Comparing Native and Hybrid Applications with focus on Features

Felix Mohammadi Kho’i & Jawed Jahid
Contact Information:
Author(s):
Felix Mohammadi Kho‘i
felix@khoi.se
Jawed Jahid
Jawed.jahid@gmail.com

University advisor:
Conny Johansson
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
ABSTRACT
Nowadays smartphones and smartphone-applications are a part of our daily life. There are variety of different operating systems in the market that are unalike, which are an obstacle to developers when it comes to developing a single application for different operating system. Furthermore, hybrid application development has become a potential substitute. The evolution of new hybrid approach has made companies consider hybrid approach as a viable alternative when producing mobile applications. This research paper aims to compare native and hybrid application development on a feature level to provide scientific evidence for researchers and companies choosing application development approach as well as providing vital information about both native and hybrid applications.
This study is based on both a literature study and an empirical study. The sources used are Summon@BTH, Google Scholar and IEEE Xplore. To select relevant articles, the Snowballing approach was used, with Inclusion and Exclusion criteria’s.
The authors concluded that native development is a better way to develop more advanced applications which uses more device-hardware, while hybrid is a perfectly viable choice when developing content-centric applications.
Keywords: iPhone, Android, PhoneGap, Native, Hybrid, Mobile Applications
# CONTENTS

## ABSTRACT

1

## CONTENTS

1 INTRODUCTION 5

2 BACKGROUND 7

3 METHOD 8

3.1 RESEARCH QUESTIONS AND RESEARCH DESIGN 8

3.1.1 LITERATURE STUDY DESIGN 8

3.1.2 EMPirical STUDY DESIGN 9

4 RESULTS 11

4.1 LITERATURE REVIEW RESULTS 11

4.1.1 SNOWBALLING 11

4.1.2 THE RISE OF HYBRID APPLICATIONS 11

4.1.3 PHONEGAP 12

4.1.4 NATIVE APPLICATIONS 13

4.1.5 COMPARISON BETWEEN NATIVE AND HYBRID 13

4.1.6 PERFORMANCE COMPARISON BETWEEN NATIVE AND HYBRID APPLICATIONS 14

4.1.7 FEATURE COMPARISON BETWEEN NATIVE AND HYBRID 16

4.2 SURVEY RESULTS 19

4.2.1 SURVEY QUESTIONS 19

4.2.2 TARGET GROUP 19

4.2.3 RESPONSE 20

5 ANALYSIS 26

5.1 RQ1 – GPS AND MAP FEATURES 27

5.2 RQ2 – ACTIVITY FEATURES 27

5.3 RQ3 – MULTI-TOUCH FEATURES 27

5.4 GPS AND MAP ANALYSIS 27

5.5 ACTIVITY ANALYSIS 28

5.6 MULTI-TOUCH ANALYSIS 28

6 CONCLUSION AND FUTURE WORK 30

6.1 CONCLUSION 30

6.2 CONTRIBUTION 30

6.3 FUTURE WORK 30

6.4 ACKNOWLEDGMENT 31

7 REFERENCES 32

8 APPENDIX A 35
1 INTRODUCTION

Nicolás S, Josune H, Gorka G (2013, p.2) wrote: “There isn’t a single solution for choosing which mobile application type will be best for every situation”.

Statistics confirms 3.6 billion unique mobile subscribers in the world at the end of 2014, and an additional of one billion subscribers predicted by 2020. Smartphones, were counting for 60% of the connections at the end of 2014. It is the developer world, driven by increased affordability of devices that will produce most of the further growth, with an expectation of an additional of 2.9 billion smartphone connections by 2020. This enormous growth, puts pressure on mobile/cellular phone manufacturers to find new approaches and application building technologies to meet the need of users. As shown in the table below, we can see that the predicted growth of smartphones is estimated to 2,659.4 million by 2019. The growth of smartphones, between 2018 and 2019 is predicted to be 9.36% while the growth between 2019 and 2020 is predicted to be 9.17% [9] [21].

Native applications are built using the native language of the platform or device it is intended to be used on. The native language for Android applications is Java and for iOS applications Objective-C and the new swift. Google’s Android, Apple’s iOS, and Windows Phone are the three leading mobile operating system. Native application are provided with features and a best overall experience. Some of the typical characteristics of native application are multi-touch, faster graphic APIs, fluid animation, built-in components and ease of use. All native application has one thing in common, they all have unopposed access to the hardware of the device and supports all user-interface (UI) and interactions are accessible in a responsive mobile operating environment. The problem with the native application is however, that the same application has to be created for different operating system which leads to increased use of resources [1] [3] [23].

Hybrid development combines the best of both the native and HTML5 worlds. Hybrid applications are developed using traditional web developing programming languages such as HTML5, JavaScript and CSS. Hybrid platforms has turned out to be a new beginning of
developing mobile applications. Once the hybrid application is built with HTML5, CSS and JavaScript, the application runs in a 3rd party container of native application to access the native packages. At the top-layer hybrid applications uses HTML5, JavaScript and CSS. This gives the possibility for traditional web developers to use existing knowledge while developing applications for mobile usage. Some of the used and known hybrid frameworks are, PhoneGap, Ionic and Cordova. Both native and hybrid can produce applications that can be downloaded for iPhone and Android at Google Play or Apple iTunes App Store. [1] [3] [23].

The aim of this study is to examine whether the functionality of the specific features, GPS-Map, Activity and Multi-touch, of hybrid application are equivalent to that of native applications and with empirical evidence conclude as to whether the features are realistic substitutes or not. If it is possible for hybrid applications to replace native applications, the hybrid approach could have comprehensive consequences in the future growth of development of smart devices in developing countries. The comparison will be based on performance equality of the features, support for drawing objects on the map, support of the Geocoder and support for the 3D-touch on the multi-touch feature [3]. This study is designed using science research methodology to collect correct information as described in section 3.1.1 and 4.1.1, combined with a survey addressing performance, supported packages and interaction through touches on the screen.
2 BACKGROUND

Ever since smartphones were introduced, phone usage has increased drastically in the world. Since 2010 the amount of mobile devices has increased more than 15% per quarter [1] [3]. Today smartphones are essentially like a small portable computer. Creating applications for smartphones have become a huge opportunity and extremely popular [22]. Not only companies create applications; hobby-developers (category of people who are developing applications for their own interest and enjoyment) are doing it as well.

The number of different operating systems for smartphones are plentiful for example android, iOS, windows phone, blackberry and other as will that are not mentioned here [28]. This makes it increasingly harder for developers and especially hobby-developers to create applications across operating system platforms. Generally, every system uses its own programming language, which makes it hard for a hobby-developer to learn all different languages and port their application to them all.

