
 <p>European Commission Executive Agency for Small & Medium-sized Enterprises (EASME)</p>	Project ID 666438 <i>H2020 EASME</i>	Project duration: Apr 2015 – Apr 2017 http://www.doremir.com
	<p>Polyfoni-X-Score Automatic music transcription of polyphonic audio</p>  <p>by DoReMIR Music Research AB</p>	

<i>Document Identification</i>			
Deliverable ID:	D3.1	Deliverable title:	Cross platform framework selection report
Release number/date:		V1 / 14.10.2015	
Checked and released by:		Bengt Lidgard	

<i>Key Information from "Description of Work" (from the Contract)</i>	
Deliverable Description	Task 3.1: Cross platform framework selection (lead: DMIR) □ To enable the multitude of possible applications that may come from the requirements phase, it is necessary to develop a new flexible platform that is easy to adapt to the specific needs of various user groups. The task involves defining the core architecture, building blocks and methodologies to be used during the forthcoming development. Several cross platform mobile application development environments (such as Phonegap, Appcelerator, Xamarin, Appear IQ) will be reviewed, with the goal to achieve the best possible user experience whilst preserving platform flexibility and independency.
Dissemination Level	PU = Public
Deliverable Type	R = Report
Original due date (month number/date)	Month 05 / 28.08.2015

<i>Authorship & reviewer Information</i>	
Editor (person/ partner):	Sven Emtell, DoReMIR
Partners contributing	
Reviewed by (person/ partner)	Bengt Lidgard, DoReMIR

Release History

Release number	Date issued	Milestone*	Release description /changes made
0.1		TOC approved	
0.5	05.10.2015	Intermediate approved	first internal release
0.8		ER proposed	changes based on reviewer's feedback
1.0	14.10.2015	ER released	final version for approval by Coordinator before release to the EC

* The project uses a multi-stage internal review and release process, with defined milestones. Milestone names include abbreviations/terms as follows:

- TOC: Table of Contents (describes planned contents of different sections)
- Intermediate: Document is approximately 50% complete – review checkpoint
- ER: External Release (i.e. to commission and reviewers);
- proposed: Document author submit for internal review
- revised: Document author produce new version in response to comments
- approved: Internal project reviewer accept the document

Table of Contents

Release History	2
Table of Contents	3
1 Executive Summary	4
2 Background	5
3 Requirements	6
4 Cross platform.....	7
4.1 Prioritized Platforms	7
4.2 Native applications.....	7
4.2.1 Platforms	7
4.3 Native cross platform tools	8
4.3.1 Xamarin.....	8
4.3.2 Unity	8
4.3.3 Qt	8
4.3.4 Appcelerator / Titanium.....	8
4.3.5 React Native.....	9
4.4 Web and Hybrid applications.....	9
4.4.1 Platforms	9
4.4.2 Graphics	10
4.4.3 Audio and MIDI.....	10
4.5 Hybrid cross platform tools.....	10
4.5.1 Apache Cordova / PhoneGap.....	10
4.5.2 Meteor	10
5 Local vs Cloud functionality.....	12
5.1 Why local?	12
5.2 Why cloud?	12
5.3 What to deploy in the cloud.....	12
6 Look and feel	13
7 Table of functionality.....	14
8 Conclusion	15

1 Executive Summary

This report is the outcome of an investigation with the goal to find a cross platform framework. The framework should be suitable for future applications to be developed by DoReMIR and also with DoReMIRs legacy and knowledge in mind.

The result of the investigation is that DoReMIR should try developing Web apps for Mac OS X and Windows and Hybrid apps (web technology inside a native container) for mobile devices.

2 Background

The objective of WP3 is to develop a cross platform application (app).

The requirements of such an app are defined as part of WP1 and described in the report D1.2 - "System technical and functional specifications".

This report describes the results of trying to select a cross platform framework meeting these requirements.

3 Requirements

The report D1.2 requests that the applications to be developed in WP3 should run on a number of different platforms. As a consequence of this the main objective has been to find a cross platform framework which fulfills also the rest of the requirements from D1.2.

Here is a reworked list of these requirements:

- Audio and MIDI support
- Great User Interface
- Printing
- Sharing
- DAW Export
- Musical Intelligence
- Offline support
- Multiple users working in the same document

Of these, the "Audio and MIDI support" together with "Great User Interface" are the most important.

4 Cross platform

The main incentive for cross platform development is to reduce development costs. Cross platform development often comes with drawbacks regarding "look and feel" and "speed" due to platforms indifferences.

Nevertheless, our goal is to find a suitable cross platform solution where these drawbacks are not threatening usability.

