Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Master in Informatics and Systems

Internship/Industrial Project

Final Report

Mobile Web Applications

Stanislava Nedyalkova

Thesis Supervisor:

Professor Viriato Marques

Instituto Superior de Engenharia de Coimbra

Coimbra, September, 2013

Acknowledgements

I would like to express my gratitude to my supervisor Aurélio Santos for his total dedication and contribution to my internship. His orientation and help have greatest impact on the success of the project.

Furthermore, my special thanks go to Victor Batista and Samuel Santos for their attention and decisions throughout the entire course of the internship, Avelino Martins for his collaboration for the Tizen push service implementation and Ricardo Silva for the iOS development for PhoneGap and Titanium.

In addition, I would like to thank Paulo Martins and all Present Technologies employees whose warm welcome made my integration into the company effortless.

Last but not least, I would like to acknowledge with much appreciation my thesis supervisor from Departamento de Engenharia Informática e de Sistemas, Professor Viriato Marques, for his availability, advices and understanding.

Abstract

This document presents the work that was elaborated at the company Present Technologies as part of the academic discipline Internship/Industrial Project for the Master's degree in Informatics and Systems, Software Development branch, at Instituto Superior de Engenharia de Coimbra.

The area of the mobile web applications has grown exponentially over the last few years turning it into a very dynamic field where new development platforms and frameworks are constantly emerging. Thus, the internship consisted in the study of two new mobile operating systems, Tizen and Firefox OS, as well as two frameworks for packaging of mobile web applications – Adobe PhoneGap and Appcelerator Titanium. These platforms are in the direct interest of Present Technology since it pretends to use them in its future projects in general and in the Phune Gaming project in particular. Since Television is one of the Present Technologies' business areas, during the course of the internship it was decided to perform additionally a study of two Smart TV platforms, namely Samsung Smart TV and Opera TV, which was considered as a valuable knowledge for the company.

For each of the platforms was performed a study about its architecture, supported standards and the development tools that are provided, nevertheless the focus was on the applications and for this reason a practical case study was conducted. The case studies consisted in the creation of a prototype or packaging of an application, for the case of the packaging tools, in order to prove the feasibility of the applications for the Present Technologies' needs.

The outcome of the work performed during the internship is that it raised the awareness of Present Technology of the studied platforms, providing it with prototypes and written documentation for the platforms' successful usage in future projects.

Keywords (Subject): Mobile Web Applications, Mobile Operating Systems, Packaging Tools, Smart TV

Keywords (Technology): Tizen, Firefox OS, PhoneGap, Apache Cordova, Titanium, Samsung Smart TV, Opera TV

Resumo

Este documento apresenta o trabalho que foi elaborado na empresa Present Technologies no âmbito da disciplina Estágio/Projecto Industrial do Mestrado em Informática e Sistemas, ramo de Desenvolvimento de Software, do Instituto Superior de Engenharia de Coimbra.

A área das aplicações web mobile tem crescido exponencialmente nos últimos anos, transformando-a numa área muito dinâmica onde novas plataformas e *frameworks* de desenvolvimento surgem constantemente. Assim, o estágio consistiu no estudo de dois novos sistemas operativos móveis, Tizen e Firefox OS, bem como duas ferramentas de *packaging* de aplicações web mobile – Adobe PhoneGap e Appcelerator Titanium. Estas plataformas são do interesse direto da Present Technologies, uma vez que esta pretende usá-los nos seus projetos futuros, em geral, e no projeto Phune Gaming em particular. Visto que a Televisão é uma das áreas de negócios da Present Technologies, no decorrer do estágio, decidiu-se realizar também um estudo sobre duas plataformas de *Smart* TV, nomeadamente Samsung Smart TV e Opera TV, que foi considerado como um conhecimento valioso para a empresa.

Para cada uma das plataformas estudadas foi realizado um estudo da sua arquitetura, *standards* suportados e as ferramentas de desenvolvimento que são fornecidas, no entanto, o foco era nas aplicações e por este motivo foi realizado um caso de estudo. Os casos de estudo consistiram na criação de um protótipo ou *packaging* de uma aplicação, para o caso das ferramentas de *packaging*, a fim de comprovar a viabilidade das aplicações para as necessidades da Present Technologies.

O resultado do trabalho realizado durante o estágio é que ele aumentou o conhecimento da Present Technologies sobre as plataformas estudadas, fornecendo-lhe com protótipos e documentação escrita para o uso bem sucedido das plataformas em projetos futuros.

Palavras Chave (Tema): Aplicações Web Mobile, Sistemas Operativos Móveis, Ferramentas de *Packaging*, *Smart* TV

Palavras Chave (Tecnologia): Tizen, Firefox OS, PhoneGap, Apache Cordova, Titanium, Samsung Smart TV, Opera TV

Table of Contents

Table of Figures

Table of Tables

Acronyms and Abbreviations

3D	Three-dimensional
3G	3rd Generation of Mobile Telecommunications Technology
ADT	Android Development Tools
AJAX	Asynchronous JavaScript and XML
AMD	Asynchronous Module Definition
API	Application Programming Interface
ASF	Apache Software Foundation
BOSH	Bidirectional-streams Over Synchronous HTTP
CDMA	Code Division Multiple Access
CDT	C/C++ Development Tools
CE	Consumer Electronics
CLI	Command Line Interface
CORS	Cross-Origin Resource Sharing
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DOM	Document Object Model
DTV	Digital Television
DVD	Digital Versatile Disc / Digital Video Disc
EFL	Enlightenment Foundation Library
EPG	Electronic Program Guide
FM	Frequency Modulation
GPL	General Public License
GPS	Global Positioning System
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HbbTV	Hybrid Broadcast Broadband TV
HD	High-Definition
HDMI	High-Definition Multimedia Interface
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
I/O	Input/Output
ID	Identification
IDE	Integrated Development Environment
IE	Internet Explorer
IME	Input Method Editor
IP	Internet Protocol
IRC	Internet Relay Chat
IRI	Internationalized Resource Identifier
ISF	Input Service Framework
ISO	International Organization for Standardization
IT	Information Technology
IVI	In-Vehicle Infotainment
JDK	Java Development Kit
JPEG/JPG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JS	JavaScript
JSDT	JavaScript Development Tools
JSON	JavaScript Object Notation
kB	kilobyte
LAN	Local Area Network
LBS	Location-Based Services
Maple	Markup Engine Platform for Embedded Systems
MB	Megabyte
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusetts Institute of Technology
MMC	Multi Media Card
MMS	Multimedia Messaging Service
MO	Mobile Origination
MT	Mobile Termination

MVC	Model-View-Controller
NDEF	NFC Data Exchange Format
NDK	Native Development Kit
NFC	Near Field Communication
OBS	Open Build Service
OEM	Original Equipment Manufacturer
OpenGL ES	Open Graphics Library for Embedded Systems
OS	Operating System
OWA	Open Web App
PDA	Personal Digital Assistants
PIM	Personal Information Management
PNG	Portable Network Graphics
PTECH	Present Technologies, Lda.
RAM	Random Access Memory
REST	Representational State Transfer
RSS	Rich Site Summary
SASL	Simple Authentication and Security Layer
SD	Secure Digital
SDB	Smart Development Bridge
SDK	Software Development Kit
SEF	Service Extension Framework
SIM	Subscriber Identity Module / Subscriber Identification Module
SLP	Samsung Linux Platform
SMACK	Simplified Mandatory Access Control Kernel
SMS	Short Message Service
SP	Service Pack
SSDP	Simple Service Discovery Protocol
SSL	Secure Sockets Layer
SSO	Single Sign-On
SSTV	Samsung Smart TV
ST	Search Target

STB	Set-Top Box
SVG	Scalable Vector Graphics
SWF	ShockWave Flash / Small Web Format
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TV	Television
TXT	Text file
UCS	Universal Character Set
UDP	User Datagram Protocol
UI	User Interface
UMTS	Universal Mobile Telecommunications System
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF-8	UCS Transformation Format—8-bit
VoIP	Voice Over IP
VT-x	Intel Virtualization Technology
W3C	World Wide Web Consortium
WebGL	Web Graphics Library
WOFF	Web Open Font Format
WPS	Wi-Fi Positioning System
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Glossary

API	Stands for “Application Programming Interface”. It represents a set of functions and protocols, which define how software components can interact.
Client application	Applications that communicate with another application in a central point called the server application, in order to use its services.
Cloud	An Internet-based computing architecture, which provides remote data storage and other computing services and resources.
Device API	An API which allows web applications to interact with the device hardware.
Framework	A set of software libraries and additional applications that can be used in applications.
Platform	Can have multiple meanings. In this report it is primary used to represent an abstract layer on which a given application runs, yet sometimes is used interchangeably with framework.
Set-top box	An electronic device, which receives digital signals and decodes them in order to be viewed on a television (TV) set. The signals can be from the TV broadcast or Internet data.
Smart TV	A TV technology, which besides the programming from the TV broadcast, provides Internet access and can run applications. Hence, it turns the TV devices into more interactive devices for the users.
Smartphone	A cellular phone, which besides the phone calls and text messaging capabilities, has Internet access and can run mobile applications. Thus, it can be considered a mobile personal computer.
Tablet	A mobile device that has Internet access and can run mobile applications. Its main input is from a touchscreen.
W3C widget package	A package, which contains all necessary files of a web application, therefore it is a complete standalone web application that does not depend on external resources.

1. Introduction

This chapter presents the description of the internship exposing its initial proposal, subsequent changes, objectives and motivation. Also presented is the structure of the report.

1.1 Internship Description

The internship was performed within the scope of the academic discipline Internship/ Industrial Project for the Master's degree in Informatics and Systems, Software Development branch, at Instituto Superior de Engenharia de Coimbra under the supervision of Professor Viriato Marques. The studies and case studies were conducted on the premises of the company Present Technologies (PTECH) in Coimbra under the supervision of Software Engineer Aurélio Santos.

The internship began on January 2nd, 2013. It was on a full-time basis for the period of seven months and thus it came to an end on July 31st, 2013. The monitoring of the internship was carried out through regular meetings and weekly progress reports.

The work performed during the internship consisted in studies of platforms for mobile web applications development, thus claiming at the end of the internship PTECH to have better understanding and knowledge of the studied platforms. Nevertheless, due to the dynamic changes in the mobile applications area and the PTECH's business strategy and philosophy to work with emergent technologies, there were several changes to the initial proposal for the internship. The initial proposal and subsequent changes are discussed in the following subsections.

1.1.1 Initial Proposal

The initial proposal provided in Appendix 8.1 consisted of the fulfillment of three main tasks as follows:

- **T1 – Mobile Operating Systems:** This task aimed at performing a study of a new mobile operating system (OS) called Tizen. The deliverables at the end of the task included a prototype for a Tizen web application and a document that describes Tizen, its architecture and application development.

- **T2 – Web Applications Packaging Tools:** This task consisted of study, evaluation and comparison of five tools for web applications packaging, namely PhoneGap, Appcelerator Titanium, Sencha Touch, Qt and Rhodes. These tools support various mobile OS's, therefore to fulfill T2 an existing application had to be packaged by using each one of the five tools, in order to test its support for the respective mobile OS. The task required elaboration of a document describing the details of each tool along with a comparison between all of them.
- **T3 – Mobile Web Frameworks:** T3 involved the study, evaluation and comparison of five frameworks for mobile application development, as follows: jQuery Mobile, jQTouch, Sencha Touch, Jo and Yiibu. The requirements for this task included development of prototypes using each one of the frameworks and a document with the frameworks' details and comparison between them.

1.1.2 Changes to the Initial Proposal

PTECH intends to start developing Tizen applications using functionality such as push notifications and the Facebook Chat service; however, due to some limitations explained in detail in Appendix 9.1.2, this functionality could be implemented only using native code. Therefore, for T1 besides the prototype for Tizen web application, two more prototypes for Tizen native applications had to be created as prove of concept, in order to demonstrate the feasibility of these applications. Additionally, for this task it was defined to conduct a study of another emergent mobile OS – Firefox OS. The Firefox OS applications are completely web and thus this new platform could be of interest to the company. To determine this it was decided to develop a prototype for Firefox OS application and elaborate a document describing the details of the Firefox OS architecture and application development. Nevertheless, for this task the Tizen study remained as major focus.

Taking into consideration the company's needs and the scope of the internship, PTECH decided to exclude Sencha Touch, Qt and Rhodes from T2. The reason for this decision was based on an evaluation research which concluded that Sencha Touch is more limited than PhoneGap in terms of supported platforms (supports only Android and iOS) and device functionality. Qt, on the other hand, seems to be an established framework for developing applications written in C++, yet its support for web applications is recent.

The Qt's support for mobile devices that PTECH is interested in will be available in a later version, thus it was not possible to analyze them during this internship. Finally, Rhodes provides an extensive set of device functionality along with good support for mobile OS's; however, it has dependencies on Ruby, which is not used in the company. This is why it was decided to focus only on the studies of PhoneGap and Titanium.

Due to the natural evolution of the PTECH's objectives and needs, it determined instead of performing the study of the mobile web frameworks from T3, to conduct studies of two Smart TV platforms, namely Samsung Smart TV (SSTV) and Opera TV. The reason for this decision was based on the fact that the mobile web frameworks from T3 became less relevant to PTECH compared to the area of Smart TV, where the company wants to concentrate. There are still many doubts if any of these mobile web frameworks will be of some use to the company, since the design of the PTECH's applications is always developed by the design team. However, such frameworks make it difficult to apply different themes. Instead, they are good for creating applications using patterns, yet PTECH rarely develops this kind of applications. Hence, the focus of this task was on SSTV and its study included description of the architecture and application development and creation of two prototypes. The study of Opera TV required elaboration of a document containing introduction to the Opera TV platform and details about the Opera TV Store and Opera TV Store applications. The practical part consisted of adapting an existing application to the platform.

