Juha Järvi

# Comparison of mobile application development technologies

Thesis
Spring 2020
SeAMK Technology

SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

## Thesis abstract

Faculty: Seinäjoki University of Applied Sciences

Degree programme: Information technology

Specialisation: Software Engineering

Author: Juha Järvi

Title of thesis: Comparison of Mobile Application Development Technologies

Supervisor: Marko Hietamäki

Year: 2020 Number of pages: 36

The purpose of this thesis was to explore the differences and limitations of native, hybrid and progressive web application development in a mobile environment. The aim was also to find out which projects these techniques are applicable and what should be considered.

tools used to develop examples and test these techniques were Android Studio and Visual Studio Code. The native application used Java as the programming language. In the hybrid and progressive web applications, JavaScript was used as the programming language. The Ionic 4 framework was also used in the hybrid application.

The result of this thesis was to find out what to take into consideration when choosing a technology for a new project and what kind of limitations certain choices bring. By comparing the differences, one could draw conclusions as to which technology is right for which application.

Keywords: Native, Hybrid, PWA, progressive web application, mobile application, Java, JavaScript, HTML, CSS, Angular, cross-platform

SEINÄJOEN AMMATTIKORKEAKOULU

## Opinnäytetyön tiivistelmä

Koulutusyksikkö: Seinäjoen Ammattikorkeakoulu

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Juha Järvi

Työn nimi: Mobiilisovellusten kehitystekniikoiden vertailu

Ohjaaja: Marko Hietamäki

Vuosi: 2020 Sivumäärä: 36

Opinnäytetyön tavoitteena oli selvittää natiivin, hybridin ja progressiivisen verkkosovelluksen eroavaisuudet ja rajoitukset mobiilisovelluskehityksessä. Tavoitteena oli myös selvittää millaisiin projekteihin kyseiset tekniikat soveltuvat ja mitä tulee ottaa huomioon.

Vertailussa käytettiin kehitysympäristönä Android Studioa ja Visual Studio Codea, joilla luotiin esimerkkejä ja testattiin kyseisiä tekniikoita. Natiivi sovelluksessa ohjelmointikielenä käytettiin Javaa. Hybridissä ja progressiivisessa verkkosovelluksessa ohjelmointikielenä käytettiin JavaScript. Hybrid sovelluksessa käytettiin myös Ionic 4 puitteita.

Opinnäytetyön tuloksena saatiin selvitettyä mitä tulee ottaa huomioon tekniikka valittaessa uuteen projektiin ja millaisia rajoituksia tietyt valinnat tuovat. Eroavaisuuksia vertailemalla voitiin vetää johtopäätöksiä siitä, mikä tekniikka on millekin käyttötarkoitukselle oikea.

Avainsanat: Natiivi, Hybridi, PWA, progressiivinen verkkosovellus, mobiilisovellus, Java, JavaScript, HTML, CSS, Angular, cross-platform

## TABLE OF CONTENTS

## Figures

## Tables

## Terms and abbreviations

| | |
|---|---|
| **API** | Application Programming Interface |
| **App Store** | Apple’s application market |
| **Cross-platform** | System that can work across multiple types of platforms or operating environments. |
| **Framework** | Platform for developing software application |
| **Google Play** | Google’s application market |
| **Hybrid** | Application that runs inside browsers WebView |
| **IDE** | Integrated Development Environment |
| **MVC** | Model View Controller means a design pattern that is commonly used for developing user interfaces. |
| **Native** | Software that is developed for use on a particular platform or device. |
| **PWA** | Website that displays like a mobile application. |

# 1 INTRODUCTION

This thesis focuses on android mobile application development and compares native, hybrid and progressive web applications. The first mobile applications on Android were created for information and daily purposes as reading mails, calculating, creating calendar events and checking forecast. These were great features for a smartphone back then, but nowadays people are doing a lot more with them. The wide variety of features that smartphones offer today are the main reason for high usage today and this is why there is such a need for more professional programmers to implement their ideas. A smartphone is more like a pocket computer. It is clear that smartphones have come closer to people's daily life in the past 10 years, it's the tool that many people find hard to live without.

## 1.1 Thesis background

In the year 2009 the first iPhone was born and around the same time first Android phones were released. Figure 1 shows that smartphones' popularity has grown a lot in the past 10 years. Today people use more their phones to make daily work easier.

The increased use of mobile devices has also increased the demand for applications. Nearly every week I hear someone having a good mobile application idea, but they do not have skills to create one or motivation to learn how to do it.

Developing a mobile application requires basic coding but nowadays there are many frameworks to choose from. Frameworks make it easier to get started with the development.

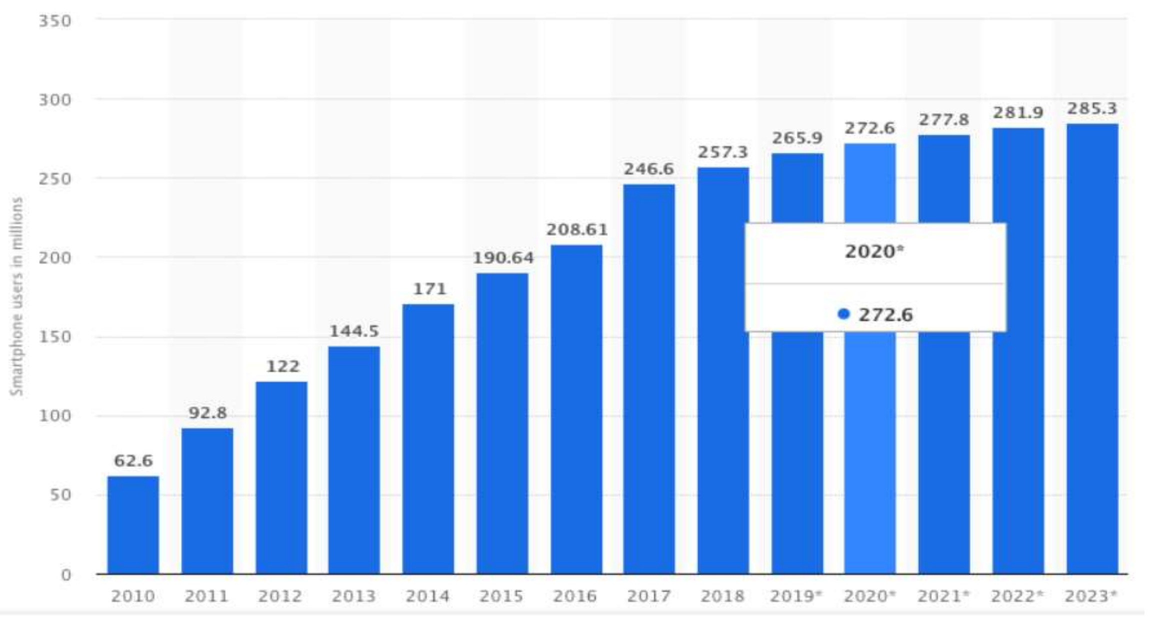


Figure 1 Popularity of smartphone usage (Mobile usage)

A common way to create mobile applications is native java coding for android or using swift for IOS. There is also a way to create a simple cross-platform application with web technologies. These cross-platform apps are using the same code base, so if you are familiar with creating websites or know something about JavaScript, it should be pretty easy.

## 1.2 Research aim and objectives

The aim of the study is to find out differences between the mobile application development technologies and help the programmer to decide which technology suits best their needs and the resources they have.

- What are native, hybrid and PWA?
- Why Hybrid and PWA might be the ways in the future of mobile applications?
- What are the benefits and disadvantages of cross-platform applications?
- How complex the technology or language is?

# 2 NATIVE APPLICATION

This chapter focuses on the native Android application. Native applications on Android use Java programming language and I have chosen Android Studio IDE to show some examples of the environment. Apple users can also use swift as the programming language and XCode IDE to create IOS native applications.

## 2.1 What is native applications

Natives are standalone applications and they are made for a certain mobile platform with the official programming language of the specific platform. For example, Android is using Java, and IOS is using Swift. Native applications communicate straight with the operating system (figure 2). With the straight connection, native application gets access to an application programming interface and through it can access the camera, sensors, and other components (comparative analysis, [referred 5.12.2019). Because of a simple straight connection between the application and operating system, the application performs as fast and smoothly as possible. Also publishing the final application in the App Store has been made very simple with native applications.