4.1 Prioritized Platforms

Here is a list of prioritized platforms (highest priority first) according to the report D1.2 - "System technical and functional specifications".

1. iOS Tablet (iPad) - Native or Hybrid
2. Android Tablet - Native or Hybrid
3. Windows - Native or Web
4. Mac OS X - Native or Web
5. iOS Phone (iPhone) - Native or Hybrid, possibly with limited featureset
6. Android Phone - Native or Hybrid, possibly with limited featureset
7. Chrome OS (Chromebook) - Web or Hybrid

For each platform, implementation options are listed after the dash sign ("-").

The terms "Native", "Hybrid", and "Web" are described in the coming chapters.

4.2 Native applications

A "Native" application is compiled for a specific platform. On Chrome OS the "Native" concept is not useful since it only runs web applications.

4.2.1 Platforms

4.2.1.1 iOS

Traditionally, native iOS development is made using Objective C or Swift where the code is then compiled for the platform. The last years a number of new tools have emerged where you write code in other languages and the code is later (translated and) compiled for the platform.

4.2.1.2 Android

The usual way to develop for Android is to write Java code which is compiled into byte code run in the Dalvik VM. Here we refer to this as "Native" although it is actually not code compiled for the platform.

N.B. Parts of an Android app can be implemented using native code languages such as C and C++, see <https://developer.android.com/tools/sdk/ndk/index.html>

4.2.1.3 Windows

A number of different development tools and computer languages can be used to produce compiled code for Windows.

4.2.1.4 Mac OS X

A number of different development tools and computer languages can be used to produce compiled code for Mac OS X.

4.2.1.5 Chrome OS

Only web applications can be run on Chrome OS, but they can contain compiled parts, see <https://developer.chrome.com/native-client/overview>

4.3 Native cross platform tools

By "Native cross platform tools" we mean tools that can produce Native applications for a number of platforms. In 6.3.x you can find some of the most popular cross platform tools according to Vision Mobile's "Cross-Platform Tools 2015", <http://www.visionmobile.com/product/cross-platform-tools-2015/>: Xamarin, Unity, Qt, and Appcelerator/Titanium. You will also find the young challenger React Native.

These tools excel when you write a fully native app. It is possible to embed a Web View running a Web app in some of these tools, but if that is the objective, the tools listed in chapter 6.4 are better alternatives.

NOTE:

The most popular Cross-Platform Tool according to the Vison Mobile report is PhoneGap / Apache Cordova, but in this report it is listed as a tool for Hybrid app development, see chapter 6.4.

4.3.1 *Xamarin*

Xamarin can be used to build native iOS, Android and Windows Store apps. Code can also be shared between iOS and Mac OS X apps. The development language is C#, see <http://xamarin.com>.

4.3.2 *Unity*

Unity is claimed to be "The best development platform for creating games". It supports the many needs of game development companies with features as multi-player interaction, recording and replays (to make others watch your playing), community, advertisements etc. Unity can be used for non-game apps too but where it really excels is for apps having extensive use of 2D (two-dimensional) and 3D (three-dimensional) graphics.

Examples of non-game apps developed using Unity are physics simulations and virtual museums. Another interesting example is Physynth which is based on Unity's physics simulators, <http://simiansquared.com/physynth/>.

Here is a report on development of non-game apps in Unity, <https://medium.com/@raquezha/unity-3d-not-just-games-aef911e3314c>

See <http://unity3d.com/unity> for more info on Unity.

4.3.3 *Qt*

Qt is a framework for creating cross platforms apps. The main language used is C++, but other languages can also be used via language bindings. Qt also has a proprietary declarative scripting language called QML. See <http://www.qt.io>.

4.3.4 *Appcelerator / Titanium*

Using Appcelerator and it's Open Source SDK Titanium you can build native apps on iOS and Android. The language used is JavaScript. See <http://www.appcelerator.com>.

4.3.5 *React Native*

React Native is a Facebook initiative letting you build native apps on iOS and Android. All code is written in JavaScript. React Native was released in march 2015 so it hasn't been out for long. See <https://facebook.github.io/react-native/>.

4.4 Web and Hybrid applications

A Web application is built using HTML, JavaScript, and CSS and runs in a web browser. A Web application is inherently cross platform since web browsers exist on most platforms.

A web browser (and the Web apps running in it) has limited access to the device/computer it is running on. A Web app may need access to the camera, the microphone, the disk drive etc but it is not obvious that the Web app is entitled to access them. Sometimes web technology makes it possible to directly access the device, but sometimes it is necessary to run the Web application in a web view inside a Native container which has the appropriate access rights. A Web app running in a web view inside a Native app, we call a Hybrid app.

