNumbers != null)
            {
                var phoneNum:ContactPhoneNumber =
                    c.phoneNumbers[0];
                c.firstPhoneNumber = phoneNum.number + "
                    " + phoneNum.type;
            }

            // alternate color flag handling
            c.alternateColorFlag = isAlternateColor;
            isAlternateColor = !isAlternateColor;

            // save the contact
            _contacts.push(c);
        }
    }
    arrcolContacts = new ArrayCollection(_contacts);

    // call the success function, if any...
    if (onSuccessFunction != null)
    {
        onSuccessFunction();
    }
}

//*****
// FILTERING CONTACTS
//*****
private var _searchPattern:String;
public function filterByName(searchPattern:String):void {
    var isAlternateColor:Boolean;
    var c:ContactExt;
    if (searchPattern.length > 0)
    {
        // filter
    }
}

```

```

        _searchPattern = searchPattern.toLowerCase();
        _contactsFiltered =
        _contacts.filter(doesNameContainSearchPattern, null);

        // set alternate color flag
        isAlternateColor = true;
        for each (c in _contactsFiltered)
        {
            c.alternateColorFlag = isAlternateColor;
            isAlternateColor = !isAlternateColor;
        }

        // create the array collection
        arrcolContacts = new
        ArrayCollection(_contactsFiltered);
    }
    else
    {
        // set alternate color flag
        isAlternateColor = true;
        for each (c in _contacts)
        {
            c.alternateColorFlag = isAlternateColor;
            isAlternateColor = !isAlternateColor;
        }

        // create the array collection

        arrcolContacts = new ArrayCollection(_contacts);
    }
}

private function
doesNameContainSearchPattern(element:ContactExt, index:int,
arr:Array):Boolean
{
    if (element.name != null)
    {
        var s:String = element.name.toLowerCase();
        return (s.search(_searchPattern) == -1 ? false : true);
    }
    else
    {
        return false;
    }
}

//*****
// PUBLIC PROPERTIES
//*****

private var _onSuccessFunction:Function;
public function get onSuccessFunction():Function
{
    return _onSuccessFunction;
}
public function set onSuccessFunction(value:Function):void
{
    _onSuccessFunction = value;
}

private var _arrcolContacts:ArrayCollection = new

```

```
ArrayCollection();  
public function get arrcolContacts():ArrayCollection  
{  
    return _arrcolContacts;  
}  
[Bindable] public function set  
arrcolContacts(value:ArrayCollection):void  
{  
    _arrcolContacts = value;  
}  
}
```