OULU UNIVERSITY OF
APPLIED SCIENCES

Claudia Caro Valadez

DEVELOPING AN APPLICATION USER INTERFACE

COMPARING THE USE OF WEB TECHNOLOGIES VERSUS QT AND QML

DEVELOPING AN APPLICATION USER INTERFACE

COMPARING THE USE OF WEB TECHNOLOGIES VERSUS QT AND QML

Claudia Caro Valadez
Bachelor's Thesis
Spring 2017
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Internet Services

---

Author: Claudia Caro Valadez
Title of the bachelor's thesis: DEVELOPING AN APPLICATION USER INTERFACE: COMPARING THE USE OF WEB TECHNOLOGIES VERSUS QT AND QML
Supervisor: Veijo Väisänen
Term and year of completion: Spring 2017 Number of pages: 41 + 1 appendix

---

The mobile applications market has been continuously growing at an increasing rate, thus the demand for mobile application development in a fast, cheap, and available for as many platforms as possible manner has increased too. Since the native application development does not offer the possibility for a multi-platform development, a view into other development methods has been assessed.

The aim of this Bachelor's thesis was to provide a wide overview of the strengths and weaknesses of different types of frameworks for a multi-platform mobile application development, in this case web-based and native frameworks. The frameworks were compared by a list of the most relevant criteria in relation to a user interface.

The comparison was made using an information context type of application for Android and iOS devices. The Ionic Framework, as an example of a web based framework, turned out to be the most straightforward development tool for this specific context of applications and from a web developer's point of view. Nevertheless, the Qt framework offers certain advantages for applications in other contexts.

Keywords: Mobile application development, User Interface, Web Frameworks, Qt, Ionic Framework, Cross-platform, Multi-platform

PREFACE

This project was done as an independent research in Oulu during the years 2015-2017 under the supervision and guidance of Mr. Veijo Väisänen and Ms. Kaija Posio.

I dedicate this thesis work to my family. To my mom, for being my best friend and believing in me. To Michal, for all his support, love and guidance during all these years.

This thesis was aimed to help in the decision process of application developers looking for an insight into the differences of frameworks and their strength and weaknesses. In this case, Qt/QML was chosen as the framework to be compared to because it is very well known as a very popular choice for desktop application development, which supports multi-platform development, including mobile and embedded devices.

Oulu, 06.03.2017

Claudia Caro Valadez

CONTENTS

VOCABULARY

ADB:	Android Debug Bridge
APK:	Android application package file
CSS:	Cascading Style Sheets
FPS:	Frames Per Second
GUI:	Graphical User Interface
HTML:	Hypertext Markup Language
IDE:	Interactive Development Environment
NDK:	Native Development Kit
NPM:	NodeJS package manager
OS:	Operating System
QML:	Qt Modelling Language. A UI markup language
Qt:	Cross-platform application framework for GUI programming
SDK:	Software Development Kit
UI:	User Interface
UX:	User Experience
WebView:	UI component to display the content of a webpage

1 INTRODUCTION

Galitz defines the term “user interface” as a “collection of techniques and mechanisms to interact with something”. The origin of the graphical user interface (GUI) dates back to the 1970s at Xerox’s Research Center. (1, p. 7, 16.)

Furthermore, the introduction of new generation mobile interfaces in 2007 made it necessary to change the user interface design, as we know it today. Terms such as responsiveness, user experience, and usability became more and more common. (2.)

Today, mobile application developers face many more issues than before. They must have a strategic approach when developing an application. For the pre-production or design phase is given much more relevance, and it is here where main issues are foreseen and handled. Deciding which technologies to use is not always straightforward as there are so many choices with different advantages and disadvantages. (2, p. 2-6.)

User interface design and mobile application development fields are evolving and it is only possible to keep up with the latest changes by checking online communities, current technologies trends, and guidelines. It is very common to find that there is no completely right answer. Each scenario has its own limitations, e.g. time to market, costs or human resources.

Currently, mobile applications are in great demand, and most customers want to get a mobile application for as many Operating Systems (OSs) as possible, cheap and quickly. There are many possibilities on the market, and web based or hybrid applications are some of them. This project evaluates which approach might be better based on certain measurements. Hence, developers who need to deliver a multiplatform application and are considering any of the frameworks compared here can make an educated decision.

2 USER INTERFACE DESIGN

2.1 Usability

Web usability can be defined, as a quality assessment of how easy it is to use a user interface. According to the Nielsen Norman Group, usability can be evaluated by five different parameters: learnability, efficiency, memorability, error recovery, and satisfaction. (3, chapter 3.)

Usability can determine the success or failure of a website or a mobile application. As currently there are so many websites and mobile applications available, users will quickly leave for many reasons. The most common reasons for this are if they find it difficult to use, they cannot get oriented easily, or they cannot find the information they are looking for. (4.)

2.2 Responsive design

This concept refers to the ability of changing the layout of a website according to a device screen size and screen orientation. The importance of this has been growing with the great amount of devices available on the market. (5.)

Having this kind of design in mind when developing a website or a mobile application has certain advantages, as the development can be done faster than having to do different versions separately, and the maintenance is much easier in the long run (5).

2.3 User Experience

User Experience (UX) is a broad term introduced first by Don Norman back in the late 1990's and it has been evolving ever since then. Don Norman and Jakob Nielsen describe that: "user experience encompasses all aspects of the end-user's interaction with the company, its services, and its products" (6).

In a survey made by Compuware about what consumers really need and want when related to mobile applications, consumers' preference for mobile applications over mobile websites was much higher due to factors such as conven-

ience, speed, and easiness to browse. Users want mobile applications that are “easy to navigate and that deliver a suite of key functionalities - - through an intuitive - - user interface” (7, p. 7). Almost 80 percent of users will try an application a second time if it failed to load the first time, but only 16 percent would try it once more. Furthermore, UX can easily be brought down when a user has problems while using an application. This discourages the user to use it again or recommend it. It also reflects negatively on the company and it would switch the user to a competitor’s company app. The user would also give a low rating on an app store. (7, p. 8-10.)

Another important factor for mobile devices users is the loading time of an application. Most users expect mobile applications to load much faster than a website. To be more specific, they expect it to launch in about three seconds or less. (7, p. 11-13.)

User experience is part of the foundation upon which an application should be designed. A good design can then be further enhanced with a good UI design implementation. Therefore, UX and UI design should be part of the process when deciding which tools to use in order to create a good product because these tools can enhance or hinder the development process.

3 MOBILE APPLICATION CONTEXT

As there are several kinds of mobile applications available, the best choice of approach is based on the purpose of the application. Applications can be classified according to different parameters. In this case they will be classified according to the context where the application is needed or used.

According to Fling (8, p. 81) a context is defined as “the surroundings in which information is processed”, and therefore an application context is directly related to the user experience. This can be seen in the way an application is presented.

Fling (8, p. 81-88) describes different application contexts according to the purpose of the application itself (see table 1).

TABLE 1. Application context (Fling, 2009, 81-88)

Context	Definition	UX	Design	Examples
Utility	Short, task-based activities. User input is minimum.	At-a-glance information	Minimal design, limited content.	Unit converters, calendars, language translation, stopwatch/timer.
Locale	Display information related to user’s location.	Location-based	Usually a map, a list of items in order of distance.	GPS navigation, traffic or public transport, fitness (i.e. exercise routine tracking), take-out food ordering.
Information	Information seeking. Information-heavy, and marketing applications.	Content-based	Allow user to flag relevant data to return later, and avoid forcing users to input too much information.	News, online directories, commerce.
Productivity	Information-heavy content and services.	Task-based	Very structured, presenting information in a defined hierarchy	Email, task management, storage cloud-service, document editors.

Immersive	**Meant to consume the user's focus**	**Full-screen**	**Full-screen, no trace of the device UI.**	**Entertainment, games, video players.**

The different application context types should be considered carefully when selecting the type of application that will be developed, as each approach may have a higher leverage according to the situation.

4 MOBILE APPLICATIONS VS. MOBILE WEBSITES

The smartphone market has been growing at a slow pace, and as of the third quarter of 2016 Android OS was on the lead with 86.8% of smartphones OSs; iOs on the second place with 12.5%; followed by a mere 0.3% from Windows Phone. This only shows a part of the mobile devices OS segmentation, as those statistics account only for the smartphone market. (9.)