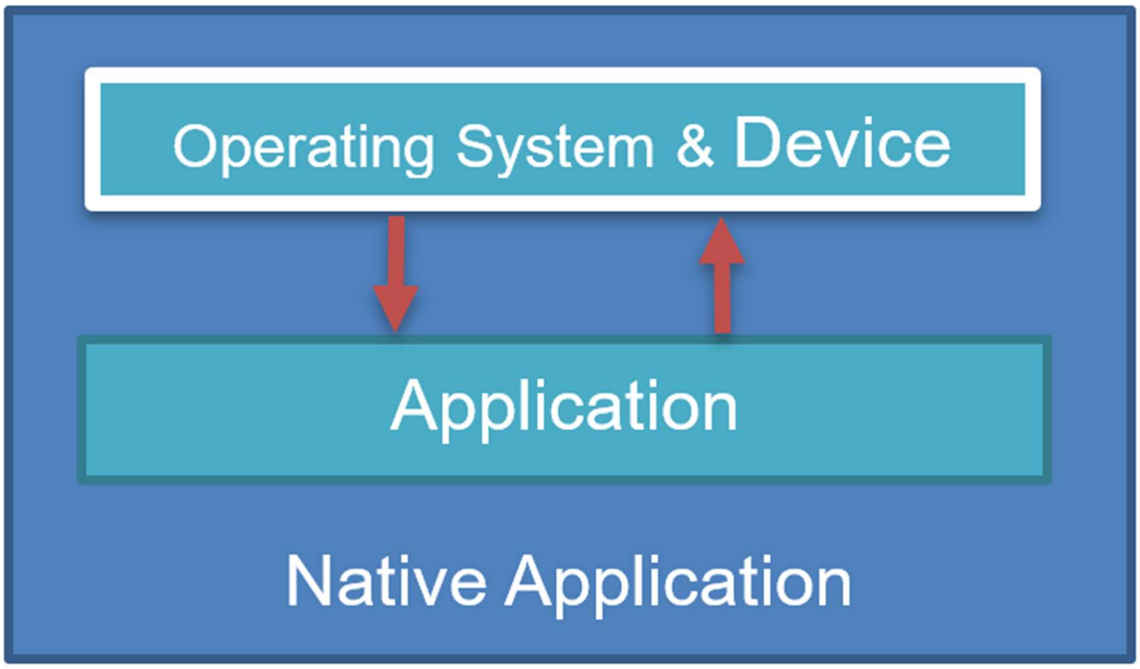
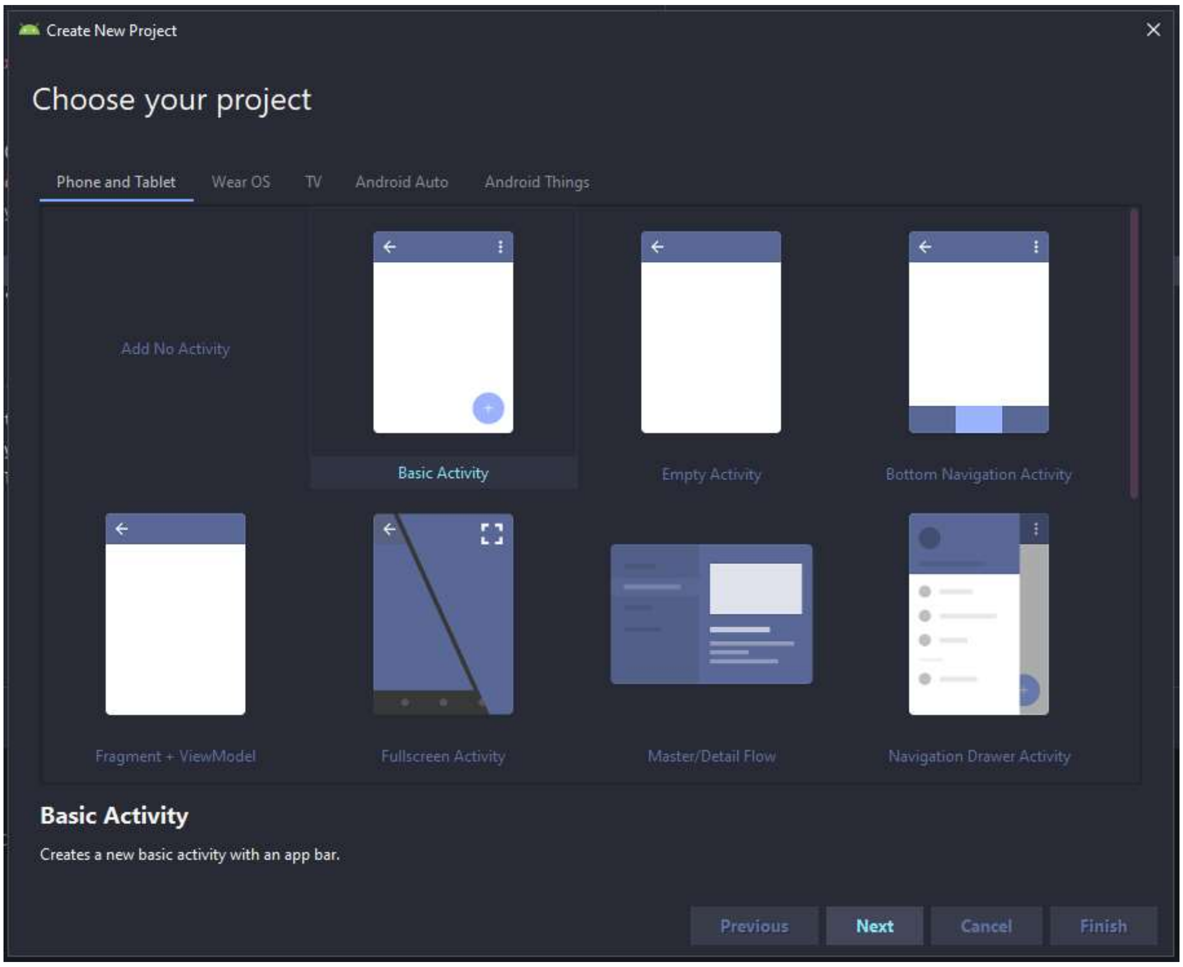


Figure 2 Native application principle

The biggest downsides of native mobile applications are probably developing and maintenance costs, because the application is made for a specific operating system. Native application development requires knowledge on both Android and IOS platforms.

## 2.2 Android Studio

Google has developed its own Android Development Environment Android Studio. Android Studio is the most common development environment for Android. Android Studio is built on the JetBrains IntelliJ development platform. Google itself has a great influence on popularity, and the platform is free for everyone. Google resources provide long-term support for Android Studio development, so it is worthwhile to try out. Google also has great documentation of the basics and its popularity provides a quick solution to problems (IntellJ idea blog [referred 20.11.2019). There are still possibilities to use IntelliJ IDEA, Unreal Engine, and other IDEs but the main IDE is Android Studio. The programming language is Java, which uses the Android Software Development Kit. Android SDK includes all the libraries for Android development.



**Basic Activity**

Creates a new basic activity with an app bar.

Figure 3 selecting a project template

Android Studio offers many prebuilt templates to get started with (Figure 3.) They are a great way to get started with Android Studio. The templates contain basic elements of Android native applications, like the navigation bar, floating buttons, and hamburger menus. Developers can see how those components are made and then copy them to their projects or just keep development with the current template.

