FH JOANNEUM (University of Applied Sciences)

**Usage possibilities of WebRTC in a cross-platform developed hybrid app**

**Bachelor Thesis**

**submitted in conformity with the requirements  
for the degree of  
Bachelor of Science in Engineering (BSc)**

Bachelor’s degree program **Internettechnik**FH JOANNEUM (University of Applied Sciences), Kapfenberg

**supervisor:** Dipl. Ing. Johannes Feiner

**submitted by:** Michael Stifter  
**personal identifier:** 1310418054

06 / 2016

**Obligatory signed declaration:**

I hereby declare that the present Bachelor’s thesis was composed by myself and that the work contained herein is my own. I also confirm that I have only used the specified resources. All formulations and concepts taken verbatim or in substance from printed or unprinted material or from the Internet have been cited according to the rules of good scientific practice and indicated by footnotes or other exact references to the original source.

The present thesis has not been submitted to another university for the award of an academic degree in this form. This thesis has been submitted in printed and electronic form. I hereby confirm that the content of the digital version is the same as in the printed version.

I understand that the provision of incorrect information may have legal consequences.

Michael Stifter Graz, 13.06.2016

**Table of contents**

[Abstract 4](#_Toc451611011)

[Kurzfassung 5](#_Toc451611012)

[1 Introduction 6](#_Toc451611013)

[List of tables 7](#_Toc451611014)

[List of figures 8](#_Toc451611015)

[List of listings 9](#_Toc451611016)

[List of abbreviations 10](#_Toc451611017)

[Bibliography 11](#_Toc451611018)

Abstract

Kurzfassung

# Introduction

Over the last years, Web Real Time Communication (WebRTC) has seen a significant rise in popularity, especially in browser-based web applications. Its biggest disadvantage to date is the fact that not all web browsers support WebRTC, although the number of supporting browsers has been continuously rising for a few years now.

This poses a problem for developers who want to use WebRTC in applications today. While nowadays there are few alternatives to web applications in terms of desktop devices, the situation is different for mobile devices. Native apps have become massively popular and deliver substantial advantages when it comes to user experience. This stems from the fact that it is possible to integrate and access many components of the user’s device, such as the list of contacts, the calendar and various sensors into an application with ease. While it is possible to develop a native app that uses WebRTC, it also increases the development effort considerably, since it is necessary to implement the same functionality on multiple platforms, such as Android, iOS and Windows Phone.

A solution to this problem could be the use of a suitable cross-platform development framework that facilitates the use of WebRTC. For a cross-platform developed mobile app, it is not necessary to develop the same application once for each platform it should support, but rather only once. The framework then generates a native app from the shared code base. However, since WebRTC is a technology that can be considered relatively new and is still under development, it is not guaranteed that cross-platform development frameworks fully support the latest version of WebRTC.

This thesis takes a deeper look into popular cross-platform mobile development frameworks and examines them on their ability to support current versions of WebRTC. To analyze this examination, a set of criteria is defined in order to identify suitable frameworks for developing cross-platform apps that use WebRTC.

The thesis is structured as follows: The first part describes various ways of implementing a mobile app and highlights the advantages and disadvantages of each method in detail. The second part discusses the history and functionality of WebRTC, together with its benefits and shortcomings. In a third step, the possibilities of using WebRTC on mobile devices are addressed. Following that, the essential insights regarding the implementation of a reference app are pointed out. Chapter 5 discusses the evaluation process and its results. The final section concludes the thesis by summarizing the essential findings and suggesting possibilities to expand the underlying work.

# Cross-platform mobile development

Beginning with the introduction of Apple’s iPhone back in 2007, mobile applications have become massively popular. Back then, it was self-evident to implement an app natively on the one hand, and on the other hand there were no other options to do so. However, the following years saw a substantially increasing number of popular mobile platforms, such as Android, Windows Phone and the aforementioned iOS. Since all these platforms use different programming languages, there was no possibility to reuse the programming code written for one platform, it had to be rewritten in the exact same way for all platforms that should be supported. Additionally, making changes to an app again meant going through all platforms and implementing the changes separately for each platform (cf. PAPER-1).