In addition, Nielsen and Budiu (10, p. 34-47) inform that users still prefer mobile applications rather than mobile sites, and that they have higher performance rates, too. This means that users can achieve certain goals, such as looking for information or purchasing a product, in a more efficient way by using a mobile application.

However, this scenario is prone to change in the future when there will be so many different platforms to develop for, and even different versions of these platforms. Mobile applications have more disadvantages besides the increase in development costs, such as a less discoverable content through a web search, and they require to be installed. This leads to mobile sites and mobile applications done with web technologies to have a certain advantage. In addition, mobile websites can have cross-platform capabilities, responsive web design can target many different screen sizes, and with the new web technologies development, such as HTML5, the capabilities of mobile sites and web mobile applications will increase. (10, p. 35-41.)

4.1 Web mobile applications

Web mobile applications, also known as Web apps, have an increasing popularity even among big companies due to mainly business reasons, such as app store policies related to content censorship, and the big share of revenues confiscated by them, too (10, p. 35-41).

Web apps are run by a browser, just like any other website. The user does not have to install them, and a bookmark is the easiest way to have direct access to them. They have some disadvantages due to the different browsers available

for mobile phones today, which leads to a platform-specificity issue. (10, p. 42-41.)

4.2 Hybrid Mobile Applications

However, there is another variant of applications for mobile phones called “Hybrid applications”. These are native applications that use the WebView of a mobile platform. To the end user, these applications are the same as a native application because they can be downloaded through an app store and there can be no noticeable differences when related to UI elements. The advantages rely on that they can be developed in a similar way as a website, but with access to the device features, such as an accelerometer or a camera. (10, p. 42-41.)

Hybrid applications allow reusing most of the code base when developing for multiple platforms, even though a developer still needs to adapt about 20% of the code to be able to comply with each platform guidelines and sometimes to access different APIs depending on the platform or hardware. This can be seen as a more efficient approach than developing two or more times the same application in different programming languages, which might even require different programmers. (11, chapter 2.)

On the other hand, some of the downsides of a hybrid applications approach, when compared to a native one, can be the reduced performance levels, the need to use third-party plugins to access device features, to get a native-like UI, and touch delays. However, even though the code of native mobile applications is compiled and optimized, the performance of hybrid applications can be improved to try to match that of the native application through good practices and the use of the right tools. (11, chapter 5; 12, chapter 1.)

5 WEB TECHNOLOGIES FOR MOBILE APPLICATION DEVELOPMENT

Mobile applications can be done with web technologies. Due to technologies, such as HTML5, CSS3 and JavaScript, and their rapid advance rate their performance and flexibility is increasing. This means that a developer does not have to develop an application for each operating system. (13, chapter 7.)

5.1 HTML

HTML stands for Hypertext Markup Language. It is used to specify the structure of the UI in web development and web apps among others. It uses tags that describe the different content along the document. The latest version is HTML5, which is supported by all modern browsers. (14, Introduction.)

5.2 CSS

Cascading Style Sheets (CSS) is a technology that works together with HTML, which is used to change the appearance of a website. CSS allows for modularity because it separates the document content from the presentation. The latest version is CSS3 and it is divided into modules. The support for CSS3 varies between modern browsers, which requires thorough testing to prevent elements from being displayed in an unwanted manner. (14, chapter 9; 15.)

Ethan Marcotte coined the term “responsive web design” in 2010. CSS3 plays an important role in responsive design, with its media queries that allow displaying different layouts depending on different environments (16, Introduction).

5.3 JavaScript

According to the Mozilla Developer Network, JavaScript is an interpreted programming language used for web development. It is also used in non-browser environments, such as Node.js. (17.)

Quick online search results show that JavaScript is also increasing in popularity among web developers, leaving behind PHP. JavaScript is also becoming more

widely used for mobile application development. This could be related to many popular hybrid application frameworks using JavaScript, such as Ionic, Mobile Angular UI and React Native. (18; 11, chapter 1.)

5.4 Web Frameworks

Web frameworks are mobile application development frameworks that use web technologies, such as HTML, CSS and JavaScript. The idea behind those is to make multi-platform development much easier, as one would not have to face a native development for each platform, and instead use only one programming language. (19, Preface.)

With the rise on the HTML5 development, many of the current application development frameworks are focusing on web technologies. Some of the most popular frameworks currently are: Ionic, Mobile Angular UI, React, React Native, and Titanium.

The great diversity between frameworks could indicate that developer's approach is towards a cross-platform development. However, there is still a vast amount to choose from and there are not enough studies that compare the pros and cons of them.

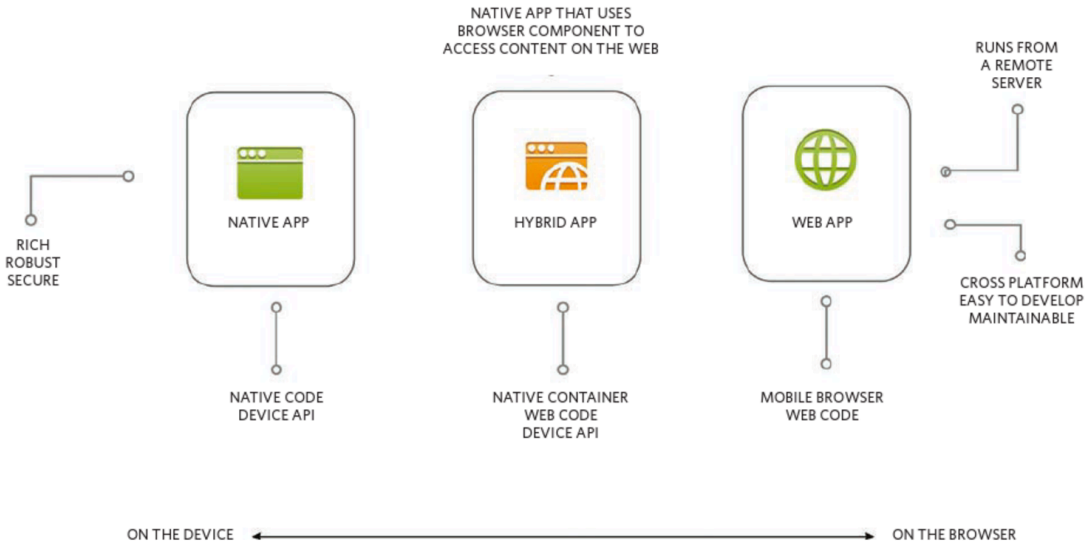

FIGURE 1 General diagram of the differences between native, hybrid, and web mobile applications (20)

6 NATIVE FRAMEWORKS

6.1 QT

Qt is an open source cross-platform framework for desktop, embedded and mobile applications. Its development dates back to 1990, and now it continues to be developed by “The Qt Project”. The framework is written in C++ and it has its own Integrated Development Environment (IDE), Qt Creator. In addition, its IDE has an integrated Qt Designer’s GUI. (21.)

Qt’s popularity as a cross-platform application framework for desktop applications could increase in the near future due to its portability to different platforms, such as a desktop, embedded devices and a mobile OS. This makes Qt/QML mobile support a very attractive technology for developers to learn.

6.2 QML

According to the Qt Company, “QML is a declarative language that allows user interfaces to be described in terms of their visual components and how they interact and relate with one another”. It is built in a way that it enables an easy and fast development of animated user interfaces, with the possibility of using any C++ libraries in the back-end. (22.)

One example of the benefits of using QML is that due to its declarative nature a property value can be an expression, which is being kept an eye on, and then when there is a change, the expression is re-evaluated and the property is set again. This reduces the programming load and actually allows the developer to focus on UI properties, such as view titles and navigation elements.

The Qt framework contains a QML module, which implements the QML language. In addition, the library “provides an API to enable application developers to extend the QML language with custom types and to integrate QML code with JavaScript and C++”. (22.)

Another key module for application development using QML is Qt Quick because it provides the developer with tools to build user interfaces. This library

provides visual elements, such as visual types and interactive types: an animation framework, a model-view support and particle and shader effects. (22.)

6.3 Qt Mobile edition

Qt Mobile was released together with Qt 5.2 back in 2013. Qt Mobile is a development package which includes a commercial license for Qt together with cloud services and a support for developers intended to help them in developing and publishing applications for Android and iOS (23).

The Qt Labs Controls library was released together with Qt 5.6 and it added support for typical UI controls for mobile applications (for example drawers) and support for different screen densities. (23.)

7 COMPARED FRAMEWORKS

A case study was conducted by comparing two cross-platform frameworks for mobile application development, i.e. Ionic Framework and QML, which are based on different programming languages and technologies.