When the first Hybrid app platforms were introduced, it essentially made it easier to create mobile-applications for all operating systems. An application can easily be created using HTML5, CSS and JavaScript and then simply port it using a Hybrid platform to any operating system you want [3] [7]. This introduction triggered many developers to switch to Hybrid app development. Development companies concluded that they already had extensive knowledge about HTML5, CSS and JavaScript, but less experience in Android, Objective-C and other smartphone languages. The step into this new world of mobile development seemed substantially easier when switching to hybrid development [3] [7].

In 2007 Facebook created a 100% native app for iPhone, but did not release an iPad version until October 2011. This version was released at the same time they re-released a new version for iPhone and Android. All three versions were now using the hybrid app approach. David Fetterman (Mobile Engineering Manager at Facebook), later gave a speech on why they had moved from native to hybrid app development [7]. In middle/late 2012 Facebook switched back to native development, Mark Zuckerberg said “The biggest mistake we’ve made as a company is betting on HTML5 over native” [28].

“All of our developers are good at HTML. Only a few of them are really good at Objective-C and Android. We are able to make our web developers the same as our client-side developers in some respects” [7].

Picking the subject “Comparing Native and Hybrid Applications with focus on Features” for this thesis study was based on an attempt to provide good understanding along with guidelines for developers and companies on which approach to use when developing mobile applications. We wanted to verify if hybrid application development could offer the same richness in features and functionality as well as performance compared to native development. Even though previous studies in this subject area exists most of them focus on performance and have not taken features and functionality into account. We believe that while performance is an important topic, features and functionality are almost equally important when developing. We choose to narrow the variety of investigated features to the GPS and Map, but also Activities and Multi-touch. The reason being that these features comes pre-installed and are highly used in many of the existing applications today [30] [31].
3 METHOD

3.1 Research questions and research design

- **RQ1** – GPS and Map Features
  - RQ1.1 – Do native application platforms have equal performance compared to hybrid application frameworks?
  - RQ1.2 – Do hybrid application frameworks have equal support for drawing objects on map compared to native applications?
  - RQ1.3 – Do native and hybrid applications support Geocoder?

- **RQ2** – Activity Features
  - RQ2.1 – Do native applications perform equal to hybrid seen to activity feature?
  - RQ2.2 – Do all platforms support sending data between activities?

- **RQ3** – Multi-touch Features
  - RQ3.1 – Do all platforms perform the same?
  - RQ3.2 – Do all platforms support 3D-touch?

This study is based on the seven research questions above. The reason why we choose GPS, Map, Activity and Multi-touch because these features generally comes pre-installed with the devices. GPS and Map feature were chosen because it requires access to specific hardware of the device and performance is highly prioritized in this case.

RQ1 was chosen because GPS and Map is used a lot in today’s applications and the aim was to examine the performance differences in native and hybrid applications. But also to see if hybrid applications provide the same features as native when developing applications.

RQ2 was selected to examine whether a feature that doesn’t use any specific device hardware could have any impact on performance.

RQ3 was chosen because interaction with smartphones can’t be avoided. For an application to be successful multi-touch performance is highly prioritized. To answer the questions above the authors did a literature study as well as conducting an empirical study through an online administrated questionnaire/survey.

3.1.1 Literature Study Design

The literature study was designed to give information about all the research questions, as well as an overview of native compared to hybrid development.

Databases used to find relevant and useful papers is illustrated on figure 2 below.

<table>
<thead>
<tr>
<th>Database</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summon@BTH</td>
<td>Year (2012-2016), Only Peer Reviewed, English/Swedish</td>
</tr>
<tr>
<td>Google Scholar</td>
<td>Year (2012-2016), English/Swedish</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>Year (2012-2016), English/Swedish</td>
</tr>
</tbody>
</table>

Figure 2 – Databases used to find literature

To find relevant and useful papers the authors used the snowballing approach. This approach is well described in “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering”, by Claes Wohlin [8].
Inclusion Criteria
Language: English or Swedish
Timeframe: 2011-2016
Title: Is the title relevant to the study?
- Yes, review Abstract, Questions, Keywords and Conclusion. Does it answer any of our questions, or contain relevant information.
- No, exclude the paper

Exclusion Criteria
Non-peer reviewed

The authors began the literature study with a starting set of six different papers. These papers where defined as relevant and useful using the Inclusion and Exclusion criteria’s.

In the article “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering”, Claes Wohlin described what the characteristics for a good start set is. In this study we chose to interpret these characteristics as:

- If the relevant papers may come from different communities, then it is important to have these covered in the start set. The reason is that the papers may be in independent clusters, i.e. in clusters of papers not referring to each other.
- It is important to get the right amount of papers in the start set. The number differs depending on the area of study, in a smaller, more focused area the amount requires fewer papers than a bigger area of study.
- If the search result of papers is too large, for example due to having to general search terms, then an alternative is to identify a number of relevant and highly cited papers.
- The start set should include different publishers, years and authors, i.e. diversity.
- The start set ought to be formulated from keywords in the research questions, preferably also take synonyms into account.

These characteristics were taken highly into consideration when defining this study’s start set. An excel spreadsheet to easily track what information was read and if it gave any useable information. The excel spreadsheet was formed though 4 fields:

- Article / Paper (URL, Name, Number)
- Topic (Performance)
- Section (3.1.1)
- Comment

This exploration phase approach later to great extent optimized the writing phase; when writing about e.g. GPS-Performance, the spreadsheet visualized references to valuable articles or papers gave valuable information about that particular topic.

3.1.2 Empirical Study Design
To empirically explore the research questions a survey was designed to capture perceptions on hybrid and native development. To get as much perspective and input as possible the survey included a few questions related to the experience level of the respondents.
This part of the study was designed mainly to collect information about RQ1, RQ2 and RQ3, but also to give a perspective of what people think about native and hybrid development.

The risks of doing this kind of study is if not enough people answer it. The authors took this risk into consideration and handled it by sending it out to a total of five forums. They also had a plan if it still wouldn’t work which was having questions that were able to be answered with the literature study as well.

To get accurate and relevant information, the aim of the survey was to get around 80 answers.

The survey was initially distributed to an Android-developing forum, iPhone-developing forum, and a PhoneGap forum. These choices were made to get as different inputs with as much variety of perspectives as possible.

The authors however considered the risk of having too few responses to the survey, which is not uncommon. To mitigate this high risk, the authors added more channels, making it a total of five forums. Worst case scenario included answering the questions based on literature studies instead.