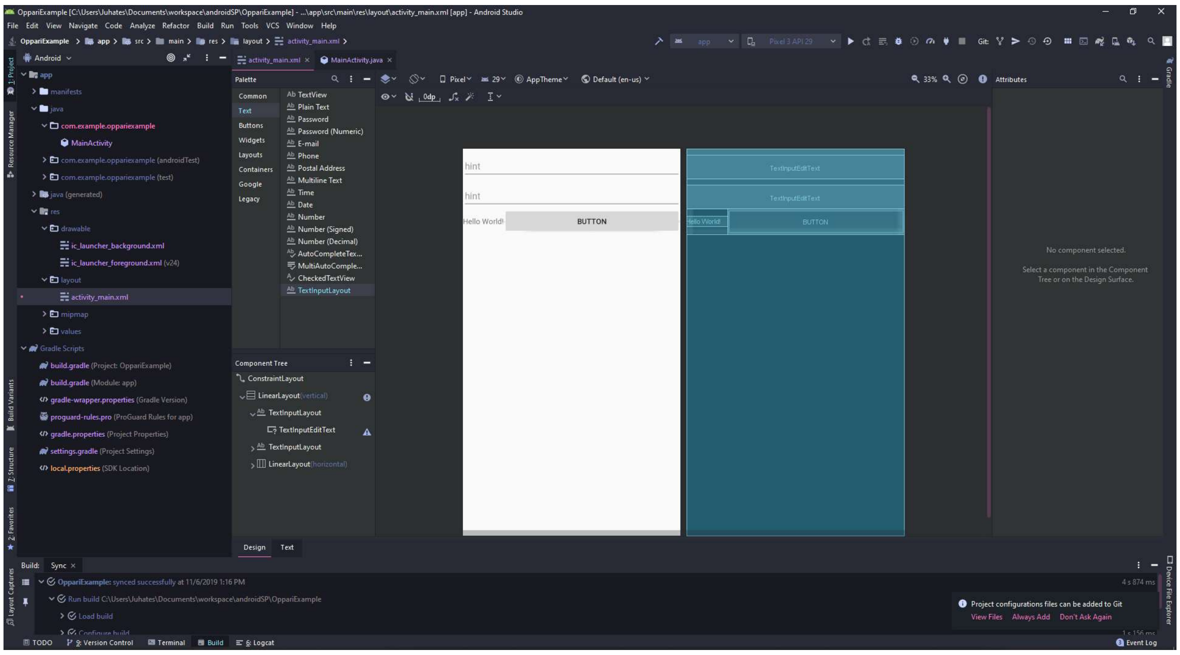

OppariExample [C:\Users\Juhates\Documents\workspace\androidSP\OppariExample] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
OppariExample > app > src > main > res > layout > activity_main.xml
Android
app
manifests
java
com.example.oppariexample
MainActivity
com.example.oppariexample (androidTest)
com.example.oppariexample (test)
java (generated)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: OppariExample)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MainActivity.java
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
TextView
Plain Text
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Number (Signed)
Number (Decimal)
AutoCompleteTex...
MultiAutoComple...
CheckedTextView
TextInputLayout
Component Tree
ConstraintLayout
LinearLayout (vertical)
TextInputLayout
TextInputEditText
TextInputLayout
LinearLayout (horizontal)
Pixel
29
AppTheme
Default (en-us)
33%
hint
hint
Hello World!
BUTTON
TextInputEditText
TextInputEditText
Hello World!
BUTTON
Attributes
No component selected.
Select a component in the Component Tree or on the Design Surface.
Design
Text
Build: Sync
OppariExample: synced successfully at 11/6/2019 1:16 PM
Run build C:\Users\Juhates\Documents\workspace\androidSP\OppariExample
Load build
Configure build
4 s 874 ms
Project configurations files can be added to Git
View Files
Always Add
Don't Ask Again
TODO
Version Control
Terminal
Build
Logcat
Event Log

Figure 4 Android Studio user interface with dark theme

The Android Studio designer mode is useful especially if you are new to mobile development. I found myself it easy to see how things work by just playing with the designer mode and using a drag and drop method. The designer also made it easy to get familiar with the components and naming (Figure 4).

# 3 HYBRID APPLICATION

Hybrid is getting more popular every day because of its agility. There are plenty of different frameworks for creating hybrid applications. In these examples, I am using the Ionic 4 framework. This chapter will quickly go through the history, architecture, components, and tools which help the developer to get started. It often takes a lot of time to find information, but ionic documentation and helpful applications make it easy to keep productive workflow.

## 3.1 What is hybrid application

When Android got popular in 2010, native applications were the main option developers had to create fast and user-friendly, fluid mobile applications for Android, but nowadays there are many other options. A hybrid application is the one, which got great advantage easily running on different operating systems, so the price and maintaining costs have decreased significantly. It is also very easy to get started if you have some web developing history because hybrids use JavaScript, HTML, CSS and Angular or React. The hybrid applications use frameworks that enable creating applications that behave like native ones. These applications are installed on the device just like native apps, so they are installed on the phone’s hard drive. The fact is that the hybrids got some downsides and they do not always perform as well as native applications. (Bakwa D, [referred 10.12.2019).

## 3.2 Hybrid application components

Hybrid applications are always made of one or more components. The ionic framework utilizes AngularJS, JavaScript, HTML and CSS programming technologies.  If a developer has previous experience with website development, Ionic may feel very much like home (Uudelleenkäytettävä mobiilisovellus, [referred 10.12.2019).

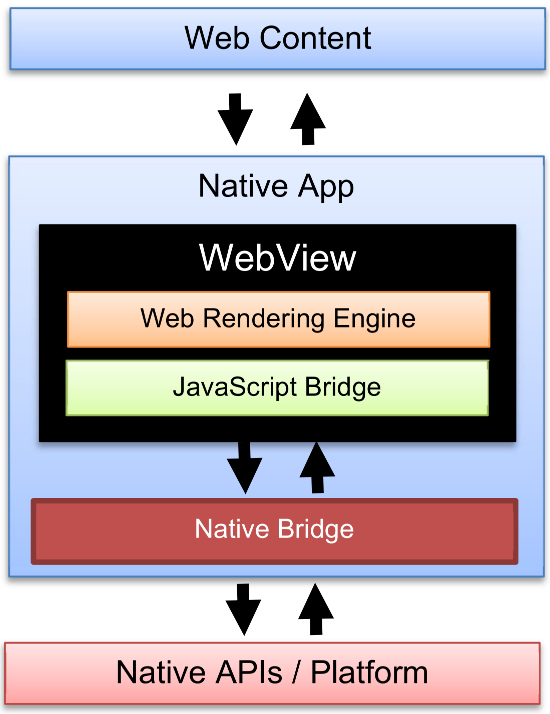


Figure 5 Hybrid application

The hybrid application runs inside webview which is inside the native application (Figure 5). The bridge connection between native and webview gives the application access to native application features like sensors and notifications.

### 3.2.1 Cordova

Cordova has been developed and maintained by Apace Software Foundation. Cordova is the new PhoneGab, if that sounds more familiar. Ionic 4 is using Cordova by deploying the final APK file or installing the software to the device. Cordova uses command-line tools for deploying the application and it does not have its developing environment. It is usually written with other web developing tools and integrated development environments example visual studio code, Atom, Sublime text, etc.

Cordova creates a full-screen WebView of the native application. The application is a standard HTML page, but when Cordova deploys the application into native, it creates a bridge between the application and the operating system. With the bridge, application can access the camera and other devices. (Ionic Glossary, [04.12.2019)

Hybrid apps use a web-to-native abstraction layer. The layer gives access to many device-specific capabilities and native APIs that are not generally accessible from the mobile web browser alone. There are also many built-in hardware functionalities in HTML5 APIs, but sometimes they are not enough. (Ionic WebView [referred 13.12.2019)

### 3.2.2 AngularJS

The first version of AngularJS was released in 2009. After that Google took the project maintenance. AngularJS is nowadays under Google's maintenance open source JavaScript library. AngularJS cannot develop server-side solutions, but it provides the MVC architectures (figure 6). AngularJS aims to facilitate application development by providing extensions and tests for application development (AngularJS, [referred 4.12.2019).

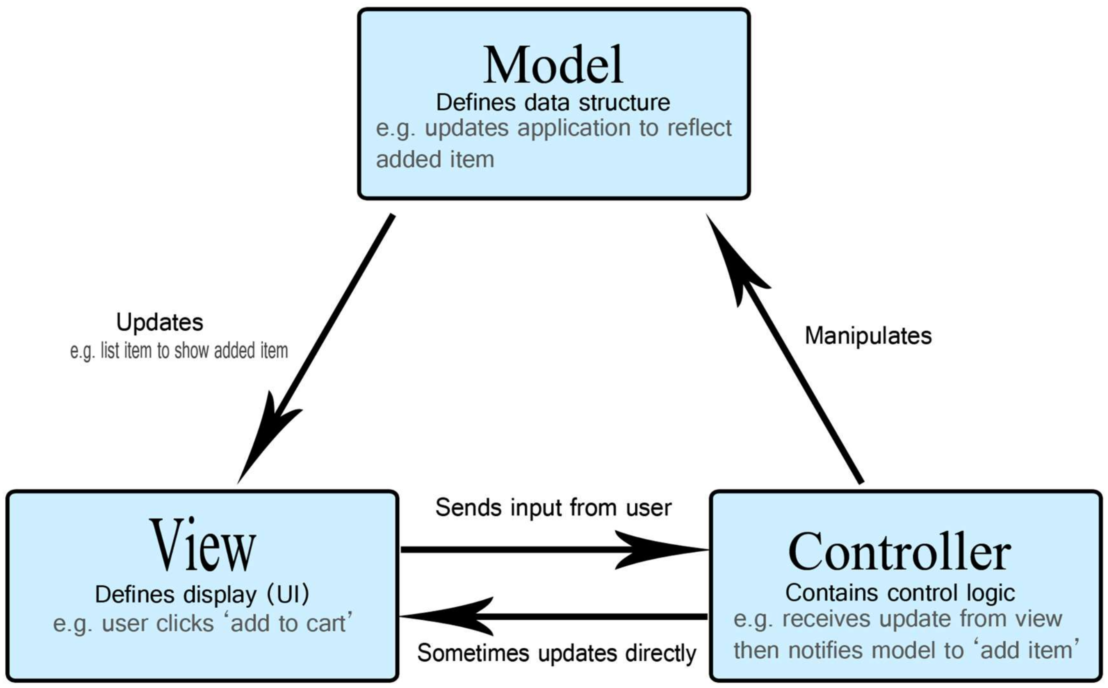

Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Figure 6 Model view controller (MVC)

AngularJS MVC contains model, view and controller. Model in AngularJS is a JavaScript object without getter or setter methods and defines the structure. It is a primitive data type such as number, string, boolean or object. View is the user interface which defines display. Controller is collection of JavaScript classes which controls the application logic. Controller receives updates from view and notifies model to react (What is MVC in AngularJS, [referred 26.3.2020).