The sample application that was developed on both frameworks has mostly UI elements, which are characteristic for an information context application.

7.1 PhoneGap and Ionic

PhoneGap is also known as Apache Cordova, through which PhoneGap is still being developed as an open source project for a cross-platform development. By itself, PhoneGap does not provide functionality for the UI generation. It is basically a packaging system for web applications. (24; 12, Module 3.)

One of the main issues that an early hybrid application development brought was the lack of a native user interface and user experience as opposed to native apps. There were not all those rich UI elements, animations and gestures that were provided by native environments. This meant that developers had to implement everything from the beginning and it led to mostly unappealing and non-standard behaving applications. All this opened the market to the development of frameworks that would provide a native-like environment where to develop hybrid applications, such as JQuery Mobile, Ionic, and React Native. (12, Module 3.)

As one of the main points of this document is to evaluate the UI development, it was decided to use PhoneGap together with Ionic Framework. PhoneGap has a support e.g. for Android, iOS, Windows phone, while Ionic Framework focuses on the look and feel of an application, and the UI interaction. (25.)

During the time frame that this thesis was written, a version 2 of Ionic Framework was released. It brings new features, such as a complete support for material design, new cross-platform components, a new plugin system called “Ionic Native”, an enhanced performance due to being built upon the new Angular

version and new development tools. However, only the version 1 will be evaluated in this document.

7.2 QT and QML

Qt currently has support for Android, iOS and Windows phone OSs. One of its main features is that it offers an easy UI development and native performance. However, it might be necessary to acquire a commercial license to access the app markets without the LGPL license requirements. (26.)

7.2.1 Signals and Slots

Most UI toolkits use the observer pattern to detect an action and react to it. JavaScript uses *callbacks*, which is a variant of it. Instead, Qt uses *signals* and *slot,* which is something that makes the application development with Qt a bit different but it also brings an easier way to handle events. “A signal is a message that an object can send, most of the time to inform of a status change”, and “a slot is a function that is used to accept and respond to a signal” (27).

7.2.2 Modules

Qt is split into modules and they can be included in the projects depending on its needs. Some modules relevant for this project are:

- Qt QML: Module that brings support for QML and JavaScript languages.
- Qt Quick: Module for GUI.
- Qt Quick Controls: Module that brings widget-like controls.
- Qt Multimedia: Module for audio, video, radio and camera functionality.

Those modules have dependencies on other Qt modules, such as Qt Core, Qt GUI, and Qt Network. (21.)

7.3 Comparison elements

This chapter presents a list of criteria and their definitions to compare the different technologies to develop mobile applications. These criteria are based on the ones proposed by Heitkötter, Hanske and Majchrzak. (27, p. 299-301.)

Resource consumption

Definition: the consumption of operational memory and storage space.

The operational memory consumption will be measured by running a “top” command over Android Debugging Bridge (ADB) from a computer while the device is plugged and running the application to be tested. This measurement will be done at two different moments: right after starting and after visiting all the application screens.

The storage space will be measured directly on each device.

Application speed

Definition: the speed of the application at the start-up and runtime.

The start-up speed will be measured by using different development tools for iOS and Android. iOS Xcode instruments will be used on it, and on Android the top command.

The runtime speed is measured subjectively through an assessment of the user experience on elements, such as responsiveness and user interaction. It will also be noted the CPU usage.

System integration

Definition: whether the technology supports a native look and feel of the operating system in question.

Supported platforms

Definition: the amount of supported mobile platforms, with special attention to whether the technology supports all the platforms to the same extent.

Debugging

Definition: whether a debugging tool is available, its amount of features, and ease of use.

Access to platform-specific features

Definition: the access to a device hardware.

In this case, it will be evaluated by getting access to a camera of the device to take a picture.

Ease of development

Definition: the availability and quality of the documentation and the learning curve. The learning curve can be understood as the subjective progress of the developer during their first contact with the technology in case.

Customisability

Definition: the ability to customize the look and behaviour of user interface components.

Responsive design

Definition: the ease of development for various screen sizes and densities.

License and costs

Definition: the license under which the framework is published, how the framework is distributed e.g. open source, free software, if the developer can create free commercial applications or whether there is an additional cost for it.

7.4 Application design

The same simple mobile application will be developed using two different technologies: Qt and Ionic. This application will consist of three different views, which will use common UI elements and access at least one hardware element, such as the camera or GPS. If possible, the application will be made with responsive design.

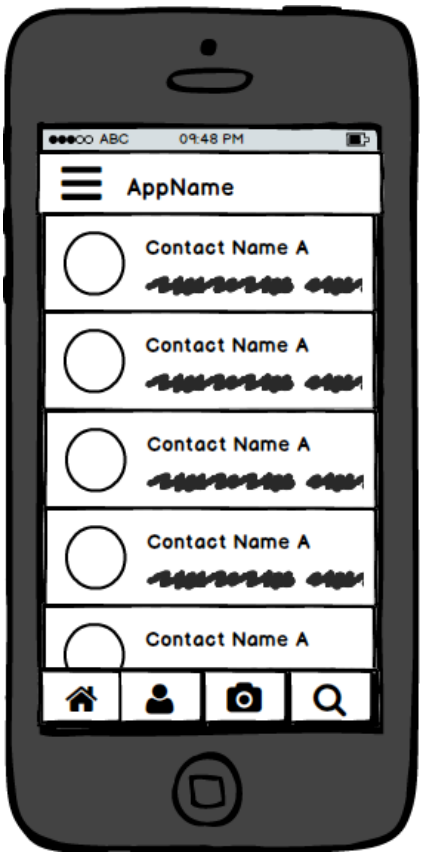

FIGURE 2. Wireframe of a list view

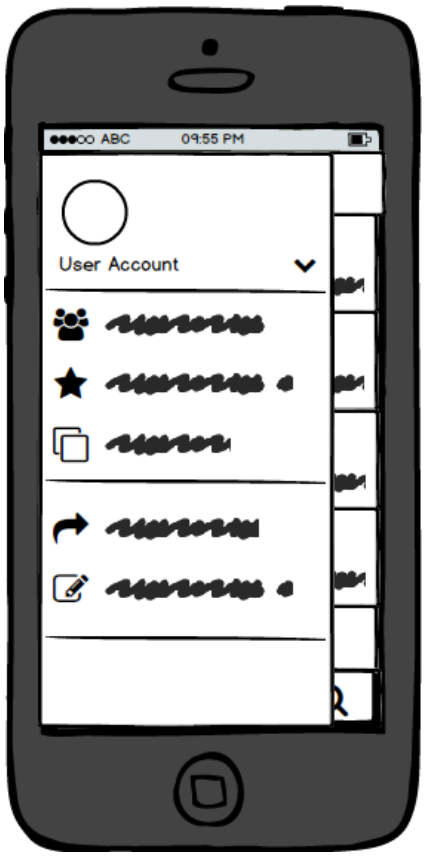

FIGURE 3. Wireframe of a left drawer

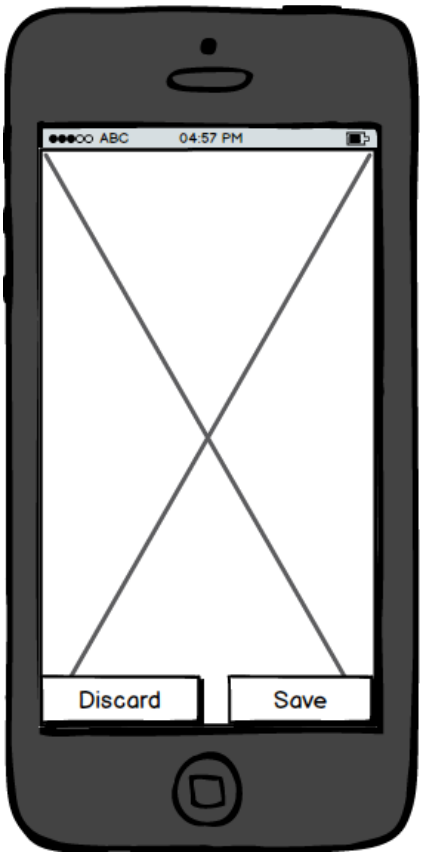

FIGURE 4. Wireframe of picture
taken with the camera of the device

7.4.1 Application UI elements examples in iOS and Android applications

After going through some popular applications that are both available for Android and iOS devices, it was found that the general trend is to use a very similar style called ‘Material Design’. However, both operating systems have different guidelines when it comes to UI components, their layout and behaviour.