The questions included in the survey can be found in Appendix A.
4 RESULTS

4.1 Literature Review Results

4.1.1 Snowballing
The snowballing approach was utilized to find relevant papers and articles for this study, our start set contained five different papers/articles. We defined both inclusion and exclusion criterions to make sure we got relevant and useful information. We customized our iterations to match our way of writing, below is a figure describing our customized iterations.

![Customized Snowballing Iterations](image)

Our start set contained 1-5 in our references. These gave us a stable foundation and a good understanding of the hybrid and native applications as well as the history of mobile development and how it went from mobile web applications to hybrid.

4.1.2 The Rise of Hybrid Applications
App-development have gone through a lot of changes since the first smartphones were released. One interesting thing that not too many people think about or even remember is
that the first iPhone which was announced on January 9, 2007 did not even contain an App Store. At that time you did not have any applications, the only app-like thing you could have was a bookmarked website which gave a shortcut from the home-screen. It was not until fourteen months later, in March of 2008, that Apple introduced their app SDK and the iTunes App Store which was a huge success. Apple’s original vision for applications drew a bright line between web applications and native applications. Apple believed that native coded applications was going to yield the best end user experiences [7][10].

Over time a variety of platforms for smartphones were introduced, e.g. Android, Blackberry, Windows. By now it started to be complex and expensive to develop native coded applications. For every platform companies and communities had to have one team for iOS, one for Android and one team for every other platform they wanted to publish applications for. Compared to Web development this was a huge difference. In Web development one doesn’t need e.g. one team for Internet Explorer, one for Chrome, one for Firefox; one simply have one (1) code to “rule them all”.

During the early phases when mobile web applications came to life and became increasingly common, they were basically websites designed to run on smartphones. The design and functionality was adopted to work on smartphones, not created specifically for them. This was a new and cheaper way to develop applications for smartphones. The drawback with these applications was that the performance was not especially good to begin with. Another disadvantage with the mobile web applications were that they required a network connection.

In 2009 at a iPhoneDevCamp event in San Francisco, Apache Cordova was developed and went to win the People’s Choice Award at O’Reilly Media’s 2009 Web 2.0 Conference. Apache Cordova was originally created by Nitobi, but was in 2011 bought and rebranded as PhoneGap by Adobe Systems. PhoneGap is today one of the most popular hybrid application platforms.

A hybrid platform is basically a platform that allows you to write one code to “rule them all”. The language differs depending on what platform you choose, but most hybrid platforms use JavaScript, HTML5 and CSS3. The code is written like the code for a website. Then it is simply thrown into the hybrid platform and out, you got applications to match your desires (iOS, Android, Windows, etc.). One of the big differences that made hybrid applications so popular compared to mobile web applications, were the ability to run offline, since they were no longer a “website”, they were now an actual application.

Hybrid applications are like a combination of mobile web applications and native applications; the main advantages are:

- Code once, deploy on all
- Ability to make native calls to hardware using the “Native Shell” though JavaScript
- Offline mode, ability to run the application without internet
- Allows a large number of users to access the application due to the multiplatform support
- Distribution through official stores

Even though mobile web applications support multi-platform it is in a restricted manner, since internet access is required [3].

4.1.3 PhoneGap

PhoneGap is an open source development framework for mobile applications. PhoneGap uses methodology “write-once”, this means the idea is to only write code once and deploy on
multiple devices as a native application.
PhoneGap contains a set of APIs to allow developers to access native functionality such as the GPS, camera and others by using JavaScript, HTML5 and CSS3. Due to this it makes application development like any other website developing, you no longer need to learn all the different native-languages, Objective-C, Java and others.
At first when using PhoneGap you could only export the application to iOS if you were using a Mac, and only export to Android if you were using Windows and so on. Later on this was resolved when PhoneGap created a cloud-compiler, which made it possible to develop and export your application to any device, without requiring any specific computer [2] [3].

4.1.4 Native Applications
Native applications are built by using of the native programming language of the device which it needs to be created for. If an application is built for iOS, it must be written in Objective-C or the new language, Swift. Applications for Android uses its native language Java. Native applications provide a development environment with tools and widgets for creating desired interfaces with native user interaction experience, which are yet not there in the case of hybrid application development tools [1] [3] [23].

Some of the characteristics that any native application would provide are:

- Best overall experience. Some of the typical process that native application would process are multi-touch, faster graphic APIs, fluid animation, built-in components and ease of use [3] [23].

- The native application multi-touch features makes it possible for the user to interact with the device with complex UI (user interface) gestures. For ex. users could double tap to zoom. Pinch-spread and other advanced gestures [3] [23].

- Depending on the different device characteristic, native applications provides fast graphic API. Animation which is an essential when providing gaming experience on the device. It is also needed for highly interactive reporting and compound computational algorithms [3] [23].

4.1.5 Comparison between Native and Hybrid
There are many different frameworks for building hybrid applications. A conscious choice of framework must be made in order to develop a hybrid application with considerations to the overall performance, and a smooth and appealing interface.

Designing the mobile interface could be achieved both using native and hybrid application. Using hybrid approach to design the mobile interface is however more flexible since the developer uses languages such as HTML5 and CSS3.
Using hybrid approach to build complex and compacted application, the developer faces design problems such as slower response, and the consumption of time.

Thus, it is easier to design such applications using native approach due to the tools and design widgets provided by the native platform technologies.

1. Native applications provide the user with a better performance and smoother experience without delay compared to a hybrid application.

2. User may need to click a specific button more than once to get a response using hybrid application, which could lead to unsatisfied end-users.
3. Building applications with large animation can also create problems in the case of hybrid applications while native application gives a greater fluidity.

4. Native application is more time consuming since the same application has to be developed for different operative systems and requires knowledge in different programming languages.

[1][3][23].

<table>
<thead>
<tr>
<th>Considerations</th>
<th>Native</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort of supporting platforms and versions</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Device capabilities access</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>User experience</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Performance</td>
<td>Very high</td>
<td>Very high</td>
</tr>
<tr>
<td>Upgrade in the client</td>
<td>Needed</td>
<td>Needed</td>
</tr>
<tr>
<td>Ease of publication/distribution</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Approval cycle</td>
<td>Mandatory</td>
<td>In some cases</td>
</tr>
<tr>
<td>Monetization in app store</td>
<td>Available</td>
<td>Available</td>
</tr>
</tbody>
</table>

*Table 2 from “Mobile Web Applications” [5] show an overall comparison between Native, Hybrid and Web applications.

Developers with native applications skills are not only expensive and harder to come across, they are also extremely specialized in native applications. It’s rarely feasible to take a group of iOS developers right away and redeploy them on an Android project because of the highly specific platform skills; workflow and pacing creates differences that doesn’t compare to web development [7].

