Master Thesis

React Native vs Xamarin – Mobile for industry

By

Dervis Avdic

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Abstract

This thesis is developed in collaboration with Tetra Pak AB. The assignment was to create a mobile application which would present information from a platform called ACT, Automation & Connectivity Tools. The mobile application would be called ACT Mobile and would also have functionality from ACT and one application which ACT provides. The ACT platform is a collection of applications that helps and facilitates the work for Tetra Pak employees. This can for example be with remote support and assistance functionality, where they don't have to be physically present. The mobile application would be developed within two different mobile frameworks in parallel. The two frameworks are React Native and Xamarin. Additional aspects to evaluate are also the maintainability of the system and performance. The result of the thesis was that ACT Mobile was further developed in Xamarin. One of the main reasons was the good knowledge of ASP.NET C# in Tetra Pak. That makes it easy for Tetra Pak to maintain and develop ACT mobile since Xamarin is based on ASP.NET.

Keywords: Mobile application, Xamarin, React Native, UI, UX, integration, Web API and Web application.

Sammanfattning

Detta examensarbete är genomfört i samarbete med Tetra Pak AB. Syftet är att utveckla en mobilapplikation som ska erhålla data från en plattform som heter ACT, Automation & Connectivity Tools. Den mobila applikationen ska heta ACT Mobile och inkludera funktionalitet från ACT och en applikation som plattformen tillhandahåller. ACT är en samling applikationer som hjälper och underlättar arbetet för anställda på Tetra Pak. Detta kan exempelvis vara med live videosamtal, där de inte behöver delta fysiskt för att kunna tillhandahålla hjälp. Den mobila applikationen ACT Mobile är utvecklad i två olika mobila ramverk parallellt. De två ramverken är React Native och Xamarin. En utvärdering av ramverken har gjorts huvudsakligen baserad på hur underhållsbart systemet är och i vilken prestanda som applikationen tillhandahåller. Resultatet blev att ACT Mobile utvecklades i Xamarin. Detta på grund av att kompetensen inom ASP.NETs C# är mycket bredare på företaget vilket leder till att vidareutveckling och underhåll blir mest tidseffektiv i Xamarin.

Nyckelord: Mobilapplikation, Xamarin, React Native, UI, UX, integration, Web API och Webapplikation.

Acknowledgments

This master thesis would not exist without the support and guidance of;

Rikard Carlsson, Manger Infrastructure – Tetra Pak AB.
I want to thank you for letting me have this opportunity.
Rikard has been my supervisor for this thesis and is also the functional product owner. He has helped me through the project, answering all my questions and has successfully managed to guide me and my work.

Jonas Melin, IT solution architect – Tetra Pak AB.
Jonas has been very helpful throughout the project and has helped me a lot in the begging and with certain difficulties with for example Web API and SOAP services. To that I am very grateful.

Carl Danell, Software developer – Tetra Pak AB.
Carl has been very helpful throughout the project and has helped me a lot in the begging and with certain difficulties with for example UI and UX. To that I am very grateful.

I am also grateful for all the help I have gotten from the team working with the ACT platform.

Christin Lindholm, Associate professor
I want to thank you for your guidance throughout my thesis and report and answering all my question regarding my thesis.

Christian Nyberg, Associate Professor
I want to thank you for answering my question regarding the thesis.

Contents

Abstract 2

Sammanfattning 3

Acknowledgments 4

1. Introduction 8

1.1. Background 8

1.2. Digitalization 9

1.3. Tetra Pak history 10

1.4. Purpose and Goal 10

1.4.1. Purpose 11

1.4.2. Goals 11

1.5. Problem definition 11

1.6. Limitations 12

2. Technical background 14

2.1. Automation & Connectivity Tools (ACT) 14

2.1.1. Service Platform 15

2.2. React Native 16

2.2.1. Advantages with React Native 17

2.2.2. Disadvantages with React Native 18

2.2.3. Redux 19

2.3. Xamarin 21

2.3.1. Advantages with Xamarin 22

2.3.2. Disadvantages with Xamarin 23

2.4. Xamarin vs React Native 24

2.4.1. Market 25

2.4.2. Availability 25

2.4.3. Compilation 26
2.4.4. Development Environment 26
2.4.5. Framework 26
2.4.6. Reusable code 27
3. Methodology and Analysis 28
3.1. Working methods 28
3.1.1. Agile & Scrum 30
3.1.2. Team Foundation Server 32
3.2. Solution of the problems 33
3.2.1. Problem solution - React Native 34
3.2.2. Problem solution - Xamarin 36
3.3. Source criticism 38
4. Results 42
4.1. Results 42
4.1.1. ACT Mobile - React Native 42
4.1.2. ACT Mobile – Xamarin 48
4.1.3. Differences between Xamarin and React Native 53
4.2. Final ACT Mobile 53
5. Conclusion 64
5.1. Result 64
5.2. Conclusion 68
5.3. Use of the prototype 70
6. Future work 72
6.1. Future development 72
References 74
Figure References 76

List of Acronyms..78

1. Introduction

This chapter gives an introduction to the thesis, its purpose, goals and problem definition. Finally, limitations of the thesis work are presented and what could be implemented in future versions is discussed.

1.1. Background

This master thesis is carried out in cooperation with Tetra Pak AB. The thesis describes how to create a mobile application for the Embedded Automation & Digitalization department which has developed the platform Automation & Connectivity Tools, ACT. The decision on which framework the mobile application should be developed in, is made by comparing and analyzing two different mobile frameworks, React Native which is a sub-library to React written in JSX (JavaScript) and Xamarin which is developed in ASP.NET.

The mobile application called ACT Mobile will communicate with the APIs that ACT is communicating with. The ACT application that is going to be displayed by ACT Mobile is determined by its possible mobile usability and demand from the users of ACT. A part of the work is therefore to investigate which application in ACT is most suitable and has the highest demand.

In parallel with this React Native and Xamarin are compared. This is done by developing ACT Mobile with both frameworks and compare them. ACT Mobile was built first in React Native where the start menu and dashboard were first designed and implemented. This was followed by designing and implementing the start menu and

dashboard in Xamarin. The reason for the parallel design and implementation was to continuously describe and note the differences of the frameworks. These observations were the basis for decisions on which framework to further develop ACT Mobile in.

Initially ACT Mobile was first developed in React Native where the design process began with a design mockup of the start menu and dashboard. This was done in order to receive feedback from the product owner and architect. The feedback determined the direction of the design process. The same was done with Xamarin. The design of ACT Mobile will be adjusted during the development especially in the beginning. Therefore, is it important to start out with an initial design which receives feedback from the product owner and architect.

The main target for the mobile application are Tetra Pak employees, Automation Specialists (AS) and Field Service Engineers (FSE) which are supporting the customers factories.

1.2. Digitalization

Our whole society is heading in a direction where the digitalization is inevitable. Within the food and beverage industry where Tetra Pak is involved there are many aspects of how to digitalize the industry in order to create efficient processing and packaging solutions. Tetra Pak has therefore established clear objectives for the whole company regarding digitalization.

This thesis examines one of the objectives which is striving to create factories which can be supported by AS and FSE based on all the data that is gathered. Therefore, creating a mobile application for the Tetra Pak employees that support the customers factories is one small step towards that objective. The focus is to be able to provide help and support without physical attendance.

1.3. Tetra Pak history

Tetra Pak AB was founded in 1951 in Lund, Sweden by Ruben Rausing. Shortly after the founding he started to create the first packaging system for dairy products, which was designed and manufactured by Tetra Pak. This was completed on the 18 of May that same year. In November 1952, the first product was created, a one deciliter cream-package [1].

In 1956 the company moved to their own factory at Råbyholm in Lund. Tetra Pak then began to create some unique packaging products for milk and other dairy products like Tetra Classic, Tetra Brik and Tetra Prisma. This got attention from outside of Sweden and this was the first time the company reached out to the world. Råbyholm is still one of their most important sites today and was their headquarters until late 20th century before the headquarter moved to Switzerland [1].

Tetra Pak is still today the world leader in food processing and packaging solutions for food. The company has more than 23,000 employees in over 85 different countries and these figures are increasing for every year. When the company was founded by Ruben Rausing in 1951 they specialized in creating and designing packaging systems. This is still a part of their income source. Today they focus in creating complete solutions for processing, packaging and distributing food products such as dairy products, juices, ice cream and cheese [1].

1.4. Purpose and Goal

Throughout the master thesis it is important to identify the purpose and to establish goals in order to facilitate the working process. The purpose and the main goals of this thesis are presented below.

1.4.1. Purpose

The purpose of this master thesis is to explore and investigate two different mobile frameworks, React Native and Xamarin. This is in order to establish an appropriate framework and develop a mobile application called ACT Mobile. This mobile application is built with base functions which are the login system and user functionalities from the original platform called ACT. Using the base functions one application from ACT is implemented in ACT Mobile for testing.

1.4.2. Goals

These are the main goals of this master thesis.

- Create a mobile application with a user interface adapted to Tetra Pak standard and which is which is audience adapted.
- Find the most suitable mobile development framework technology to create the application through a comparison between React-Native (JSX/JavaScript) and Xamarin (ASP.NETs C#).
- Investigate which application in ACT would bring most value in a mobile application in terms of usability and rapid access.
- Create a scale-able and maintainable integration with the ACT platform

1.5. Problem definition

The problem definition of the master thesis is presented below and is going to be used during the whole working process.

- How to collect feedback from the target user?
- Which comparison metrics are suitable when selecting between React Native and Xamarin?
- How to investigate which application in the ACT platform is going to be implemented as a web application?

- Which possibilities and limitations exist to create an integration between the mobile application and ACT platform?
- How should the mobile application behave when users are in the factory and when out of range?

1.6. Limitations

One of the limitations of the master thesis is that not all of the mobile development frameworks are used in the comparison. This is because React Native and Xamarin are frameworks used for cross-functionality development and this thesis doesn't investigate frameworks that use native development, such as Swift/C for XCode or Java for Android Studio. There are more cross-functionality development frameworks on the market, but they have a much smaller market share than React Native and Xamarin.

All the applications in the ACT platform are not implemented in ACT Mobile. This is due to the time aspect and complexity of understanding how the applications in ACT are structured and implemented. There was an original idea of creating a priority list of ACT applications to be transformed to mobile applications in ACT Mobile, but this was not completed. The reasons for this are presented in section 5, Conclusion.

2. Technical background

This chapter gives an introduction to the technical background of the thesis work. It describes ACT, Service Platform, Xamarin and React Native.

2.1. Automation & Connectivity Tools (ACT)

Automation & Connectivity Tools, called ACT is a platform developed by the Embedded Automation & Digitalization department of Tetra Pak. It is mainly used by automation engineers and field service engineers around the world to maintain different factories that are connected to the ACT platform. The ACT platform displays a variety of applications that their users need when working with an issue. Some of the applications within the ACT platform are Connectivity dashboard, Remote Service Unit (RSU) dashboard, Remote support and Remote assistance.

The ACT platform is installed on Tetra Pak's servers, where it can be access by the Tetra Pak LAN or by the RSU if the user is located at a customer site. Figure 1 presents how the RSU is located on the customer site and the Tetra Pak LAN. This design has been chosen, because the user is not always present at the customer site. Therefore, there must be an access point to the customer site through the RSU and out to the global Tetra Pak server.

The information that ACT presents is gathered by the RSU on the customer site. This data contains information about how the factories are operating and if there is any alarm about shutdown. The data that is being generated is sent to the server at Tetra Pak Lund and is then displayded by the ACT GUI. The data is continuously updated

through VPN from the stationed RSU server all the way up to the server located in Tetra Pak Lund. By this server set up there is a possibility to send and receive data.