1.2 Objectives and Motivation

After the changes the internship had the following objectives:

- Study of two new mobile OS's – Tizen and Firefox OS;
- Study and comparison of two web applications packaging tools – PhoneGap and Appcelerator Titanium;
- Study of two Smart TV platforms – SSTV and Opera TV.

The motivation behind these objectives was the PTECH's business strategy and need to acquire knowledge of the studied platforms so they can be used in its projects. To further understand the need of conducting the studies, following is a brief description of the company.

PTECH is a Portuguese IT company founded in 2000. Its mission as stated on the company's web site[1] is "research and development of innovative services and applications for the worldwide market, using emergent and state-of-the-art technologies". PTECH covers four key business areas as follows: Mobile Solutions; Internet Services and Applications; Enterprise Applications and Television. Regarding the Mobile Solutions area, the company is already competent in many mobile platforms, such as Android, iOS, Symbian, Windows Phone to name a few, and is willing to spread its knowledge with new emergent platforms and frameworks. In the foreseeable future the company's strategy is to focus on the Smart TV area.

Currently, PTECH is developing a multiplayer gaming platform called Phune Gaming that allows casual games to be played online against real users. Phune Gaming is intended to be used on various device categories including mobile devices, desktop and Smart TV. At this moment the platform is targeted at Android and iOS; however, it is contemplated to support more platforms. Therefore, the studies of the various platforms during the internship allowed PTECH to gain more knowledge supported with practical experience and documentation, thus it can use them in its projects and especially in Phune Gaming.

1.3 Report Structure

The report is organized in the following order:

- Chapter 1, Introduction, describes the internship, its initial proposal and changes and, in addition, the objectives it aims to achieve and the motivation behind them.
- Chapter 2, Background, presents brief background information about the Mobile and Smart TV areas in order to introduce the reader to the technologies that are discussed in the following chapters.
- Chapter 3, Mobile Operating Systems, reports the studies of Tizen and Firefox OS, including their overviews, architectures, supported APIs and the tools that are available for the application development. Also described are the applications that each platform supports. The chapter additionally provides an introduction to

[1] http://www.present-technologies.com/profile.jsp

the case studies that were conducted for the two mobile OS's and the conclusions of the studies.

- Chapter 4, Web Applications Packaging Tools, presents the details of the two web applications packaging tools – PhoneGap and Titanium. The PhoneGap study includes: the PhoneGap overview and history, its comparison with Apache Cordova, the additional services provided by PhoneGap and the description of the PhoneGap case study. The Titanium study, by contrast, presents the Titanium architecture, supported application types and their specific features and introduction to the Titanium case study. The chapter further contains a comparison of the APIs of the two packaging tools, and the conclusions section draws the analogy between PhoneGap and Titanium.
- Chapter 5, Smart TV, reports the studies that were performed for SSTV and Opera TV. For SSTV are presented the platform's architecture, supported APIs and the contents of the latest Software Development Kit (SDK). Also described are the SSTV applications, their possible display types and contents, and information about the application testing and publishing. The SSTV study concludes with a description of the SSTV case studies and final conclusions. Regarding Opera TV its section introduces the overview, architecture, supported web standards and available tools. Furthermore, the Opera TV study presents the Opera TV Store, its architecture and applications. Finally are the description of the Opera TV case studies and the conclusions of the study.
- Chapter 6, Conclusions, summarizes the work that was performed during the internship exposing its achievements, limitations and difficulties. The chapter concludes with an outline for the future work.
- Chapter 7, Bibliography, makes available the bibliography and references that were used during the elaboration of the report.
- Chapter 8, Appendices, presents the report's publicly available appendices.
- Chapter 9, Confidential Appendices, contains the confidential appendices.

2. Background

This chapter presents some background information in order to introduce the reader to the technologies that were employed during the internship. For the sake of good organization and readability the chapter is divided in two sections – Mobile and Television. Nevertheless, it should be noted that the applications that were used in the Smart TV case studies are mobile web applications that were repurposed to be displayed on a TV screen and controlled by the TV controller as its main input interface.

2.1 Mobile

Over the last two decades mobile devices have undergone significant changes in their sizes and available functionality. Thus, the today's smartphones are devices that combine the functionality of yesterday's mobile phones and personal digital assistants (PDA). This gives the users opportunity to use the mobile phone's voice calling and text messaging and beyond this to be able to manage their personal information, like contacts, notes, and calendar and at the same time have Internet access, features typical for PDAs. Some of the functions that a smartphone offers are: media players, web browser, digital camera, GPS navigation, high-resolution touch screen, sensors and many others.

A mobile device is operated by a mobile OS, which typically consist of five functional layers that from bottom to top are described as follows [89]:

- Kernel – the lowest layer is composed of hardware drivers, file system, memory and process management;
- Middleware – this layer includes libraries for device management, communication and messaging engines, multimedia codecs, web page rendering engines, security subsystems, among others;
- Application execution environment – the layer consists of various components for application management and APIs for implementing device functionality in applications;
- User Interface (UI) framework – represents a set of Graphical User Interface (GUI) components that are specific to the mobile OS, such as buttons, input fields, tab bars, dialog boxes, menus, and many more;

- Application suite – makes available a set of built-in applications, for example contacts, photo gallery, calendar, browser and messages.

Alongside the application suite other applications (sometimes referred in the texts as apps) developed by third party developers can be installed. Thus, the mobile OS provides these applications with access to the device resources and user data, such as sensors, camera, battery, geo location, contacts, among others.

Nowadays, there are numerous mobile OS's in the market. Table 1 summarizes the mobile OS's that were used in the case studies conducted during the internship. Also presented in the table are the main characteristics of each OS.

Table 1 Mobile OS's overview

OS	Principal Developer	Source model	Programming Language	Tools	Package format	App Store
Android[2]	Google	Open source	Java, C++	Android SDK, NDK	apk	Google Play
BlackBerry[3]	BlackBerry	Closed and proprietary	Java, Web	BlackBerry, NDK BlackBerry WebWorks	cod, bar	BlackBerry App World
iOS[4]	Apple Inc.	Closed and proprietary	Objective C	iOS SDK and Xcode	ipa	App Store
Firefox OS[5]	Mozilla	Open source	Web	Simulator, Remote Debugger	zip	Firefox Marketplace, and others
Tizen[6]	Samsung, Intel	Open with proprietary components	Web, C++	Tizen SDK	wgt, tpk	Tizen Store
Windows Phone[7]	Microsoft	Closed and proprietary	C#	Windows Phone SDK	xap	Windows Phone Store

As can be seen in the table the different mobile OS's use specific programming languages for the app development. Thus, the apps that are developed for a particular OS

[2] http://www.android.com/
[3] http://us.blackberry.com/
[4] http://www.apple.com/ios/
[5] http://www.mozilla.org/en-US/firefox/os/
[6] https://www.tizen.org/
[7] http://www.windowsphone.com/en-us

are known as native apps. They are normally installed as a binary executable file (consider Java, Objective C, C++ and C#) and when they are launched they interact directly with the OS, therefore they have full direct access to the mobile OS APIs. Another type of mobile apps exists, whose apps are created with web technologies (e.g. HTML, CSS, and JavaScript) and hosted on a remote server; thus they are called web apps. The web apps run within the browser of the device and therefore they have access only to a limited set of device or OS-specific APIs, which are exposed by the browser. The benefits of the mobile web applications are that they do not require downloading from the app store and installing the app and they provide instant application updates for all users; yet they need a constant network connection.

There is third type of applications called hybrid apps, which are a combination of native and web parts. The application logic is usually written entirely with web technologies and the native code is used to create a container which is a web browser view that displays the contents of the web app. A web browser view, referred to as webview in the text, is a native UI component that renders web pages similarly to a regular web browser; however, unlike the regular browser it does not have navigation controls. The webview has a full access to the OS's APIs and thus it represents a bridge between the device API's and the web app. Developers can develop this bridge themselves or instead can use a framework, referred to as packaging tool in this report, such as PhoneGap or Titanium.

The difference between these three types of mobile applications in terms of interaction with the device APIs is illustrated in **Figure 1**[8].

[8] Image courtesy of [48]

Figure 1 Mobile app types

In order to prepare an application for deployment and distribution its contents have to be packaged in an archive file with specific file format. The packaging of a web application usually includes only the creation of an archive file with all source files and other resources, for example images, audio and configuration files. The native applications packaging, by contrast, implies the compilation and sometimes linking of the source code in order to generate a binary executable file.

After the apps are packaged they can be distributed on the mobile OS respective online app store.

2.2 Television

Smart TV, also known as connected TV, is a technology that emerged in 2010 and provides the following key features: full Internet access, ability to install applications and ability to connect with other devices on the network such as desktop computers, smartphones and tablets. The Internet connection allows users to surf the Web right from their connected TV sets. Besides the Internet navigation, the application support brings the functionality a smartphone user is already familiar with to the TV screen. However, since the TV is usually placed in the living room it is more social device than a smartphone, and this should be considered in applications when sensitive information has to be entered.

With Smart TV the TV no longer provides only passive programming. It allows interaction and thus the viewer is much more engaged with the content. Some modern

Smart TV sets have integrated camera, microphone, voice and motion sensors, therefore users can have video calls or control the device via voice and motion commands.

A Smart TV normally has available an SDK, tools and an app store and thus provides a platform for application developers to develop and distribute their apps. The apps are web applications written in HTML5 and related web technologies. The HTML5 standard enables some benefits for the TV consumers such as [33]:

- Web contents can be stored locally which enables apps to remain functional even without Internet connection;
- The WebSockets technology provides a full communication channel between the Smart TV device and the server. This is beneficial for real-time delivery, such as weather updates and live sports results;
- The HTML5 audio and video tags do not require plugins to launch audio and video and this allows to provide almost identical experience on mobile and TV environment;
- The WebGL standard allows creation of 3D graphics for more compelling contents.

The Smart TV technology can be incorporated in TV sets, set-top boxes (STB), Blu-ray players, game consoles and other players. Some Original Equipment Manufacturers (OEMs) that unveiled their Smart TV devices are: Samsung, LG, Sony, Toshiba, Philips, Panasonic and Vizio. Nevertheless, unlike the Mobile area in the Smart TV area no clear leaders exist thus far.

3. Mobile Operating Systems

This chapter presents the studies of the mobile OS's Tizen and Firefox OS. Since PTECH already intends to develop applications for Tizen, the focus of the chapter is on Tizen while the Firefox OS study aimed at familiarizing the company with the new OS.

3.1 Tizen

This section exposes the Tizen OS. It should be noted that Tizen is a new platform that is still under development and as result it undergoes various changes and updates. The study was performed in January 2013; therefore the information in the text presents the state of the platform at the time of the study. However, due to the emerging needs of PTECH, later on the study was reestablished and the Sections: 3.1.2.1 Architecture; 3.1.2.2 API, 3.1.3.3 Native Applications, 3.1.4 Tizen Case Study and 3.1.5 Tizen Conclusions were updated corresponding to the changes in the Tizen SDK as of April 2013.

3.1.1 Tizen Overview

Tizen (pronounced Tie Zen) is a new open source standards-based mobile OS that supports multiple device categories and provides an environment for developing applications for end users. The Tizen Project is hosted by the Linux Foundation[9] and its engineering governance is provided by the Technical Steering Group composed of Intel and Samsung. The focus of the Technical Steering Group is on the platform development and delivery, while the Tizen Association[10] is responsible for the Tizen's in-market support and industry presence. The current members of the Tizen Association are: Huawei, Intel, NEC Casio, Fujitsu, NTT DOCOMO, Orange, Panasonic, Samsung, SK Telecom, Sprint, Vodafone and KT.

[9] http://www.linuxfoundation.org/
[10] http://www.tizenassociation.org/

3.1.1.1 Tizen Objectives

The main objectives of Tizen are as follows:

- Tizen is intended to serve the industry as a strong independent device platform that is supported by a collaborative governance structure. The Tizen Project is hosted by the Linux Foundation which has a great influence and ability to attract companies to the project.
- The Tizen Project aims to build a completely open OS, from the core, up through the core applications and user interfaces. Currently, there are several other OS's that claim to be open; however they are normally controlled by a single entity and some of their components are proprietary.
- Tizen is designed to support a variety of devices such as smartphones, tablets, notebooks, Smart TVs, and In-Vehicle Infotainment (IVI) systems.
- The Tizen development will be completely transparent and rely on application developers' feedback in order to improve the platform.

3.1.1.2 Tizen History

In February 2010 during the Mobile World Congress, Intel and Nokia announced MeeGo, a Linux-based free OS that was designed to target a variety of devices such as smartphones, notebooks, Smart TVs, and IVI systems. The aim of MeeGo was to merge the efforts of Intel and Nokia on their former projects Moblin and Maemo, respectively, into one new common project [54]. Subsequently, Nokia announced their strategy to use Windows Phone for future smartphones and abandoned MeeGo; however they continued developing MeeGo/Harmattan for the smartphone device Nokia N9. Intel collaborated with Samsung on a new mobile OS, Tizen, which would be based on Samsung Linux Platform (SLP), a platform that Samsung previously provided to the LiMo Foundation [41]. In September 2011, Intel announced that MeeGo would be transitioning to Tizen in 2012 and in January 2012 the first Tizen source code and SDK preview were made available.

In **Figure 2**[11] is presented the family tree of Tizen. It illustrates the Tizen's parentage from components of MeeGo, SLP and the Samsung's Bada platform. Mer[12] is an open, mobile-oriented software distribution that is aimed at device manufacturers. It is based on the work from MeeGo and plans to share effort with the Tizen project.