Moreover, hybrid applications are being slower and suffer poor performance compared to native applications due to the fact that hybrid applications must be run via native container. However, some of the worlds’ largest companies in the business like Facebook, LinkedIn and Netflix have changed to hybrid approach building applications. Of course there will be cases where native application is more suited than hybrid applications but for a very wide different content-centric applications, hybrid applications are performing better. However, building hybrid application means that the developer must rely on mobile app development frameworks and tools and its capability, provided feature to build the application. This means that if the chosen feature is not up-to-date, hybrid developer may face disadvantages and would not be able to implement the features that are not provided by the chosen framework [7] [23].

4.1.6 Performance comparison between Native and Hybrid applications

Performance is a cutting edge difference to take into consideration when developing, and not just when it comes to mobile applications. One of the drawbacks of hybrid applications is that they generally don’t have as good performance as native applications. Since hybrid applications first have to go through the hybrid platform and compile the code, before it can communicate with the native features. This prolongs the execution-time compared to native applications, which communicate directly with native features. Below is a figure describing PhoneGap When we asked the question why the used given hybrid’s method call flow path [2] [17].
In the article “Mobile multiplatform development: An experiment for performance analysis” Luis C, Alberto S, and Giancarlo S, made a performance comparison between an Android application and a PhoneGap application, shown below.

<table>
<thead>
<tr>
<th>Measured Job</th>
<th>Arithmetic Mean (milliseconds)</th>
<th>Standard Deviation</th>
<th>Geometric Mean (relative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access to accelerometer</td>
<td>0.7136</td>
<td>2.0021</td>
<td>0.9984</td>
</tr>
<tr>
<td>Launch sound notification</td>
<td>18.4835</td>
<td>26.7481</td>
<td>13.3665</td>
</tr>
<tr>
<td>Trigger vibrator</td>
<td>1.5134</td>
<td>3.2222</td>
<td>1.2234</td>
</tr>
<tr>
<td>Request data from GPS</td>
<td>2.1881</td>
<td>809.2352</td>
<td>6.7244</td>
</tr>
<tr>
<td>Request network information</td>
<td>1.1015</td>
<td>1.01419</td>
<td>1.2052</td>
</tr>
<tr>
<td>Write a file</td>
<td>4.7146</td>
<td>7.9221</td>
<td>9.2085</td>
</tr>
<tr>
<td>Read a file</td>
<td>13.3036</td>
<td>255.7381</td>
<td>13.8829</td>
</tr>
<tr>
<td>Retrieve data from contact list</td>
<td>95.8686</td>
<td>1841.4689</td>
<td>13.8747</td>
</tr>
</tbody>
</table>

As shown in table 3, the native applications performed a lot better than hybrid applications. Values show that the web application only succeeded in executing equal or better than native applications in only one instance (Launch sound notification), 35% faster. For the rest, native applications were faster than hybrid applications [2].

**GPS**

The performance of the GPS differs significantly between hybrid and native applications. Table 3 show us that the GPS performance between an Android application and a PhoneGap application, differs a lot. Even a simple task as “request data from GPS” on a hybrid application takes 809.23 milliseconds to execute, while that’s not much it’s still a big difference compared to native.
4.1.7 Feature comparison between Native and Hybrid

Features are always hard to compare because there are several third-party plugins out there giving extra support and more features, but we will compare the original features without any third-party plugins.

The GPS-features differ quite much, below we created a diagram illustrating the number of features per platform.

Both iOS & Android contain 26 different GPS features while PhoneGap only contains three. This is a major difference which should be highly taken into consideration when choosing development approach, in case you develop an application that uses the GPS. Whilst PhoneGap only contains 3 features not all features are needed or relevant for every user [17] [18] [19].

Android contains a great feature “getBestProvider(Criteria criteria, Boolean enabledOnly)”, which basically lets you check which is the best provider that meets the given criteria. Another great feature in Android is the “getProvider(String name)”, if combining these two function, you are able to determine and use the provider that gives the best information for your need [17].

According to the iOS Developer API, you have no possibility to choose or see what type of provider is used to determine location. If you want to see what provider is in use when developing iOS, you’ll have to manually check the “accuracy”-property, if it’s lower than 40 meters, in case that return true, the application is using the GPS. Although manually picking GPS-provider might not be highly desired, it’s still a good to have [18]. PhoneGap neither contains the function to choose provider; by default it tries to retrieve the location using a network-based method. You can however apply the parameter “enableHighAccuracy”, then it will try to use more accurate methods, such as the GPS [19].

Android contains a function called “getLastKnownLocation(String provider) - Returns a Location indicating the data from the last known location fix obtained from the given provider”. Even though this location can be out-of-date, for example if the device was turned off and moved, it still is a very usable function. This function is supported neither on iOS nor PhoneGap.

Both Android and iOS that allows you to monitor an “area” and triggers if the device enters that area. On Android it’s called “addProximityAlert” and on iOS it’s called...
“startMonitoringForRegion”. These functions can be very useful, but are not provided on PhoneGap. The alternative to monitor a region on PhoneGap would be to constantly get location updates from the device and constantly check if the device did enter an area. This however decreases performance [17] [18] [19].

Just like the GPS the Geocoder can be extended with multiple different third-party plugins, but here we will only compare the original features.

![Graph showing number of Geocoder features](image)

*Figure 5 – Number of Geocoder-features [17] [18] [19]*

As you can see above, PhoneGap does not support Geocoder originally. You’ll have to install third-party plugins to get the Geocoder when using PhoneGap, for example Google Maps Geocoder. Both Android and iOS contains a Geocoder, iOS contains 5 different Geocoder features while Android contains 4. Both of the platforms have the same functionality but with different names. Nonetheless one excellent feature iOS supports are the “cancelGeocode – Cancels a pending geocoding request”. This feature can be particularly useful if there is a need to cancel a request, for example when the internet access is broken.

Multi-touch features provide the possibility of interaction with smartphones in different ways via screen, e.g. double tapping on screen to zoom, one finger press and lift to select and other advanced on screen movements. Unlike hybrid applications, native applications come with multi-touch features included. Hybrid applications must depend on third party plugins to gain access to use multi-touch features.

In the case of android applications when the pointers (mouse, pen, finger, trackball) touches the screen simultaneously four different main touch events are generated by the system:

- **Action_Down** – for the first pointer that touches the screen. This starts the gestures.
- **Action_Pointer_Down** – for extra pointer that enters the screen
- **Action_Move** – Responsible for when changes happens while a press gesture
- **Action_Pointer_Up** – is send when a non-primary pointer goes up
- **Action_Up** – is sent when the last pointer has left the screen
In the case of android, these are the main multi-touch events that are generated by the system. To keep track of the individual pointers within a MotionEvent, it is archived by using the pointers index and/or pointers ID.