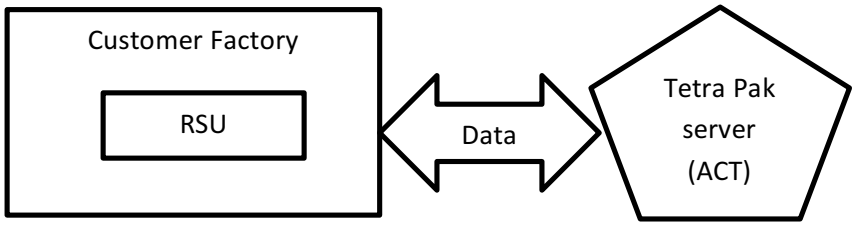

Figure 1 - RSU overview

2.1.1. Service Platform

Originally the ACT platform is a user interface upgrade from the previous platform called Service Platform. The Service Platform is the Tetra Pak generic channel for all electronics service product tools functionality in customer operational environment. It supports all common needs like user handling, single sign-on, language support, offline operations, multi device support and standard user interface. It is built on top of the Tetra Pak secure standard way of networking the customer with VPN technology.

The Service Platform is seen at figure 2 where the workspace lets the user navigate to and start the desired services. Service Platform provides an all-in-one window integrated layout, where all the windows and panels are integrated into a single larger application window.

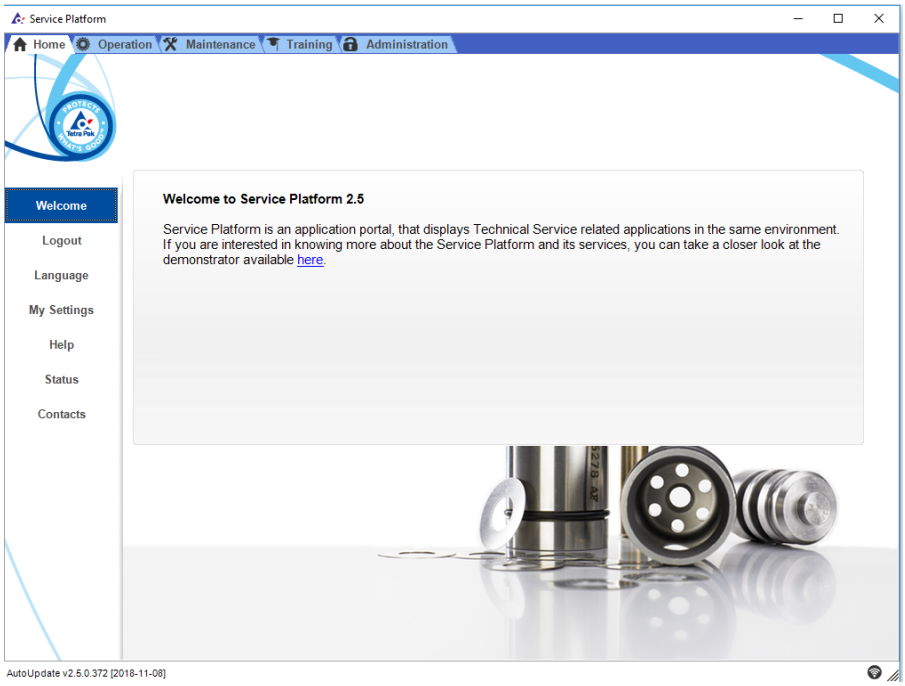

Figure 2 - Service Platform UI

2.2. React Native

React Native is a JavaScript framework which is used to write natively rendering mobile applications with cross-functionality. This means that applications developed by React Native can be used on both iOS and Android. React Native is originally based on Facebook's JavaScript library called React. This was created in order to build user interfaces for browsers and later on Facebook created a mobile framework called React Native. Web developers which have experience with JavaScript and React can easily write native applications for all mobile devices at the same time [2].

The behavior of React Native is very similar to React applications because they both use a combination of JavaScript and XML markup, known as JSX. React Native invokes the native rendering of APIs in Objective-C for iOS and Java for Android. In that way the mobile

application will render using real mobile user interface components and therefore look like any other mobile application [2].

2.2.1. Advantages with React Native

One of the largest advantages is that React Native renders using the platforms host standard APIs. This native rendering technique is not used by other cross-platform frameworks. Other frameworks that want to achieve something similar use a combination of JavaScript, HTML and CSS and renders their views using Web Views, which is a simplified web browser inside the mobile application [2].

It can therefore render native user interface elements on Android, iOS and Windows Phone views. Performance is important for mobile applications and React Native can achieve a high performance by working separately from the main UI thread. React Native renders views by changing the props or state. Props are so called properties that every component contains. A React Native application consists of components and each component represent a part of the GUI that is displayed on the mobile phone. It can also include some functionality to perform operations within the mobile application. This is an effective way to communicate between the components which render the user interface [2].

React Native and React change and modify content on the GUI by changing the state and props. React Native changes the state and props by mutating the user interface libraries provided by the mobile devices. This is done by the mobile manufactures to ensure that the correct content is displayed and that the mobile application developed doesn't creates performance issues. React changes the state and props by using HTML and CSS markup [2].

An additional advantage of the development environment in React Native is that the changes in the mobile application can instantly be previewed. This is important, because developers need feedback on

their projects. There is also a strong debugging tool integrated in React Native which uses a browser to present the debugging information. During the development the developer can debug the mobile application simultaneously as the debugging data is presented on a web browser, see figure 3 [2].

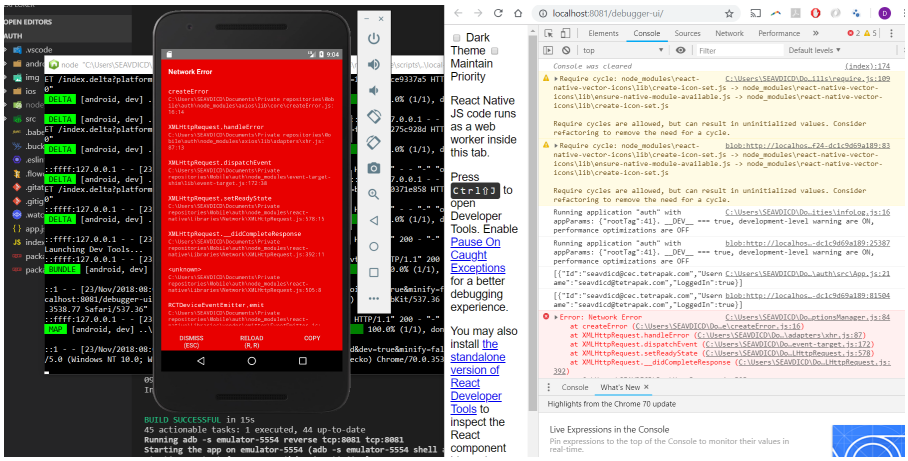

Figure 3 - Debugger for React Native

Reusing code with React Native is simple. React Native applications are built with components that can be reused within the application. All of the code can't be reused, because it targets different platforms. Facebook and Microsoft are both working on reusability, because it can save a lot of time during development. Within a large development project increased reusability could lead to large cost and time reductions [2].

2.2.2. Disadvantages with React Native

The largest risk developing with React Native is the framework's lack of maturity. The framework was released in March 2015 to iOS and

to Android in September 2015 and is therefore in an early stage of development. Another disadvantage is the lack of experienced React Native developers. Facebook, which founded React Native are continuously working on creating a sustainable and maintainable framework. [2].

One more disadvantage of React Native is also that because of its early stage there is not so much information on the framework and the solutions it offers. The traditional way of programming has been with Java for Android and Objective C for iOS [2].

React Native is built with JavaScript XML (JSX) which is a typescript language built on JavaScript. It is a different way of programming because is both involves code structure and design. New React Native users can perceive the framework to be difficult in the beginning and therefore can have a steeper learning curve [2].

2.2.3. Redux

React Native is suitable for writing small applications, but when the application is starting to grow and becomes more complex it will get complicated to handle all the state and props if there are a lot of components. Therefore, a library called Redux has been developed to make it easier to handle this complexity. [3].

Nowadays, user interfaces are becoming more complex and need quite a lot of maintenance. Routing is often implemented on the client so that we don’t need to refresh a browser in order to see some change. Before this, the application had to refresh the whole page. Routing on the client side is beneficial for performance, but it means that the client must handle more states compared to server-side routing. Managing all these states can be very difficult if not handled correctly [3].

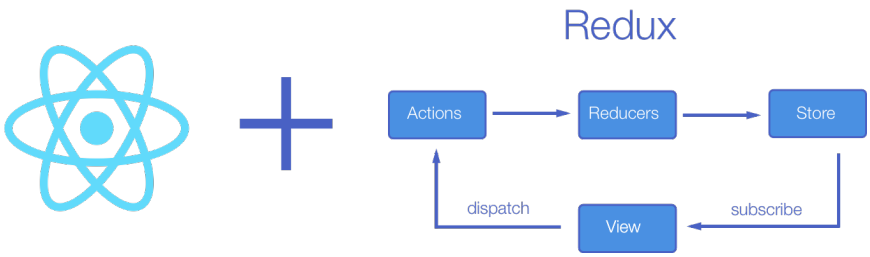

Figure 4 – React + Redux

Figure 4 describes how React and Redux operate together. Actions are JavaScript objects which describe the state changes in the application for the different components. Then there are action creators which basically behave like functions that take in parameters and return actions where they are supposed to change something. Reducers which receive actions and current state, return a new state and send it to the store. The store behaves like the core of Redux and stores and guards the states of the application. All of the components within the application can subscribe to state changes in the store and dispatch actions to it. Because of this every component can communicate with each other, rather than going to the parent node each time and further out. See figure 5 for more clarification [3].

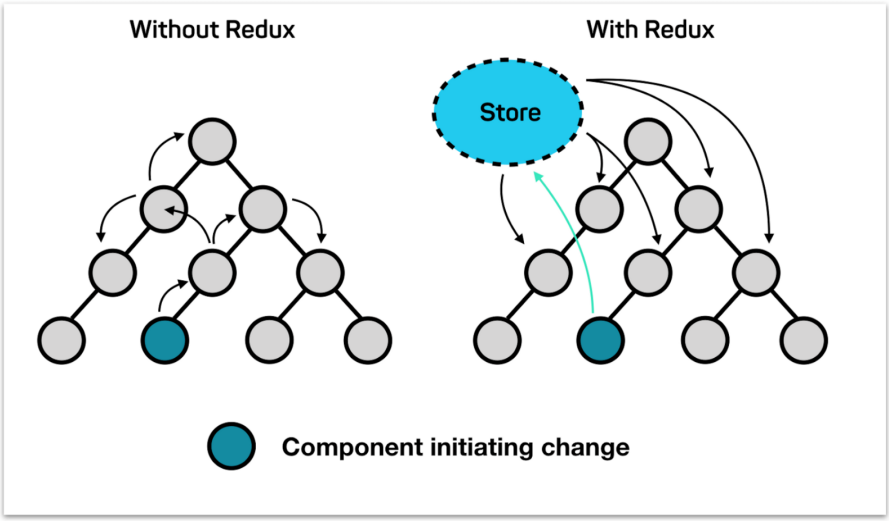

Figure 5 - Redux structure

2.3. Xamarin

Xamarin is a development platform which enables cross-platform development on iOS, Android and Windows phone through writing

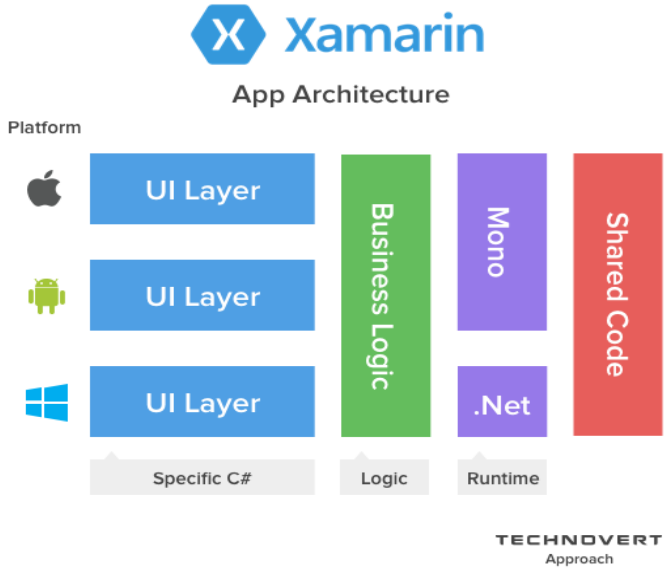

Figure 6 - Xamarin

C# code. The Xamarin platform ports .NET to the iOS and Android operating system and it also partly supports the Windows Phone, see figure 6 above [4].