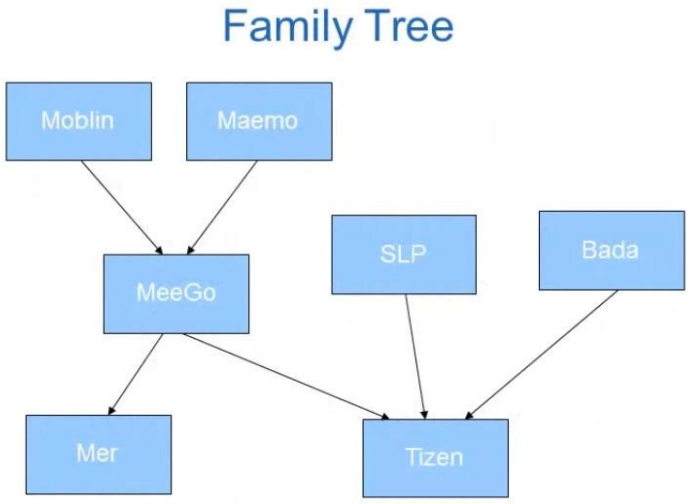


Figure 2 Tizen family tree

3.1.1.3 Tizen Versions

Tizen 1.0 Larkspur was released on April 30th, 2012. The platform consists of the following components [86]: Application Framework, Graphics and UI, Multimedia, Web, Messaging, Location, Security, System, Base, Connectivity, Telephony, Personal Information Management (PIM), Kernel.

Tizen 2.0 Alpha SDK and source code were released on September 25th, 2012. This release has many improvements and additional features and tools such as:

- Enhanced Web UI framework that provides new functionalities from HTML5 and W3C APIs;
- New Tizen Device APIs – Download, Notification, and Power;
- Changes to the Web UI Framework related to the subservices and JavaScript algorithm, page specification and widgets;
- The Tizen Device APIs – System and Contact, were altered;

[11] Image courtesy of [5]
[12] http://merproject.org/

- Many bugs in the Web UI framework were fixed;
- New features were added to the Core system components – Application, System and Telephony;
- Additional functionalities to the tools – Emulator, Emulator Manager, Web Simulator, UI Builder, JavaScript editor, etc.;
- New platform SDK that helps platform development based on Open Build Service (OBS).

The complete list of the new and changed features in this release, as well as the fixed bugs and known issues can be consulted in [88].

3.1.1.4 Devices

Tizen 2.0 Alpha is targeted towards smartphones; however the platform is designed to provide support for multiple device categories, as follows [31]:

- **Smartphones –** the smartphone technologies of Tizen include a flexible UI, 3D window effects, advanced multimedia, location-based service frameworks, sensor frameworks, multi-tasking and multi-touch capabilities and support for scalable screen resolution.
- **Tablets -** for tablets is provided a touch-optimized UI with a suite of built-in applications for Web browsing, personal information management (PIM), and media consumption.
- **Netbooks –** a rich user experience and improved performance will be available for netbooks.
- **IVI devices –** IVI devices provide navigation and entertainment services in vehicles, such as cars, buses, airplanes, etc. Tizen has an open source project called Tizen IVI[13], which will enable the development of applications for IVI devices. Thus, users can have Internet and multimedia experience while travelling.

[13] https://wiki.tizen.org/wiki/IVI

- **Smart TVs –** for smart TVs, Tizen provides a complete open standards-based Linux stack, optimized for living room devices, such as TVs, STBs and Blu-ray players.

3.1.2 Tizen Architecture, API and SDK

In this subsection are presented the Tizen architecture, available APIs and contents of the Tizen SDK.

3.1.2.1 Architecture

As shown in **Figure 3**[14], the Tizen architecture is composed of four subsystems that are: Web framework, Native framework, Core and Kernel.

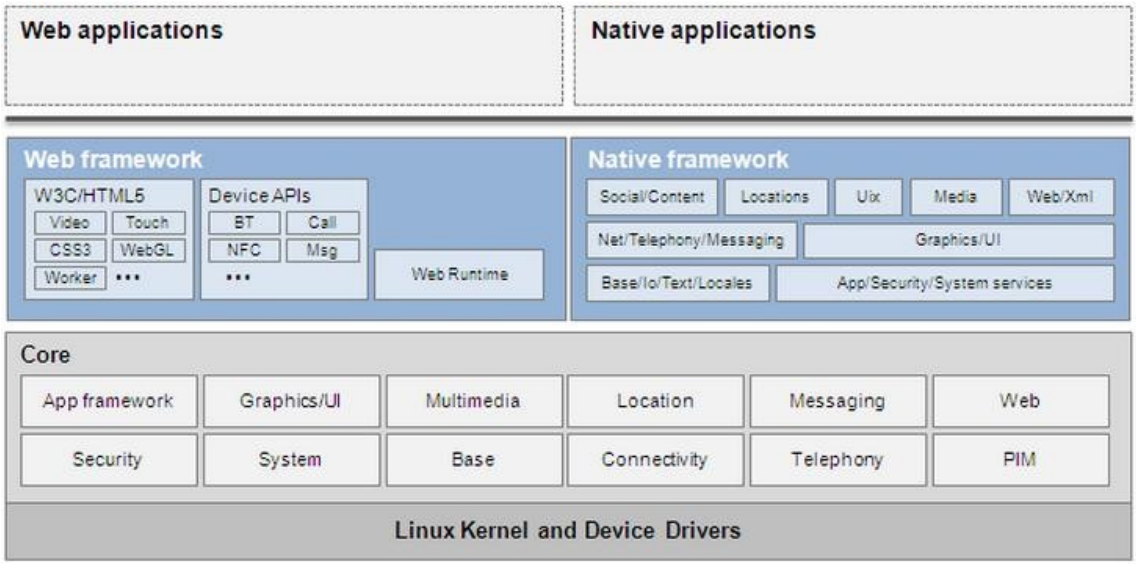


Figure 3 Tizen architecture

❖ **Web framework**

The Web framework enables the development of web apps. It supports many HTML5 functionalities, for example: video, audio, 2D canvas, WebGL, CSS3, geolocation, vibration, WebSocket, and Web worker; and in addition it makes available various device APIs for device functionality such as alarm, messaging, Bluetooth and Near Field Communication (NFC).

[14] Image courtesy of [28]

❖ **Native framework**

The native framework allows the creation of native applications. It provides numerous services divided into the namespaces presented in Section 3.1.2.2.5. This framework additionally supports some standard open source libraries, such as: glibc, libxml2, libstdc++, OpenAL, OpenGL ES and OpenMP.

❖ **Core**

The Core subsystem consists of open source libraries and a set of APIs that are used by the Web and Native frameworks. The Core comprises the following components: Application Framework; Base; Connectivity; Graphics and UI; Location; Messaging; Multimedia; PIM, Security; System, Telephony; Web. Their details are presented in Appendix 8.2.1.

❖ **Kernel**

The kernel subsystem consists of Linux kernel and device drivers.

3.1.2.2 API

For the development of web and native applications Tizen provides the following API references:

- Web framework:
  - Tizen Web Device API;
  - Web UI Framework;
  - W3C/HTML5 API;
  - Supplementary API;
- Native framework:
  - Tizen Native API.

These API references are described in the following subsections.

3.1.2.2.1 Tizen Web Device API

The set of Tizen Web Device APIs enables web applications to access device functionality such as: alarm; application launching; Bluetooth; calendar; call history;

contacts; downloads; file system; messaging; multimedia contents; NFC; notification; power; system information and settings. The description of the APIs in the set is provided in Appendix 8.2.2.

3.1.2.2.2 Web UI Framework

The Web UI framework provides services for creating GUI widgets, events, and different animations and effects. The services are based on a template making use of the JavaScript projects jQuery, jQuery Mobile and Globalization and work on a WebKit-based web browser. Thus, the Web UI framework provides the following:

- Web widgets – include Tizen and jQuery Mobile widgets;
- Web themes – make available CSS themes and resources;
- Loader – supports the loading of configurations, for instance Web theme and internationalization set-up.

3.1.2.2.3 W3C/HTML5 API

Tizen supports APIs that are part of several W3C specifications, for functionality such as: Communication, Device, DOM, Forms and Styles, Graphics, Location, Media, Performance and Optimization, Security, Storage, UI and Widgets. A table with the Tizen support for W3C APIs for these functionalities is presented in Appendix 8.2.3.

3.1.2.2.4 Supplementary API

Tizen supports several non-W3C specifications that are widely used, such as WebGL, Typed Arrays, FullScreen API and viewport Mega Tag. The description of these specifications can be found in Appendix 8.2.4.

3.1.2.2.5 Tizen Native API

For the development of native apps, Tizen provides the following C++ namespaces: App; Base; Graphics; Io; Locations; Messaging; Net; Security; Social; Ui and Web. Their detailed list along with description is presented in Appendix 8.2.5.

3.1.2.3 SDK

The Tizen SDK is a comprehensive set of tools for developing Tizen applications. It includes platform binaries and libraries, header files, Integrated Development Environment (IDE), tools and sample applications. The main contents of the SDK folders are presented in **Table 2**.

Table 2 Tizen SDK folder contents

Folder	Contents
documents	General Tizen documentation
ide	Tizen IDE
tools	Tools available through the IDE
platforms	Tizen libraries, samples, and public header files
Install-manager	Tizen SDK Install Manager
license	Tizen SDK license

3.1.2.3.1 Tizen IDE

The Tizen IDE is based on the JavaScript Development Tools (JSDT) and Eclipse C/C++ Development Tools (CDT) [28]. The SDK provides all the necessary plugins for the Tizen IDE. As can be seen in **Figure 4**, the IDE has different wizards and tools that are useful for application creation and debugging.

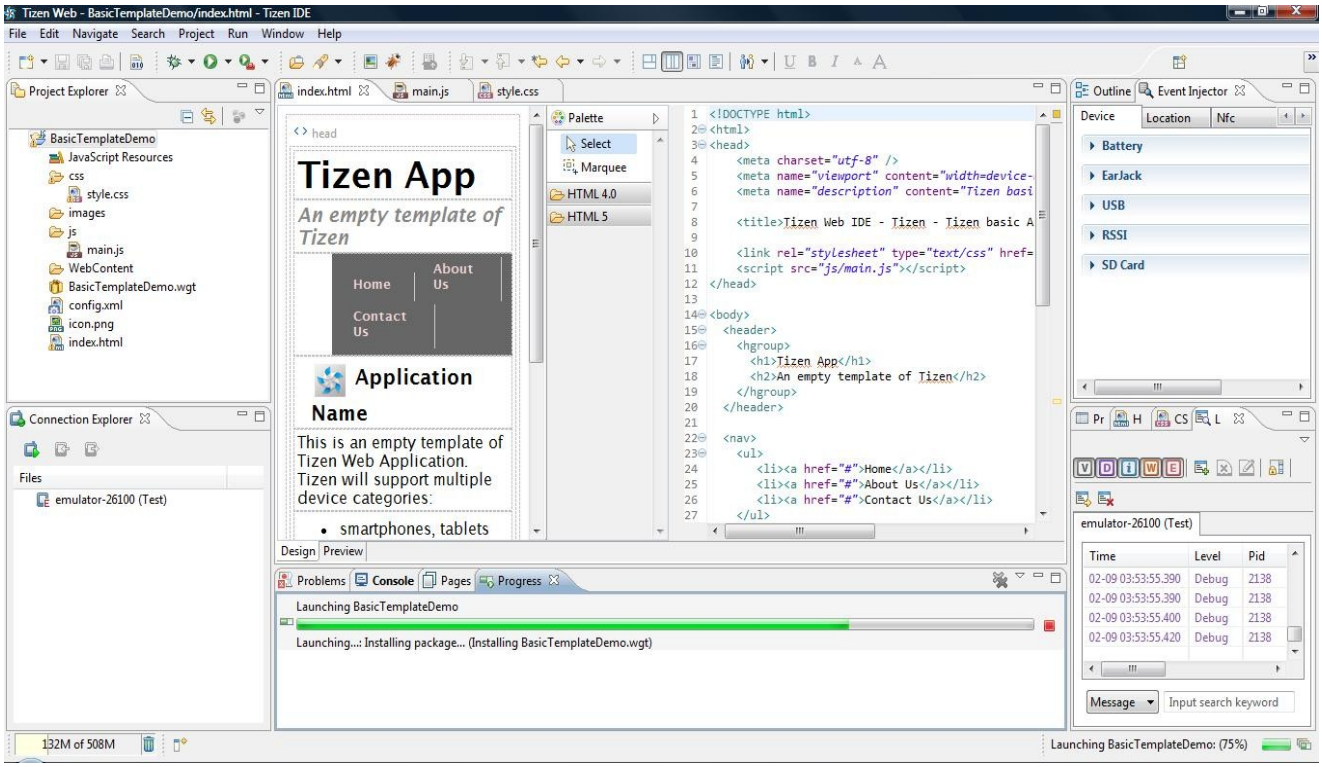


Figure 4 Tizen IDE

3.1.2.3.2 Tools

The Tizen SDK provides several standalone tools that are useful for testing and debugging purposes. These tools are presented in the following subsections.

3.1.2.3.2.1 Web Simulator

The Tizen Web Simulator is a light-weight tool for testing mobile web applications. It is based on the Ripple-UI Framework (a browser-like component that provides emulation services) and runs on Google Chrome. The Web Simulator has the following features:

- Supports running and debugging HTML5 applications;
- Uses JavaScript back-end to simulate Tizen Web APIs;
- Includes various configuration panels for simulating different events, such as device events, phone calls and messages.