A MotionEvent stores every pointer information in an efficient manner in an array. The MotionEvent class in Android is responsible for keeping track of movements (mouse, pen, finger, trackball), and may depending on type of device hold either absolute or relative movement data (or other data) of the pointer. Most of the MotionEvent methods used take the pointer index as parameter, not pointer id, to interact with pointers. The MotionEvent class includes 174 different public methods that can be used to interact with events generated by pointers [17].

Like the Android platform, iOS API also provides the possibility to access the multi-touch features and its methods. Generally, most of the touch events in iOS can be managed with standard controls and movement recognizers in UIKit (user interface kit). The gesture recognizer is used to separate the recognition of the touch from the action that the touch produces.

Unlike the Android platform, a subclass of the UIResponder class has to be implemented and created in order to access the multi-touch events of the iOS platform and implement custom touch event handlers.

There are four main classes that can be used to access the multi-touch features. Any of the following classes can be used [18]:

- **UIView** - Subclass UIView to implement a custom drawing view.
- **UIViewController** - Subclass UIViewController if you are also handling other types of events, such as shake motion events.
- **UIControl** - Subclass UIControl to implement custom controls with touch behavior.
- **UIApplication** or **UIWindow** - This would be rare, because you typically do not subclass Disapplication or UIWindow.

The iOS platform recognizes the touches as a part of multi-touch sequence unlike in Android. During a multi-touch sequence, series of event massages are sent to the target responder. In order to receive and handle these incoming messages, the responder object’s class must use the following UIResponder methods [18]:

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;

As described above the iOS and Android multi-touch features provide different ways of handling multi-touches. If we compare the main methods of the features, we find that the two platforms works similarly when handling the touch events, regarding the method aspects.
4.2 Survey Results
As described in the method, we combined our literature study with an empirical study as well, our choice was to do a survey. The empirical study aims to get a more comprehensive perspective, different views and to broaden the input and base for the resulting study. We got a total of 54 responses.

4.2.1 Survey Questions
To get the most relevant and rewarding answers we carefully designed every question in the survey to match our needs.
We designed and produced the questions over a total period of 2 weeks, including analysis and preparation.

We designed the survey to contain three different parts:
- Personal Questions
- Overall Programming
- Native or Hybrid depending on given approach

First we asked some personal questions to get an understanding of the respondents. Some examples of the questions we asked here were:
- How old are you?
- What’s your degree?
- How many years have you been developing applications? (any programming)
- What is your primary motivation for developing applications?

Given these questions we wanted to establish what type of developer this was.

The second part (Overall Programming) only contained the question “Which approach do you mainly use when developing applications, native or hybrid?” depending on the answer the user gave here, we designed the survey to redirect the user to either the hybrid- or native-part.
This were made to ensure the accuracy of the survey and not to allow for example a hybrid developer to answer native questions.

The third part (Native or Hybrid) contained specific questions regarding either native or hybrid depending on the development approach picked.
The questions here were mainly about the features GPS & Map, Activities and Multi-touch.
Some of the questions asked were:
- How satisfied are you with the GPS and Map performance?
- Are you missing the option to draw certain objects to the map?
- How do you like the 3D-touch feature?

We tried to make as many multiple choice questions as possible to make it easier to analyze the answers later.

The details of the questionnaire can be found in Appendix A.

4.2.2 Target group
Our target group basically was people developing applications. To secure diversity and targeting the right communities we picked different channels to publish the survey carefully.
In the start we planned to only publish the survey on:
- Android Developer Forum [12]
- iOS Developer Forum [13]
PhoneGap Developer Forum [14]

We reckoned these were good places to maximize the combinations and diversity of native iOS and Android developers, as well as hybrid PhoneGap developers.

However, we immediately got into a showstopper; according to the rules you are not allowed to post surveys or basically any marketing things in those forums. The iOS forum rejected our thread even before publication. The strategy had to change, we turned and added more places/channels to post the survey at. We chose to publish the survey at the following places to replace the rejected Apple forum:
- AndroidForums.com [15]
- MacRumors Forum [16]

Unfortunately, on both these forums we got rejected as well. During this initial phase we were troubled, we only had 9 answers after almost 2 weeks. To resolve this and get the additional and sought after responses we targeted some companies in hope of getting some answers from their developer communities. We targeted four different development-companies. Three of the companies were in Gothenburg, Sweden and one in Karlskrona, Sweden. We also sent out the survey on our LinkedIn accounts.

4.2.3 Response

Even though the initial difficulties receiving answers on the survey, we still managed to get a total of 54 responses. It was not as much as anticipated but in the end it gave enough input to base analysis upon.

The age of all the responders varied a lot.

![Figure 6 – Age of survey responders](image)

As you can see above the age ranges on the respondents are wide, which added an additional dimension to the results from the survey.
74.1% of all the responders had 5+ years of overall programming experience, whilst only 29.6% had been developing applications for 5+ years. 40.7% had developed applications for 0-1 years.

The predominant to develop applications were “Personal Interest” with a total of 38/54. The second largest motivation was “Profit” with a total of 19/54. The least motivation to develop applications for our responders were “Brand Recognition” with only 3/54.

To the questions “Which market does your company/organization operate in?” which was a multiple choice question, 44 people answered iOS and 46 answered Android, and another 8 answered Windows, while 0 people operated in Blackberry.

To the question “Which approach do you mainly use when developing applications, native or hybrid?” there were more respondents using native over hybrid.

The survey result indicates a prevalence to use native approach in general, it becomes evident that hybrid platforms are gaining development ground.

**Native users**

Out of the 30 people who used the native approach preliminary, there were 8 of those who were fluid in both Swift / Objective-C & Android-Java, while 11 people were only fluid in Android and 8 only fluid in Swift / Objective-C.

To the question “How satisfied are you with the GPS and Map performance?” we got the results below.
Most of the respondents were satisfied with the GPS and Map performance when using the native platforms. There were only 2 respondents who were less satisfied with the performance.

When we asked the question “Are you missing the option to draw certain objects to the map” there were only 2 people thought this was missing, one person who were developing both native and hybrid applications answered “No (Not in Native API’s)”.

The Geocoder results were a bit different where 17 people only selected 3/5 in satisfaction to the Geocoder support. Eight people choose 4/5 and four choose 5/5 and one person choose 1/5.

The question “Are you missing any GPS and Map features?” got good response, only one person who said “Yes, offline support” other than that everyone was satisfied with the existing features.

The respondents were satisfied with the performance regarding Activities including sending data between them, 13 people submitted 3/5, eight people submitted 4/5 and seven people choose 5/5, only two people said 2/5.