Under Android and iOS layer there is Mono runtime. This is the bridge between C# and the native Android and iOS APIs. This enables the development to use Android and iOS user interface, notifications and all the features included in the phone. This is to create a development environment as if you were to develop a native application. Xamarin's takes advantage of .NET for the features regarding data types, generic types and garbage collection, Language-Integrated Queries, asynchronous programming patterns and Windows Communication Foundation communication. In order to connect all this to the Xamarin this is managed by a linker to only

include the referenced components. Xamarin forms which are used in this master thesis have a layer on top of the other UI bindings which provides a fully cross-platform UI library, see figure 7 [4].

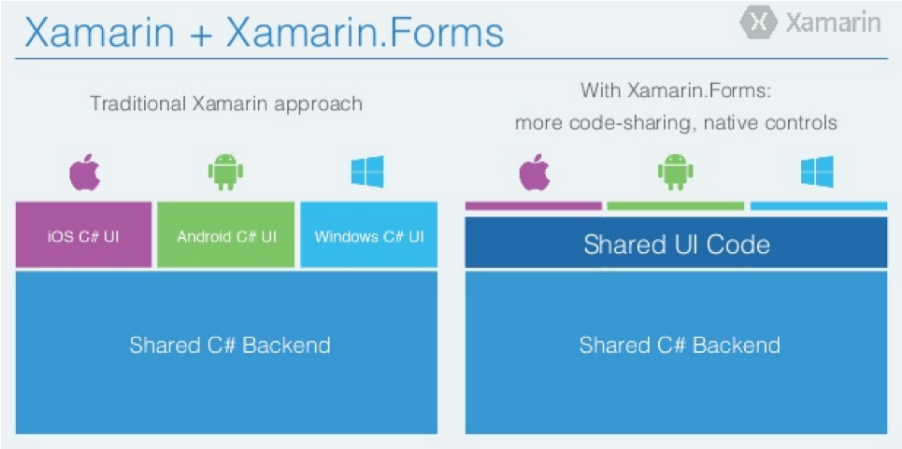

Figure 7 - Xamarin Forms

2.3.1. Advantages with Xamarin

One of Xamarin's largest advantage is the ability to reuse code. This can create an effect on the development, because it saves time and energy from the developers. Xamarin is also built with C# which is a modern and advanced language that is well known among the development and software business. Therefore, it is easy to find information and solutions regarding C# problems [5].

Due to the popularity of developing mobile application with Xamarin, it facilities business that emphasized modern mobile development. One reason is the fast development speed without compromising on cost. The speed of developing the first beta version can be rapid and it would also target all the different mobile operation systems, such as iOS, Android and Windows.

An additional benefit with Xamarin is the shared code base. When hiring a Xamarin developer, the developer has a good overview of the API, web services and input validation, because all is included within the framework. When hiring software developers for developing mobile application, costs could be reduced. This is due to the developer competence, because the developer creates one mobile application and it targets all the mobile operating systems. If the mobile application is built natively, targeting each mobile operating system would require three developers, one for each OS [5][7].

The Xamarin transforms UI components into platform-specific elements which results in a real native application and not in a hybrid application which could have impact on performance and user experience. This is important in today's mobile development [5][7].

One more practical advantages with Xamarin is that the framework allows for unit, integration and system testing due to the integration with .NET framework. Today's pressure on getting the product to the market is very critical. A lot of companies strive for automated testing on the devices and Xamarin opens the possibility to test the product for performance aspects and general bugs [6].

Much fewer developers use Xamarin than the native frameworks for Android and iOS. Therefore, a learning platform called Xamarin University has been developed. Experts on Xamarin and mobile applications can provide live and online classes for developers [6].

2.3.2. Disadvantages with Xamarin

One disadvantage with the Xamarin framework is that the installation files must include linkage and referencing for all mobile OS to work correctly. This can lead to large file sizes of 3 megabytes to 15 megabytes. Due to the file size the time for downloading the application or initializing it may be considerable [5].

User interface, UI is one of the most important parts of mobile development. One of the most time-consuming tasks when Xamarin

is used is the UI development. Although Xamarin has a high ratio in shared code base, there is some time-consuming portion of coding for each independent platform [5].

Xamarin is under a license cost just like Microsoft Visual Studio. They are all commercial tools that is licensed and it comes with a cost [8, page 29].

Google and Apple are gradually releasing new updates and features to their mobile framework and it can then occur some delay in order to transform those releases into Xamarin's framework. It can create issues for the developers working with Xamarin, when the users are experiencing new features with native applications, but not with Xamarin developed mobile applications [8, page 29].

2.4. Xamarin vs React Native

The demand for mobile applications has been growing since their launch in 2008. Due to this high number of users many applications are needed to meet the user's need. In March 2017, there are over 2.8 million Android applications and 2.2 million iOS applications. Therefore, it becomes interesting to find a framework which can meet Time to Market demands and create applications which are maintainable, testable and stabile. Earlier in the creation of applications native language has been used to create the applications, but now there are frameworks for creating applications which can be run on all platforms. Examples of this are the competing Xamarin and React Native [9].

In section 2.4 Xamarin and React Native are compared. These two frameworks are the most mentioned when talking about cross-functional frameworks and to compare and evaluate them is the main purpose with this thesis [10].

2.4.1. Market

The market share for React Native is increasing as the figure 5 shows. There is a lot of different companies that uses React Native in their software development and companies like Guardian, Tesla and Facebook are some of the companies which use React and React Native in their software development [9].

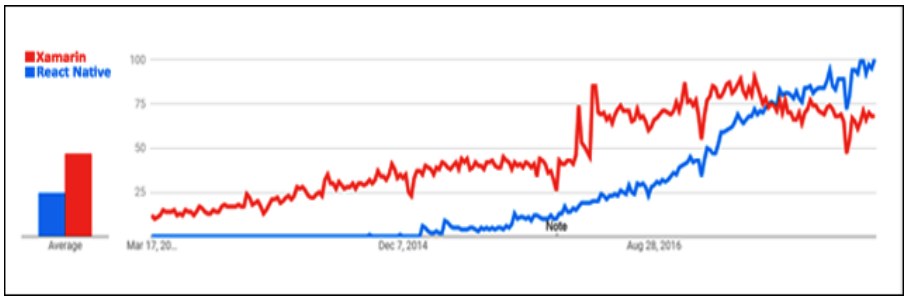

Figure 8 - Market share for Xamarin vs React Native

Xamarin is 6 years old and has therefore a longer history than React Native. Over 15000 companies like CA Mobile and Story use Xamarin. According to Google (see figure 8) Xamarin's market share is decreasing. Why this is happening will be discussed in the conclusion of this report [9].

2.4.2. Availability

As mentioned earlier there is a cost for Xamarin on enterprise level, but there is a limited free version in which some features can be accessed in order to test the framework. React Native is a free tool which is available to anybody that wants to create applications with it [9].

2.4.3. Compilation

JIT, Just in Time compilation is not possible when developing mobile application with iOS with React Native. This is because React Native interprets the JavaScript code and uses the JavaScript Core library in the iOS and Android in order to process its content [9].

Xamarin uses ASP.NETs C# which enables both JIT and AOT, Ahead of Time compilation, but only AOT compilation is used [9].

2.4.4. Development Environment

With React Native the developer can choose which Integrated Development Environment, IDE they want to work with. Examples of IDE are Visual Code, Visual Studio and Atom [9].

When working with React Native on an Apple computer, the developer is restricted to only compile the iPhone simulator on the Apple device. Xamarin has some beneficial functions regarding this. Xamarin enables the development project to be created on a Windows machine, but complied on an Apple computer in an iPhone simulator [9].

2.4.5. Framework

React Native is essentially developed from React, therefore React Native uses the one-way data flow which is included into React. This is similar to the JavaScript web development and therefore is more suitable to what the JavaScript developer is used to [9].

Xamarin follows the Model View ViewModel pattern called MVVM style. Developers which have experience with ASP.NET web development are familiar with this pattern and can therefore easily understand the syntax and structure [9].

2.4.6. Reusable code

React Native has complete components which are ready to use and include into the application and this comes with great documentation of what the components are and how they perform [9].

Xamarin follows ASP.NETs NuGet store where there are components stored. This is a market store for all the external libraries and extensions for Visual Studio users. There are some improvements on the documentation, because the framework is older and more stakeholders have contributed [9].

The problem that occurred while developing the mobile application in both frameworks are presented in under section 3. In the conclusion, section 5 there is also the final discussions about why one of these frameworks was most suitable for this thesis

3. Methodology and Analysis

This chapter is about the methodology and analysis used during the thesis work. It also describes which problems/challenges this thesis has encountered and how they have been solved.

3.1. Working methods

The work was divided into several parts as presented in figure 9. The first part consisted of collecting information about React Native and Xamarin. This was done to be able to decide in which framework to start developing ACT Mobile. Because the author of this thesis has less experience with React Native it would probably take longer time to develop ACT Mobile with React Native than with Xamarin. Therefore, the development started with React Native. Further on the development of ACT Mobile was done in parallel. When the development of ACT Mobile in React Native was completed, the development process continued with just Xamarin. The development was done in parallel

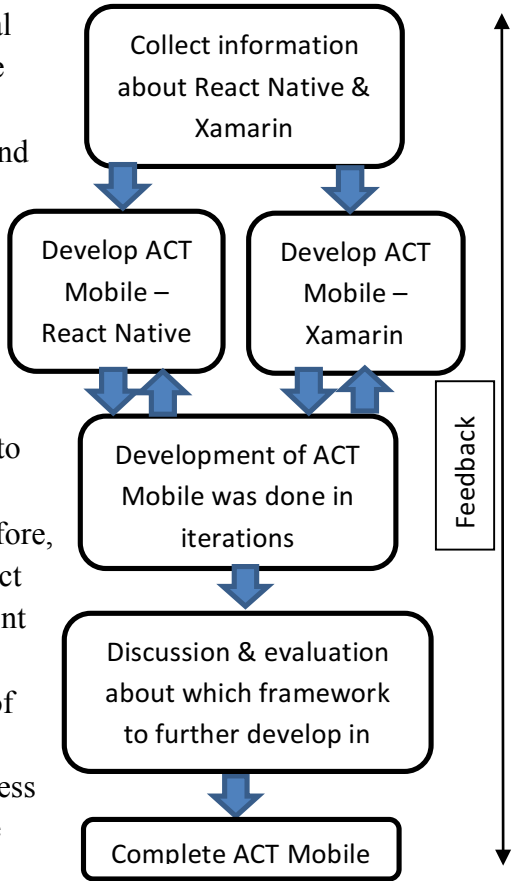

Figure 9 – Development process

because the evaluation and analysis of performance and usability could be done at each step.

Thereafter a long discussion was held with the architect and product owners to evaluate which mobile framework is most suitable to further develop in. During the discussion, it was decided to base the evaluation on the following metrics: maintainability, scalability and the ability to integrate with Tetra Pak environments data transport protocols. There were also discussions if the knowledge about the frameworks could be found in at the organization and if not, which are the costs for learning or buying the knowledge in terms of hiring.

Throughout the whole development process there were appointed meetings to collect feedback from the product owner and architect, but there were also spontaneous meetings to discuss the progress and collect new feedback.

In parallel with the processes of developing the application there was a need of structure to create an effective workflow for the thesis work. The workflow process was therefore built on an Agile work process called Scrum, see figure 10, which will be described in section 3.1.1 Agile & Scrum. Briefly explained it's an iterative and incremental agile software development framework for product development.