### 3.2.3 WebView

WebView is an embedded browser that displays a "native" application. The browser itself is a native application that UI (User Interface) contains an address bar, navigation keys and other components required for use. The other part of the browser is an engine similar to WebView. The interface tells WebView what content should be displayed in the window and WebView executes it (Understanding WebViews [referred 15.12.2019).

Hybrid applications use WebView to display content. The only "native" thing in hybrid applications is maintaining WebView and displaying the required content there. Besides JavaScript is capable of using a native application programming interface within WebView, so the hybrid application is not as limited as the Web App alone (Understanding WebViews, [referred 15.12.2019). However, Hybrid applications can be referred to as technically native applications for this reason

## 3.3 What is Ionic

There are plenty of different frameworks and every day comes a new one, but they all have advantages and disadvantages. The moment when I am writing this thesis, Ionic 4 seems a promising choice, so that’s why I picked it up for an example.

Ionic is a rapidly growing open-source library that provides powerful, high-quality system-independent interfaces for mobile devices, websites, and computers. Ionic uses JavaScript, AngularJS, HTML and CSS programming languages. Ionic utilizes the Cordova framework. The ionic framework focuses on user interfaces such as buttons, animations and element placing. Ionic is easy to get started and there is plenty of information on the internet (Ionic Intro [referred 15.12.2019).

## 3.4 History of Ionic

The Ionic framework is today one of the most popular cross-platform development technology and releases some enhancements every six months. New versions of Ionic keeps still the same concept, so development doesn't suffer from this (figure 7).

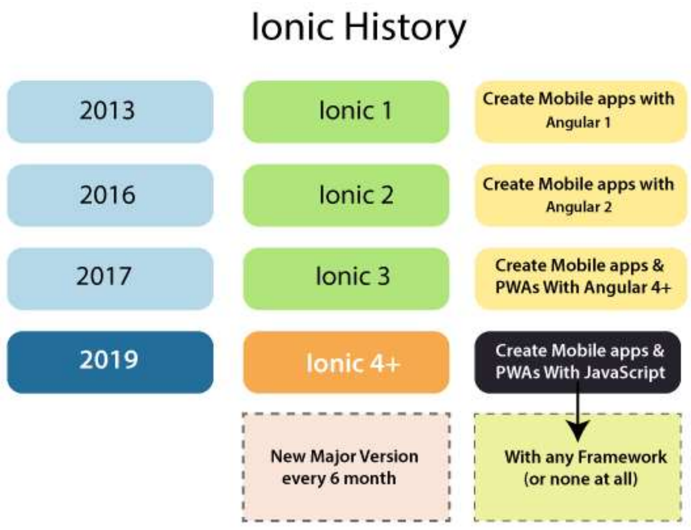

Ionic History
2013
Ionic 1
Create Mobile apps with Angular 1
2016
Ionic 2
Create Mobile apps with Angular 2
2017
Ionic 3
Create Mobile apps & PWAs With Angular 4+
2019
Ionic 4+
Create Mobile apps & PWAs With JavaScript
New Major Version every 6 month
With any Framework (or none at all)

Figure 7 History of Ionic (Javatpoint)

In 2013, Ionic 1 used Angular 1 and its directives. Web components were not in use yet (Ionic history, [referred 15.12.2019).

In 2016, Ionic 2 was released using Angular 2. This was just a newer version of the Angular but completely different from the older version, so Ionic 2 was no longer compatible with the old Angular (Ionic history, [referred 15.12.2019).

In 2017, Ionic 3 was released with a lot of new features but mainly focused on Angular 4. The Ionic development team found that Ionic 3 is not an optimal environment to continue operating as it is entirely limited to Angular (Ionic history, [referred 15.12.2019).

In 2019, Ionic 4 was released, focusing on web components. Web components allow you to create native application-specific applications that are run in WebView. Ionic 4 utilizes JavaScript, AngularJS, HTML and CSS Web technologies. It supports all browsers that allow access to traditional HTML elements and can be used with any web framework. Ionic 4 can be now used with any other framework (Ionic history, [referred 15.12.2019).

## 3.5 Security

You can never fully trust the content you download from a website. However, this does not cause any problems with WebView.

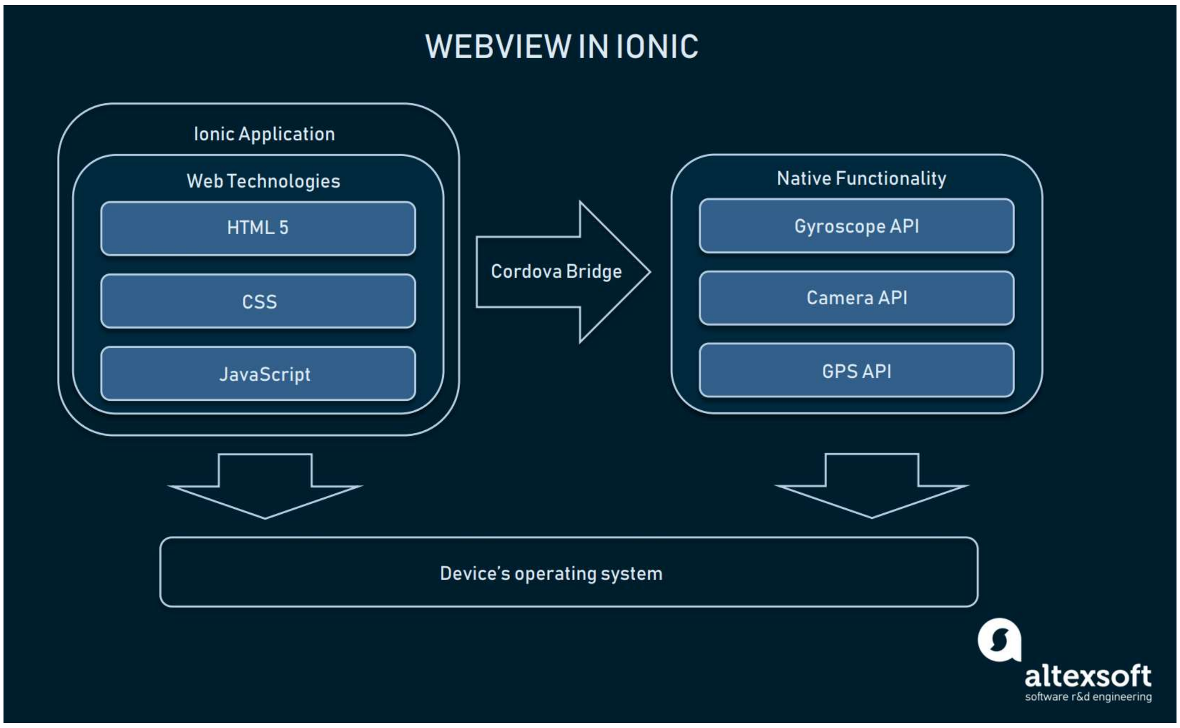


Figure 8 Ionic WebView (Ionic WebView)

The content of WebView is under the control of the developer, so the likelihood of malicious code getting inside WebView is low. Because of this, developers have been given several different options for override with standard data security settings and through this get the web code and native application to talk to each other (Figure 8). This connection is called a bridge. The bridge enables JavaScript to call the device API, the device programming interface, and use sensors, storage, contacts, and other components (Ionic webview, [referred 15.12.2019).