The question regarding the performance on Multi-touch were also very high, nine people selected 3/5 while 11 selected 4/5 and 10 selected 5/5. Not a single person selected either 1/5 or 2/5.

The last question “How do you like the 3D-touch feature” people answered as below.
The results were neutral, with a total of 14 people answering 3/5.

**Hybrid users**

We asked the question “Which platform/s are you using”, we got these results. We did this question as a multiple answers question, so the responder could choose multiple frameworks.

Based on our results the most used framework when it comes to hybrid was IONIC with a total of 13 responders. On a close second was Cordova, with a total of 12 persons. The least used framework was jQuery Mobile with only four people.

Five people were using other frameworks than the ones listed here. “When we asked the question, “Why are you using the chosen platform/s?” We got the results given below. This was a multiple choice questions which means the responder could choose multiple reasons here.

Viewing the results above people select platform based on which is the easiest to start working with.

When we asked the question “Have you done any research on other platforms beyond the one you mainly use?” 66.7% answered yes, and 33.3% answered no. Like the question before this was not what we expected either, we thought the responders for sure would’ve done research on other platforms as well.

Just like we asked the native users if they were satisfied with the GPS and Map performance, we asked the hybrid users the same question, the results are below.
These results are not very good and especially not when comparing to the native results. When comparing this to the results given by the native responders, we can draw the conclusion that native applications have better GPS and Map performance than hybrid applications.

To the question “Are you missing the option to draw certain objects to the map” there were only one respondent who answered Yes here.

The question “Are you satisfied with the Geocoder support” the responders answered as shown below.

These results were not that big of a difference from native, the responders were neutral here.

To the question if the user were missing any GPS and Map features most of the responders choose No. One person was missing “offline support” and two persons replied “Without plugins a lot of features are missing”.

Sending data between activities satisfaction were also quite neutral responses, two people answered 1/5, one answered 2/5, 12 people answered 3/5, eight answered 4/5 and one answered 5/5.

Just like the activities question the responders were neutral when we asked how satisfied they were with the Multi-touch performance. One person answered 1/5, 2 people answered 2/5, 15 people answered 3/5, five people answered 4/5 and one answered 5/5.

We also asked how they liked the 3D-touch feature and also here were the responders quite neutral.

- **1** – 1 person
- **2** – 2 persons
• 3 – 19 persons
• 4 – 1 person
• 5 – 1 person

As an open question response we got the following answer from one responder “The most "lications we develop in my teams are premium-applications therefore we only develop in native”.
5 ANALYSIS

Comparing the result of the survey, native application is the primary choice when using the GPS and Map in applications. Using GPS and Map feature requires access to specific hardware and native application are able to provide that.

Based on the results of this study, native development is the preferred choice when developing more advanced applications that require performance and access to specific hardware for example GPS, Multi-touch. While hybrid applications have come a long way, they still do not perform as well as native applications since the hybrid application must be run in a native container, which in turn implies sending data through an extra layer.

Hybrid contains most of the needed and standard features, but has still is far from being mature enough to offer the “unusual” but sometimes needed features. One example of this is the GPS-function called “addProximityAlert” on Android or in iOS, called “startMonitoringForRegion”, that returns information about device region. A developer can always write the application to instead manually check if the user/device enters a region, but utilizing the GPS feature function gives better performance as well as “better looking” code. Even though there are features missing in hybrid the survey respondents were still overall satisfied with the features given in hybrid platforms.

Based on what the survey respondents answered, the developers who use native approach are generally more satisfied with features and performance. As shown in 4.2.3 the result show that some features are missing from hybrid that exists only in native environment. And some doesn’t perform as well as native, and some are overall not as good as native.

When developing regular applications for example a simple To-do list, hybrid development is preferred; it’s quick to learn, and the developer gives the basic understanding of how applications are build. Basically, as long as developer are not accessing/utilizing device hardware in the application, it will be just fine with hybrid development.

Product maintenance and future development is a consideration worth mentioning. Based on the thesis results hybrid platforms offers a less complex maintenance steady state. In a steady state mode, the applications and products are to be error corrected, further developed with added user features, recompiled and republished, etc. With the native approach the developers are forced to process the same amount of work for all platforms, whereas on hybrid only once. This may not be an issue for smaller changes but for larger updates and especially for an application that aims to have a longevity beyond shorter periods, this is an important input to your technical choices.

Also, if you as developer is new to application development and are unsure of whether to learn hybrid or native, then hybrid is a perfectly viable way to go. Today you will have good support and documentation how to develop. There exist many third-party plugins with easy access to satisfy your needs. As long as you are creating “simple” applications, not utilizing much device-hardware features, like GPS, Multi-touch etc, hybrid covers your needs. To use device-hardware features with hybrid is an option as long as the application calls for those functions semi occasional. If you e.g. create a simple application which checks the user/devices location hourly only hybrid is viable, but if the location is checked every minute and then referenced further then native is a better choice.

The thesis study bases its validity on the referenced articles and papers, along with a survey. Should any source be outdated or erroneous, this study also shows erroneous results. To mitigate this risk, the authors have cross checked information and input with multiple
sources but there might still be validity threats. Another validity threat is the survey, if given respondents haven’t answered correctly, the results will be erroneous. The target group can be a validity threat as well. If the hybrid respondents have based their answered on any third-party plugins, that would be a validity threat as well, since we have based our study on original features and haven’t taken any third-party plugins into account.

5.1 **RQ1 – GPS and Map Features**

**RQ1.1 – Do native application platforms have equal performance compared to hybrid application frameworks?**
Since GPS and Map feature requires access to specific hardware, native applications are performing significantly greater than hybrid applications when using the GPS and Map feature. Hybrid applications are deficient in performance when using GPS and Map due to reason that hybrid applications are wrapped in a native container and at least one pull-request is always required which resulting in inferior performance.

**RQ1.2 – Do hybrid application frameworks have equal support for drawing objects on map compared to native applications?**
Regarding drawing certain objects to the map, based on our results and findings we could not find any significant discrepancies. Since PhoneGap does not contain any original map of its own; Android uses Google Maps originally and iOS uses Apple’s map originally, comparisons are not fully conclusive. The analysis deems therefore the platforms similar for this particular usage.

**RQ1.3 – Do native and hybrid applications support Geocoder?**
Native applications are supporting Geocoder unlike hybrid applications. Hybrid applications do not come with pre-installed Map and are missing Geocoder. Hybrid applications has possibility to access support for Geocoder through third-party plugin.

5.2 **RQ2 – Activity Features**

**RQ2.1 – Do native applications perform equal to hybrid seen to activity feature?**
Since Activity feature does not require access to specific hardware, both hybrid and native application are performing equal when using activity feature.