There may be other reasons to make parts of a Web app native too. Examples are introductory screens, login screen etc.

4.4.1 *Platforms*

4.4.1.1 **iOS**

The usual way to deploy apps on iOS is via the App Store. To accomplish this for a web app, it can be run in a Web View inside a native app. Apache Cordova/Phonegap makes it possible to access native device functionality such as the camera from JavaScript.

4.4.1.2 **Android**

Just as on iOS, a web app can be run inside of a native container on Android. Apache Cordova/Phonegap makes it possible to access native device functionality such as the camera from JavaScript.

4.4.1.3 **Windows**

Web apps are usually run directly in a Web browser on Windows. Nevertheless Apache Cordova/Phonegap makes it possible to access native device functionality such as the camera from JavaScript. Windows 10 also introduces the possibility to create a Windows app from a Web app, see <https://blogs.windows.com/buildingapps/2015/03/02/a-first-look-at-the-windows-10-universal-app-platform/>

4.4.1.4 **Mac OS X**

To deploy a web app on the Mac App Store, the app could be run inside a web view of a native app in a similar way as on iOS. On Mac OS X there are less reasons to do this though so it is more common to just run the web app "as is" in a Web Browser.

4.4.1.5 **Chrome OS**

Only web apps can be run on Chrome OS, but they can contain compiled parts, see <https://developer.chrome.com/native-client/overview>

Chrome OS also makes it possible to package a Web app to make it look more like a desktop app running in it's own window etc.

4.4.2 Graphics

Usability is extremely important. For a web app to be useful, there must be almost no difference in "look and feel" from a native app. This is especially important on touch interfaces like tablets and phones. Computers and devices are getting faster for every new hardware release and at the same time improvements in software are made. One example is that the WKWebView introduced in iOS 8 and Mac OS X 10 is considerably faster than its predecessors UIWebView (iOS) and WebView (Mac OS X). With iOS 9 Apple introduced SFSafariViewController, which can give an even better user experience sharing cookies, autofill etc.

There are a number of UI frameworks for mobile web apps and they are usually part of or easily integrated with the web frameworks mentioned in 6.5

4.4.3 Audio and MIDI

WebAudio and WebMIDI are exciting technologies bringing audio and MIDI to web apps. They seem to have a good feature set but questions remain regarding performance and browser support, especially on mobile devices.

Things we need to further investigate:

- WebMIDI is not supported on any browser in iOS. Can MIDI input/output be handled natively in a Hybrid app while other things like a sample synth is running in WebAudio? Can we get latency short enough for monitoring while playing?
- Is WebAudio performance good enough on Tablets? How many simultaneous plain audio tracks can be played? How many simultaneous tracks can be played with sample player? Do sample player tracks have to be translated to plain audio files to get good enough performance?
- Microphone input is not possible in WebAudio on iOS. How can we best record using native functions? Do we need direct interaction between the native container and the WebView or is it enough to have the native container interact with the WebView via the cloud?

4.5 Hybrid cross platform tools

The number of web frameworks which can be used either alone or in combination with others to produce Hybrid apps is overwhelming.

They can produce Hybrid apps, where all or part of the app is a Web View displaying content (HTML, JavaScript, and CSS) loaded either locally from the app itself or from the web.

Some web frameworks we have looked at are: Interact.js, Ionic, jQuery Mobile, React and Sencha Touch.

The two we found most interesting are listed below.

4.5.1 Apache Cordova / PhoneGap

Apache Cordova and it's commercial sibling PhoneGap Apache Cordova is a set of device APIs that allow a mobile app developer to access native device functions such as the camera or accelerometer from JavaScript. It can also be used to package apps in native containers for distribution via App stores. With Apache Cordova / PhoneGap you usually implement a Web app as a thin native container with a single Web View where all the user interaction happens. As an alternative you can mix a Web View with native components.

4.5.2 Meteor

Meteor is a JavaScript app platform, offering a complete full-stack framework for delivering web and mobile apps entirely in JavaScript. It has a number of great features like:

-
- *Real-time.* Real-time bidirectional communication using Web Sockets between the client and the server
 - *Data on the Wire.* Meteor doesn't send HTML over the network. The server sends data and lets the client render it.
 - *One Language.* Meteor lets you write both the client and the server parts of your application in JavaScript.
 - *Database Everywhere.* You can use the same methods to access your database from the client or the server.
 - *Latency Compensation.* On the client, Meteor prefetches data and simulates models to make it look like server method calls return instantly.
 - *Full Stack Reactivity.* In Meteor, realtime is the default. All layers, from database to template, update themselves automatically when necessary. This means that synchronization between multiple devices is built-in.
 - *User accounts and sessions.* User accounts and sessions are available out-of-the-box.