A screen shot of the Simulator is shown in **Figure 5**.

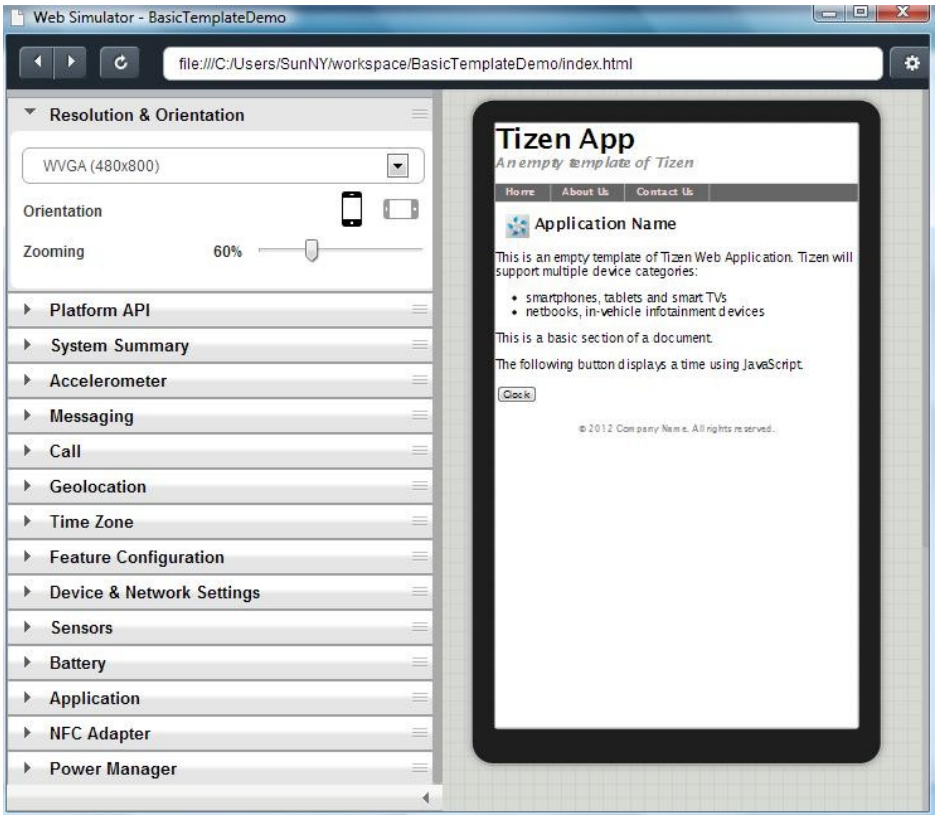


Figure 5 Tizen Web Simulator

In the current version of the Simulator, the supported Tizen Web Device APIs are: Tizen, Alarm, Application, Calendar, Call, Contact, Geocoder, Location-Based Services (LBS), Media Content, Messaging, NFC, System Information, Time, and Power [28].

3.1.2.3.2.2 Emulator

The Emulator is a virtual mobile device that can be used to test Tizen applications before deploying them to an actual device. The Emulator includes virtual CPU, memory, and various peripherals.

A screen shot of the Emulator is presented in **Figure 6**. The Emulator can be started either through the Emulator Manager or using the command line, where different start-up options can be defined. While the application is running on the Emulator, various functions can be performed, such as using multi-point touch, network features, file sharing, among others. The Emulator can be controlled by control keys and menus. It supports various media formats and codecs and OpenGL ES acceleration; however,

compared to physical target devices it has some limitations and differences, which are described in detail in [28].

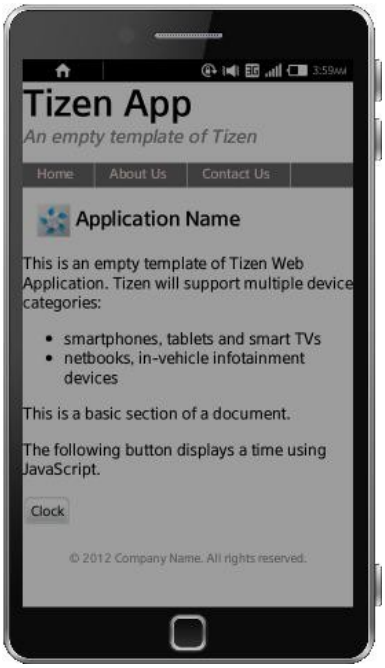


Figure 6 Tizen Emulator

3.1.2.3.2.3 Emulator Manager

The Emulator Manager, illustrated in **Figure 7**, is a tool that enables to create emulated devices and customize their hardware aspects, including display resolution and density, and RAM size. The Emulator Manager allows the creation of multiple Emulators which is useful in order to test multiple environments.

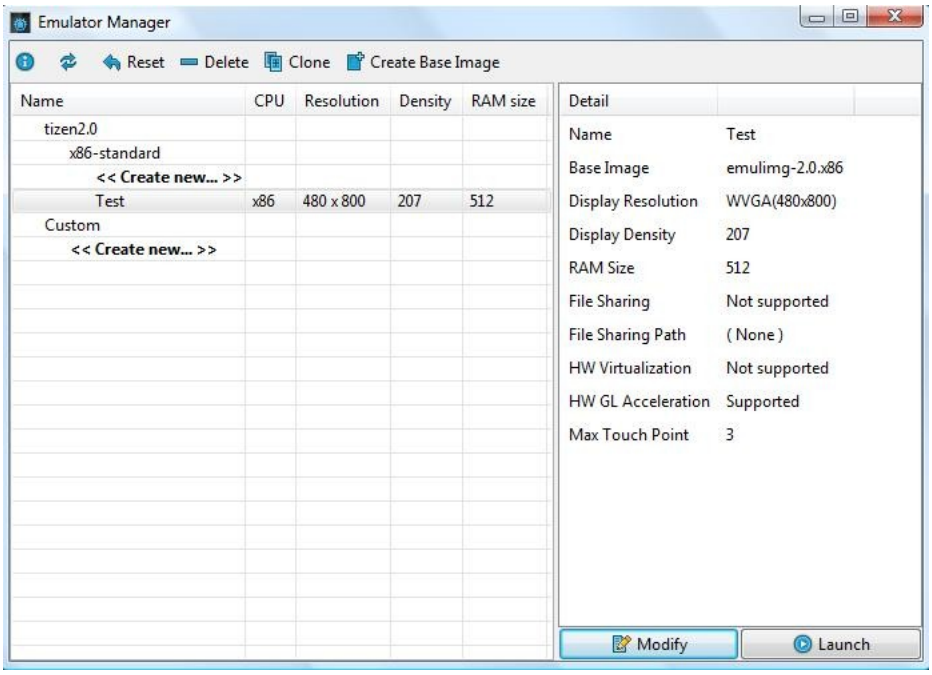


Figure 7 Emulator Manager

3.1.2.3.2.4 Smart Development Bridge

The Smart Development Bridge (SDB) is a command-line tool that manages multiple device connections. It provides basic commands for application development such as: file transfer; remote shell command; port forwarding for a debugger; viewing, filtering, and controlling device log output. For each connected device, the SDB creates a serial number, which is a string that uniquely identifies an Emulator or an actual device. By using this serial number, commands can be sent to a specific device in a list of connected devices. In order to use the SDB with an actual device, the device has to be set to the SDB mode.

The SDB commands can be seen in **Figure 8** and a detailed description for each command is available in [28].

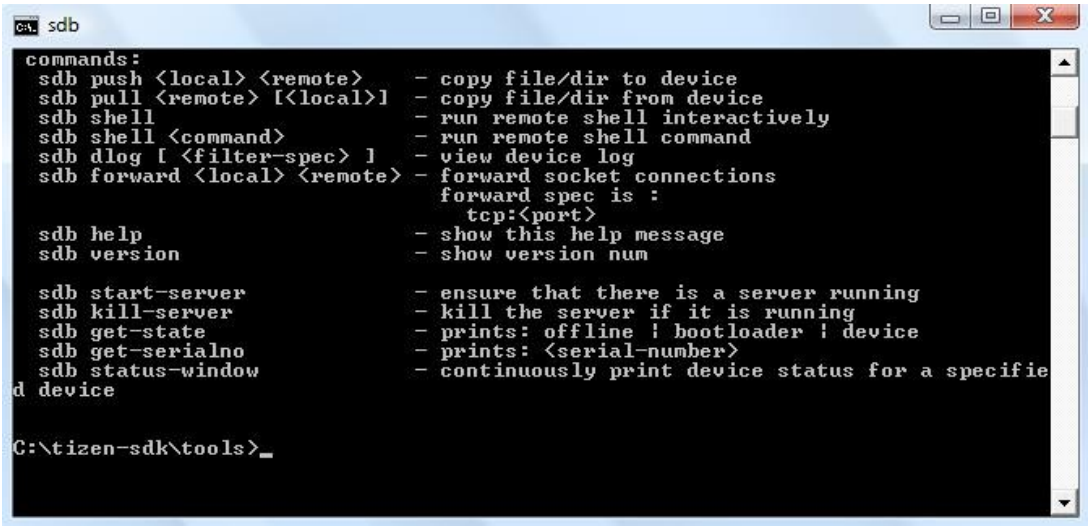


Figure 8 Smart Development Bridge

3.1.3 Tizen Applications

This section presents the Tizen applications which can be web, native and hybrid.

3.1.3.1 Application Life Cycle

The lifecycle of a mobile application is illustrated in **Figure 9**[15].

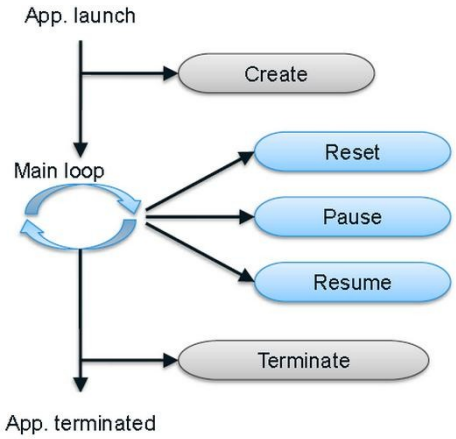


Figure 9 Application lifecycle

[15] Image courtesy of [12].

In the figure the *main loop* represents the app run time and the ellipses *create*, *reset*, *pause*, *resume* and *terminate* are the callback functions that can be implemented for an application. Each of these callbacks is invoked as follows:

- Create – when the application is launched to facilitate the creation of the window and data allocation;
- Reset – when the app is restarted during run time;
- Pause – when the window of the application is sent to the background;
- Resume – when the application window is restored back to the foreground;
- Terminate – after the main loop is executed to terminate the application.

During execution the app passes through several states as presented in **Figure 10**[16].

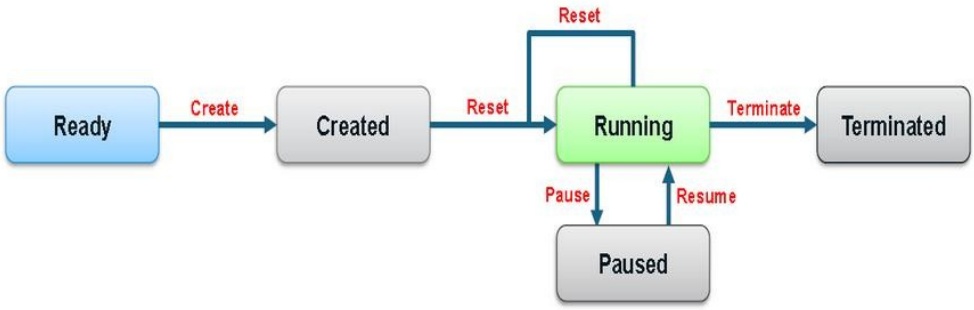


Figure 10 Application states

When the application is launched it is in a *ready* state. Then, the *create* callback is called and the app enters in a *created* state where it is initialized. In the *running* state the application runs in the foreground of the screen and receives user inputs. Hence, it can be restarted by calling the *reset* callback. Moreover, the app can be paused passing it to the *paused* state and resumed, which returns it to the *running* state. The app is in a *terminated* state after calling the *terminate* callback.

[16] Image courtesy of [12].

3.1.3.2 Web Applications

A Tizen web application is a composition of HTML (based on the HTML5 standard), CSS and JavaScript files. These files are distributed in a W3C widget package[17] that can be installed on a device.

The Tizen IDE provides several templates for the creation of Tizen web projects. The available templates in the current Tizen SDK are: Basic, Tizen Web UI Framework, jQuery Mobile, and Tizen Web UI Builder. Developers can create Tizen web projects by using these templates, by using their own user templates, or without using any template. Depending of the chosen template, the Tizen IDE automatically creates the files and folders needed for the project. A Basic Template project is illustrated in **Figure 11**. Although this is the recommended project structure, the developers are allowed to customize the project the way it serves best for their needs.

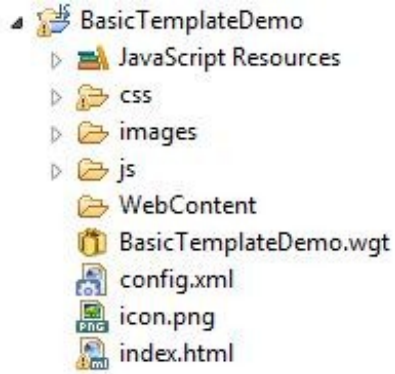


Figure 11 Tizen web project structure

Most of the contents in the figure are self-explanatory with exception of the WebContent folder, intended for the multimedia files used in the project, and the BasicTemplateDemo.wgt is the final widget package. The package is built using the IDE and after that it is ready to be installed on the Emulator or on an actual device. The configuration of the widget is set in the config.xml file. Each widget has exactly one configuration file and it is located at the root of the package. More details about the config.xml file can be found in Appendix 8.2.6.