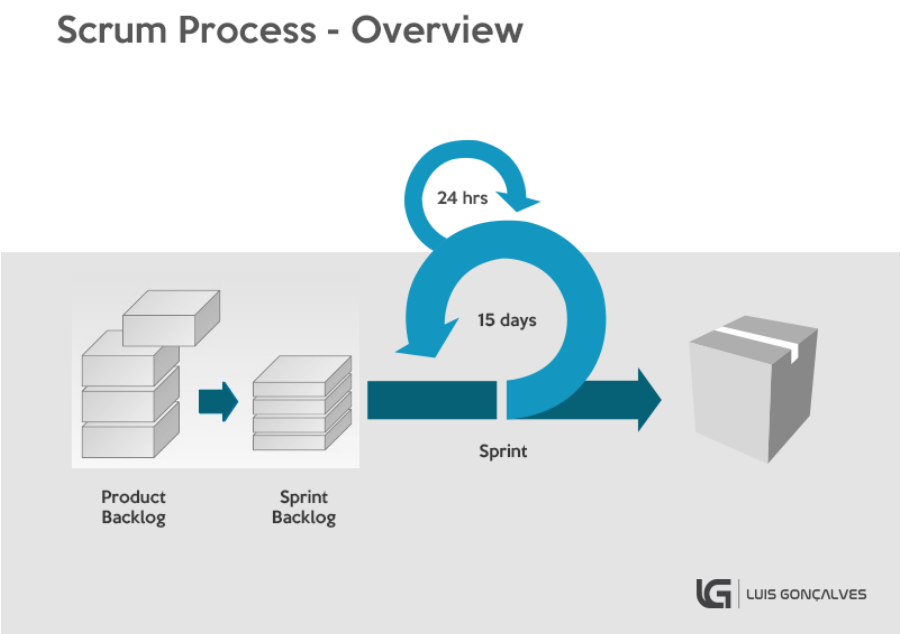

Figure 10 - Scrum work process

3.1.1. Agile & Scrum

Scrum is an iterative process which was followed in this thesis and development process. The iterations are divided into two weeks of work, where one week is called one iteration and both weeks is called one sprint. After completion of one iteration there was a discussion about the progress so far, if any problem has occurred and what should be implemented in the next iteration. In this thesis, there were 8 iterations completed. In order to keep track of all the tasks and assignments there was a backlog implemented in Team Foundation Server, see figure 11. More information about Team Foundation Server in section 3.1.2.

The backlog that contained all the assignments and tasks was separated into two sections, a product backlog and a sprint backlog. A product backlog is where all the tasks are created and stored. In the sprint backlog, there is only tasks which are supposed to be completed within the iteration. This is to separate what is most prioritized. All the tasks that should be implemented for the whole thesis were divided into features and one feature can have multiple stories, see figure 11. This work is done before the implementation in order to prioritize what should be developed first and which should be included into the current sprint. When the correct stories are included in the sprint the development of the mobile applications begins.

The stories that are included into the sprint can also have something called tasks which describe what should be done in detail. This is done in order to create more traceability and control of what is developed and if something happens, what went wrong. This is also an advantage when working in teams with multiple people, because then all the developers can have knowledge of what the others are working with.

Scrum is a branch of agile development, which is a set of standards for developing software. The requirements, or the prioritized tasks evolve within self-organization teams and are often managed by a Scrum Master [10].

Scrum is also a framework for some of the largest companies around the world where Tetra Pak is included. Scrum is usually used for larger project developments, where developers take advantage of the iterative process. This is to enhance the work process and create efficiency, which has been proven when comparing against other product development methods. One of the largest advantages is that new demands from the customer can easily be prioritized in the original priority list of tasks. If this occurs there is a possibility to change the backlog priority in order to meet the customer demands [10].

3.1.2. Team Foundation Server

There are different applications available on the market to handle the features, stories and sprint overview and they come with a variety of license costs. Some of them are also free that allow in-purchase functionality. Tetra Pak has created a standard for handling code, tests, requirements and project management. This is in order to give an overview of products and allow traceability in customer products and to control the test environment. This is also a tool available for management to analyze the product installations, how many bugs and incidents that have been reported. Team Foundation Server can also be integrated with the code writing editor from Microsoft, Visual Studio. Tetra Pak is therefore using Visual Studio together with TFS for optimal integration and performance when working with the code and requirements. Figure 11 presents a general work day of TFS with all the features and included stories. To the left in the figure the current sprint is displayed.

Within the TFS there is also existing a priority of the stories in terms of what should be implemented or done first. The stories are usually called requirements in a water fall working process.

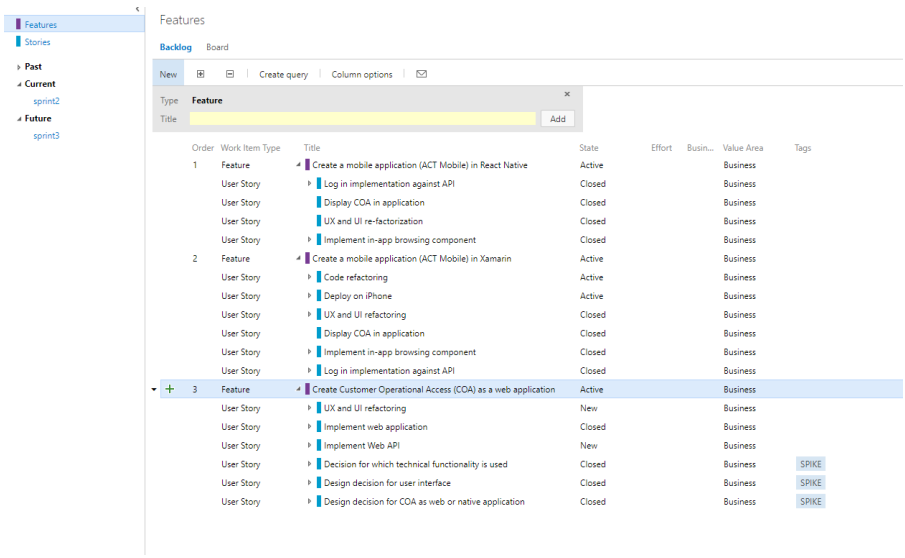

Figure 11 - TFS work process

3.2. Solution of the problems

Problem solution has been an extensive part of this thesis. The first problem emerged when the mobile application was created with React Native due to the knowledge level in React Native. It has similarities with JavaScript, but still has a very different structure in rendering information and establishing logic functions. The knowledge of this framework was low in Tetra Pak, therefore a lot of information gathering for answers resulted in an extensive Internet search and testing different code examples. The problems that are presented below are more practical problems that emerged when the knowledge about React Native was gained. All of the problems below were solved and the complexity of each problem is presented with the most challenging problem first.

The problems that occurred in developing the mobile application, ACT Mobile in React Native (JSX/JavaScript):

- Create a mobile application which can be scalable

- Create an architecture that is maintainable and enable scalability
- Limitations in integration with the ACT platform

The development of ACT Mobile created some challenges, where the authors experience in C# was not an issue, but there arised some problems regarding the architectural structure. This was because the earlier experience was in web development and there are some differences when developing a mobile application in C#. All the problems below were solved and the complexity of each problem is presented with the most challenging problem first.

The problems that occurred in developing the mobile application, ACT Mobile in Xamarin (C#):

- Create a mobile application which can be scalable
- Limitations in integration with the ACT platform
- Limitations in integration with the ACT platforms messaging protocol
- Deployment of ACT Mobile to iPhone

3.2.1. Problem solution - React Native

The development of React Native created some challenges due to the author's lack of knowledge about JavaScript and React. Therefore, there was some struggling in the beginning, but with persistency, information search and testing the lack of knowledge quickly transformed into an effective development process, because of research and testing code examples. It was starting to get more fun and React Native's advantages were clearer. One of the first problems that did occur during the development was to handle requests between components within the application. A component renders a part of the GUI interface and the communication between the components can

quickly become very complex if the structure is inefficient. Discussions regarding how to scale the applications architecture raised some discussion if the application was to become larger.

As presented in figure 5 in section 2.2.3, there is a figure that explains the differences between the state through all components. The figure starts with the parent on top and explains how there exists a possibility to have a store which holds all the states. The problem with implementing Redux in an already developed system can be time consuming however this could lead to an efficient system. There was some lack of competence with React at Tetra Pak and it would lead a cost, because there is a need of investing in learning React. Additionally, the cost would become greater if Redux also was necessary for further development. There was a decision not to invest in Redux, because it would only create a more complex application which was not necessary at this stage of development. This decision was taken together with the product owner and architect of the ACT. The benefit of using Redux when implementing ACT Mobile revealed itself when the components had to communicate with each other in a more complex situation.

This also leads into the second issue where the architecture had to become maintainable and flexible in order to add new functionality into the mobile application. ACT Mobile was created as a native platform which should offer the different applications that ACT platform can provide. These applications in the ACT platform would then be created as web applications and could be presented both in the ACT Mobile, but also in the ACT platform. ACT Mobile will therefore grow and would become more complex which means that it's important to invest in architecture. The ACT Mobile was established as a platform which held the native functionality of login system, user and navigation functionality.

There occurred some limitations regarding the integration against ACT platform. To begin with, ACT platforms messaging protocol

against log-in system was built with Simple Object Access Protocol (SOAP). There is a more recent process to handle this, by Rest API which is very commonly used in mobile application. React Native does not support SOAP services. Therefore, the SOAP service had to be transformed into a Web API which then could be used by the mobile application to authenticate and authorize certain users into ACT Mobile.

A testing process for the ACT Mobile in React Native was not implemented neither was any test completed, due to lack of time. There were tasks in the TFS to create a test environment, but the focus was on create a first version of a mobile application.

3.2.2. Problem solution - Xamarin

As the development with ACT Mobile began with React Native it continued with Xamarin. The first version of ACT Mobile in Xamarin was faster due to the author's knowledge level in C#. It was easier to understand how Xamarin wants the developer to build the application and because of the author's earlier experience in C# there opened possibilities to create a robust code structure.

How to create a mobile application with scalability to enable more functionality from the ACT platform was one of the challenges with developing ACT Mobile in React Native. This was also solved by creating an application from the ACT platform as a web application. The web application is then presented in the ACT Mobile and the ACT platform. This solved the scalability problem, because every application that would get included is created as a web application and then it could be presented in both the mobile application and ACT platform. There could emerge some problem regarding the performance of displaying a web application and the solution to that is to create the application from ACT as a native functionality. It would prevent the application to be displayed into both ACT Mobile and ACT platform.

Limitations of the ACT platform were identified as with the React Native. ACT was built with SOAP services and there is support for SOAP services in Xamarin, because it is partly a Microsoft product and Microsoft develops SOAP services. When the configuration of communication with the SOAP service was complete there was an ability to communicate with the ACT platform in order to retrieve information about the user. Another problem that occurred during development, was that the requests from SOAP to the ACT platform were encrypted with Web Services Enhancements (WSE) 3.0 for Microsoft .NET. It is a common practice to encrypt secure the information that is being sent between the applications because ACT holds sensitive information about Tetra Pak employees. Therefore, the WSE enables developers and administrators to apply security policies to web communication running on the .NET Framework 2.0.

Xamarin only supports the .NET standard and not the .NET Framework 2.0 as WSE was implemented on and it was not possible to communicate with the ACT platform through SOAP services. The SOAP services had to be transformed into a Web API. In order to keep the security intact there are built in features which can encrypt information sent between the HTTP-requests in order to secure the Tetra Pak user information.

When the development of the application started, there was a need for testing it in a physical mobile device. Earlier on, the testing was done with the emulator of an Android device in Visual Studio. The testing of the mobile application was limited due to network problem regarding the Web API. It was not located on a global server since sensitive information is sent. Due to the limitation of Android devices, there were only iPhones to test on because Apples devices are an industrial standard throughout the company, both mobile devices and tablets.

Visual Studio is a Microsoft product and therefore, they want to limit the testing on Apples devices. There was a possibility, but it included

an Apple laptop. If an Apple laptop was used the process started with connecting the iPhone to the MacBook through the mobile application development tool called XCode, which is used to develop mobile application for Apples devices. Then a contract is created that enables the application to be tested on iPhone for 6 days and be stored on the iPhone. Further on the project on Visual Studio was opened and remotely connected to the MacBook through the PC computer.