## 3.6 Ionic tools

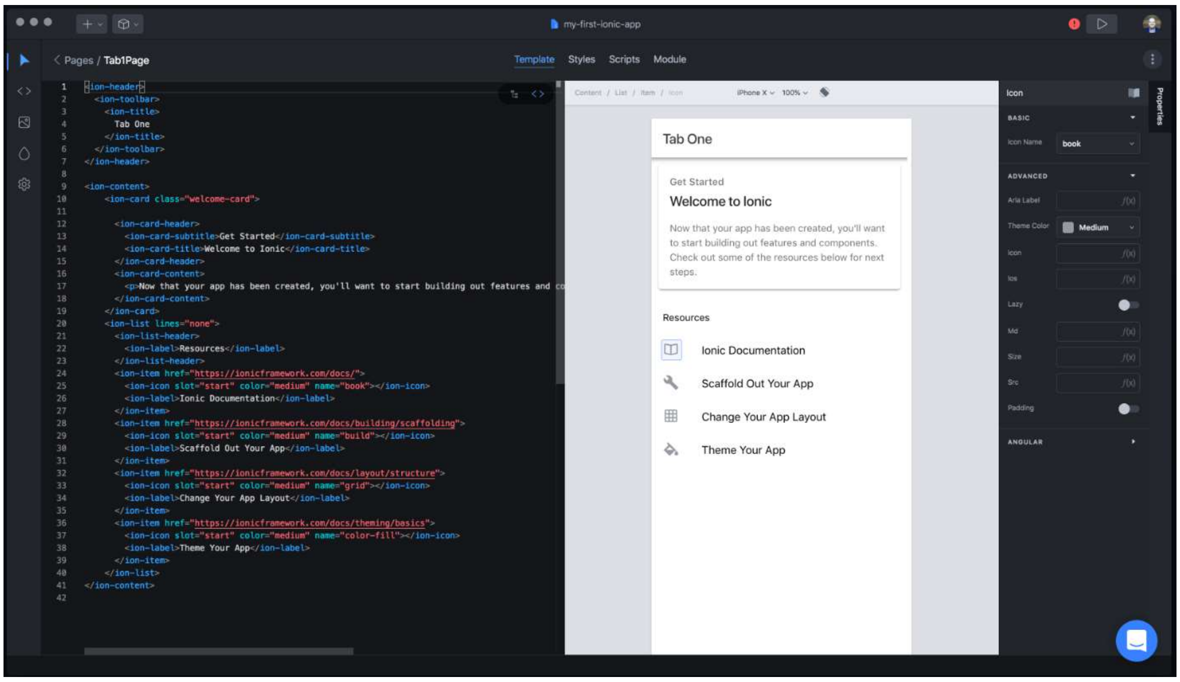
Ionic offers a lot of helpful tools for hybrid application developers to be more productive and speed up things. Programmers act differently and some of them like to use only command line tools etc. Anyway, everyone should give a chance for this kind of tool because if they are done right, they help to get a job done faster.

### 3.6.1 Ionic Creator

Ionic Creator makes it easy to create applications. The main focus isn't in the coding. The program is built into a web interface where you can directly move components with your cursor like drag and drop style, into the application and see the appearance changes in real-time. The application generates HTML code compliant with the Ionic framework in the background. Creating an interface is easy. Ionic Creator is free for public projects, but you must obtain a license for private and commercial use.

### 3.6.2 Ionic Studio

Ionic Studio was created for Ionic hybrid application development. IDE (Integrated Development Environment, [referred 20.12.2019) supports TypeScript, automatic code completion, de-bugging and more. besides, Ionic Studio offers many pre-programmed components that can be easily added to your project (figure 9).



```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Tab One
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
    <ion-card class="welcome-card">

      <ion-card-header>
        <ion-card-subtitle>Get Started</ion-card-subtitle>
        <ion-card-title>Welcome to Ionic</ion-card-title>
      </ion-card-header>
      <ion-card-content>
        <p>Now that your app has been created, you'll want to start building out features and co
      </ion-card-content>
    </ion-card>
    <ion-list lines="none">
      <ion-list-header>
        <ion-label>Resources</ion-label>
      </ion-list-header>
      <ion-item href="https://ionicframework.com/docs/">
        <ion-icon slot="start" color="medium" name="book"></ion-icon>
        <ion-label>Ionic Documentation</ion-label>
      </ion-item>
      <ion-item href="https://ionicframework.com/docs/building/scaffolding">
        <ion-icon slot="start" color="medium" name="build"></ion-icon>
        <ion-label>Scaffold Out Your App</ion-label>
      </ion-item>
      <ion-item href="https://ionicframework.com/docs/layout/structure">
        <ion-icon slot="start" color="medium" name="grid"></ion-icon>
        <ion-label>Change Your App Layout</ion-label>
      </ion-item>
      <ion-item href="https://ionicframework.com/docs/theming/basics">
        <ion-icon slot="start" color="medium" name="color-fill"></ion-icon>
        <ion-label>Theme Your App</ion-label>
      </ion-item>
    </ion-list>
</ion-content>
```

Figure 9 Ionic Studio

The idea behind Ionic Studio is to make programming as easy and efficient as possible over time. Ionic Studio does not offer a free trial version but requires a current monthly license, which will vary in price depending on the content (Ionic Studio, [referred 20.12.2019).

### 3.6.3 Ionic DevApp

Ionic DevApp is an application designed for Android and IOS that allows you to run the application directly on your mobile device in real-time. DevApp uses a network connection for data transfer and the devices must be on the same LAN. This is very useful if you need to give some demo experience for a customer or someone else.

Through DevApp, the application can be tested as fast as in the browser using localhost or Ionic Lab. These ways are much faster, installing the actual APK of the app on your phone.

### 3.6.4 Ionic Lab

Ionic Lab is an add-on to the Ionic Framework that can emulate Android and IOS applications side by side (Figure 10). In the lab, an application can be easily emulated on screens of different sizes.

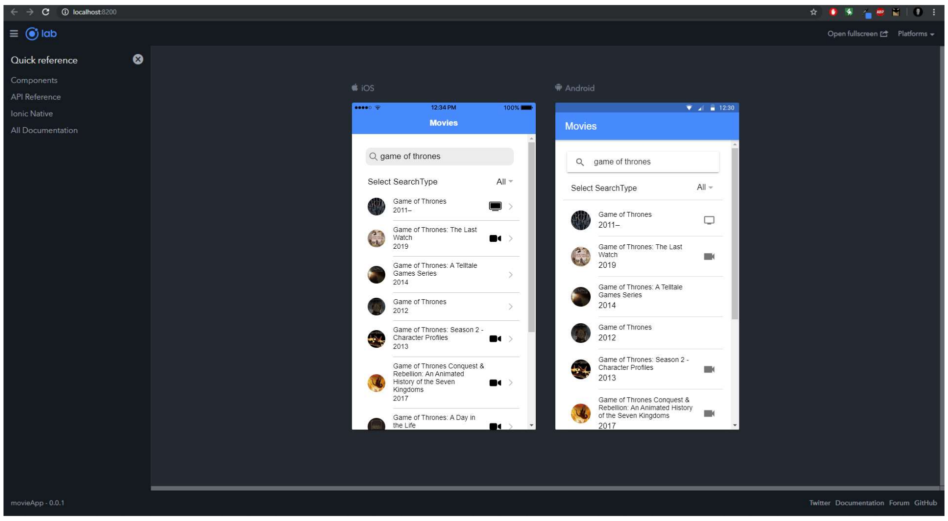


Figure 10 Ionic Lab

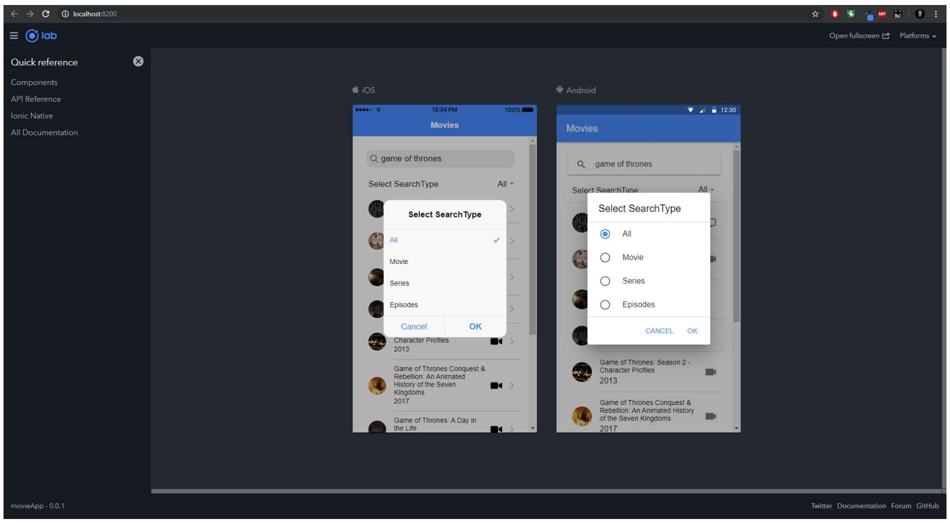


Figure 11 Ionic Lab styles

The Ionic Lab is updated in real-time for both operating systems, so the developer can easily track changes he makes. In the picture (figure 11) you can see how the Ionic Framework downloads its styles for IOS and Android. (Ionic lab, [referred 20.12.2019)