An example of how the same application UI components look and are arranged in a different manner can be seen in the WhatsApp application (Figure 4).

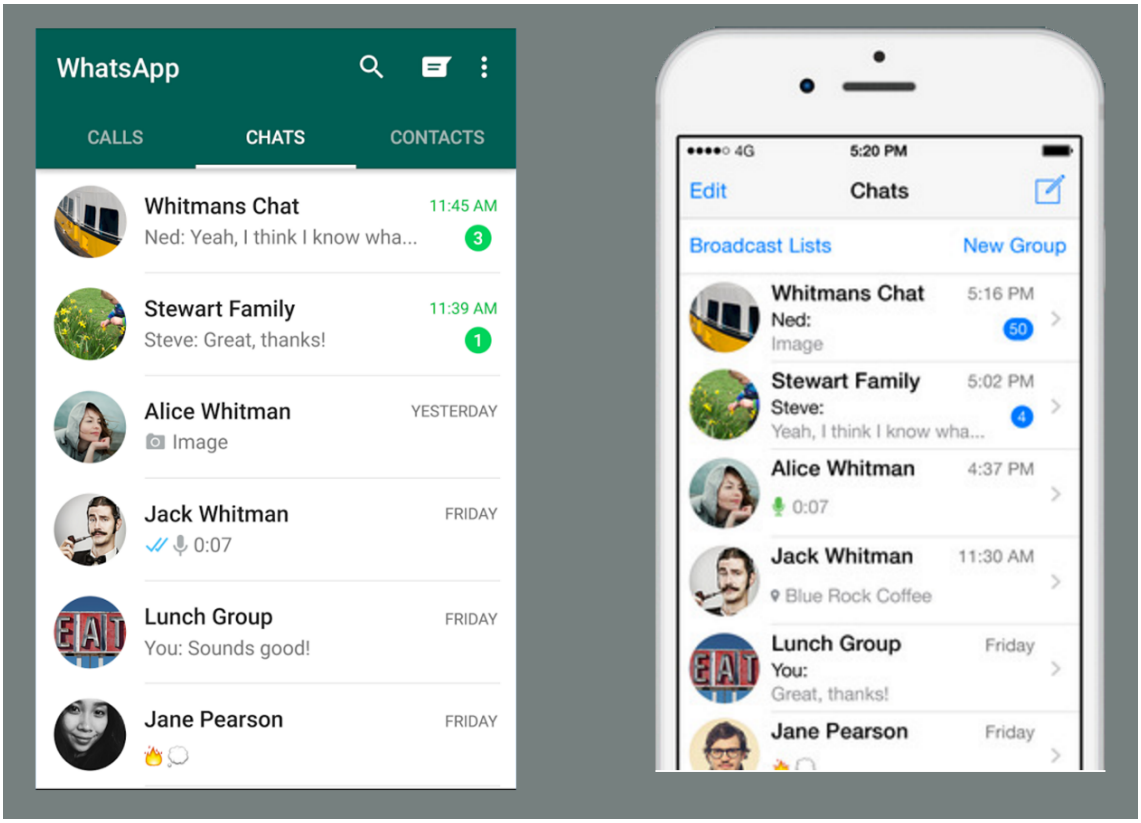

FIGURE 5. Screenshot of WhatsApp application list view for Android and iOS OSs (from left to right) (29)

7.5 Application Development Process

With either framework, to be able to develop an application for iOS, the following software is needed:

- OS X Operating System version 10.9 or greater
- Xcode
- iOS SDK
- If you want to install the application in a device other than a simulator:
    - Apple Developer license
    - Device running iOS 8 or higher

To be able to develop an application for Android, Java Development Kit 7 and Android SDK are needed.

7.5.1 Ionic Framework

To get an Ionic development environment setup, the following software needs to be installed:

- Node.js
- Cordova and ionic command line tools (available through NPM)
- Platform-specific tools for Android
- Platform-specific tools for iOS

To start a project, it is possible to get a ready-made application template. In this case a ‘sidemenu’ template was used. It comes with a central view and a left drawer.

After that, multiple views were created with the simple HTML code and by using Ionic CSS classes and directives and Angular directives. The navigation rules were configured by using the Angular routing.

During all the process, the application was easily tested in a local browser. This makes things faster as the changes can be seen immediately and the application does not need to be deployed to a mobile device. Due to this, debugging is also done through the browser developer tools, which are very familiar to web developers. Debugging on a mobile device through the browser is also possible very easily.

Angular services were used to emulate the data received from a server. To use camera of the device, all that was needed was to install Cordova’s camera plugin and to configure it with a few lines of code as seen in Figure 6.

```
.controller('CameraCtrl', function($scope) {
  $scope.upload = {};

  function getPicture(sourceType) {
    navigator.camera.getPicture(
      function(imageData) {
        $scope.upload.picture = imageData;
        $scope.$apply();
      },
      function() {
        $scope.upload.picture = undefined;
        $scope.$apply();
      },
      {
        quality: 80,
        encodingType: Camera.EncodingType.JPEG,
        destinationType: Camera.DestinationType.DATA_URL,
        sourceType: sourceType,
        saveToPhotoAlbum: false
      }
    );
  }

  $scope.takePicture = function() {
   getPicture(Camera.PictureSourceType.CAMERA);
  };

  $scope.selectPicture = function() {
    getPicture(Camera.PictureSourceType.PHOTOLIBRARY);
  };
})
```

FIGURE 6. Camera plugin configuration in an Ionic application

To be able to use the camera on an iOS device, one has to set the appropriate privacy camera permission with a description in the info.plist file.

Finally, the application was tested on Android and iOS devices by generating release builds and installing them on the devices.

Application source code and deployment

The application repository is available on GitHub:
https://github.com/crcvv/ThesisIonic

The following steps are needed to deploy the application on an Android device:

1. Connect the device via a USB, which needs to have the development mode enabled.
2. Build the application by executing the following command in the root directory of the application: `ionic run android`

The following steps are needed to deploy the application on an iOS device:

1. Build a release version by executing the following command in the root directory of the application: `ionic build ios --release`
2. Open the project in Xcode.
3. Connect Xcode to an Apple developer account and also register the device.
4. In Xcode, set the build configuration scheme to “Release”.
5. Connect the device to the computer, and select it from the Scheme toolbar menu in the project editor.
6. Press the “play” button to install the application to the device.

7.5.2 Qt/QML

To setup the development environment for Qt the following was needed:

- Qt 5
- Qt creator IDE
- Android NDK (to compile native code)

After installing all the required software, the paths to Android SDK and NDK need to be specified.

The development with Qt required reading the documentation and following tutorial examples to learn how to use the elements required for the applications.

To start the project, a new Qt Quick Controls 2 application was created with Android and iOS kits. This generates a UI file that can be used for the main view.

Afterwards, each view was built from the ground up from QML components that come from Controls 2 and from Qml-Bootstrap libraries (30).

The camera comes from the Qt Multimedia module, which comes with ready-made functions that do not require much extra configuration other than to assign where the camera view will be displayed and which buttons will control it as can be seen in Figure 7.

```
id: cameraViews

ColumnLayout {
    spacing: 10
    anchors.fill: parent

    VideoOutput {
        autoOrientation: true
        Layout.fillHeight: true
        Layout.fillWidth: true
        source: camera
        focus: visible
    }

    ButtonDefault {
        text: "Take picture"
        Layout.fillWidth: true
        icon: FontAwesome.icons.fa_camera
        class_name: "royal large"
        iconRight: false
        onClicked: camera.imageCapture.capture()
    }
```

FIGURE 7. Camera placement in the view in the Qt application

The contrast of QML code against JavaScript relies on its declarative nature, and it is visible on the way the application code is structured. In the views the components are declared as blocks that have properties. This code resembles a mix between CSS and JavaScript (See Figure 8). There is sometimes a bit of logic code included in the views, for example on buttons that trigger a view switch.

```
ButtonDefault {
    text: "Send email"
    anchors.bottom: parent.bottom
    anchors.bottomMargin: 0
    anchors.left: parent.left
    anchors.right: parent.right
    class_name: "energized large"
    iconRight: false
    icon: FontAwesome.icons.fa_envelope
}
```

FIGURE 8. QML button declaration in a sample view in the Qt application

The UI logic in QML is written in JavaScript, while the functional part of the application is written in C++. This has the advantage of a native performance and

a static type checking. The disadvantage is that the programmer must know two languages. In comparison, in Ionic the whole application is implemented in JavaScript, unless one decides to write a custom plugin. In such case, it would be difficult to do and it would need to be done separately for every platform.