This enabled the mobile application project on the PC computer to remotely connect to the MacBook which simulated ACT Mobile on the iPhone. The application started and could now be used on the iPhone which had the certificate to run it. The problem was that the ACT Mobile was never used on the iPhone, because the Web API that handled the login was not located on a global server. It was never resolved, because a large organizational process was needed in order to locate the server to enable outside web requests. This could create a good opportunity to test the application in a physical device but was never done.

A testing process for the ACT Mobile in Xamarin was not implemented neither was any test completed, due to lack of time. There were tasks in the TFS to create a test environment, but the focus was on creating a first version of a mobile application.

3.3. Source criticism

The source criticism is very important. For the history facts Tetra Pak's home page was used when collecting information. This is safe and reliable information, because the source is Tetra Pak and when talking about the history there is no hidden agenda with trying to sell something. When writing about React Native, Redux and Xamarin it is best to find information from an impartial source. If collecting information from Facebook which developed React Native, then Facebook would write out the information for their market benefit and Microsoft would do the same, which has connections to Xamarin

even though they are not the founders. Therefore the need of sources which have an independent perspective is the most valuable source. The investigation in the sources reliability has been assessed by the questions; Who is behind the source? Is it a company? Is it an organization? Is it a private person? Is it someone you rely on? [12].

After these questions has been asked, there is a next step of questions that can be asked and those are; Is it to inform? Is it to convince me into something? Is it to sell me something? It is important to examine whether the company wants me to buy their product or use their framework and that can be accomplished by describing how complex and secure the product is and reflect the benefits. Discussions and opinions that come from sources where people are professional software developers have a greater reliability. Therefore, the information from those kind of sources is consistent and can be used. If the source has connection with the founders, they will probably present more benefits. There is then the possibility to combine the source with another in order to establish a reliability.

When analyzing a source of information something that is very important is to analyze how the information is presented. If the information includes both drawbacks and benefits and is written with an interest to teach, then the information is more reliable. If the source only wants to sell something or only present the benefits, there could be some suspicions regarding whether it is to teach or only to sell. For example, regarding Xamarin, the source where the benefits and drawbacks where extracted were from a blog. Often blogs have a theme and therefore contain articles based on partial perspectives. The important part of this is to abstract the relevant information and compare with real life experience. If there is any doubt, don’t use the information until it is confirmed with another similar source. Ask the questions; Who is behind the source? Is it someone that can be trusted? Is it a company? [12].

When it comes to the comparison between the frameworks, Xamarin and React Native there are more questions raised about the sources. This is for the sources 5, 6, 7, 9 and 10. Questions regarding where the sources come from are very important, because it could be the companies behind writing for the frameworks advantages. The question; Is it to convince me into something?

When assessing the different sources, the most central part is also to assess the authors. Which kind of articles do they write and how do they write them. Do they have a background of being partial in their writing? Then that information should be compared to what is known to happen in software development. The advantage within this thesis is that the assessment of the information is already confirmed from previous programming knowledge and therefore increases the reliability with the source. That is information which can be extracted [12].

In this report, there is a section about product development, Agile & Scrum. Here the source is CPRIME, which could be strengthened by adding a book about Scrum. CPRIME is a company that delivers a service to progress structure in product development. They clarify what scrum and agile is in terms of how it is managed out in work field. The purpose of the page creation is though unclear. Their goal could also be to slowly affect the reader and convince the reader to purchase their service. One perspective could also be to try their service, because of their great quality and customer service. The source was chosen because it gave a realistic information about product development with agile [12].

4. Results

This chapter presents the results of a comparison between a mobile application prototype ACT Mobile created in React Native and in Xamarin.

4.1. Results

The result of this thesis includes the comparison of React Native and Xamarin and the result of the prototypes that was created by both mobile framework. Both protoypes include a log-in system for authorized Tetra Pak users. If log-in is successful there is a navigation page for the application which is available for each user. The navigation page for ACT Mobile is different on each mobile framework. The navigation design on Xamarin can be seen in figure 15. ACT Mobile navigation design for React Native can be seen in figure 13 and is presented with a start page and then the page is changed to the current application. These two different navigation designs were tested in order to see how the user experiences different design approaches.

More in detail about each application is described in the following sections. Section 4.2. ACT Mobile presents which mobile development framework ACT Mobile was further developed in. Due to copyrights and ownership of the mobile application any code is not visible and presented in this thesis.

4.1.1. ACT Mobile - React Native

The first prototype of a mobile application that was developed was with React Native. One of the first things that was implemented is the log-in system, see figure 12. Here the Tetra Pak user creates an authorization with their Tetra Pak email and password. The

authentication is based on a rest call to the Web API created for this thesis. It then validates the user and sends back a response. The response is interpreted by the application in order to process the user to the navigation page or to try log in again.

When trying to log in to the system the application also handles all requests with activity indicators in order to create visual feedback to the user. This feedback can for example be a spinning wheel. This is very important when interacting with a mobile application, because there is no other visual feedback that is sent back to the user and therefore it can be unclear about how different activities progress.

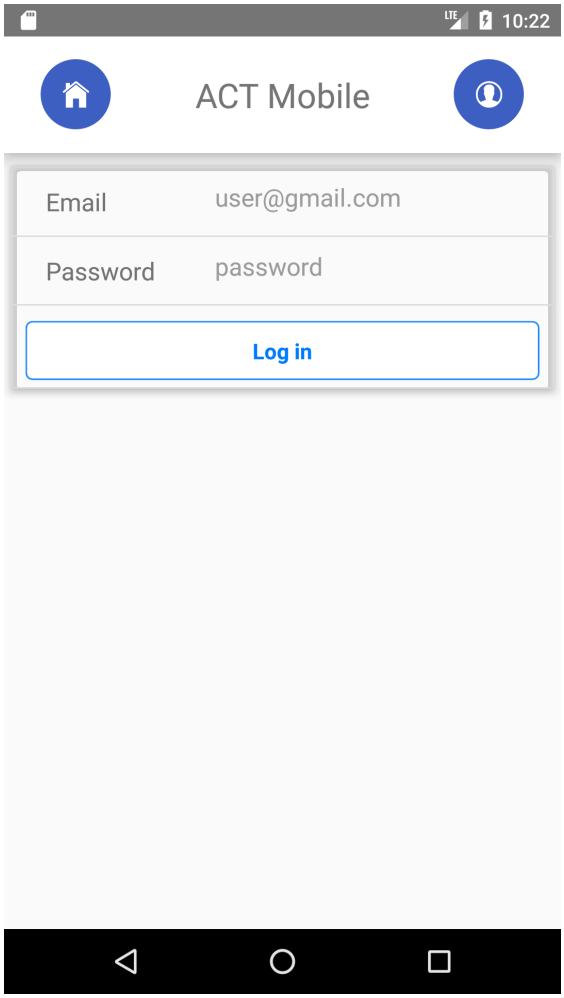

Figure 12 - ACT Mobile - React Native Log-in

If the user is not successfully logged in, the input fields for the email and password becomes empty and the user can try to log in again. The user is allowed unlimited attempts to log in. If the user is successfully logged in, then the user is redirected to the navigation page, see figure 13.

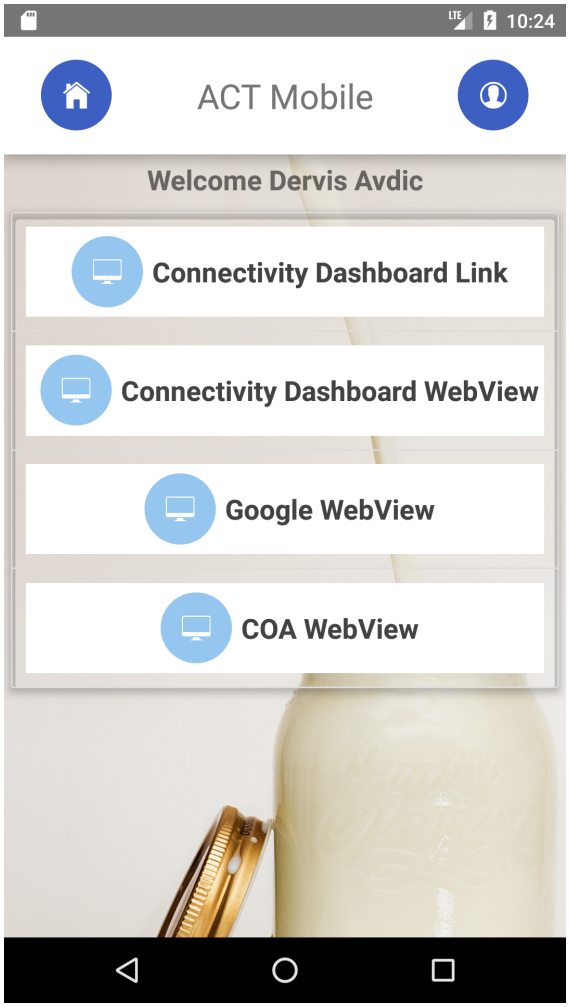

Figure 13 - ACT Mobile - React Native Navigation

Figure 13 is presenting the available applications for the logged in user. The navigation consists of 4 applications. The first application is a Microsoft Power Business Intelligence application which collects information about site alarms and present their data. The second application presents the same content with an exception that is in a Web View. This means that the application is opened within a frame inside the application and the first application opens a browser. The third application presents Google within the application in a frame

and not in another browser window, see figure 14. The fourth and last application is the Customer Operation Access application. This presents information about customer sites around the world with purpose to display how the site is running and if any alarm has been detected. It is also presented within a web-view.

The navigation page also presents a welcome text where the mobile application communicates with the Web API. Here the state is changed by a prop on the welcome text. The complete navigation page is rendered by using states. Dependent on which application is chosen the state is changed and therefore the whole page changes.

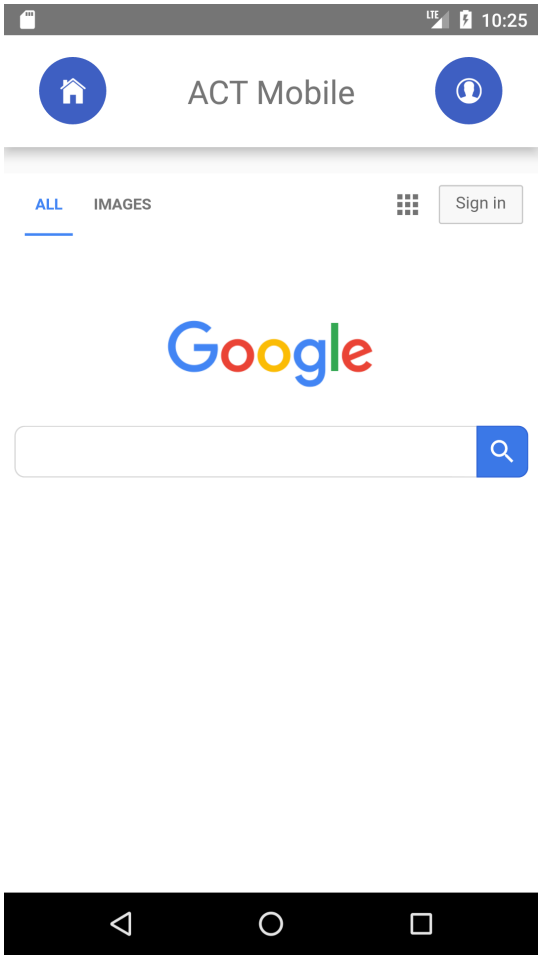

Figure 14 - ACT Mobile - Google Web View application

In figure 14 the application of Google Web View is opened and presented within its frame. This is an example of how the web applications from ACT is presented in the application. Some of the functionality from ACT is transformed into a web application and can be shown as websites inside the application. Other information and functionality is created natively because of performance and interaction and that is an example of the log-in system and navigation functionality.

4.1.2. ACT Mobile – Xamarin

The second application which was developed during the thesis was the ACT Mobile with Xamarin. One of the first things that was implemented was the log-in system, see figure 15. The log in system is developed as in React Natives log-in system in corresponding JavaScript. The only differences are the design with the background color and a logotype of Tetra Pak in Xamarin's ACT Mobile. A Tetra Pak user creates an authorization with their Tetra Pak email and password. The authentication is based on a rest call to the Web API created for this thesis, which confirms if the user is valid or not and sends back a response. The response is interpreted by the application in order to process the user to the navigation page or to try log in again.