**RQ2.2 – Do all platforms support sending data between activities?**
Both hybrid and native applications has support for sending data between activities.

5.3 **RQ3 – Multi-touch Features**

**RQ3.1 – Do all platforms perform the same?**
When using the multi-touch feature native applications are more suitable than hybrid application since native applications has direct and unlimited access to the device hardware. Hybrid application must send request to native container which in turn takes time and resulting in lower performance.

**RQ3.2 – Do all platforms support 3D-touch?**
Native applications support 3D-touch per default. Hybrid application must depend on third-party plugins to use 3D-touch.

5.4 **GPS and Map analysis**
The GPS and Map functions/features are performing significantly better on native, albeit hybrid applications are getting increasingly competent.
When developing premium applications and more advanced applications our results show us that native is the ideal choice. You get faster response, and better overall performance, as
well as numerous useful features which do not exist in hybrid platforms. When using hybrid application approach, developer must depend on third party features.

If you however are developing a bit lighter applications where performance is not the primary goal of the usage, hybrid is more suitable, because of the high compatibility.

In figure 4 (4.1.7) it shows that both Android and iOS contains 26 different GPS features while PhoneGap only contain 3. When comparing this to the hybrid developers survey responses which can be found in 4.2.3 where all but one answered “No” to the question “Are you missing any GPS and Map features?” Our analysis of this tells us that even though there are fewer features in PhoneGap, it still captures the most used and important features.

When analyzing the PhoneGap performance analysis table in 4.1.6 and the survey results regarding the question “How satisfied are you with the GPS and Map performance” the majority of the survey respondents seemed satisfied in regards to the substantial difference in execution time. Even though on paper the performance can differ quite a lot between the technologies it isn’t always noticeable when actually using the function.

In the literature part of the study we found that PhoneGap do not contain any Geocoder originally, you’ll have to use third-party plugins for example Google Maps. Despite this the survey results which can be found in figure 14 (4.1.7) show that the hybrid developer respondents were quite satisfied with the Geocoder support. When comparing the native and hybrid developer’s responses in the survey, we found that the native users are more satisfied with Geocoder support than the hybrid users.

5.5 Activity analysis

When it comes to Activity there is not as big a difference as the GPS and Map. Developing PhoneGap applications is quite different when it comes to Activities. In Android for example you create Activities for every “page”, but in PhoneGap you simply change content via JavaScript or create a new html-page.

Both native and hybrid survey respondents answered almost the same to the question “How satisfied are you with the performance of Activities?” our analysis states that there is little difference between hybrid and native concerning Activities. Based on the results the Activity-features of the hybrid framework can be a viable substitute to native applications.

5.6 Multi-touch analysis

Already at a quick glance at the results from the survey, shown in figure 15, native application development is preferred when using multi-touch functionality. All 30 respondents using native approach regarded the multi-touch feature to have an average or higher performance; a satisfaction score much higher than the other group of developers using hybrid approach. In this group of 24 respondents the performance satisfaction score was 62.5% in average.

Looking specifically at the distribution of the responses of the two groups it becomes even more evident that native approach is considered superior to hybrid; more than 33% gave top
(5) grade to native application compared to the top given grade of the hybrid approach which was only 4%.

None of the native application satisfaction grades were below average level but in the case of hybrid approach 12% of the participants gave a grade which was beneath average.

The reason why the developers deemed the native multi-touch functionality to perform better than hybrid multi-touch, could be explained by the fact that native applications has direct and unlimited access to the device hardware which in turn gives better performance.

As described in section 4.1.7 hybrid applications does not support multi-touch by default. Multi-touch can however be accessible through third-party plugins. Depending on how the plugins are designed, different multi-touch gestures could be implemented. Why hybrid multi-touch doesn’t perform as well as native, could be explained by that hybrid applications have to run inside a native container to gain access to the device hardware. That means hybrid application has one extra layer (shown in figure 1) to send data which in turn gives inferior performance.
6 CONCLUSION AND FUTURE WORK

6.1 Conclusion
The authors conclude that native application development still is superior choice for developing advanced and more complex applications, especially if access to device-hardware is required, as well as when the aim of the application is to provide high performance and superior user experience. The authors also conclude that, for novice application developers and hobby application developers, the hybrid approach is perfectly viable choice since hybrid applications are easy to start with, does not requires platform specific skills, provides a lot of features and gives a good understanding on how applications development works.

Hybrid applications are also preferable if the main factor is to target multiple platforms and time to market is the critical factor. Hybrid platforms also show a great advantage when looking at maintainability since only one code base has to be written and maintained, and then deployed anywhere.

While hybrid applications have come a long way, they are still not mature enough to replace native applications. There are several features missing when developing hybrid applications. If we compare the GPS features in native and hybrid we can clearly see that native still is a more viable choice. Native have better performance as well as much more features.

Hybrid applications are also preferable if the main factor is to target multiple platforms and time to market is the critical factor. Hybrid platforms also show a great advantage when looking at maintainability since only one code base has to be written and maintained, and then deployed anywhere.

Even though hybrid applications are good when time is of the essence since you have the “code once, deploy anywhere”. There still are some time consuming tasks and choices to be done before you can have a complete application. You have to research the existing hybrid platforms and choose one that fits your needs and desires. Another downside with hybrid is that if you are building a more complex application, third-party plugins will be used, then you have to research them and learn them before you can code. Today there are several popular hybrid platforms, which can take some time to compare and find the right platform to use, this also applies to third-party plugins. This issue does not exist when developing native, since there only are one platform per language.

6.2 Contribution
While it exists a lot of performance comparison papers between native and hybrid, there’s not as many paper comparing features. This study guides you and shows you which development-approach is preferred, considering performance but mainly focusing on features. It gives a good perspective when to use hybrid or native, and what the pros and cons are with them.

This study gives a basic understanding about when to use native development and when to use hybrid development. It provides a new view from a new angle when comparing both performance but mainly features in native and hybrid development.

6.3 Future Work
Taking this study to the next level would include performing own performance comparisons as well as developing own applications in both hybrid and native to see how and when to use different features.

Another thing that could be made is to include Mobile Web Applications into the comparison. As well as expand both hybrid platforms and native platforms compared.
Taking hybrid third-party plugins into account and comparing them more would also be a way to take this study further.

6.4 Acknowledgment

We would like to take this opportunity to express our profound gratitude and deep regard to our examiner Conny Johansson for his exemplary guidance, valuable feedback and constant encouragement throughout the duration of this examination paper. His valuable suggestions were of immense help throughout our examination paper. His perceptive criticism kept us working to make this examination paper in a much better way. Working under him was an extremely knowledgeable experience for us.