Application source code and deployment

The application repository is available on GitHub:
https://github.com/crcvv/ThesisQt

To deploy the application on an Android device the following steps are needed:

1. Open the project in Qt Creator.
2. Select Android/debug and press the “play” button to install the application to the device.

To deploy the application on an iOS device the following steps are needed:

1. Open the project in Qt Creator.
2. Select iphoneos/release and press build to create the Xcode project.
3. Open the project in Xcode.
4. Connect Xcode to an Apple developer account and also register the device.
5. In Xcode, set the build configuration scheme to “Release”.
6. Connect the device to the computer and select it from the Scheme toolbar menu in the project editor.
7. Press the “play” button to install the application to the device.

8 RESULTS AND CONCLUSIONS

The frameworks described in the section 5 were compared according to the criteria in the section 6. This was done by developing an application for Android and iOS OSs with each framework and trying to make the views as close to the design as possible and with as little variation as possible between each other.

The fulfilment of each comparison element was graded on a scale from 0 to 5, with 0 meaning that the element was not present, 1 “very poor”, and 5 “very good”. In addition, problems and notes were collected during the app development.

The application testing in devices was done for Android on a Samsung Galaxy Note 3 with Android 5.0, and on iOS on an iPad (4th generation) with iOS 10.2.1.

All four applications UI elements in the different views can be seen in Appendix 1.

8.1 Ionic Framework

TABLE 2. Evaluation of Ionic Framework

Criteria	Grade	Notes
Resource consumption	5	Storage space in Android: 4.00 MB Storage space in iOS: 10.9 MB
Application speed	5	Start-up speed Android: 1.6 s Start-up speed iOS: 2.72 s Runtime speed–Android: Average low CPU usage of 13%; when in idle state the CPU usage is 0%, and during transitions between application screens, the CPU usage peaks to around 39%. Runtime speed–iOS: The CPU usage is low when idle (below 5%). The CPU usage peaks to around 60% for about 300ms during transitions between application screens. (100% corresponds to all cores fully utilized)

System integration	5	The framework supports current style trends for Android and iOS and also gives support to make platform-specific changes.
Supported platforms	5	Ionic Framework currently supports iOS 7 and up, and Android 4.1 and up.
Debugging	5	Debugging can be done through a web developer console (i.e. Google Chrome's developer tools, Firefox) on a local web deployment, or by accessing the application running on the phone. This approach is no different from debugging a website, thus enabling real-time and quick debugging
Access to platform-specific features	5	The framework has plugins that allow for the developer to access these features in both, Android and iOS without much effort other than to include the plugin in the project and specify certain parameters. In this specific case, when using the camera plugin the way it is prepared, it allows for direct platform access and not an embedded view to take the picture.
Ease of development	5	Documentation is available through their official website where there are many UI examples available. In addition there is an online community and there is support by the company members in some cases. The learning curve was quite flat due to previous experience with Angular, JavaScript, CSS frameworks and HTML. There is a very active community developing plugins through the Cordova project and The Ionic Framework team keeps developing new features.
Customisability	4	The framework comes already with many pre-made UI components and with a uniform Material Design for Android devices and iOS Styles for iOS devices, which vary on the platform specific guidelines. The framework allows for variables customization (i.e. colour changes), but it also allows for specific style changes for each supported platform through AngularJS and SASS.
Responsive design	5	The framework is designed for responsive design that works on multiple screen sizes and densities. There is no need for developer input in this subject.
License and costs	5	Ionic framework is a free and open source project, distributed under MIT license. It is based above another open source project called Apache Cordova. Developers are allowed to create free or commercial applications without additional costs. (25.)

8.2 QT/QML

TABLE 3. Evaluation of Qt/QML

Criteria	Grade	Notes
Resource consumption	1	Storage space in Android 5.0: 41.52 MB Storage space in iOS 10.2.1: 42.7 MB
Application speed	5	Start-up speed–Android: 1.0 s Start-up speed–iOS: 4.8 s Runtime speed–Android: Average low CPU usage of 13%; when in idle state the CPU usage is about 10% or below. The CPU usage peaks go up to 46% during transitions between application screens. Runtime speed–iOS: Very fluid and fast transitions between views. The CPU usage is low when idle (below 5%). The CPU usage peaks to 40%, exceptionally to 50% during transitions between application screens. (100% corresponds to all cores fully utilized)
System integration	4	Back in 2015, when this thesis was in its initial phase, QML didn't have much support in its libraries for standard mobile UI components such as drawers, panes, popups, etc. They did have Qt Quick Controls, a QML library, but it was developed with desktop devices in mind. It was until June 2016 that Qt Quick Controls 2 was released with support for mobile and embedded devices, aiming at a better performance and lower memory consumption. However, they did lack native style integration until a few months later they released support for default, material (based on Google's Material design) and universal (based on Microsoft Universal design) styles. It is important to note that the UI elements included in this library are just a few of the wide variety offered by other frameworks. When doing a more through search, there was a public GitHub repository "qml-bootstrap" that offers more UI components, and a ionic framework-like style for all UI components. (30; 31.)
Supported platforms	5	It supports both Android and iOS platforms. The Qt Creator seems to be out of date to work with the latest version of Xcode, which is necessary to test the application being developed on a device. However, after some workarounds it was possible to make it work without any need for the developer to make changes in the application code.

Debugging	5	Qt uses a debugger through Qt Creator, which has basic features to see runtime and build errors. It is quite simple to use and similar to web debuggers.
Access to platform-specific features	5	Qt provides libraries that enhance access to hardware, in this case, the camera. It also allows for high customization of the UI.
Ease of development	4	For a person with a web development background and just basic experience with C++, the learning curve can be quite steep. It is true that QML does resemble CSS, but it is not so easy to get a grasp on. There is a graphic designer for UI design called Qt Creator, which in theory should make UI design by far easier. During the development of the sample applications Qt creator design interface didn't make development much faster. However, it does help when trying to visualize the code without need for launching the application either in desktop or mobile version.
Customisability	5	Qt/QML allows for as much customization as raw HTML and CSS. However one has to build most UI elements from scratch, which can be cumbersome. Although, after further research, the open-source community has developed UI components based on other popular frameworks, which allows for a wider variety and ease of development.
Responsive design	5	The framework is designed to be responsive by default and to work properly on different screen layouts and densities.
License and costs	5	Qt has a commercial and open source licenses. Developers are free to develop open source free or commercial applications without any cost under LGPL3, GPL2 or GPLv3 licenses. However, for a commercial non-open source application, they must purchase a Qt commercial license per developer, either a subscription license or a perpetual one. (32.)

8.3 Conclusions

From a web developer's point of view, developing a mobile application with a JavaScript-based framework felt easier due to the familiarity with web technologies and how Ionic Framework CSS components for UI implementation use is very similar to CSS frameworks, such as Bootstrap.

Another strength of web frameworks, such as Ionic Framework, is the platform-based customisation. It includes a certain style that is similar to what each OS guidelines have, but it also offers the possibility to add a completely different style for any platform via AngularJS and dynamic templates loading. (33.)

In relation to performance and speed, the applications behaved well in both operating systems. On Android, there were no noticeable differences between Qt and Ionic applications. The average CPU consumption was low and the subjective user experience was good in both cases. On iOS, the Qt application had a noticeably better user experience, the transitions seemed immediate and there were no noticeable delays. The Ionic application on iOS had a slightly higher CPU usage peaks during view transitions, which could explain the slightly noticeable delay. Thus, the performance would not be a decisive factor in this case.

Qt is very popular among Linux OS users for the development of desktop applications for multiple platforms. However, it is missing discoverability as Linux users are a minority against the large database of the two other major OSs, Windows and OS X. Nevertheless, Qt has a lot of potential when it comes to applications different from the information context, such as utility, productivity and immersive contexts. This is mostly due to performance requirements of such applications because a web technologies-based application would not be able to meet the logic demands behind them.

Some of the downsides of using the Qt framework are the size of the application and license costs. The large size of Qt applications is due to the inclusion of all libraries. This could be improved if there were many Qt applications, which could share those libraries, and therefore, there would be need for them to be installed just once. When it comes to licensing, a commercial license must be purchased per developer if an application will be published with a license other than LGPL3, GPL2 or GPLv3 licenses.