When trying to log in to the system the application also handles all requests with activity indicators in order to create feedback to the user. Therefore, when trying to log in the system will complain if the fields are empty, no authority or wrong credentials.

The Android project emulator is a fictive mobile device inside the computer and therefore lacks Internet connection and GPS location. Therefore, the bar at the top indicates that no Internet is available. When starting the iOS project, a simulator is started which is a simulation of the iOS application. The simulation has the same access rights and functionality as the computer it is simulated on. Another example of confirming that the emulator is a fictive mobile device inside the computer is that when writing localhost in order to retrieve the local IP-address on the emulator it cannot interpreted. This is because the emulator is a real device and a mobile device do not contain a localhost attribute. The simulator understands it, because it is a simulation on a computer and therefore has access to the computers local IP-address.

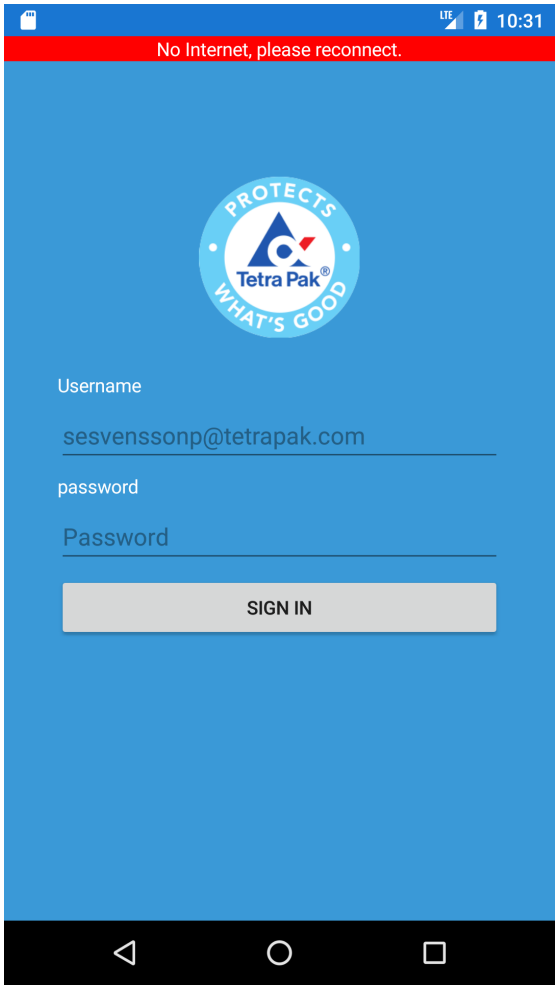

Figure 15 - ACT Mobile - Xamarin

If the user is not successfully logged in, the input fields for the email and password becomes empty and the user can try to log in again. If the user is successfully logged in the user is redirected to the navigation page, see figure 16.

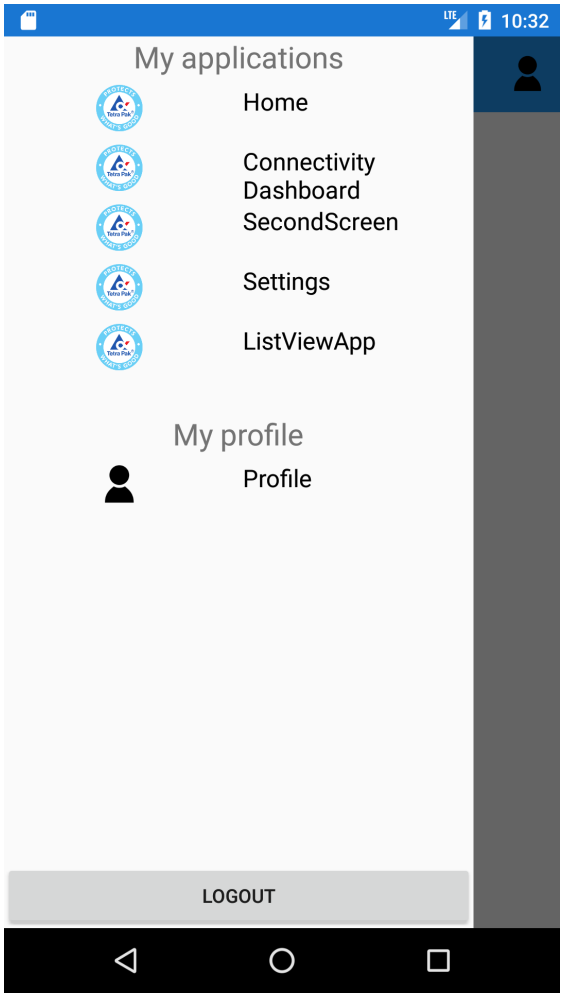

Figure 16 - ACT Mobile - Xamarin Navigation

Figure 16 presents a different navigation design than figure 13 which is the navigation design selected for the React Native application. The different design approach for Xamarin was made in order to collecting more feedback about which navigation approach is most user friendly. More information about the differences is presented in 4.1.3 Differences between Xamarin and React Native. Within the

navigation design there is also a separation between the applications and the profile functionality as can be seen in figure 16. This is in order to present a clear view of the application regarding the ACT Mobile and the functionality regarding the user.

The applications which is included in ACT Mobile are built with Web View functionality which is the same as with React Native. An application is presented as a frame inside the ACT Mobile, see figure 17. This is the long-term goal for every application, because the ACT Mobile doesn't have to generate a new browser in order to display the information. It enables ACT Mobile to easier transport data between the web view and the application itself.

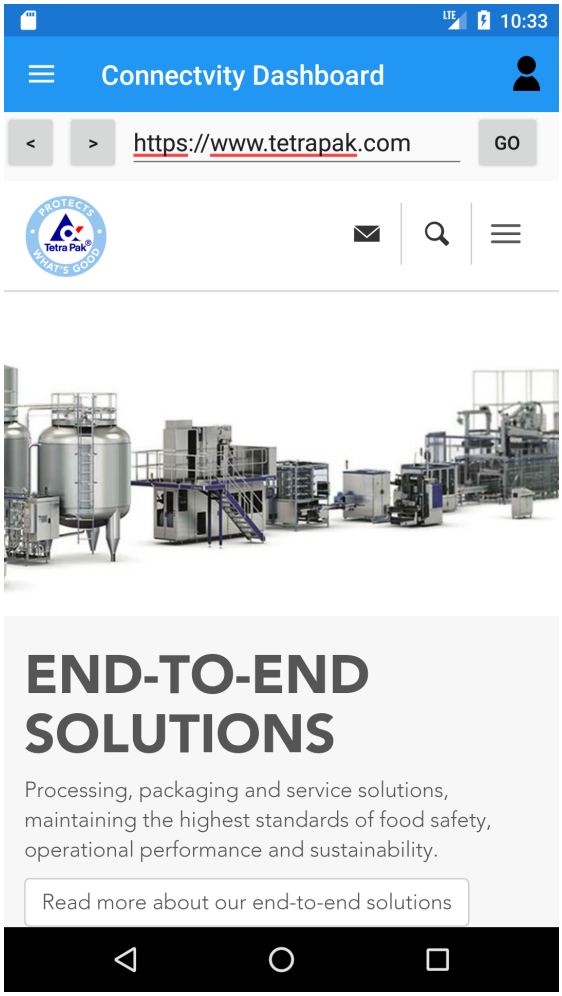

Figure 17 - ACT Mobile - Xamarin WebView

Figure 17 displays the content of a Web View inside the Xamarin application. As described with React Natives web view, this is an example of how the web applications from ACT is presented in the application. Some of the functionality from ACT is transformed into a web application and can be shown as websites inside the application.

4.1.3. Differences between Xamarin and React Native

This section explains and summarizes the differences of Xamarin and React Native found during the parallel development phase. The log in system is the first encounter with both versions of ACT Mobile. Here the functionality behind the log in system is equal with the exception that it developed in two different environments and therefore the syntax of the programs become different. Further on the design has some minor differences where React Native has white background and Xamarin dark blue.

When successfully logged in to both applications the navigation design on Xamarin can be seen at figure 15. This is possible to perform on all of the different pages inside ACT Mobile and the available functionality is the applications and user settings which are separated in order to facilitate the experience of the user. ACT Mobile navigation design for React Native can be seen at figure 13 and is presented with a start page and then changes page to the current application by replacing the start page. The user settings are available at the top header where there is a home and user button for the corresponding functions.

Other differences in the web view component which both applications have implemented is that in the Xamarin application there is a navigation header at the top, see figure 17. Here the user can enter a different web link and go forward and backwards. This is not implemented in the React Native application, because it was not necessary due to the decision to use Xamarin in the future.

4.2. Final ACT Mobile

After developing the ACT Mobile in both Xamarin and React Native there was a final decision on which framework to use in the future. This was Xamarin, due to maintainability, scalability and further

development competence. Section 5 Conclusion explains maintainability, scalability and further development competence in more detail. The application had some refactoring to the design as can be seen if figure 15 and 18 are compared. In the log-in page the background color was changed in order to follow Tetra Pak's standard for colors. The placeholder for the username is changed and this is because the Tetra Pak user don't longer log in with the domain name (@tetrapak.com). This was made because the only authorized users are able to log in to ACT Mobile are Tetra Pak employees. Therefore, the system adds the domain name for the user, so that the user doesn't have to waste time. When typing the password, the user can also press the icon with an eye. This is a peek functionality where the user can reveal the password behind the password characters. It is a convenient function because writing on a mobile device can be more difficult than when typing on a keyboard.

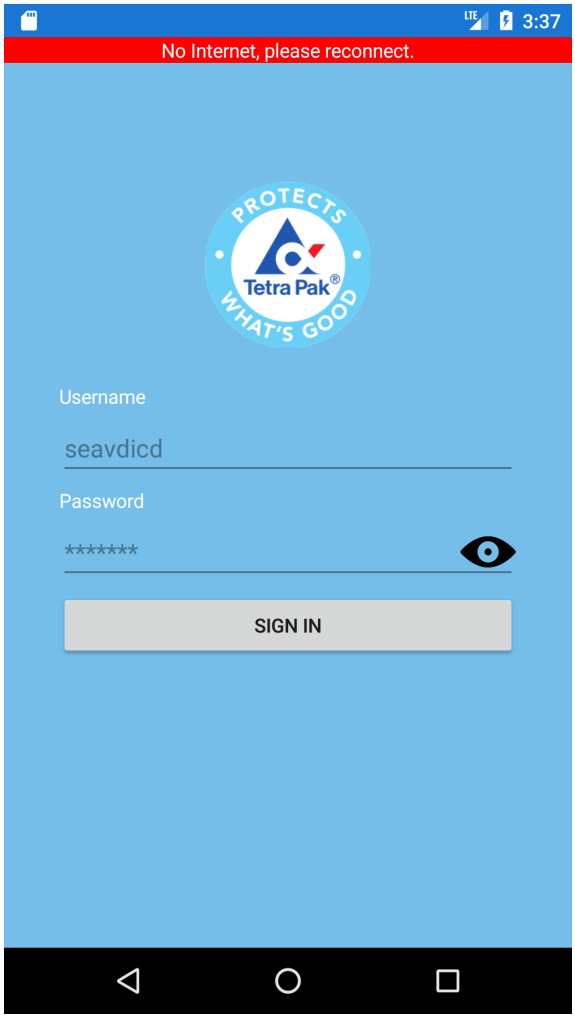

Figure 18 - ACT Mobile - Log in

There is also a home page implemented and that is the start page for the ACT platform, as can be seen in figure 19. Here is also a welcome text which call the rest API in order to retrieve information about the logged in user.