A Tizen application can be tested on an actual device or on the Emulator. As an alternative, a web app can be tested in the Web Simulator; however, as it was mentioned

[17] http://www.w3.org/TR/widgets/

before (Section 3.1.2.3.2.1) in the current version of the Tizen SDK, the Web Simulator does not support all Tizen Web Device APIs.

3.1.3.3 Native Applications

Tizen supports the development of native apps written in C++ or C (partially supported). The package file format of the Tizen C++ applications is the zip archive format with package file extension ".tpk". The Tizen IDE provides the following project templates: Empty Application; Form-based Application; Library; OpenGL Application; Service Application; and Tab-based Application. The structure of a Form-based project is illustrated in **Figure 12**.

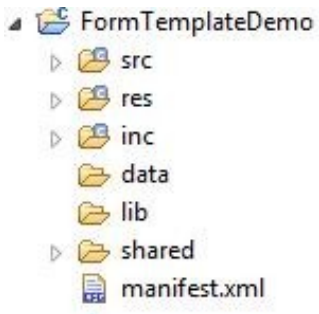


Figure 12 Tizen native project structure

The manifest.xml file contains all application-related information including application ID, version, type and privileges. The privileges allow the application to use features and services of the privileged APIs, which are Tizen native APIs responsible for handling platform and user-sensitive data. More information about the manifest.xml file is available in [28].

3.1.3.4 Hybrid Applications

Tizen supports hybrid applications containing web and native parts. The package format of the hybrid apps is the zip archive with file extension ".wgt". However, in the current release of the Tizen SDK the development of hybrid apps is not available.

3.1.4 Tizen Case Study

For the practical part of the Tizen study three distinct case studies were conducted. The first one explored the Tizen web applications, more specifically a conversion of an existing web app into a Tizen web app and implementation of a Tizen device functionality using a Tizen Web Device API. The second and third case studies included implementation of two prototypes for Tizen native apps using Tizen Push Messaging service and Facebook Chat service, respectively. PTECH considers the three case studies confidential; therefore their description is presented in Appendix 9.1.

3.1.5 Tizen Conclusions

The following conclusions can be extracted from the Tizen study presented in the previous sections:

- Tizen is an open source platform targeted at a wide variety of device categories.
- The Tizen project is backed by leading device manufacturers, chip suppliers, and mobile operators.
- It combines the effort on other projects such as MeeGo, SLP, and Bada as it was confirmed that Samsung will merge Bada with Tizen [75].
- Tizen provides a Web Framework allowing developers to use HTML5 and well-known Web technologies to develop applications that are portable across multiple devices.
- Tizen also offers a Native Framework, which enables the creation of interactive native application written in C++ or C (partially supported).
- Hybrid applications containing native and web parts will be supported in the future.
- The Tizen project relies on application developers' feedback to improve the platform. It provides a vast documentation, many tools for creating and debugging, and other resources that facilitate the process of development and distribution.

Compared to the dominating platforms in the mobile industry, Android and iOS, in terms of application development only Tizen offers the possibility to create apps that are completely web. Android and iOS do not support this option without using external

frameworks (e.g. PhoneGap). Furthermore, Tizen also allows the creation of native apps, thus it does not lose over the other two platforms in terms of performance.

The participation of many handset manufacturers and carriers in the Board of Directors of the Tizen Association, as well as the combination of the efforts on MeeGo and Bada, are considered as some of the ingredients that may guarantee the success of Tizen [32]. However, whether this will happen will be known this year when Samsung launches its first smartphones based on the Tizen OS.

3.2 Firefox OS

This section presents the Firefox OS study. It exposes the state of the platform as of March 2013. Nevertheless, similarly to Tizen, Firefox OS is a new OS that is still under development; therefore some of the information in the text may be subject to change at some later point.

3.2.1 Firefox OS Overview

Firefox OS, also known by its project name Boot to Gecko or B2G, is a new open-source mobile OS developed by Mozilla. The platform is based on a Linux kernel and a Gecko-based runtime engine and it is free from any proprietary technology [49]. The Firefox OS applications are web pages written in HTML, CSS, and JavaScript and they have enhanced access to the hardware and services of a mobile device.

Currently, 17 operators are committed to the project as follows: América Móvil, China Unicom, Deutsche Telekom, Etisalat, Hutchison Three Group, KDDI, KT, MegaFon, Qtel, SingTel, Smart, Sprint, Telecom Italia Group, Telefónica, Telenor, TMN and VimpelCom [55].

Firefox OS is oriented to low-cost devices proving that they can have the same features and performance as high-end smartphones. The first devices running Firefox OS are manufactured by Alcatel (TCL), ZTE and LG. They are powered by Qualcomm Snapdragon mobile processors. The first devices will be available to consumers in Brazil, Colombia, Hungary, Mexico, Montenegro, Poland, Serbia, Spain and Venezuela [55].

3.2.2 Firefox OS Architecture, API and Tools

This subsection presents the platform's architecture, an overview of the supported APIs and the tools that can be used during the development process.

3.2.2.1 Architecture

The Firefox OS architecture is illustrated in **Figure 13**[18].

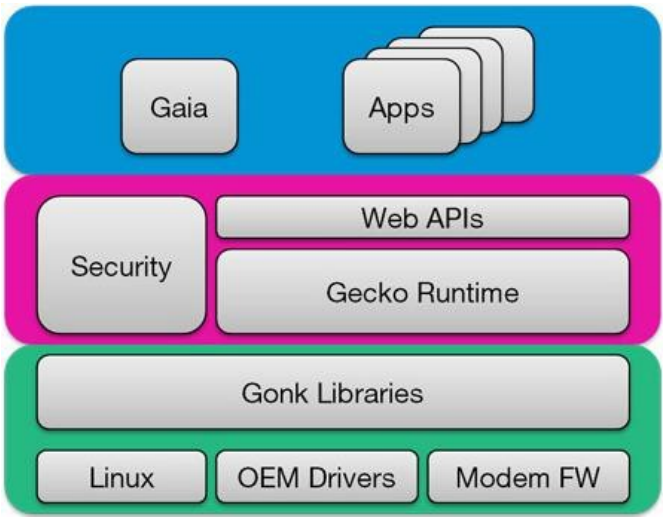


Figure 13 Firefox OS architecture

As can be seen in the figure the platform is composed of three main components as follows [39]:

- Gaia – the top layer of the architecture is the UI of Firefox OS. It contains the home screen, lock screen, telephone dialer, camera and other apps. Gaia is entirely written in HTML, CSS, and JavaScript, and its interface with the lower layers is only through Open Web APIs. Third party applications can be installed beside Gaia.
- Gecko – this is the application runtime that provides all support for HTML, CSS and JavaScript. Therefore, some of the Gecko's components are: a layout engine, a JavaScript virtual machine, networking and graphics stacks and porting layers.
- Gonk – the Firefox OS's bottom layer consists of Linux kernel and Hardware Abstraction Layer (HAL). The kernel and some of the userspace libraries are

[18] Image courtesy of [35]

common open source projects, including Linux, libusb and bluez. Other parts of the HAL are shared with the Android project, such as camera and GPS. Gonk is a porting target of Gecko, which means that there is a port of Gecko to Gonk. Thus, since the Firefox OS project has full control over Gonk, Gecko has direct hardware access on Gonk; however, it does not have this access on other platforms.

3.2.2.2 API

Firefox OS supports the following sets of APIs:

- Firefox OS Device APIs – this set of APIs exposes device features to the Firefox OS apps. Thus, it includes functionality such as: alarm; browser; contacts; geolocation; notifications; radio; storage.
- General Web APIs – consist of standard Web APIs that are supported by the Firefox browser, some of which are: DOM events; device orientation; history; HTML5 audio and video tags; network requests; online and offline events; touch events.
- Firefox Marketplace Services – provides APIs that support publishing and managing apps on the Firefox OS Marketplace.

The list of the available APIs and their descriptions can be seen in Appendix 8.3.1.

3.2.2.3 Tools

In this subsection are presented some of the available tools that can be used to test Firefox OS applications.

3.2.2.3.1 Firefox OS Simulator

Firefox OS Simulator is a tool that simulates a mobile phone environment. It is installed as an Add-On on the Firefox desktop browser. As can be seen in **Figure 14** the Simulator is a complete simulation of Firefox OS providing a Home button and numerous pre-installed apps. Nevertheless, some device APIs are dependent on hardware features;

therefore, although they work on an actual device, they may not work properly on the Simulator.

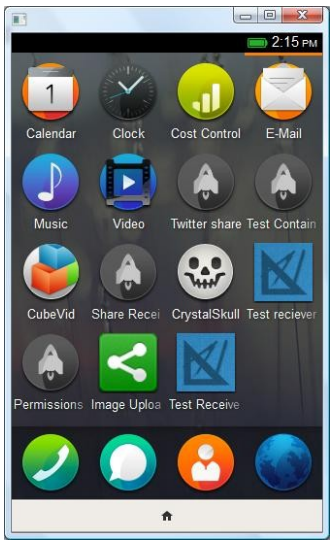


Figure 14 Firefox OS Simulator

3.2.2.3.2 Simulator Dashboard

The Simulator Dashboard, shown in **Figure 15**, is a manager tab on the Firefox desktop browser that serves to launch the Firefox OS Simulator and install applications on it. In addition, the Dashboard provides a remote console for the Simulator and a port number for the Remote Debugger.

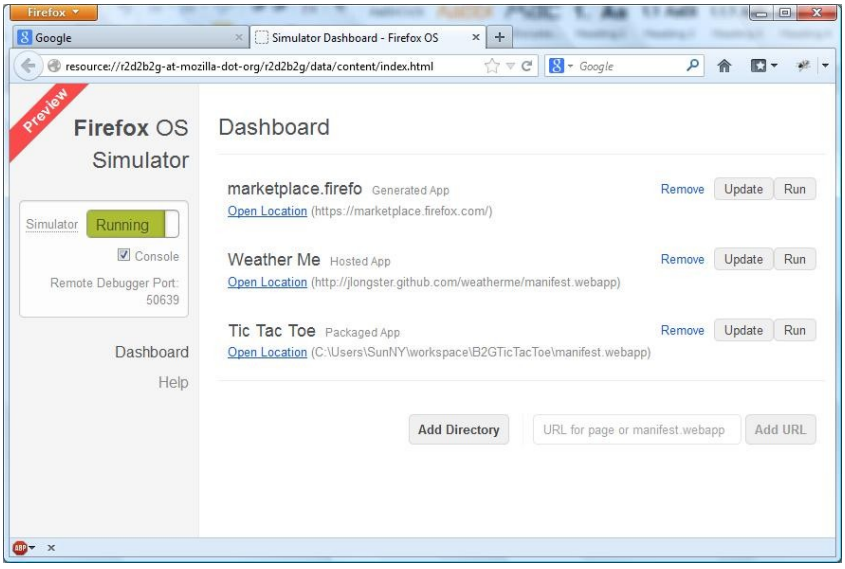


Figure 15 Simulator Dashboard

3.2.2.3.3 Remote Debugger

The Remote Debugger delivers the Firefox development tools to debug apps running on the Firefox OS Simulator or on an actual device. The main difference with the web content debugger is that the Remote Debugger runs on its own window, as can be seen in **Figure 16**. In order to use the Remote Debugger the Firefox desktop browser and the Firefox OS Simulator or the device need to be set up. However, currently the device debugging is disabled [79]; therefore, only the Simulator can be used to debug apps. More details about the Remote Debugger's features can be found in [25].

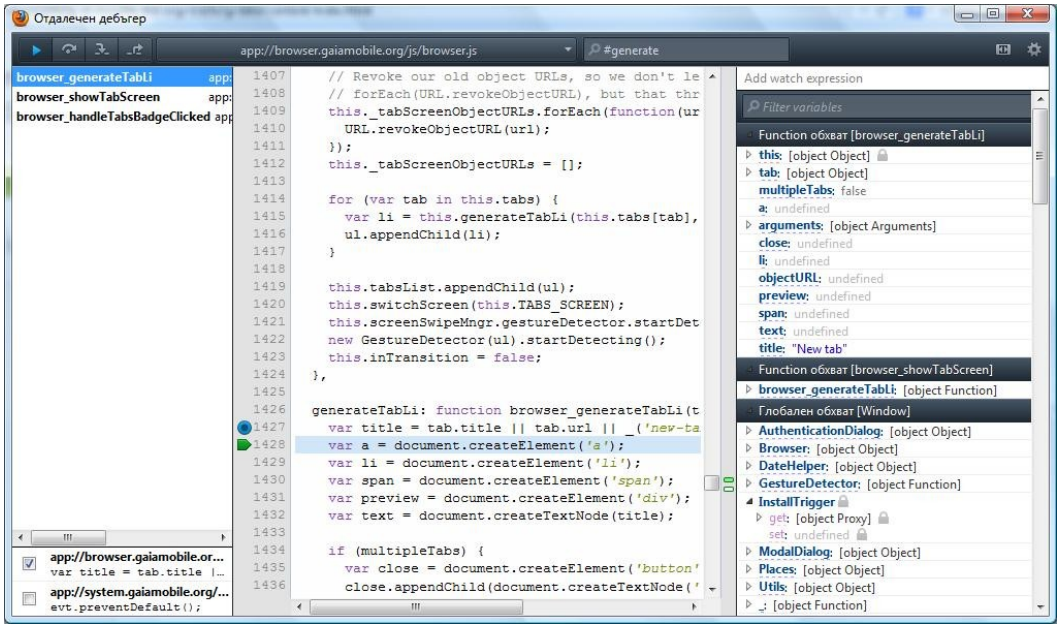


Figure 16 Remote Debugger

3.2.2.3.4 App Validator

The App Validator [19] is an online tool that can be used before submitting an app to the Firefox Marketplace. The tool checks the manifest file of an app and shows the errors (if any) or warnings that should be considered. The App Validator is shown in **Figure 17**.