While developing the application with Qt/QML, there was a steeper learning curve due to the use of C++ language and QML. Nevertheless, once the basic functionality was learned, the UI development process was very similar to that

of raw HTML and CSS. There are some pre-built UI elements that come from Qt Controls 2.0, but they are not enough or at least not as diverse as what Ionic Framework offers by default. Nevertheless, Qt framework's versatility should not be dismissed as it enables to develop fast and responsive multi-platform applications with access to platform-specific features, with a native performance.

In conclusion, the Ionic Framework has more strengths than weaknesses for the information context mobile applications development, especially when it is focused on the UI design.

9 DISCUSSION

In this project a sample application, which matches mostly information context applications, was developed. This means that the results can only be generalized for this group of applications. For example, Qt/QML can be a much better tool for creating utility, productivity and immersive applications when compared to Ionic Framework, which is mostly aimed at informative context applications. In such cases Qt/QML has an advantage against native development, which is the shared code base among platforms.

As most programmers start as web developers, it is very common and easier for them to transit to application development with web technologies-based tools. In addition, there are large companies, such as Google and Mozilla, who put a lot of effort into making browsers faster and also increasing the amount of features that they support.

In addition, there is a high demand for applications of an information context type, which basically have the character of a website. Therefore, web-based frameworks, together with a packaging tool, make the application development straightforward.

REFERENCES

1. Galitz, W. 2007.The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. United States of America: Wiley Publishing Inc.

2. McWherter, J. & Gowell, S. 2009. Professional Mobile Application Development. United States of America: John Wiley & Sons, Inc. Date of retrieval 15.02.2016 http://site.ebrary.com.ezp.oamk.fi:2048/lib/oamk/reader.action?docID=10593160&ppg=13

3. Pratas, A. 2014. Creating Flat Design Websites. United Kingdom: Packt Publishing Ltd. Date of retrieval 3.03.2016 http://proquest.safaribooksonline.com/book/web-development/html/9781783980048

4. Nielsen, J. 2012. Usability 101: Introduction to Usability. Date of retrieval 3.03.2016 https://www.nngroup.com/articles/usability-101-introduction-to-usability

5. Schade, A. 2014. Responsive Web Design (RWD) and User Experience. Date of retrieval 7.03.2016 https://www.nngroup.com/articles/responsive-web-design-definition

6. Norman, D. & Nielsen, N. 2017. The Definition of User Experience. Date of retrieval 31.01.2017 https://www.nngroup.com/articles/definition-user-experience/

7. Compuware Corporation. 2017. Mobile Apps: What Consumers Really Need and Want. Date of retrieval 31.01.2017 https://info.dynatrace.com/rs/compuware/images/Mobile_App_Survey_Report.pdf

8. Fling, B. 2009. Mobile design and development. Sebastopol, CA: O'Reilly Media, Inc.

9. International Data Corporation. 2016. Smartphone OS Market Share, 2016 Q3. Date of retrieval 20.01.2016 http://www.idc.com/promo/smartphone-market-share/os;jsessionid=37194528FE9FEB76C1B2F01C0FE466B7

10. Nielsen, J. & Budiu R. 2013. Mobile Usability. United States of America: New Riders.

11. Panhale, M. 2016. Beginning Hybrid Mobile Application Development. Date of retrieval 17.01.2017 http://proquest.safaribooksonline.com/book/programming/mobile/9781484213148

12. Saleh, H., Holmes, E., Bray, T. & Yusuf, S. 2016. Mobile Application Development: JavaScript Frameworks. Date of retrieval 26.01.2016 http://proquest.safaribooksonline.com/book/programming/javascript/9781787129955

13. Sheehan, M. 2015. Developing Mobile Web ArcGIS Applications. United Kingdom: Packt Publishing. Date of retrieval 7.03.2016 http://proquest.safaribooksonline.com/book/web-applications-and-services/9781784395797

14. Coremans, C. 2015. HTML: A Beginner's Tutorial. Brainy Software. Date of retrieval 14.03.2016 http://proquest.safaribooksonline.com/book/web-development/html/9781771970181

15. Refsnes Data. 2016. CSS Introduction. Date of retrieval 14.03.2016 http://www.w3schools.com/css/css_intro.asp

16. Jehl S. 2014. Responsible Responsive Design. New York: Jeffrey Zeldman. Date of retrieval 14.03.2016 http://proquest.safaribooksonline.com/book/software-engineering-and-development/9780134077987

17. Mozilla Developer Network. 2016. JavaScript. Date of retrieval 9.04.2016 https://developer.mozilla.org/en-US/docs/Web/JavaScript

18. Bouwkamp, K. 2016. The 9 Most In-Demand Programming Languages of 2016. Date of retrieval 18.01.2017 http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2016/

19. Saleh, H. 2014. JavaScript Mobile Application Development. United Kingdom: Packt Publishing. Date of retrieval 18.04.2016 http://proquest.safaribooksonline.com/book/programming/javascript/9781783554171

20. Lal, R. 2013. Digital Design Essentials: 100 Ways to Design Better Desktop, Web, and Mobile Interfaces. Osceola, US: Rockport Publishers. Date of retrieval 31.01.2017 http://site.ebrary.com.ezp.oamk.fi:2048/lib/oamk/detail.action?docID=10724268

21. The Qt Company Ltd. 2016a. About Qt. Date of retrieval 15.02.2016 http://wiki.qt.io/About_Qt

22. The Qt Company Ltd. 2016b. QML Applications. Date of retrieval 14.03.2016, http://doc.qt.io/qt-5/qmlapplications.html

23. The Qt Company Ltd. 2013. Introducing Qt Mobile. Date of retrieval 03.01.2017, http://blog.qt.io/blog/2013/12/12/introducing-qt-mobile/

24. Adobe Systems Inc. 2016. Adobe PhoneGap. Date of retrieval 23.05.2016 http://phonegap.com

25. Drifty Co. 2016a. Ionic Documentation Overview. Date of retrieval 23.05.2016 http://ionicframework.com/docs/overview

26. The Qt Company Ltd. 2015. Qt Documentation. Date of retrieval 23.10.2015 http://doc.qt.io/

27. The Qt Company Ltd. 2016c. Qt for Beginners. Date of retrieval 03.02.2017 https://wiki.qt.io/Qt_for_Beginners

28. Heitkötter, H., Hanshke, S., and Majchrzak T. 2012. Comparing Cross-Platform Development Approaches For Mobile Applications. Web Information Systems and Technologies, 299-311.

29. WhatsApp Inc. 2016. Date of retrieval 23.05.2016, https://www.whatsapp.com

30. Koumondji, B. 2015. Qml-bootstrap. Date of retrieval 01.12.2016, https://github.com/brexis/qml-bootstrap

31. The Qt Company Ltd. 2016d. Qt 5.7 released. Date of retrieval 01.12.2016 http://blog.qt.io/blog/2016/06/16/qt-5-7-released/

32. The Qt Company Ltd. 2016e. FAQ. Date of retrieval 29.11.2016 https://www.qt.io/faq

33. Drifty Co. 2016b. Platform Customization. Date of retrieval 23.01.2017, http://ionicframework.com/docs/platform-customization