# 4 PROGRESSIVE WEB APP

The third category represents progressive web applications. Web apps act like a native mobile application but its really a regular web page. Web apps lack functionally if compared to native. There is no bridge access to API which could be used to access device components and functionalities. Browsers and web apps are becoming more advanced and now PWA's can do some similar functionalities as native apps. Sending push notifications, access device hardware, vibration and touch gestures. A few years back, PWA's were supported only by Google Chrome, but nowadays that's not a problem anymore Annie Dossey, [referred 20.12.2019).

## 4.1 What is PWA

Progressive Web Applications are web pages that act like a native application and gives better user experience than normal websites. Because PWA is just a regular website that runs on a server, it will work on all devices regardless of the operating system. The main requirement is that the device can run a browser. Updating is easy when there is only one codebase and it can be done on the fly (Annie Dossey, [referred 20.12.2019).

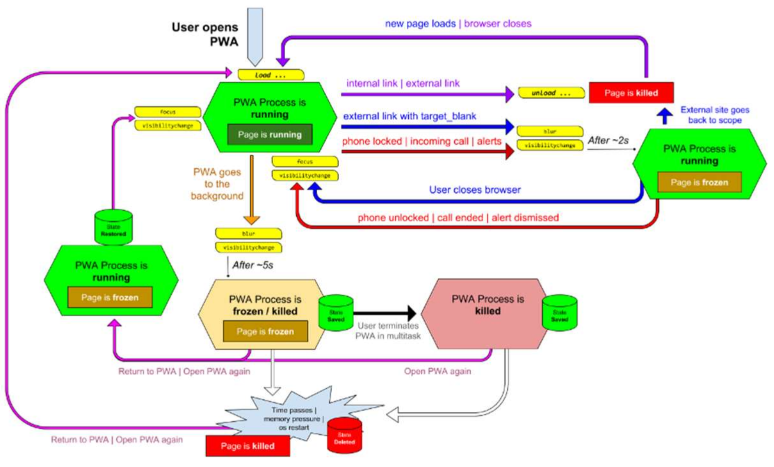

User opens PWA
new page loads | browser closes
load ...
PWA Process is running
Page is running
internal link | external link
unload ...
Page is killed
focus
visibilitychange
external link with target_blank
External site goes back to scope
phone locked | incoming call | alerts
blur
visibilitychange
After ~2s
PWA Process is running
Page is frozen
PWA goes to the background
focus
visibilitychange
User closes browser
phone unlocked | call ended | alert dismissed
State Restored
blur
visibilitychange
After ~5s
PWA Process is running
Page is frozen
PWA Process is frozen / killed
Page is frozen
State Saved
User terminates PWA in multitask
PWA Process is killed
State Saved
Return to PWA | Open PWA again
Open PWA again
Time passes | memory pressure | os restart
Return to PWA | Open PWA again
Page is killed
State Deleted

Figure 12 The new Lifecycle (Maximiliano Firtman)

New PWA lifecycle prevents iOS reload problems. iOS instance was terminated every time the user closed the application and it restarted from scratch. That was making PWA unusable on iOS but after 12.2 IOS update, this new lifecycle prevents that kind of issue (Figure 12). Application stores the current state of the program in local storage to retrieve it later in case of a full reload (Maximiliano Firtman, [referred 20.12.2019).

## 4.2 PWA Advantages

Web apps use the same code base on all devices so maintaining and updating are easy and more efficient than native and hybrid apps. Also running the app doesn't really require any specific specs, basically, the only requirement is that the device must be able to run with a web browser example Google Chrome. Web apps don't use standard operating system protocols and don't require approval from the marketplace, so releasing the application is very simple and makes maintaining easy (Maximiliano Firtman, [referred 20.12.2019).

## 4.3 Disadvantages of PWA

Web apps are much more restricted when it comes to features and hardware. That affects the overall user experience, because there is less responsiveness, and performance is poor if compared to native apps. There are many more steps to get into the web app, because user needs to open a web browser, search the URL and maybe log in, if there is no cookie data available. Some people are more familiar with these web sites and they know how to save a shortcut on the desktop, but still very few do that (Maximiliano Firtman, [referred 20.12.2019). Web apps usually have poor discoverability because they are not listed in the app store, but usually, this kind of applications are made by companies that already have specific clients. So, app marketing is quite easy.

# 5 COMPARISON OF TECHNOLOGIES

This chapter handles native, hybrid and PWA technologies and compares my thoughts and the results of Comparative analysis research. The results of table 1 are based on the comparative analysis research article and the results of table 2 are based on my thoughts about these technologies. Table 1 and Table 2 differ a little bit from each other because I found out that there are still a few important things to add, but the basics are the same.

- Supported mobile platforms, means operating systems of mobile devices on which it is possible to install and start an application (Comparative analysis, [referred 4.1.2020).

- Programming languages – languages in which the main code of the application is written. They differ depending on the platform for which the application is developed (Comparative analysis, [referred 4.1.2020).

- Official documentation and community of programmers – official documentation is available on the web sites of the mentioned IDEs and frameworks. Detailed documentation is extremely important for programmers, especially for those who are beginning to work with a certain technology. Alongside documentation, one of the key elements is the community of programmers who use the same technologies. In the evaluation of official documentation and community, mark 1 means that documentation is not transparent or detailed and that it is difficult to be found, so the community itself is not wide and it is almost impossible to find an answer to any question posed (Comparative analysis, [referred 4.1.2020).

- Application installation location – where the application will be installed on the device.

- Complexity of installation – getting started with a new framework or a language is always a challenge. Mark 1 means that the language needs a lot of training and earlier experience in that specific programming language or framework. Mark 5 means that basic knowledge of programming is enough to get started easily.

- Application speed – how fast an application performs. Mark 1 means long waiting times and slow performance. Mark 5 means responsive and fast performance.

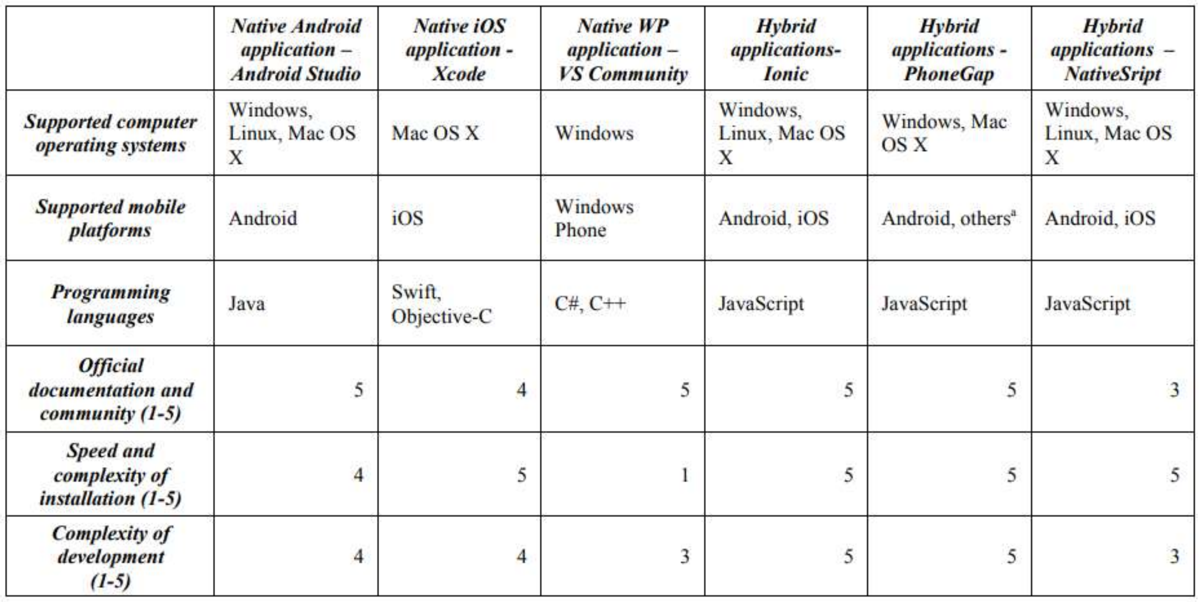
- User experience – almost the same as application speed. User experience focuses more on how the application feels in hand and how native-like the app animations and usage are. Mark 1 means lags and stuttering with bad responsiveness. Mark 5 means native fluid experience with a fast performance like native apps usually have.