[19] https://marketplace.firefox.com/developers/validator

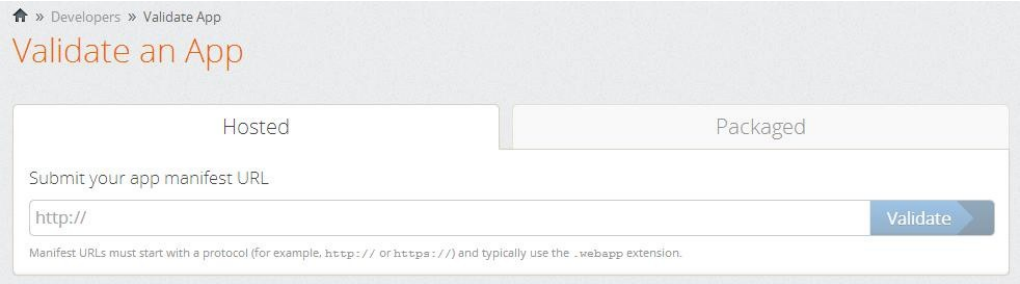


Figure 17 App Validator

3.2.3 Firefox OS Applications

The Firefox OS applications are Open Web Apps (OWA), which are web apps written in HTML, CSS, and JavaScript. OWAs are intended to be standardized and therefore to be able to work on any browser, OS or device. The apps can be distributed through the Firefox Marketplace or any other website. Moreover, it should be noted that web apps can be published as any website; however there are several characteristics that make them different from a normal website such as: apps are installed by the user and can run offline; they are self-contained and do not require a browser window [44].

In order to install a website as a web app on a device, it has to have a manifest file.

❖ **App Manifest**

The app manifest is a JSON file which contains information that the browser needs to interact with the app. The manifest has the name manifest.webapp and is placed at the root of the application. The file may provide information for the app's version; icons; permissions required by the app; locale strings to name a few; however the only required fields are the name and description for the app. For more details about the manifest file can be referred to [11].

❖ **App Types**

The Firefox OS apps can be two main types as follows:

- Hosted app – an app that is run from a server at a given domain and alternatively it can be installed on a device using JavaScript code;
- Packaged app – a zip file containing all app resources such as HTML, CSS, and JavaScript files. This allows the zip file to be downloaded and installed on a device.

Both app types must have a valid manifest file. However, since the packaged apps can have access to some sensitive device APIs, they must be verified by the app store where the apps are distributed. There are three types of packaged apps based on their accessibility to the device APIs:

- Plain packaged app – a regular app that is packaged in a zip file. It is signed on the marketplace, yet since it does not have access to certain sensitive device APIs it is not verified following any special process;
- Privileged app – an app that is approved by an app store (e.g. Firefox Marketplace) in order to have access to sensitive APIs;
- Certified apps – an app that is used for critical system functions, such as the default dialer or the system settings app on a device. Therefore, it must be approved by the OEM or carrier to use all device permissions explicitly (without requiring the user's authorization).

3.2.4 Firefox OS Case Study

The objective of the Firefox OS case study was to explore the conversion of an existing web app into Firefox OS app and to implement one of the Firefox OS Device APIs, in order to estimate the effort needed for porting the PTECH's apps to the new OS. After performing the case study, we can conclude that the process was very straightforward. Nevertheless, PTECH considers the practical part of the Firefox OS study confidential, and therefore its description is presented in Appendix 9.2.

3.2.5 Firefox OS Conclusions

Firefox OS is a new free open source mobile OS which is Web-based and runs OWAs, developed with HTML5 and related technologies. The OWAs are part of the Mozilla's vision for Free Web aiming to standardize them and thus making the apps runnable on every browser, OS or a device. HTML5 and the other open web standards are likely to attract more developers to start developing apps for the new OS. Hence, Mozilla does not intend to create a new ecosystem, instead it wants to allow web developers to leverage their skills to create mobile apps that run on diverse devices. This means that even

existing websites and web apps can be easily transformed into Firefox OS apps as we proved in our case study.

Firefox OS apps can be published on Firefox Marketplace or any other website creating multiple marketplaces. This is an important concept for the openness of the platform, because the multiple marketplaces will permit the developers to have a direct relationship with their customers and, additionally, carriers to bill the consumers for the apps they download. This is why Firefox OS is embraced by so many carriers. However, having multiple marketplaces raises some security and privacy concerns. Nevertheless, Mozilla assures that the users can expect security and privacy since Firefox OS is designed to protect the users from malicious apps as well as applications from one another [56].

In our opinion, Firefox OS has all the potential to compete with other Web-based OS's, like Tizen for example, taking into account that it is developed by Mozilla, the company that develops the successful Firefox web browser. However, competing with platforms already established in the market like Android and iOS, will be challenging since these two OS's are still the leaders in the market with 92 percent of global smartphone shipment during the first quarter of 2013 [6]. In terms of market position even the Mozilla's goal is not to achieve "global domination" [57]. Rather, its goal is to provide an open source and low-priced mobile OS option to the end users and to prove that HTML5 is powerful enough to allow functionality competitive with the native applications of other mobile OS's.

Mozilla already presented the first official Firefox OS devices, Alcatel One Touch Fire and ZTE Open, at the Mobile World Congress in Barcelona [51]. They are expected to be available to the consumer later in 2013.

4. Web Applications Packaging Tools

This chapter presents the studies of the web applications packaging tools Adobe PhoneGap and Appcelerator Titanium. It describes the state of the tools as of May 2013; however, since both, PhoneGap and Titanium, are under active development, some of the information presented in the text may be subject to change at some point in the future.

4.1 PhoneGap

PhoneGap is a free open source framework for developing cross-platform mobile applications. It currently supports eight mobile OS's as follows: Android; Bada; BlackBerry; iOS; Symbian; Tizen; webOS and Windows Phone. PhoneGap has been downloaded over 1 million times and it is used by over 400 000 developers [2].

The PhoneGap applications are hybrid apps, thus they are created with standard web technologies such as HTML5, CSS3 and JavaScript; and they use webview providing them with enhanced access to device APIs. The web resources are hosted locally in the application and not on a remote HTTP server. As illustrated in **Figure 18**[20] the UI of the app occupies 100% of the display's width and height. PhoneGap uses the native webviews of the supported OS's and since there is difference in the rendering engines, the UI of a PhoneGap app may not be consistent on the different platforms.

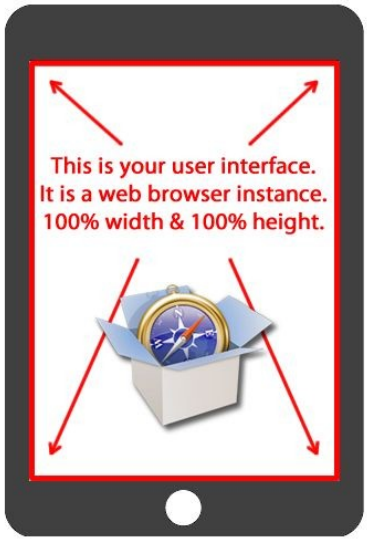


Figure 18 PhoneGap app UI layer

[20] Image courtesy of [71]

To enable device functionality in applications, PhoneGap provides a standard set of JavaScript APIs. These APIs handle the communication between the web app and the native OS, as illustrated in **Figure 19**[21]. Since the JavaScript APIs are built on web standards, the apps should be portable between the different OS's with minimal or no changes. Nevertheless, PhoneGap supports only a limited set of native APIs; therefore, for a device functionality that is not exposed yet, PhoneGap allows the developers to create native plugins that provide the access to the required functionality.

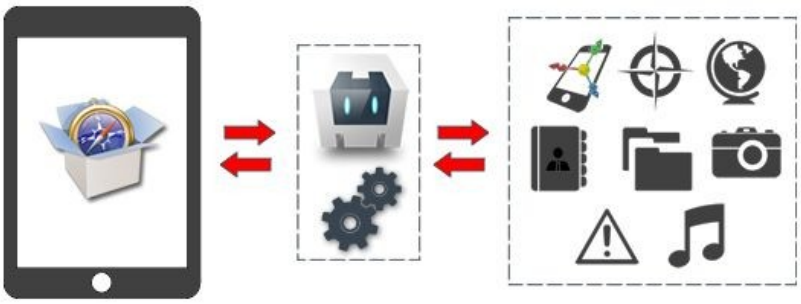


Figure 19 PhoneGap API

A PhoneGap app is developed using web technologies; however, the application package is a binary archive in the format that the mobile OS uses. The final app can be distributed through the appropriate app store (refer to **Table 1**).

4.1.1 PhoneGap History

PhoneGap and PhoneGap Build were originally developed by Nitobi Software. In 2011, Adobe Systems Incorporated acquired Nitobi [3] and as a consequence of the acquisition they donated the PhoneGap codebase to Apache Software Foundation (ASF) for incubation. This contribution was provoked by Adobe/Nitobi's desire to provide the PhoneGap code with proper stewardship, to maintain open and transparent governance, and moreover to facilitate the contribution of other large organizations [70]. Nevertheless, as a result of the donation, PhoneGap had to be renamed. Initially it was called Apache Callback; however, this name was considered too generic [38]. Later, it was changed to Apache Cordova after the Cordova Street in Vancouver, Canada, where

[21] Image courtesy of [71]

the Nitobi's office was situated when they developed PhoneGap. In October 2012, Apache Cordova became a top level project within the ASF [1].

Currently, Cordova has contributors from Adobe, BlackBerry, Google, IBM, Intel Corporation and other independent contributors [20].

4.1.2 PhoneGap vs. Apache Cordova

Many sources use the names PhoneGap and Cordova interchangeably; however, when speaking about the open source project more technically correct is to use the name Cordova, since PhoneGap is the Adobe's distribution of Cordova and Cordova is an Apache project, as shown in **Figure 20**[22]. From PhoneGap explain the relationship between Cordova and PhoneGap as Cordova is the engine that powers PhoneGap [70]. Both, Apache Cordova and PhoneGap, are licensed under the Apache License, Version 2.0.

Figure 20 Apache Cordova and PhoneGap logos

Comparing the distributions of Apache Cordova and PhoneGap for version 2.6.0, the PhoneGap distribution[23] includes several supporting compiled files, such as the cordova.jar required for Android and the CordovaStarter-x.x.x.zip for the Windows Phone template. Otherwise, if the Cordova distribution [1] is used these files have to be built manually.

[22] Image courtesy of [45]
[23] http://phonegap.com/

4.1.3 PhoneGap Services

Besides the functionality that corresponds to the one of Cordova, PhoneGap provides some optional services. These are described in the following subsections.

4.1.3.1 PhoneGap Support

PhoneGap allows subscriptions for paid technical support. The PhoneGap Support service comprises several packages that are divided based on factors, including number of developers, office hours and response time. More information can be found on [72].

4.1.3.2 PhoneGap Build

PhoneGap Build is a paid service for mobile application packaging. As illustrated in **Figure 21**[24], the service packages a web app in the Cloud for all supported mobile OS's; therefore, the developer does not need to install the SDK for each target OS and build the projects manually. The application can be submitted to PhoneGap Build by uploading the application's files or by referencing a GitHub[25] repository. The PhoneGap Build service additionally provides a free plan with one private app and unlimited number of open source apps.

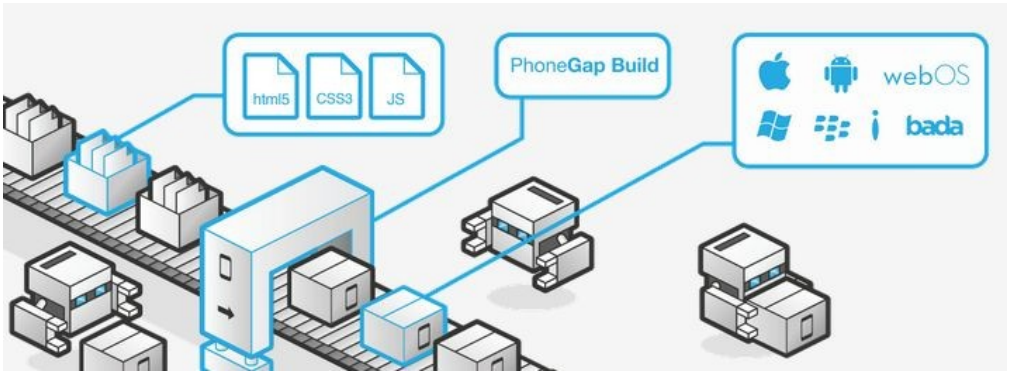


Figure 21 PhoneGap Build

[24] Image courtesy of [4]
[25] https://github.com/

4.1.3.3 Hydration

Hydration is a service that can be enabled through PhoneGap Build, as shown in **Figure 22**. It improves the compilation time by pushing updates directly to the application installed on a device. This is achieved by compiling a native binary that serves as a container for the mobile app and checks for updates directly from the Cloud account. Hydration is intended to benefit the development process. Thus, when a developer uploads a new build, the tester is notified for the update upon restart of the application. If the tester accepts the new version, the app is updated automatically without the need of uninstallation and re-installation.