APIs, MDD?

Another solution to this problem is cross-platform mobile development. It enables developers to write code for an app only once and, subsequently, generate native applications for all desired platforms from that code base. In most cases, the code is written using web technologies, such as HTML5, JavaScript and CSS. Incidentally, PAPER-2 points out, it was the original plan for apps for the iPhone to be written using these tools. In the end, however, Apple decided that third-party apps for their operating system have to be written natively in Objective-C, which was followed by Swift in 2014.

Introduction, motivation

PAPER-1

PAPER-5 [In this context, the challenge for web developers is to de-

velop di\_erent versions of their applications that are cus-

tomized to suit the speci\_c characteristics of the di\_erent

platforms, yet provide a consistent set of features and ser-

vices across all versions.]

* API

-🡪 Model Driven Development? (PAPER-6)

## Differences to native app development

A large difference between a native app and a cross-platform developed app is the fact that native apps are usually compiled, which in most cases results in faster execution times because the programming code is translated into machine code before the execution of the program. Cross-platform developed apps, on the other hand, mostly use interpreted languages such as JavaScript, which executes its code instructions step-by-step without compiling them first (cf. PAPER-2).

PAPER-2

Difference native/compiled versus web/interpreted code

Sensor/device access

User experience (push notifications, access to phonebook, contacts)

Ability to use app offline (HTML5 application cache – PAPER-3)

## Motivation

Pro / con

PAPER-7

## Approaches

PAPER-1 defines four different categories for cross-platform developed apps: Web, hybrid, interpreted and generated apps. All four approaches will be discussed in detail in this section.

PAPER-1

PAPER-9

### Web apps

Web apps are applications that run within a web browser. Typically, they use HTML5 and JavaScript. The advantage of web apps is that nowadays, almost any smart mobile device has a web browser installed, thus providing a broad range of dissemination. One disadvantage of web apps is the limited access to the device’s sensors, file system and features like contact list and calendar. Native apps, on the other hand, can exploit the device’s full potential when it comes to these features.

Unlike native apps, web apps do not need to be physically installed on the device and, furthermore, also do not have to be upgraded when a new version is available. At the same time, this becomes a disadvantage when users are not connected to the internet. In this case, the web app is not accessible to the user (cf. PAPER-1). There are modern HTML5 technologies like the Application Cache (AppCache) to eradicate this problem. AppCache allows developers to store programming logic and data on the user’s device. However, this technology requires substantial additional programming effort (cf. PAPER-3).

### Hybrid apps

Hybrid apps are a combination of native apps and web apps. They are “primarily built using HTML5 and JavaScript, and a detailed knowledge of the target platform is not required” (PAPER-1). The essential difference to web apps is that they are running within a native app container. The code is still executed by a web browser, but can be bundled together with the application, thus removing the necessity of an active internet connection to download the programming logic. With hybrid apps, it is also possible to access the device’s special features through APIs provided by the cross-platform development framework (cf. PAPER-1).

### Interpreted apps

Interpreted apps use pre-defined commands during the development process to use native user interface components. This means that on the Android platform users will interact with typical Android-styled buttons, while on iOS users will interact with iOS-styled buttons, without any effort of the developer. Despite this advantage in user experience, the developer is completely dependent on the used framework. This could especially pose a problem when a new version of an operating system is released, because it is not clear if the app will automatically have access to new features or if all previously used components will look and behave the same way (PAPER-1).

### Generated apps

This type of cross-platform developed apps use the code to generate native apps from it. They benefit from a high overall performance due to the use of compiled native code. One downside of generate apps is the increased build time that has to be carried out each time a change is made to the app (cf. PAPER-1).

## Cross-platform development frameworks

PAPER-4

Important criteria for choosing a framework

### Apache Cordova (PhoneGap)

### Xamarin

### Titanium

### Ionic

# WebRTC

# Prototype development

# Evaluation

## Setup

## Method

## Results

# Outlook

# Conclusion

List of tables

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

List of figures

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

List of listings

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

List of abbreviations

Bibliography