- Development and maintaining cost – The cost of developing the product is generally known but maintaining and updating are usually not considered. After the product is finally ready for publishing, someone should keep it alive and updated as time goes by and that is not cheap. Mark 1 means developing and maintaining is very laborious. Mark 5 means easy updates and cost-effective development.

## 5.1 Comparative analysis results

Comparative analysis research made simple mobile applications with each technology. Their goal was to show the principle of functioning of those six different development tools and different IDE's. Computers they used for development was running Windows 7 professional with 4GB of RAM and OS X Yosemite with 8GB of RAM. Computer specs have a major effect on development speed when programming and compiling software. They used Android Studio 2.1, XCode 6.4 and Visual Studio Community 2015. Hybrid application development they used Ionic version 1.7.14 and PhoneGap version 6.3.0 for Android and iOS (Comparative analysis, [referred 20.1.2020).

Table 1 Comparative analysis comparison of technologies (Comparative analysis, [referred 20.1.2020)



| | *Native Android application – Android Studio* | *Native iOS application - Xcode* | *Native WP application – VS Community* | *Hybrid applications- Ionic* | *Hybrid applications - PhoneGap* | *Hybrid applications – NativeSript* |
|---|---|---|---|---|---|---|
| ***Supported computer operating systems*** | Windows, Linux, Mac OS X | Mac OS X | Windows | Windows, Linux, Mac OS X | Windows, Mac OS X | Windows, Linux, Mac OS X |
| ***Supported mobile platforms*** | Android | iOS | Windows Phone | Android, iOS | Android, others[a] | Android, iOS |
| ***Programming languages*** | Java | Swift, Objective-C | C#, C++ | JavaScript | JavaScript | JavaScript |
| ***Official documentation and community (1-5)*** | 5 | 4 | 5 | 5 | 5 | 3 |
| ***Speed and complexity of installation (1-5)*** | 4 | 5 | 1 | 5 | 5 | 5 |
| ***Complexity of development (1-5)*** | 4 | 4 | 3 | 5 | 5 | 3 |

Referring to the comparative analysis research, native Android applications are not as good as Hybrid Ionic framework applications. The difference is not that big, but the hybrid is still much more agile, and it uses generic web application technologies, so many programmers might find it easy to get started with (Table 1). The speed and complexity of installation and also development is slower on a native application. Running and testing a native application is also slower because Android Studio converts the application APK and installs it into the device. Hybrid apps run real-time on the web page, so all changes can be seen immediately after saving the modified files.

## 5.2 Results of the study

Table 2 compares the basic differences between native, hybrid and PWA programming technologies. There are minor differences in the application speed between a native, hybrid and PWA but It will directly affect the user experience which is one of the most important things in the application.

Table 2 Comparison of development technologies

| | **Native Android** | **Hybrid Ionic** | **PWA** |
|---|---|---|---|
| **Programming language** | Java | JavaScript | JavaScript |
| **Supported mobile platforms** | Android | Android, iOS | Android, iOS |
| **Application installation location** | Local | Local | Web browser |
| **Access to API** | Full access | Full access | Lacks access to NFC, contacts and proximity sensors |
| **Getting started / Complexity of installation** | 3 | 4 | 4 |
| **Complexity of development** | 3 | 4 | 2 |
| **Application speed** | 5 | 4 | 3 |
| **User experience** | 5 | 3 | 2 |
| **Official documentation and community** | 4 | 4 | 3 |
| **Development and maintaining cost** | 2 | 4 | 5 |
| **Security** | 5 | 4 | 3 |

Official documentation and overall documentation have a great impact on the development and maintaining costs if the developer team does not have earlier experience in the framework or current language. Access to API restricts using PWA if the application needs access to the gyroscope or some API specific sensors. This makes it difficult to evaluate different technologies because there are so many factors that affect it. However, this comparison is made from a beginner programmer's perspective.

### 5.2.1 Native applications

Native applications behave the same way on Android and iOS. The native application is made with a specific programming language and runs on a certain operating system, for example, Apple iOS and other manufacturers like Samsung, LG and Google use Android. Apple uses swift and objective-C with XCode IDE. Android uses Java programming language with Android Studio. That is why it is the most inefficient way to create apps for multiple platforms. However, a native application performs and gives the best user experience of all these technologies and it is hard to start arguing about it when everything works fast and smoothly without problems. The native applications will be installed on the local hard drive of a phone so offline usage is also possible. A local install speeds up the loading times because it is not dependent on internet speed or connection quality. One of the most important benefits in the native application is straight communication with API, which permits to use a gyroscope and other device sensors. I noticed that setting up the Android Studio environment was easy but after that there were hard times with Android API. According to the Comparative analysis research (Comparative analysis, [referred 22.1.2020), getting started was not so difficult as with hybrids and PWA but I disagree. Maybe because I am used to working with Web development technologies and frameworks, it was hard to get started with native programming. However, everything makes much more sense in Java architecture if compared to JavaScript which is used by the hybrids and PWAs. About the security, a native has often been seen as the most secure option. That is because all the development has been done into the infrastructure of the app, encrypted and obfuscated (What about security, [referred 22.1.2020).

### 5.2.2 Hybrid applications

Hybrid applications use JavaScript programming language, so it’s is easy to get started with if the developer has some previous experience in web development because most hybrids use JavaScript as the programming language. JavaScript has a large community and Ionic documentation is good. Problems are usually easy to solve just using Google.

The hybrid application will be installed in a local storage, so it takes some space but nowadays, that is not a problem because of the large hard drives that phones have. Local install gives also the opportunity to use the app in an offline mode without an internet connection. Hybrids are cross-platform apps, so the codebase is the same on android and iOS. Cross-platform apps are easier to update and maintain because there is no need to have multiple knowledge of different languages or frameworks like native applications have. If Programmer have earlier experience in web technologies like HTML, SASS, JavaScript and bootstrap framework it may feel like a home with Ionic framework.

Hybrid applications perform very well nowadays. A user might not even notice the difference between a hybrid and native. The programmer does not need to think about the different styles for iOS and Android. All animations and application themes come automatically from the Ionic framework if the HTML has been done right. Ionic tools that I tested were Ionic Creator, Ionic Studio and Ionic DevApp. Those tools help a lot in the development process. DevApp was a great tool to see real-time changes on the device and made the user interface designing fun and easy. In my opinion the only reason to create native applications might be the requirements to use all the functions of the mobile device through the API. There are still some functionalities that hybrids cannot perform but natives can.

### 5.2.3 Progressive web applications

A progressive web application runs on a web page in a web view. That means it i has not been installed inside the phone's local memory and it does not take any space. That is why PWA is dependent on the internet connection to work properly. The application can run from a browser cache, but all functionalities are not available then. The programming language is JavaScript with HTML and CSS. PWA is also easy to get started especially if the programmer has some earlier experience in web development. A PWA application runs inside the browser so any device with a browser can run the application. There is only one codebase like in hybrid applications and the app runs on a server-side, so updating is done there.  Users do not need to download any updates on their phones.

PWA is lacking API access. There is no Bluetooth connection, NFC, proximity sensor, ambient light, wake lock or access to contacts (Stack Overflow, [referred 18.1.2020). User experience on PWA is good but not as good as with native or hybrid. Responsiveness and overall performance are slower than those of the native. There are so many ways to create progressive web applications with all available frameworks that documentation may be messy when the programmer starts the learning process. Documentation depends on the path that the programmer chooses. A hybrid application wrapper gives almost the same security capabilities as native. Progressive web applications are often seen as a less secure option if compared to native applications. This is because the hybrid apps run inside a container. The container is essentially a normal web browser which can be exposed to some of the features of the underlying platform. PWA’s security is better than that of normal web applications because of HTTPS. Security protocols ensure that there are no exchanges between the server and the client. Still PWA is less secure than hybrid and native applications (PWA vs Native App, [referred 18.1.2020).