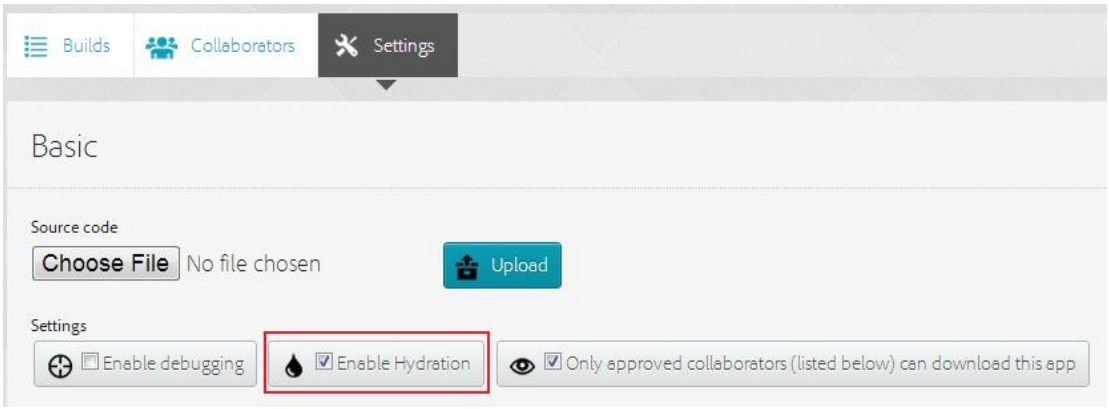


Figure 22 Enable hydration

Hydration can be enabled for a new application or for an existing one. Nevertheless, it currently supports PhoneGap version 2.0.0 or greater and only the platforms Android and iOS [47].

4.1.4 PhoneGap Case Study

The practical part of the PhoneGap study included packaging of a web app for the following target OS's: Android; iOS; Tizen; BlackBerry and Windows Phone. PTECH decided to package the Phune Gaming client application and required to use Cordova, thus the PhoneGap case study would help the company to choose between Cordova and PhoneGap. By the same token, PTECH required the app to be packaged with PhoneGap Build in order to have some estimation between the effort needed for packaging manually with Cordova and by using the automated tool. However, PTECH considers the case study confidential and therefore it is presented in Appendix 9.3.

4.2 Appcelerator Titanium

Appcelerator Titanium is a free open source platform for developing native mobile applications written in JavaScript. Titanium is developed by Appcelerator Inc. and it was first introduced in December 2008 [10]. Its goal is to help developers use their JavaScript skills to create native mobile apps that run across multiple platforms [83]. Titanium supports the following platforms: Android; BlackBerry; iOS; Mobile Web and Tizen. The Titanium SDK in addition provides the Alloy framework, which allows mobile web applications to be designed under the Model-View-Controller (MVC) design pattern.

Currently, Titanium has about 55 000 mobile applications deployed on 137 million devices [84]

4.2.1 Titanium Architecture

Titanium consists of the following components:

- Titanium SDK – includes a set of Python scripts and other supporting tools that interact with the native SDK tools;
- Titanium Mobile APIs – comprise JavaScript-based APIs, which expose access to native APIs;
- Titanium Studio – an Eclipse-based IDE that supports creating, testing, debugging and distributing mobile apps;
- Modules – enable to extend the Titanium's core functionality in order to add support for device or OS-specific APIs. Developers can create their own modules and distribute or sell them through the Appcelerator Marketplace;
- Appcelerator cloud services – the core API makes available various backend services, such as the analytics, which provide basic usage statistics about how often the application is used and on which platforms.

Titanium exists as a bridge between the application source code and the mobile OS. This architecture is presented in **Figure 23**[26].

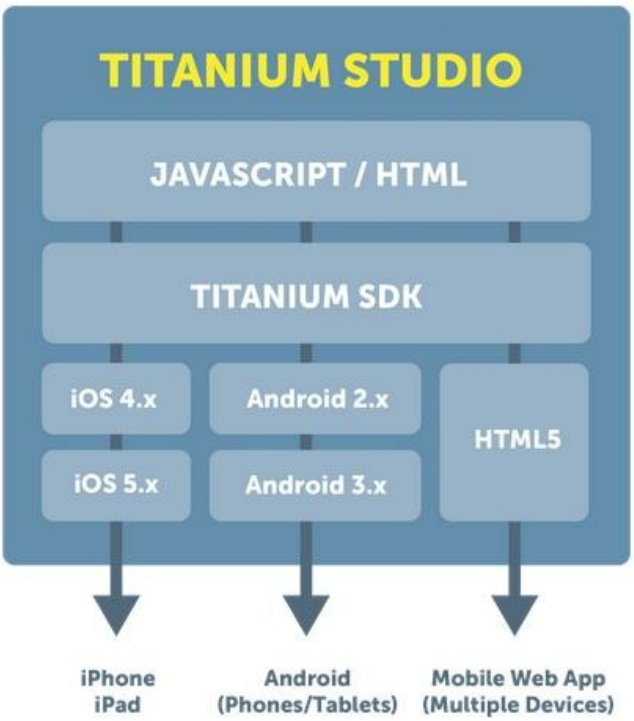


Figure 23 Titanium architecture

At the top layer is the developer's application written in JavaScript and at the bottom layer is the native OS (Android, BlackBerry, iOS and Tizen) or the browser (for the Mobile Web platform). Between the top and the bottom layers resides the Titanium SDK with the Titanium APIs. Therefore, the developer writes the app in JavaScript calling the Titanium APIs and the Titanium Bridge, called Kroll, translates the calls into their native equivalents.

Thus, at runtime the application consists of three major parts – the JavaScript source code, the platform-specific implementation of the Titanium API in native language and a JavaScript interpreter that evaluates the JavaScript code at runtime [18]. When the JavaScript code is evaluated, a proxy object, which is a native object that exposes a JavaScript API, is created. Then, this proxy object is returned to the JavaScript layer and the JavaScript engine creates a corresponding JavaScript object [83]. Therefore, the proxy object exists in the native and in the JavaScript layer and serves as a bridge between them.

[26] Image courtesy of [83]

This is illustrated with the code snippet in **Figure 24**[27].

```
// 1. Create a proxy object
var win = Ti.UI.createWindow();
// 2. Set a property
win.title = "Hello World";
// 3. Call a method on the proxy.
win.open();
```

Figure 24 Example code illustrating proxies

After the execution of the code, the creation of the proxy object and the invocation of its methods are visualized in **Figure 25**[28]. It can be seen that there are two objects – the JavaScript object and the native proxy object.

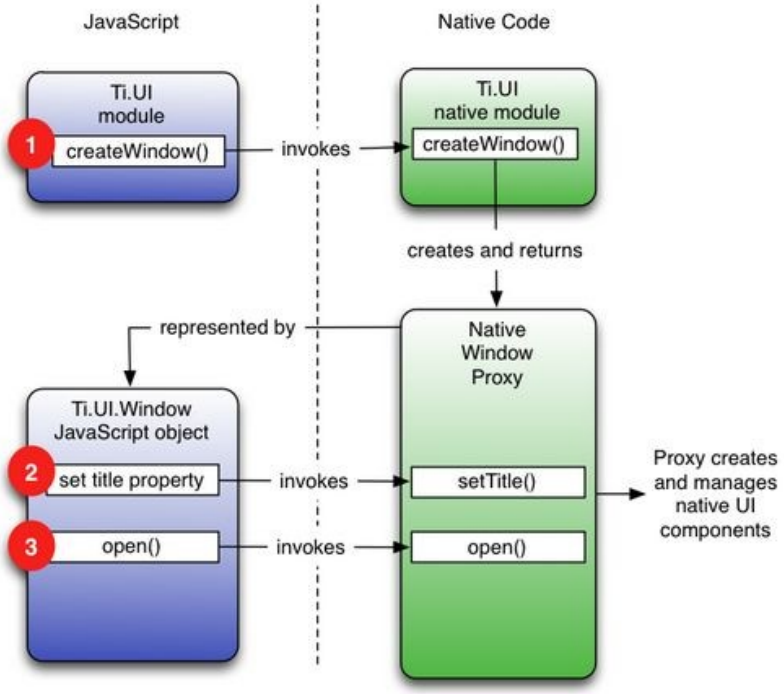


Figure 25 Diagram of the executed code

It should be noted that Titanium does not require a webview, although it can be used, and the JavaScript code is not cross-compiled to Java or Objective C; however, it is evaluated at runtime [18].

[27] Image courtesy of [83]
[28] Image courtesy of [83]

4.2.2 Titanium Applications

The regular Titanium applications are native apps since the JavaScript APIs translate the developer's code into native API calls. Nevertheless, Titanium additionally provides support for web and hybrid app development. The following subsections present features specific for the Titanium apps.

4.2.2.1 Native Applications UI

Although Titanium provides cross-platform mobile application development, the UI of the native apps is not identical across all platforms in order to preserve the native look and behavior that the target OS users are accustomed to. This is illustrated in **Figure 26**[29], which demonstrates the UI differences of an example application running on the iOS and Android.

Figure 26 UI differences

[29] Image courtesy of [83]

4.2.2.2 Mobile Applications

The mobile web apps implement Titanium functionality, such as: common UI elements, animations and 2D matrix operations; HTTP network access; local storage and cache; add-on modules in the form of CommonJS and Asynchronous Module Definition (AMD) modules. Titanium tends to provide this functionality whenever the browser of the supported devices permit and its access is identical on all platforms. Nevertheless, the mobile web apps have some limitations due to restrictions imposed by a vendor, platform and mobile browser. These limitations are as follows [83]:

- Access to some platform-specific components (e.g. iOS local notifications, Android activities);
- Access to native UI controls;
- Full operation without a constant network connection;
- Universal access to hardware sensors (e.g. the camera);
- Access to some components that depend on the OS support (e.g. calendar, contacts);
- Support for Titanium+Plus modules for iOS and Android, which are written in native language and cannot run in a browser.

4.2.2.3 Hybrid Applications

Titanium makes available the Ti.UI.WebView API, which provides access to the platforms' native webview components. The webview can be positioned and resized the way it serves best for the developer's needs and thus other UI components can be added to the remaining space. The web contents displayed in a webview can be hosted on a remote server or locally in the application resources. The major benefit of using remote web contents is the ability to update particular parts of the app without the need to update the whole application. The local web contents, on the other hand, enable to use the app even offline and furthermore this approach can decrease the loading time of the application.

A bidirectional communication can be established between the native Titanium code and the JavaScript code running within the webview. This functionality is limited to the local

web contents; yet to interact with a remote web content the platform provides the `evalJS()` method which can be used to inject JavaScript code into a webview.

When using webviews, it should be taken into consideration that the webviews are the heaviest native UI components, they take time to render and can affect the performance of the application [83].

4.2.3 Titanium Case Study

Similarly to the PhoneGap case study, the Titanium case study included packaging of the Phune Gaming client application for all supported platforms. PTECH is interested in the hybrid app and therefore we used the project template, which allows apps with webview to be developed. Nevertheless, since this project further provides the option to create mobile web app, we decided to explore it as well, because, as expected, it would not require too much effort. PTECH considers the case study confidential; therefore, it is presented in Appendix 9.4.

4.3 API Comparison

Table 3 presents the comparison of the APIs provided by Cordova and Titanium. The first column lists the APIs in alphabetical order and the second and third column indicate with check marks (✓) the Cordova and Titanium support, respectively. When the two platforms provide similar functionality the check marks are presented in bold to increase the readability of the table. In the cases when Cordova and Titanium use different API names for similar functionality, these are presented in the format: Cordova API / Titanium API.

Table 3 API comparison

API	Cordova	Titanium
Accelerometer	✓	✓
Analytics		✓
Android		✓
Calendar		✓
Camera, Capture, Media / Media	✓	✓
CloudPush (Push Notifications)*		✓

API	Cordova	Titanium
Compass, Geolocation	✓	✓
Connection / Network	✓	✓
Contacts	✓	✓
Device, Events / App, Platform	✓	✓
Facebook*		✓
File / Filesystem	✓	✓
Gesture		✓
Globalization / Locale	✓	✓
Map		✓
Notification	✓	✓
Splashscreen	✓	
Storage / Database	✓	✓
Stream		✓
UI (Create native user interface)		✓
XML		✓
Yahoo		✓

* Cordova provides plugins for this functionality.

The Titanium Android API enables the access to some specific Android features (e.g. the Android Intents). Furthermore, it should be noted that some of the Titanium APIs were not considered; hence they are not presented in **Table 3**. The reason for their exclusion is that they are specific for Titanium and since the table does not present a textual description, they could cause some misunderstanding. Therefore, for the complete list of APIs and respective description, as well as which target OS they support should be referred to Cordova API Reference [7] and Titanium API Documentation [83].

4.4 Packaging Tools Conclusions

This chapter presented the studies of two packaging tools for cross-platform mobile app development – PhoneGap and Titanium. Both platforms are open source and are licensed under the Apache License, Version 2.0. Moreover, they are similar in that they require the use of JavaScript; expose access to device and OS-specific features and if any feature is not available, they allow the developers to create the missing functionality themselves as plugin/module. The PhoneGap and Titanium apps can be installed on a device and they are distributed via the app stores of the target mobile OS's.

Although at first the two platforms seem to be very similar, they differ in many aspects. The PhoneGap applications are hybrid apps – they are written in HTML, CSS and JavaScript that run within a native webview component. Titanium provides support for hybrid apps, yet it also supports the development of native apps written in JavaScript and mobile web apps that are accessed from the browser of a device. Therefore, PhoneGap aims at offering option web developers to reuse existing web apps to make them portable across multiple platforms. While this is possible with Titanium as well, Titanium wants primarily to provide a way for web developers to use their JavaScript skill in order to write native apps instead of learning a specific native programming language – Java, Objective C, or others. In terms of hybrid apps, the PhoneGap apps are displayed in full screen, whereas with Titanium the webview can be resized and positioned the way it serves best the developer's needs. Therefore, Titanium seems to be more powerful and flexible platform.