We would also like to give our sincere gratitude to all the friends and respondents who filled in the survey, without which this research would be incomplete.
7 REFERENCES


http://ac.els-cdn.com/S1877050912004516/1-s2.0-S1877050912004516-main.pdf?_tid=e0e95538-e541-11e5-8b4e-00000aab0f6b&acdnat=1457450861_925347bf095a2b2a60a9f708ee83417


http://research.ijcaonline.org/volume118/number15/pxc3903463.pdf


http://www.idc.com/prodserv/smartphone-os-market-share.jsp


10. “iPhone (1st generation)”, Accessed 2016-04-02  
https://en.wikipedia.org/wiki/IPhone_(1st_generation)
   https://en.wikipedia.org/wiki/Apache_Cordova


   http://developer.android.com/training/gestures/multi.html

21. Number of smartphone users worldwide from 2014 to 2019 (in millions)

   http://www.statista.com/topics/1002/mobile-app-usage/

   http://www.comentum.com/phonegap-vs-native-app-development.html

   https://developer.salesforce.com/page/Native_HTML5_or_Hybrid_Understanding_Your_Mobile_Application_Development_Options

25. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach, Dec 2012. Accessed 2016-04-03.
   http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6420693

   http://ieeexplore.ieee.org/xpls/stamp.jsp?tp=&arnumber=6642422

28. Facebook switches to native apps -

29. Mobile operating system, accessed 2016-05-09
   https://en.wikipedia.org/wiki/Mobile_operating_system

30. List of features in Android, accessed 2016-06-10

31. iOs, accessed 2016-06-14
    https://en.wikipedia.org/wiki/iOS
8 APPENDIX A

Personal Questions

- How old are you?
  o 20-25
  o 25-30
  o 30-35
  o 35-40
  o 40+

- What's your degree?
  o Bachelor Degree
  o Master Degree
  o Doctorate Degree
  o Other

- How many years have you been programming? (any programming)
  o 0-1
  o 1-3
  o 3-5
  o 5+

- How many years have you been developing applications?
  o 0-1
  o 1-3
  o 3-5
  o 5+

- What is your primary motivation for app development?
  o Profit
  o Extension of company portfolio to attract more external clients/requests?
  o Brand recognition?
  o Personal interest
  o Company directive
  o Other

- Which market does your company/organization operate in?
  o iOS
  o Android
  o Windows
  o Blackberry
  o Other

Overall App Programming

- Which approach do you mainly use when developing applications, native or hybrid?
  o Native
  o Hybrid

Native Questions

- Which native languages are you fluid in?
How satisfied are you with the GPS and Map performance?
 o 1
 o 2
 o 3
 o 4
 o 5

Are you missing the option to draw certain objects to the map?
 o Text

Are you satisfied with the Geocoder support?
 o 1
 o 2
 o 3
 o 4
 o 5

Are you missing any GPS and Map features?
 o Text

Are you satisfied with the performance of Activities-features (including sending data between Activities)?
 o 1
 o 2
 o 3
 o 4
 o 5

What do you think about the performance of Multi-touch feature?
 o 1
 o 2
 o 3
 o 4
 o 5

How do you like the 3D-touch feature?
 o 1
 o 2
 o 3
 o 4
 o 5

Hybrid Questions

Which platform/s are you using?
 o PhoneGap
 o Cordova
 o jQuery Mobile
 o IONIC
 o Other
• Why are you using the chosen platform/s?
  o Easy to start working with
  o A lot of features
  o Security
  o Performance
  o UI-Design
  o Easy to do testing
  o Easy to debug
  o Other

• Have you done any research on other platforms beyond the one you mainly use?
  o Yes
  o No

• How satisfied are you with the GPS and Map performance?
  o 1
  o 2
  o 3
  o 4
  o 5

• Are you missing the option to draw certain objects to the map?
  o Text

• Are you satisfied with the Geocoder support?
  o 1
  o 2
  o 3
  o 4
  o 5

• Are you missing any GPS and Map features?
  o Text

• Are you satisfied with the performance of Activities-features (including sending data between Activities)?
  o 1
  o 2
  o 3
  o 4
  o 5

• What do you think about the performance of Multi-touch feature?
  o 1
  o 2
  o 3
  o 4
  o 5

• How do you like the 3D-touch feature?
  o 1
  o 2
  o 3
9  APPENDIX B

How old are you? (54 responses)

What's your degree? (54 responses)

How many years have you been programming? (any programming) (54 responses)
How many years have you been developing apps? (54 responses)

What is your primary motivation for app-development? (54 responses)

Which market does your company/organization operate in? (54 responses)

Overall Programming

Which approach do you mainly use when developing apps, native or hybrid? (54 responses)
Native Application Questions

Which native languages are you fluid in? (30 responses)

- Swift/Object: ~16 (53.3%)
- Android/Java: 19 (63.3%)
- Other: 8 (26.7%)

How satisfied are you with the GPS and Map performance? (30 responses)

- 0: 0 (0%)
- 1: 2 (6.7%)
- 2: 8 (26.7%)
- 3: 7 (23.3%)
- 4: 9 (30.0%)
- 5: 0 (0%)

Are you missing the option to draw certain objects to the map? (30 responses)

- No
- No
- No
- No
- No
- No
- No
- No
- No
- No
- No
- No
- No
- No
- No
- Yes
- No
- No
- No
- No
- No
- No
- No
- No
- No
Are you satisfied with the Geocoder support? (29 responses)

No
Yes
No (not so satisfied)

Are you missing any GPS and Map features? (30 responses)

No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
No
Have you done any research on other platforms beyond the one you mainly use? (24 responses)

- Yes: 33.3%
- No: 66.7%

How satisfied are you with the GPS and Map performance? (24 responses)

- 1 (4.2%)
- 2 (8.3%)
- 3 (20.8%)
- 1 (4.2%)

Are you missing the option to draw certain objects to the map? (24 responses)

- No
- No
- No
- No
- No
- No
- No
- No
- Maybe
- haven't used it yet
- No, still possible
- Not really
- Have not tested that yet.
- not used map
- No idea? have not used it...
- Not so far
Are you satisfied with the Geocoder support? (24 responses)

- Yes
- No
- Not yet
- Haven't worked with Map or GPS
- Haven't used this feature much
- Haven't tested it
- Haven't tried

Are you missing any GPS and Map features? (24 responses)

- Yes
- No
- Haven't used it yet
- Without plugins a lot of things are missing. Luckily there are many plugins and you can write your own.
- I miss some
- Haven't tested that yet
- Not used
Are you satisfied with the performance of Activities (including sending data between Activities)?

(24 responses)

What do you think about the performance of Multi-touch? (24 responses)
How do you like the 3D-touch feature? (24 responses)