# 6 Conclusions

It is often difficult to decide which the technology to use for developing an application. Programmers often choose the language or framework that is familiar to them, so the project can be started quickly. That is understandable, of course, but later this may bring problems and cause big expenses in maintenance. Therefore, it is really important to determine which are the requirements before starting a project. What kind of customer group the application will be targeted to, which operating system it should run on and which resources are available?

## 6.1 Native application

A native application is the best option if the application is planned to run only on a particular operating system. Native apps are faster and more responsive because they are usually built lightweight and run on that specific platform. The app will have access to all device components, for example, a camera, gyroscope and Bluetooth and NFC through API.

If the application needs to be run on multiple operating systems, then native development cost is significantly higher compared to hybrid and progressive web applications, because the application needs specific programmers and knowledge for different platforms and languages. Also maintaining and updates must be done separately for each platform. However, native applications are the most secure ones and provide the best user experience after all, so If money is not a problem, a native application can still be an option.

## 6.2 Hybrid application

Hybrid applications are quite close to the native ones today. There are plenty of frameworks that are based on web development technologies. Most hybrid frameworks use JavaScript and AngularJS or React. That helps a lot to get started if the developer has earlier experience in web development. Nowadays it is easy to find people to carry out such a project because the web is everywhere. Documentation is good and that is why it is easy to solve problems.

The ionic framework brings many useful tools to help developing. The application is updating real-time on the web browser or a real smartphone with an ionic dev tool installed. These things are small but speed up developing when programmer can see responsiveness and UI updating in real-time. Hybrids use the same code base on every operating system. Therefore, maintaining and updating the application is cost-effective and easy. Studying JavaScript gives an opportunity to try several other frameworks that are available.

### 6.3 PWA application

The development team only needs to build the software for a single operating system. There is no need to hire multiple different programmers to create software with different languages. The development team only needs to build it for a single operating system, which helps a lot with updates, testing and maintaining. The same code base runs on all operating systems so maintaining is efficient and powerful. Users do not need to download updates, because the application runs on a web site.

Apps can be used on any device that runs a web browser regardless of the operating system as long as there is internet connectivity. Unlike conventional applications, PWAs can be easily shared with a URL-link and they give the user a quick access to test the application anywhere.

PWAs are cross-platform apps and use the same codebase on all operating systems. The content on web applications can be customized to match different operating systems and frameworks, like Ionic loads automatically current operating system native animations and fonts, so depending on the running device, the application gets the native look.

# REFERENCES

Angular Docs v.1.7. Angular Modules [Online publication]. [Ref. 3. December 2019]. Available at: https://docs.angularjs.org/guide/module

Angular Docs. Architecture Overview [Online publication]. [Ref. 15 December 2019]. Available at: https://angular.io/guide/architecture

Annie Dossey. 20.7.2019. A Guide to Mobile App Development :Web vs. Native vs. Hybrid [Online publication]. [Ref. 4. January 2020]. Available at: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/

Bakwa D. Dunka, Dr.Edim A. Emmanuel, Dantala O Oyeyinka. December 2017. Hybrid mobile application based on ionic framework technologies [Online publication]. [Ref. 1. December 2019]. Available at: https://www.researchgate.net/publication/322397904_HYBRID_MOBILE_APPLICATION_BASED_ON_IONIC_FRAMEWORK_TECHNOLOGIES

Comparative analysis. Vilček,T. Tomislav, J. Faculty of Humanities and Social Sciences, Osijek, Croatia. Comparative analysis of tools for development of native and hybrid mobile applications. [Online publication]. [Ref. 10.11.2019. Available at: https://ieeexplore.ieee.org/. Requires access.

Harnil Oza. 17.3.2019. Google Launches Android Studio 3.5 [Online publication]. [Ref. 12 December 2019]. Available at: https://www.hyperlinkinfosystem.com/blog/google-launches-android-studio-35-whats-new

https://stackoverflow.com/questions/55693406/can-pwa-access-contacts-gps-or-use-the-phone-camera

Ilpo Andström. 2.5.2016. Hybridimobiilisovelluksen kehitys [Online publication]. [Ref. 12 December 2019]. Available at: https://www.theseus.fi/bitstream/handle/10024/114497/ilpo_andstrom.pdf?sequence=1&isAllowed=y

IntellJ Idea Blog. 16.3.2013. IntellJ. [Online publication]. [Ref. 11 December 2019]. Available at: https://blog.jetbrains.com/idea/2013/05/intellij-idea-and-android-studio-faq/

Ionic Docs. 09.12.2019. UI Components [Online publication]. [Ref. 14 December 2019]. Available at: https://ionicframework.com/docs/components

Ionic Glossary 29.01.2020 Cordova [Online publication]. [Ref. 20 February 2020]. Available at: https://ionicframework.com/docs/faq/glossary#cordova

Ionic Intro 11.02.2020 Cordova [Online publication]. [Ref. 24 February 2020]. Available at: https://ionicframework.com/docs/intro

Ionic Lab. 11.2014. Introducing Ionic Lab [Online publication]. [Ref. 16 December 2019]. Available at: https://ionicframework.com/blog/ionic-lab/

Ionic Studio. 2019. Visual Mobile App Development [Online publication]. [Ref. 10. January 2020]. Available at: https://ionicframework.com/studio

Ionic WebView. 09.12.2019. WebView [Online publication]. [Ref. 5 January 2020]. Available at: https://ionicframework.com/building/webview

Javatpoint. Ionic history [Online publication]. [Ref. 22 December 2019]. Available at: https://www.javatpoint.com/ionic-history

Matti Väkiparta. Spring 2017. Android sovelluskehityksen perusteet Android Studiolla. [Online publication]. [Ref. 10. November 2019]. Available at: https://www.theseus.fi/bitstream/handle/10024/123499/Matti_Vakiparta_Opinnaytetyo.pdf?sequence=1&isAllowed=y

Maximiliano Firtman. 26.3.2019. What's new on iOS 12.2 for Progressive Web Apps [Online publication]. [Ref. 12. November 2019]. Available at: https://medium.com/@firt/whats-new-on-ios-12-2-for-progressive-web-apps-75c348f8e945

Mobile usage. 26.9.2019. 75 Mobile statistics to get you amped for 2020 [Online publication]. [Ref. 20. March 2020]. Available at: https://www.highervisibility.com/blog/mobile-statistics/

MVC. 18.3.2019. Model view controller example [Online publication]. [Ref. 10. December 2019]. Available at: https://developer.mozilla.org/en-US/docs/Glossary/MVC

PWA vs Native App. 17.8.2018. Security. [Online publication]. [Ref. 10. January 2020]. Available at: https://blog.magestore.com/pwa-vs-native-app/

Stack overflow 18.8.2016 What features do Progressive Web Apps have vs Native apps and vice-versa, on Android [Online publication]. [Ref. 15. February 2020]. Available at: https://stackoverflow.com/questions/35504194/what-features-do-progressive-web-apps-have-vs-native-apps-and-vice-versa-on-an

Stack Owerflow. 30.9.2019. PWA access [Online publication]. [Ref. 17. January 2020]. Available at: Angular Docs v.1.7. Angular Modules [Online publication]. [Ref. 3. December 2019]. Available at: https://docs.angularjs.org/guide/module

Technology stacks. 25.9.2019. Do you have a technology stack to create your mobile apps [Online publication]. [Ref. 13. January 2020]. Available at: http://naija-techninja.blogspot.com/2015/09/do-you-have-technology-stack-to-create.html

Timo Salola. Spring 2016. Uudelleenkäytettävä mobiilisovellus. [Online publication]. [Ref. 10. December 2019]. Available at: https://www.theseus.fi/bitstream/handle/10024/110261/Salola_Timo.pdf?sequence=1&isAllowed=y

Understanding WebViews. WebView 101 [Online publication]. [Ref. 27 December 2020]. Available at: https://www.kirupa.com/apps/webview.htm

What about security. 14.3.2018. Hybrid or native: but what about security. [Online publication]. [Ref. 5. may. 2020]. Available at: https://blog.onegini.com/native-vs-hybrid-apps

What is MVC in AngularJS. 4.12.2019. AngularJS MVC Architecture. [Online publication]. [Ref. 26. may. 2020]. Available at: https://intellipaat.com/blog/tutorial/angularjs-tutorial/mvc-angularjs/