APPENDICES

USER INTERFACE VIEWS ON EACH OPERATING SYSTEM

QT UI ON ANDROID

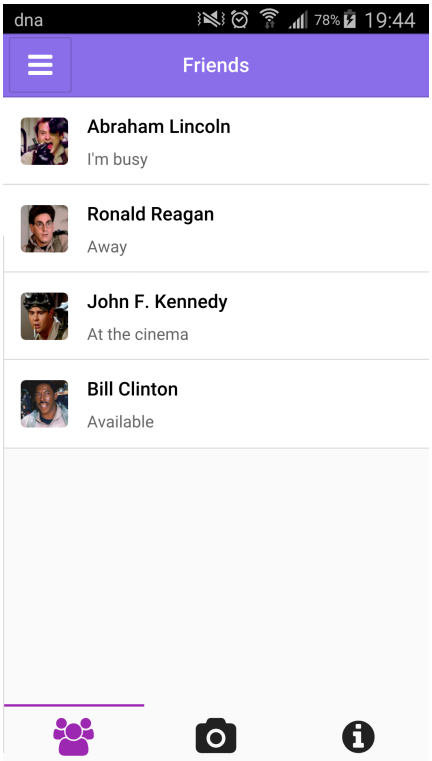

FIGURE 9. List view

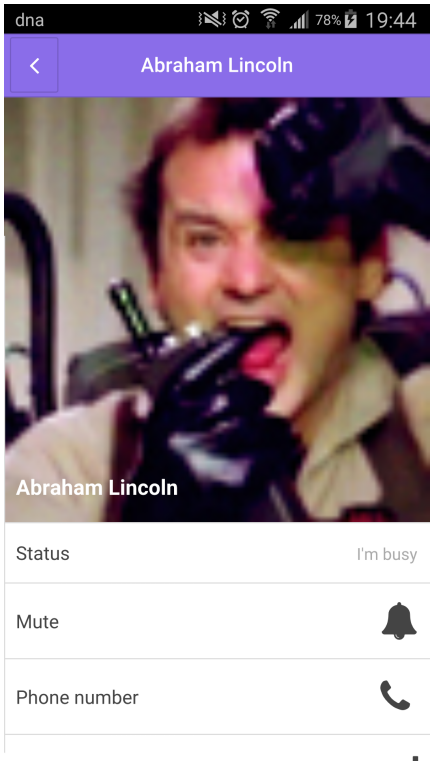

FIGURE 10. Detail view with back button

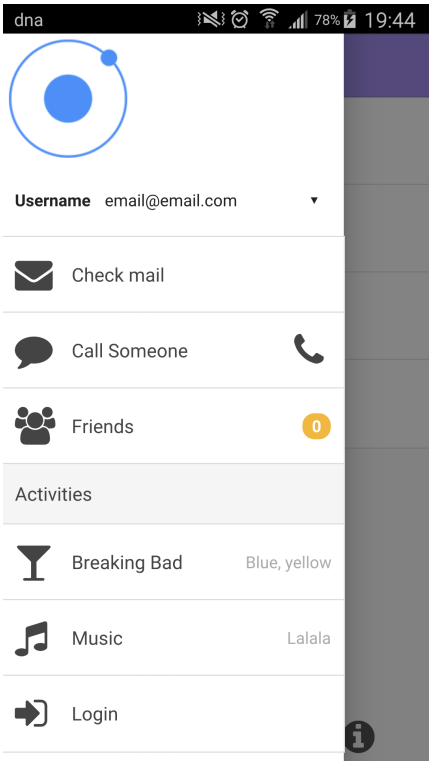

FIGURE 11. Left drawer view

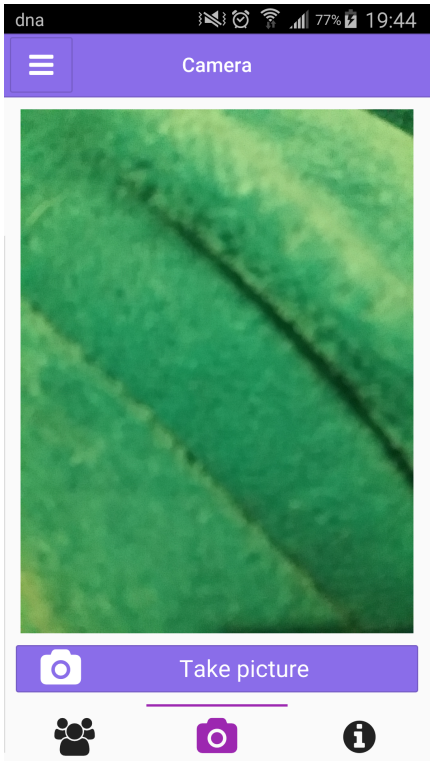

FIGURE 12. Camera view

QT UI ON IOS

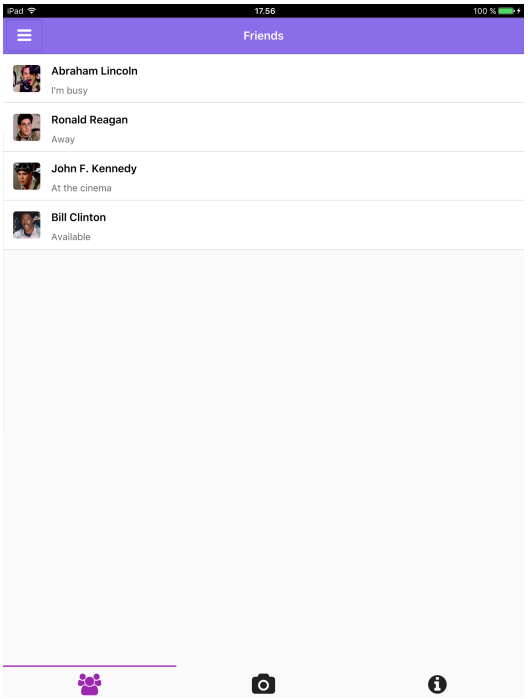

FIGURE 13. List view

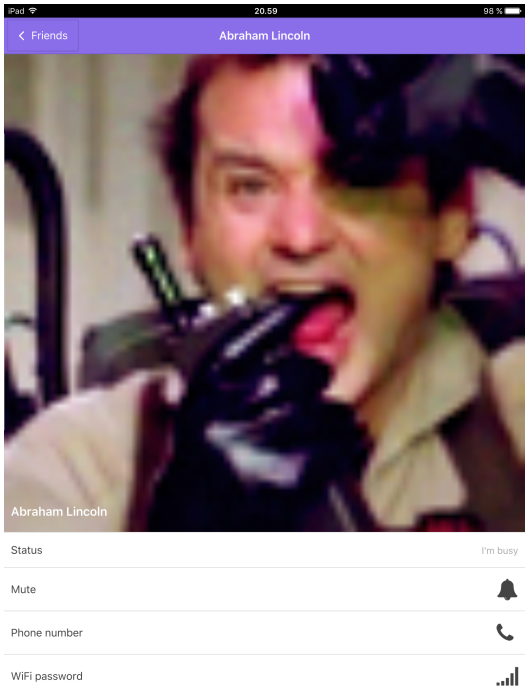

FIGURE 14. Detail view with back button and previous view name

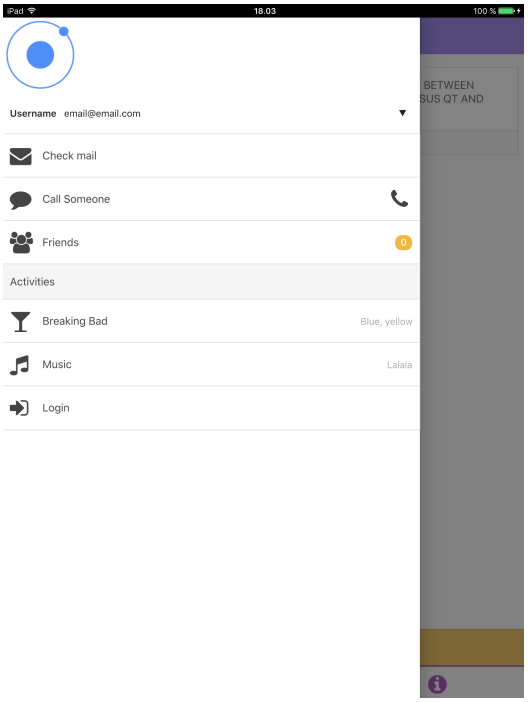

FIGURE 15. Left drawer view

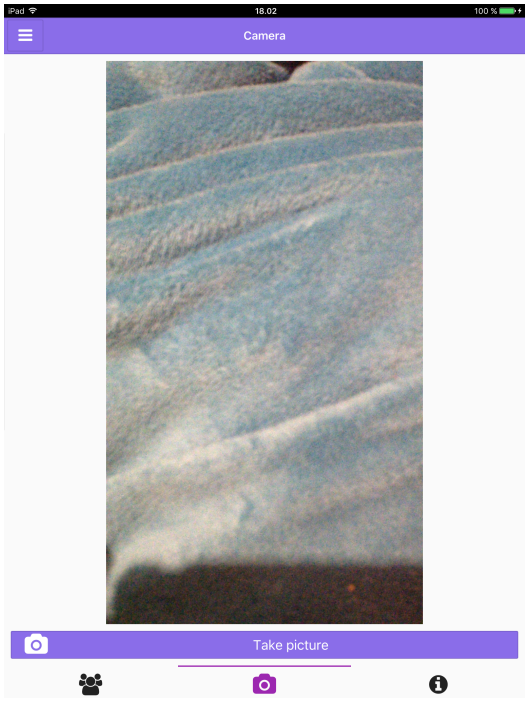