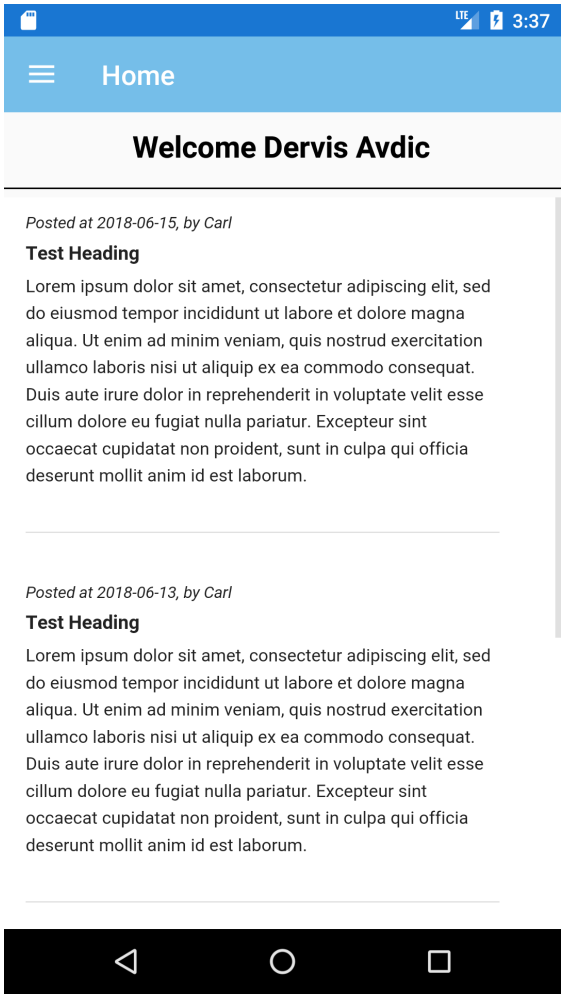

Figure 19 - ACT Mobile - Home

The navigation page had some design modifications as seen in figure 20. The icons are redesigned with more suitable images that represent what the application does. It is also done to create a connection between an image and an application. It facilitates the for the user, which is confirmed by feedback from the product owner and architect. Customer Operation Access and Connectivity are two

applications that present a web application within an iframe and the Customer Site locator is a beta application that was developed as a native application.

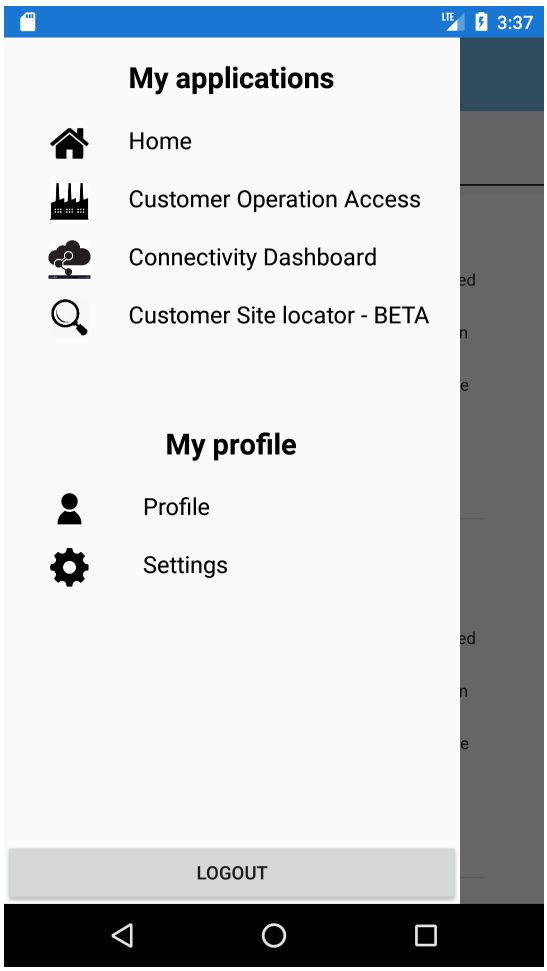

Figure 20 - ACT Mobile - Navigation

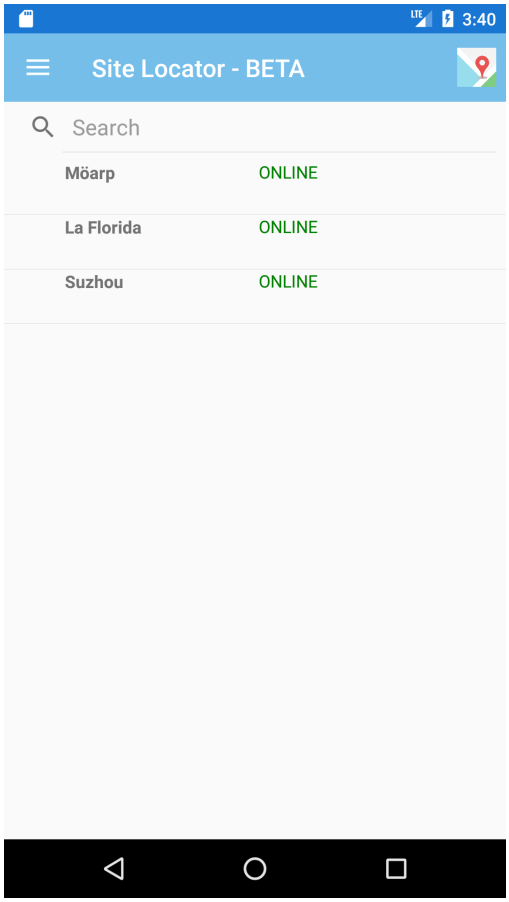

Figure 21 - ACT Mobile - Site locator BETA

Site locator is a native application developed as a proof of concept to another application called Customer Operation Access which also can be seen at figure 20. It displays all the available customers and the user can then navigate to their RSU dashboard in order to overview the status of the specific details of a factory.

In this function there is all the available and online factories and the user can click on them in order to go to the RSU dashboard. The RSU

dashboard is not visible in order to maintain customer secrecy, see figure 21.

Another design approach to navigate between the sites is to implement Google Maps as seen in figure 22. Here all the available factories are presented as needles on a map and the user can see the name and address of a factory. If it is clicked, then the user is redirected to the RSU dashboard. This design approach is an advantage to the user, because it doesn't have to search for the site in some tree structure or in a list. If the user has some clue of where the site is located, it can easily trace the site on a map. It helps the user see the sites from a whole new design perspective.

Figure 22 - ACT Mobile - Google Maps

The profile function is implemented, see figure 23 and if the user clicks on the profile is reveals all the information about the logged in user and there is the possibility to change password, see figure 24.

Figure 23 - ACT Mobile – Profile

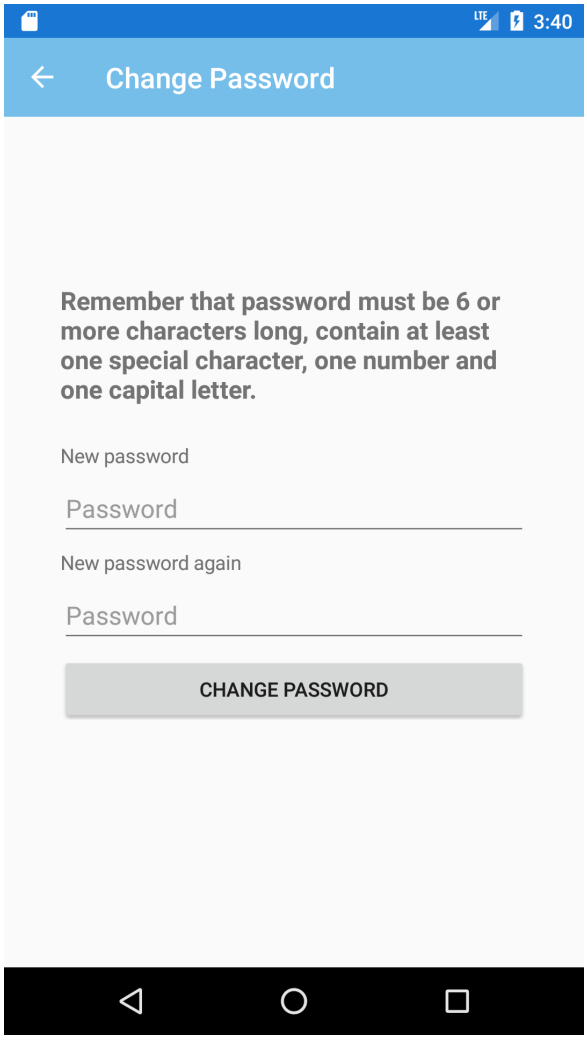

Figure 24 - ACT Mobile - Profile - Change password

There is also a page for user settings. This is supposed to collect all the common settings for the logged in user, see figure 25.

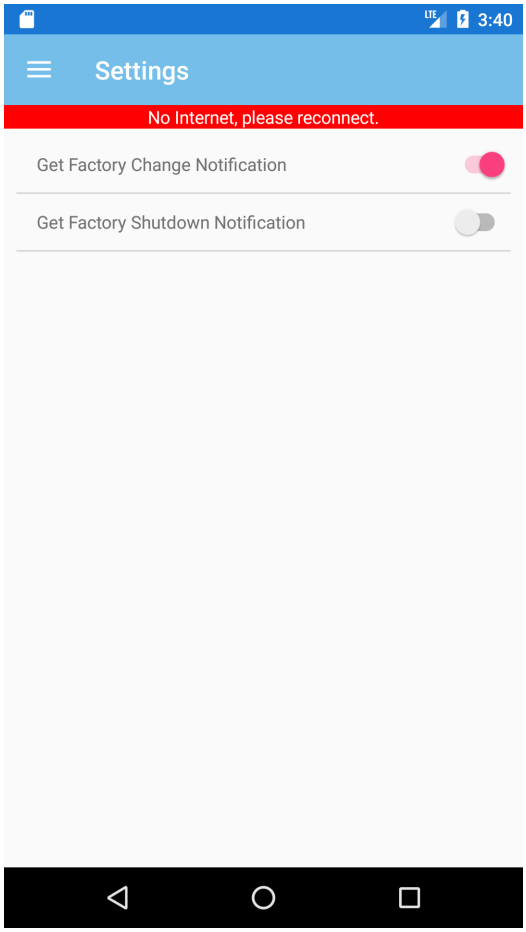

Figure 25 - ACT Mobile – Settings

Further on the ACT Mobile has not been tested with automatic tests, unit tests or integration tests. The user interaction and design features have been tested where discussions has been held with the product owner and architect. Unit and integration tests are something that is prioritized in future work and is something that is necessary if the mobile application reaches the market.

5. Conclusion

This chapter contains the conclusions of the thesis work and the answers to the problems addressed in this thesis. It also contains how the result will be used by Tetra Pak.

5.1. Result

This section answers the problems specified in section 1.4.

- How to collect feedback from the target user?

The feedback was received by the thesis author from the product owner and the architect on an appointed meeting. Here the discussions regarded the architect, user interface, UX and how to create a design that is in align with Tetra Pak's colors.

Further on the collecting of feedback was completed by displaying demo versions of the applications every other two week. This was done to synchronize the design, user interface and UX effort with the people that is supposed to use and maintain the application in the future. If ACT Mobiles new functions were not discussed with the product owner and architect there would be a possibility to create redundant functionality which is time and energy consuming.

When ACT Mobile was demonstrated feedback was received regarding new requirements or modifications. This feedback was received by the product owner and the architect and discussed in order to analyze if new functionality is necessary and suitable to the applications architecture. Feedback that was received which was not given by the architect and product owner was always discussed with the product owner and architect in order to align with the future of ACT Mobile in terms of functionality and user interface.

If feedback was given by the developer, the feedback would be transformed into tasks in the backlog. This is a benefit because every feature that is stored in the backlog is also confirmed by product owner and architect. The product owner confirms that the features is needed by the customer and the architect is confirming that the function is needed for further development and that is possible with the current architectural setup.

- Which comparison metrics are suitable when selecting between React Native and Xamarin?