5 Local vs Cloud functionality

Application functionality can be deployed locally in each client or centrally in the cloud. There are pros and cons for each choice.

5.1 Why local?

- usually faster for the end user
- less load on cloud servers

5.2 Why cloud?

- more secure (harder to reverse engineer)
- easier to upgrade (all users use the same version - upgrade once for all users)
- no need to rewrite legacy code

5.3 What to deploy in the cloud

Our initial plan is to deploy the following components in the cloud.

- monophonic audio analysis
- polyphonic audio analysis

6 Look and feel

A native app usually tries to blend in by using the same type of UI Elements (menus, icons, buttons etc) as the hosting operating system.

Web apps on the other hand stem from the rich flora of home pages, where individuality frequently has been a goal. Therefore Web apps often have their own idiom when it comes to color and form. Lately as Web apps due to new web technologies have been more capable, some more tool like apps strive to mimic the UI Elements of the hosting operating system.

7 Table of functionality

This table maps the requirements from chapter 3 to the native frameworks from chapters 4.3.x and Web and Hybrid technology respectively.

	Appcelerator	Qt	React Native	Unity	Xamarin	Web	Hybrid
Audio and MIDI	x ¹	x ¹	x ¹	x ²	x ¹	x ³	x ¹
Great User Interface	x	x	x	x ⁴	x	x	x
Printing	x	x	x	x	x	x	x
Sharing	x ⁵	x ⁵	x ⁵	x ⁵	x ⁵	x	x
DAW Export	x	x	x	x	x	x	x
Musical Intelligence	x ⁶	x ⁶	x ⁶	x ⁶	x ⁶	x ⁶	x ⁶
Offline support	x	x	x	x	x	-	x ⁷
Multiple users working in the same document	x	x	x	x	x	x ⁸	x ⁸
Current staff competence	-	-	-	-	-	x	x
Making sure all users are on the same version	x ⁹	x ⁹	x ⁹	x ⁹	x ⁹	x ¹⁰	x ¹⁰

¹ = can use the native capabilities of the platform

² = no built-in midi support (but can probably be developed as a PlugIn)

³ = Support for Web Audio and Web MIDI depends on browser. Web Audio support exists in many browsers while Web MIDI support only exists in a few. The most obvious shortcomings are that Web MIDI and Microphone input is not supported on any iOS browsers. See the following two links.

Web Audio API: <http://caniuse.com/#search=web%20audio>

Web MIDI API: <http://caniuse.com/#search=web%20midi>

⁴ = Unity has great 2D and 3D graphic capabilities

⁵ = Needs some kind of web presence when sharing on Facebook etc

⁶ = Depending on feature it may be implemented locally or in the cloud. If implemented locally the implementation may vary.

⁷ = Web technology is by design not offline, but a hybrid app has access to a native container where it can be implemented.

⁸ = Using Meteor as we plan to do will greatly simplify synchronization

⁹ = Forced upgrade to a new version can be built but needs user interaction

¹⁰ = By design, since client code is loaded from the server every time the app is run. Not available in offline mode.

8 Conclusion

In September 2012, Facebook's Mark Zuckerberg said:

- "The biggest mistake we've made as a company is betting on HTML5 over native."

This related to Facebook's unsuccessful mobile strategy where clients were made using HTML5.

Since then the speedup of hardware is manifold and the progress in Web technologies has been extremely fast.

There are also a number of strong reasons to use cross platform Web technology instead of cross platform Native technology:

- We know Web technology and can therefore get a flying start
- We can reuse more code
- It's easier to deploy updates
- Developing back- and front-end using the same language is a great thing

A conservative person might say -"It can't be done, look at Facebook.", but products from companies like SoundTrap and Hansoft and the showcase <http://midwinterlightup.com> and others at <http://audiocrawl.co> show something very promising, regarding user experience and WebAudio.

Our take (and hope) is that Web/Hybrid technology is ready for prime time in products like the ones we are aiming at!

The plan is to develop Web apps for Mac OS X and Windows and Hybrid apps for mobile devices.

Two technologies we believe in are Apache Cordova / PhoneGap (see chapter 4.5.1) enabling us to access native functionality from Web apps on mobile devices, and Meteor which gives us a platform with a lot of useful and leading edge functionality out-of-the-box (see chapter 4.5.2).

Our next step is to start implementing a concept app where we use Apache Cordova / PhoneGap and Meteor. During this quest we'll find out if we are on the right track!