FIGURE 16. Embedded camera view

IONIC FRAMEWORK UI ON ANDROID

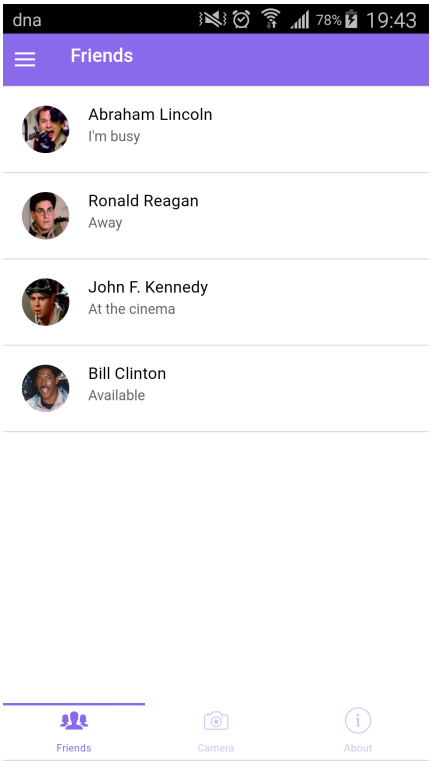

FIGURE 17. List view

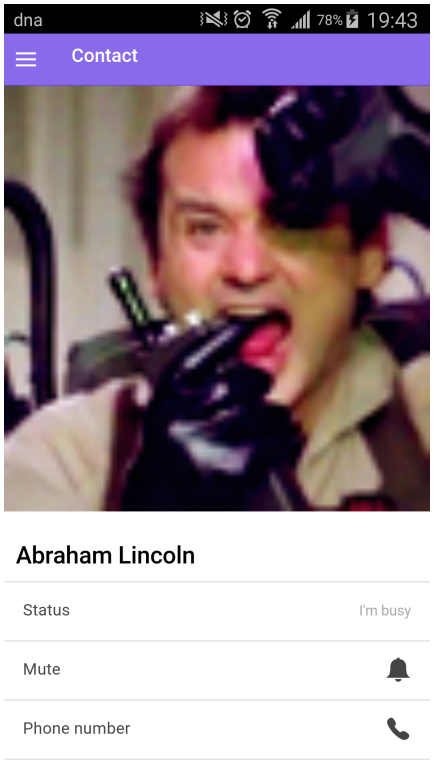

FIGURE 18. Detail view without back button

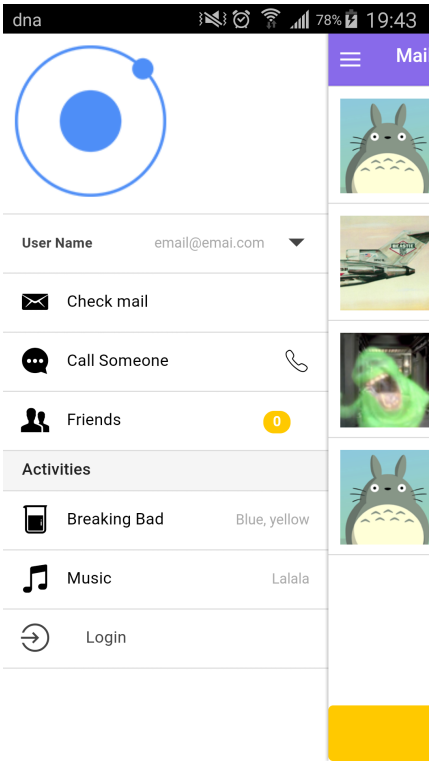

FIGURE 19. Left drawer view

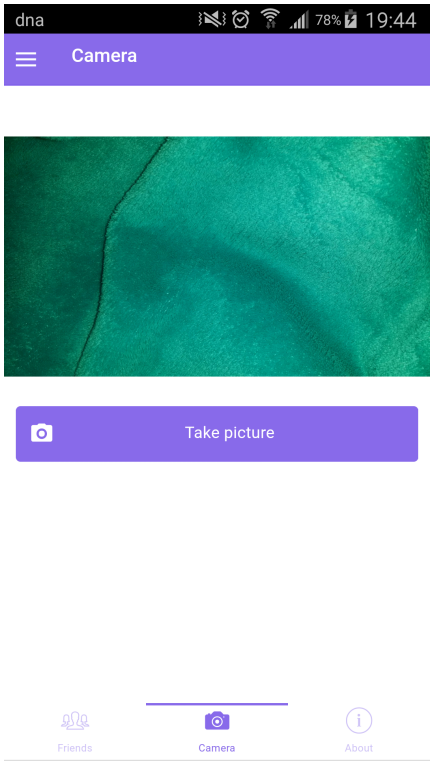

FIGURE 20. Camera view

IONIC FRAMEWORK UI ON IOS

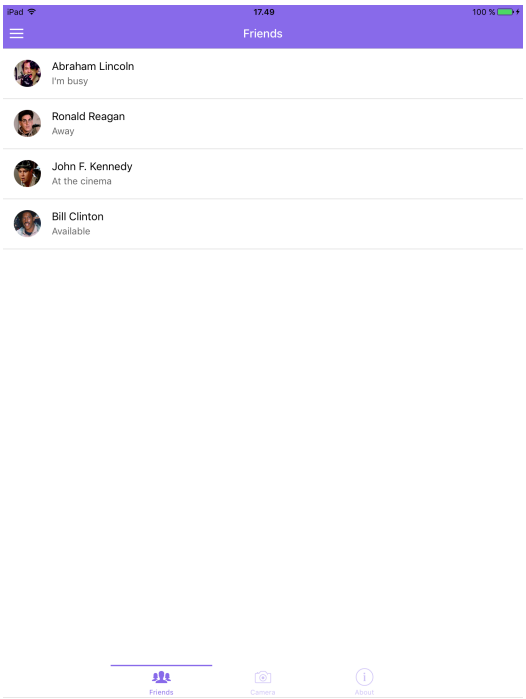

FIGURE 21. List view

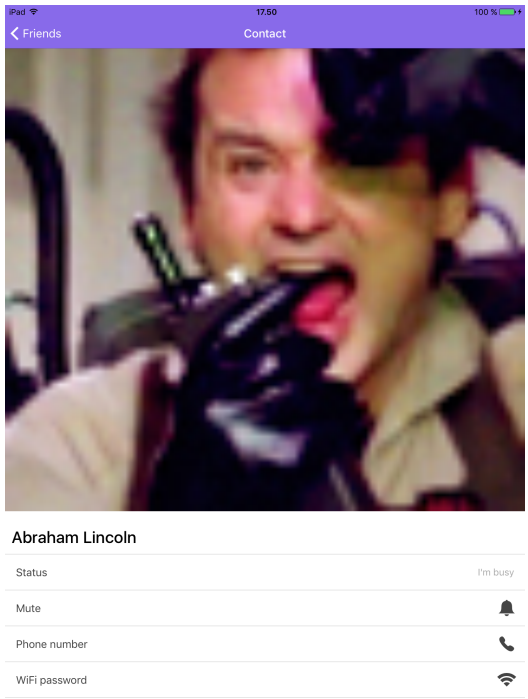

FIGURE 22. Detail view with back button and previous view name

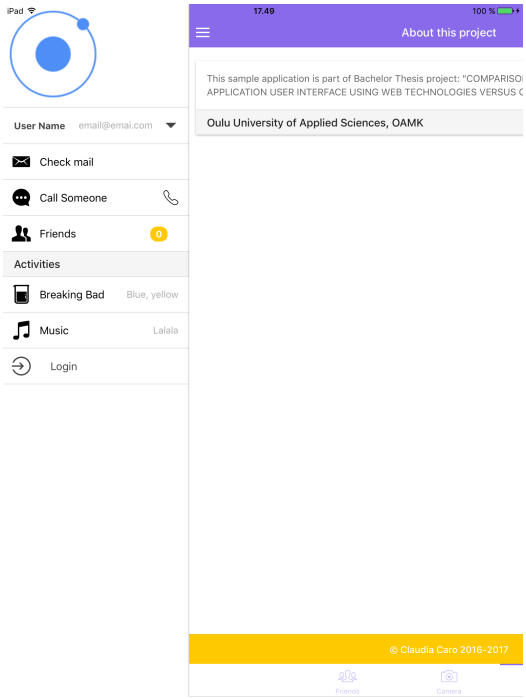

FIGURE 23. Left drawer view

FIGURE 24. Camera view