There is a variety of metrics that could be used to evaluate which mobile framework is most suitable for future development. Some of them are number of code rows written. reusability of code and scalability. All these metrics are important for comparing React Native and Xamarin. Due to secrecy, there is not an exact number of differences in code rows, but there was more code written inside the Xamarin project then in the React Native project. This shows that the React Native framework has better code reusability then Xamarin. Although Xamarin Forms is developed in order to increase its reusability React Native still wins and this is because of the component structure. Scalability is a metric that is very important, but was not the most crucial because the long-term perspective for the ACT Mobile was unclear at Tetra Pak. Therefore, the scalability was not prioritized.

The metrics that were prioritized and most important was the further development and maintainability of ACT Mobile. The knowledge behind the architecture of ACT Mobile is very important. This is due to maintainability and further development. If there is lack of knowledge within the architecture, maintainability and further development can become very time consuming and costly in terms of money. This can also be seen for larger software productions for mobile development. When creating something, make sure that the product can be maintained and further developed if handed to some

other organization. Reusability is something that was discussed for code but can also be applied to complete products. If the whole organization knows what is developed within the company, there is a possibility to share knowledge and experience.

The developers that are going to create further applications and add new functionality must have the competence to proceed this. Here at Tetra Pak the most used coding language is ASP.NETs C# and they are a Microsoft oriented company. This means that they are heavily invested in Microsoft environment. They also have different mobile development departments around the world working in Xamarin. Therefore, the development was being proceeded in Xamarin.

- How to investigate which application in the ACT platform is going to be implemented as a web application?

This problem was solved by discussion with the product owner and the architect and all developers involved in developing applications in ACT. People that also has connection with business units and towards the customer was also involved to receive information about which demands the customers have on mobility solutions. Therefore, the decision was taken by the thesis author, the architect and the product owner about which application is most suitable for a mobile device. What suitability means in this context is which application can be presented in a mobile device and have most usability for the customer.

Initially there was a discussion about creating a priority list for every application in ACT that would be developed as a web application. Due to lack of time no priority list was created.

To develop the log in system and the dashboard was time-consuming. That was considered more important and this is something that was discussed along the way together with the product owner and architect.

Therefore, Customer Operation Access was chosen as the application that should be converted to a web application. The reason behind this was that the application had firstly an outdated user interface and the tree structure was very complex. Customer Operation Access application was created as a web application in JavaScript, HTML and CSS. The application is responsive, so it is presented in a tablet and mobile device equally well as within desktop environment. The COA application was also created as a native functionality inside ACT Mobile and this was to demonstrate the power of native functions. The tree structure that was complex could be more efficient in terms of performance if ACT Mobile could avoid rendering HTML. It was shown that it was efficient with native functionality, but the benefit of creating a responsive web application is that it can also be shown in the ACT platform. The decision was to keep both the web application and the native function in order to evaluate both approaches.

- Which possibilities and limitations exist to create an integration between the mobile application and ACT platform?

ACT platform uses SOAP services to send and receive data and there is not support for that in React Native and with Xamarin there existed an integration. The problem which occurred further on was that the SOAP Service is secured and encrypted with a web service called WSE 3.0, more information is in 3.3.2 Problem Solution – Xamarin. WSE 3.0 did only support .NET framework and it differentiated from Xamarin's framework, which was .NET standard which is a different framework only for mobile solutions. The conclusion was therefore that there is a limitation between integrating ACT Mobile and ACT platform due to the differences in their framework.

Possibilities that occurred for the integration aspect was that some web application could be possible to present within the application as

a website, but this would work on a traditional mobile web browser such as Safari, Google Chrome and Mozilla Firefox.

- How should the mobile application behave when users are in the factory and when out of range?

This was implemented back-end as a native function inside the application. There is mocked data which generated geographical coordinates, longitude and latitude for the different sites. The mobile device could then through its own GPS support recognized where the device is located and calculate the vicinity to the site. The reason behind the mocked data is because there is no Web API supporting the data communication of the sites geographical positions. It is only stored in a database and must be transformed into a Web API and that comes with a security risk of leaking vulnerable information to the Internet.

5.2. Conclusion

This solution is presented based on evaluation of the two frameworks Xamarin and React Native. It presents material about the differences between React Native and Xamarin, but also how to evaluate which framework is most suitable. These metrics are maintainability, scalability and accountability. One of the most important discussions are also in which environment the mobile application is developed. The thesis discusses the organization or company that is developing the mobile application and formulates questions that should be asked and answered before any development is started.

There must be a long-term plan for scaling the application and how to add new and existing functionality. There must also be an architectural basis for the application which enables the possibility to handle a lot of data and functionality. Furthermore, there should exists the possibility to test the mobile application before released to customers.

The maintainability is an important aspect and there is a great need for a plan for how to maintain the application. This is because if there is not a strategy for this, the mobile application can grow without the application is maintained. This can lead the mobile application to crash in the worst-case scenario. Therefore, a plan must be implemented in order to avoid this risk. To create a sustainable strategy for the maintainability the competence must be allocated, within the company or from external resources such as consultants. What is the cost of hiring consultants to maintain the application or can the company solve the problem themselves within in-house competence if it exists. The in-house competence should be located and allocated before taking this decision to avoid redundancy.

The accountability, the responsible within the organization for the mobile application is something that is also very important, but sometimes is not prioritized. The responsibility could for example be very confusing and have multiple ownership within the company. It can therefore be complicated when different stakeholders within the company begin to make demands for their own benefit. Then the mobile application creates a different purpose than what it was created for. It is therefore important to ask the question; Who is responsible? If the responsibility question is solved it is easier to integrate other demands from different stakeholders and create a mobile application that could be used by multiple stakeholders and employees within the company. An additional benefit that is generated from a solid responsibility-strategy is the efficiency of locating other competence within mobile application development. This is because there is a structured overview of which kind of application each stakeholder develops and designs. Each stakeholder can avoid that code and design are duplicated in order to save time and money.

5.3. Use of the prototype

This prototype will be used and commissioned by Tetra Pak along their new ACT platform MES in 2019. They are then able to offer customers a mobile version of the ACT platform with functionality that is suitable and user-friendly.

This is also a benefit for their branch within the organization, because they can market their product and describe their capacity and responsibility within their branch. It is also presenting a step into the digitalisation which is one of the company's digital strategy set for the coming years.

6. Future work

This chapter of the thesis will be about future work in Tetra Pak with the ACT Mobile.

6.1. Future development

The ACT Mobile prototype works in Tetra Pak LAN. This is because the Web API is only installed on Tetra Pak due to the risk of uncovering valuable information. In the future, there is the possibility to install the Web API on a server which can be access from outside Tetra Pak LAN. Then there would be a possibility to use the ACT Mobile where ever the employees are located, if connect to Wi-Fi and Internet.

Further on ACT Mobile is going to be included with more functionality from the ACT platform. Before implementing anything, there must be a discussion regarding if the functionality is mobile friendly and suitable for a mobile device. A mobile application doesn't always have to be the solution. This is because there exists web applications that achieve the same performance and usability if they are responsive within the browser. The cost and time of creating a mobile application can therefore be avoided.

It is important for other companies that also want to implement a mobile solution that they first analyse the needs. If the decision is to develop a mobile application, it is also important to utilize the native functionality with the mobile device. This can for example be Global Positioning System, Accelerometer, Gyroscope, Magnetometer or Proximity sensor. It is not necessary to utilize these native functions, but then it is more time effective and cost-friendly to create a responsive website to present information.

If a company has some content regarding not only plain information there is also a discussion about how to handle those functionalises inside the application. In ACT Mobile Customer Operation Access is presented as a web application because it is more convenient to show the application on a desktop device. If the application is more suitable and user-friendly with some natively functionality such as the GPS and Google Maps, then maybe the application should be transformed into a native application also in order to exploit the functions maximum.

Consequently, the analysis of why the mobile application must be developed is very important and what it should be used for.

References

[1] Tetra Pak AB. *Tetra Pak History.* [ONLINE] Available at: http://www.tetrapak.com/se/about/history [2018-10-05]

[2] B. Eisenman. 2017. Learning React Native – *Building Native Mobile Apps with JavaScript*. Second edition. Published by: O'Reilly Media, Gravenstein High North, Sebastopol, United States of America.

[3] D, Bugl. 2018. Learning Redux – *Build consistent web apps with Redux by easily centralizing the state of your application.* First edition. Published by: Packt Publishing Ltd, Birmingham, UK.

[4] D. Hermes. 2015. Xamarin Mobile Application Development – *Cross-Platform C# and Xamarin.Forms Fundamentals*. First edition. Published by: Apress.

[5] Red Bytes. *Advantages and Disadvantages of Xamarin in Mobile App Development.* [ONLINE] Available at: https://www.redbytes.in/advantages-and-disadvantages-xamarin-mobile-app-development/ [2018-11-27]

[6] Quora. *What are the advantages and disadvantages of developing apps on Xamarin cross-platform?* [ONLINE] Available at: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-developing-apps-on-Xamarin-cross-platform [2018-10-05]

[7] Gerasimov, V.V., Bilovol, S.S. and Ivanova, K.V., COMPARATIVE ANALYSIS BETWEEN XAMARIN AND PHONEGAP FOR .NET.

[8] N. Panigrahy. 2015. Xamarin Mobile Application Development for Android – *Develop, test and deliver fully featured Android applications using Xamarin.* Second edition. Published by: Packt Publishing, Livery Place, Birmingham, UK.

[9] Cabot. *Xamarin vs React Native: Which is Better for Cross-Platform App Development?* [ONLINE] Available at: https://www.cabotsolutions.com/xamarin-vs-react-native-which-is-better-for-cross-platform-app-development [2018-10-05]

[10] DZone. *Xamarin vs. React Native – Comparison of Two Cross-Platform Development Tools.* [ONLINE] Available at: https://dzone.com/articles/xamarin-vs-react-native-comparison-between-two-cro [2018-10-05]

[11] CPRIME. *What is agile? What is Scrum?* [ONLINE] Available at: https://www.cprime.com/resources/what-is-agile-what-is-scrum/ [2018-12-04]

[12] IIS. *Källkritik på Internet* by Kristina Alexanderson. [ONLINE] Available at: https://www.iis.se/lar-dig-mer/guider/kallkritik-pa-internet/ [2018-12-06]

Figure References

[Figure 1] Creator: Dervis Avdic [2018-01-27]

[Figure 2] Creator: Dervis Avdic [2018-11-29]

[Figure 3] Creator: Dervis Avdic [2018-11-29]

[Figure 4] https://cdn-images-1.medium.com/max/1600/0*95tBOgxEPQAVq9YO.png [2018-12-06]

[Figure 5] https://blog.codecentric.de/files/2017/12/Bildschirmfoto-2017-12-01-um-08.53.32.png [2018-12-06]

[Figure 6] https://technovert.com/xamarin/ [2018-11-29]

[Figure 7] https://www.sitepoint.com/build-cross-platform-android-ios-uis-xamarin-forms/ [2018-11-29]

[Figure 8] https://trends.google.com/trends/explore?date=today%205-y&geo=US&q=react%20native,xamarin [2018-11-29]

[Figure 9] Creator: Dervis Avdic [2018-01-28]

[Figure 10] https://luis-goncalves.com/what-is-scrum-methodology/ [2018-12-04]

[Figure 11] Creator: Dervis Avdic [2018-12-17]

[Figure 12] Creator: Dervis Avdic [2018-12-17]

[Figure 13] Creator: Dervis Avdic [2018-12-17]

[Figure 14] Creator: Dervis Avdic [2018-12-17]

[Figure 15] Creator: Dervis Avdic [2018-12-17]

[Figure 16] Creator: Dervis Avdic [2018-12-17]

[Figure 17] Creator: Dervis Avdic [2018-12-17]

[Figure 18] Creator: Dervis Avdic [2018-12-17]

[Figure 19] Creator: Dervis Avdic [2018-12-17]

[Figure 20] Creator: Dervis Avdic [2018-12-17]

[Figure 21] Creator: Dervis Avdic [2018-12-17]

[Figure 22] Creator: Dervis Avdic [2018-12-17]

[Figure 23] Creator: Dervis Avdic [2018-12-17]

[Figure 24] Creator: Dervis Avdic [2018-12-17]

List of Acronyms

APP	Application
API	Application Programming Interface