PhoneGap and Titanium further differ in their support for target mobile OS's. PhoneGap has a larger spectrum of target OS's since at the time of the study it supports Android, Bada, BlackBery, iOS, Symbian, Tizen, webOS and Windows Phone; whereas Titanium, provides support for Android, BlackBery, iOS, Mobile Web and Tizen. The difference in the supported OS's can be explained with the scope of the device APIs. PhoneGap supports a smaller set of API that access device features thus it is easier to support new platforms. The scope of the Titanium API, on the other hand, is greater and therefore to implement the Titanium API on a new platform requires greater effort.

Regarding the development process the two platforms take different approaches. Cordova (and PhoneGap) requires the creation of separate projects for each target platform. Titanium, by contrast, provides the Titanium Studio, which centralizes the

interaction with the native tools and thus creates a single codebase for all target platforms. Nevertheless, PhoneGap overcomes the problem with the separated projects by providing PhoneGap Build.

In terms of performance both frameworks have their issues. Some developers have complained for performance problems related to memory leaks when developing complex applications with Titanium [93]. The problem of PhoneGap is that the apps are not native and the users may feel the difference if the app appears more as a web page than an application. Thus, PhoneGap apps for iOS may not be approved on the App Store if the apps do not satisfy the performance requirements.

Finally, as a conclusion we must say that like all technologies for software development, both platforms have their strengths and weaknesses. Nevertheless, considering the hybrid apps, which are in the interest of PTECH, in our opinion, PhoneGap is the better choice for the PTECH's needs. The reason for this is the support for more target platforms and that the PhoneGap functionality is more focused on the hybrid apps, whereas Titanium focuses on the native apps. In our case studies using the Phune Gaming client application we created the Cordova-based applications easier and obtained better results. The development with Titanium, on the contrary, was particularly problematic. Finally, in our practical experience we did not work with the native functionality neither for Cordova nor for Titanium, nevertheless if this is required we expect to be easier to make modifications or additions to the native functionality of Cordova, since its implementation is simpler compared to the one of Titanium.

5. Smart TV

This chapter presents the studies of the two Smart TV technologies that were addressed during the internship – SSTV and Opera TV. The SSTV study was conducted in February, 2013 and the one of Opera TV in June, 2013. Hence, since the two platforms are under active development, some of the information in the text may be subject to change at some later point in time. Furthermore, SSTV is in the direct interest of PTECH; therefore, the focus was on it, while the Opera TV study aimed at introducing the company to the new platform.

5.1 Samsung Smart TV

SSTV is a platform that extends the user experience beyond the traditional TV providing the users with access to web contents. A user can navigate the Internet and obtain information about news, weather, sports, and much more on the TV screen. Nevertheless, the experience is not the same as viewing a web page on a personal computer due to the difference in the screen resolution, hardware specification and using the remote controller as the main interface with the TV.

SSTV provides a place on the TV, called Smart Hub, which centralizes the management of users' accounts and all the smart contents, thus allowing an instant access to applications, videos, photos, web browser and others. It allows search by using the remote controller, a wireless keyboard or voice, and also displays recommended applications to the users. The Smart Hub is presented in **Figure 27**[30].

[30] Image courtesy of [34]

Figure 27 Smart Hub

5.1.1 Samsung Smart TV Architecture, API and SDK

This subsection presents the various modules of the SSTV architecture, and also introduces in a concise manner the supported APIs and the SDK's contents.

5.1.1.1 Architecture

Figure 28[31] illustrates the architecture of SSTV, which is composed of the following components: Application Manager; Mapple Browser; Common Modules; Device APIs and Internet@TV.

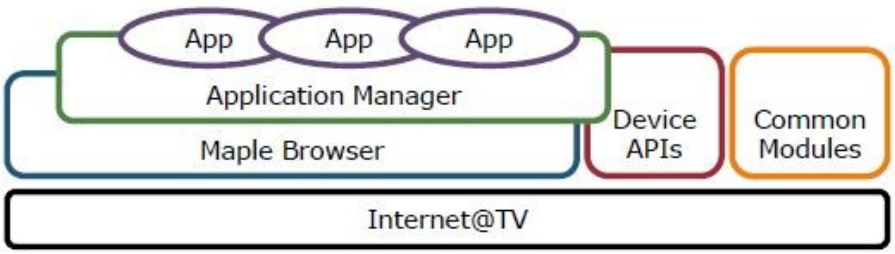


Figure 28 SSTV architecture

[31] Image courtesy of [69]

- **Application Manager**

The Application Manager is responsible for the management of applications for tasks, such as installing, running and deleting apps and also assists in the functions of the Common Modules component. Furthermore, the Application Manager handles the management of the user accounts. It supports the Single Sign-On (SSO) module, which gives to the users the convenience to enter once their account information and then the Application Manager encrypts and saves the user data, and sends it to applications whenever it is required.

- **Maple Browser**

The SSTV platform uses a Markup engine Platform for Embedded Systems (Maple), which is a browser engine for Consumer Electronics (CE) devices [76]. However, the 2012 devices have an entirely new browser engine based on WebKit [9].

- **Common Modules**

The common modules provide some general purpose objects that can be used in an application, such as: TVKeyValue; Widget; Input Method Editor (IME) and SSO. These objects are needed in order to run and display applications on the TV screen, recognize remote controller events, use plugins, and communicate with the Application Manager. The list of the Common Modules' objects can be seen in Appendix 8.4.1.

- **Device APIs**

The Device APIs is a set of APIs that provide access to some middleware features of the digital TV (DTV), including: audio; download; filesystem; network; player; screen; time and video. The full list of Device APIs is provided in Appendix 8.4.2.

- **Internet@TV**

This component in the figure indicates that the platform has integrated Internet capabilities, which are required in order to run SSTV applications.

5.1.1.2 API

Besides the Device APIs, SSTV provides APIs for functionality, such as: advertisement; in-application purchase; interaction between the TV and a mobile device; convergence

application, among others. The summary of the supported APIs is presented in Appendix 8.4.3.

5.1.1.3 SDK

The SSTV SDK is available for Windows, Linux and Mac OS and it contains all tools and APIs necessary for developing applications for SSTV. The latest SDK version is 4.0 and provides the tools for creating applications for 2013 TV platform, such as:

- Eclipse-based IDE;
- Visual editor;
- JavaScript debugger;
- Smart TV Emulators for TV models: 2011, 2012 and 2013.

The changes in the various SDK releases are available in [80] and the technologies supported by the different TV platforms can be consulted in [81].

5.1.2 Samsung Smart TV Applications

The SSTV applications are web applications that run on a DTV connected to the Internet. However, unlike the regular web pages, the SSTV applications can implement the SSTV APIs in order to use features that are specific for TV. This makes the apps more TV-oriented which contributes to the better TV experience for consumers.

Figure 29[32] illustrates a comparison between a web page and a Smart TV application. As can be seen the main difference is in the screen resolution, hardware specifications and the usage of the remote controller as the main user interface.

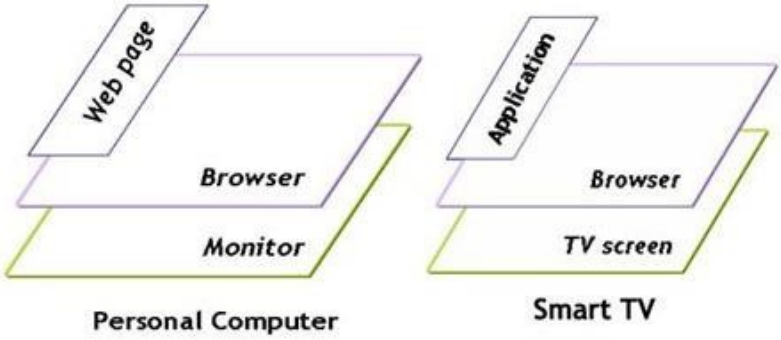


Figure 29 Web page vs. SSTV application

[32] Image courtesy of [77]

5.1.2.1 Application Display Types

In terms of positioning on the screen, an application can be full screen, single-wide or ticker, as shown in **Figure 30**[33]. The full screen applications occupy the full screen area, while the single-wide apps are displayed only on a part of the TV screen. The ticker type allows the application to remain on the bottom of the screen while the user does other things on the TV. It should be noted, however, that the apps launched in Europe can be only full-screen applications.

Figure 30 Application display types

5.1.2.2 Project Types and Application Structure

The SSTV SDK makes available the following project types:

- Basic – permits the creation of applications using the Visual Editor tool;
- JavaScript – provides SSTV JavaScript APIs for greater control over application tasks and processes;
- Flash – allows Flash functionality to be used in the application.

As can be seen in **Figure 31**[34] a SSTV application is composed of HTML, CSS and JavaScript files, the config.xml file and other resources.

[33] Image courtesy of [77]
[34] Image courtesy of [69]

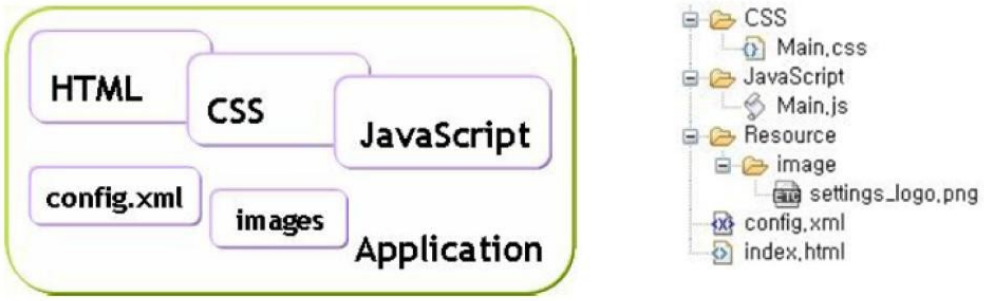


Figure 31 Application file structure

The config.xml file contains information related to the settings of the application and the operating environment. This information is used by the Application Manager in order to manage the user accounts, control the application's version and set the environment. The configuration of the config.xml file is provided in Appendix 8.4.4.

5.1.2.3 Testing and Publishing

A SSTV application can be tested on an Emulator or on an actual SSTV. Although the Emulator simulates running an application on the TV, it is recommended to test the app on an actual TV, since there are some aspects in which the TV differs from a computer environment, such as [26]:

- Less memory is available on TV;
- The response to the remote controller's keys may have different timing;
- Only certain keys are allocated to the application;
- Audio and video playback may have different behavior;
- The application may behave differently if the TV has a different browser version than the browser of the Emulator.

In order to test an application on the TV and to prepare it for publishing, it has to be packaged using a SSTV IDE. After the app is packaged and tested it can be submitted to Samsung Apps[35] where it goes through a certification process. Once the application is certified and published, the developer can upgrade it or remove it from the marketplace.

[35] http://www.samsungapps.com

5.1.3 Samsung Smart TV Case Study

For the SSTV study two case studies were conducted. The first one included the implementation of the remote controller's key events handling in an existing JavaScript application in order to control it using the remote controller. The second case study aimed to examine the SSTV convergence application, which is a composition of two applications – a TV application and a client application that runs on an external device, such as a smartphone, desktop, tablet, and interacts with the TV application. PTECH considers the two case studies confidential and therefore they are presented in Appendix 9.5.

5.1.4 Samsung Smart TV Conclusions

SSTV is a platform that provides to the consumers many services that go beyond the traditional TV allowing the user to access web content from the TV, play games, purchase items, talk with friends on social networks or via Skype video calls, command the TV using motion , voice control or external mobile device. Therefore, SSTV makes the TV experience more engaging for the consumers.

From the developer's point of view, the platform provides SDK with all APIs and tools needed to develop SSTV applications. It also makes available a developers website containing many articles and tutorials about the various functionality supported by the platform. Nevertheless, it has its weaknesses, since we find the organization and the search engine not very easy to procure the information that is looking for. Moreover, it lacks information about the SSTV architecture, a sufficient explanation for the API's functions and available examples. However, besides these weaknesses we can conclude that the platform provides satisfactory information to start developing SSTV apps.

In our case study we experienced several challenges related to the development tools. SSTV provides two IDEs – Eclipse-based IDE available in SDK 4.0 and SSTV IDE available in previous SDK versions. The Eclipse IDE requires to be run with administrator privileges in order to set the SSTV perspective. In addition, its packaging did not work properly and therefore we had to use the SSTV IDE to package the applications. Furthermore, there were some issues with rendering the pages on the Emulators. Samsung provides a SDK Emulator Image for Virtual Box, which crashed frequently, thus we used

the Emulator provided by the SDK; however, it did not display the app's images. In order to solve this issue we had to specify the absolute paths in the source code.

Another less pleasant aspect is that the developers are not able to upgrade their TVs to test applications developed with functionality that is not supported in their TV models, as it was in our case. Fortunately, Samsung seems to be working in this direction as they are developing an external device, called Evolution Kit, which enables a 2012 SSTV to evolve into the new 2013 SSTV [78].

5.2 Opera TV

Opera TV is a platform that provides web functionality and environment for running web apps on connected TV devices that integrate Opera Device SDK. The Opera Device SDK is a toolkit intended for OEMs and therefore it is not available for app developers. It allows device manufacturers to build custom HTML5 and CE-HTML rendering based on the Presto engine.

The Opera Device SDK powers a wide range of devices, such as: TVs, STBs, Blu-ray and other media players. Some of the brands that integrate the SDK in their devices are: Panasonic; Philips; Toshiba; Sharp; Boxee TV among others [74].

5.2.1 Opera TV Architecture, Web Standards and Tools

This subsection presents the Opera TV architecture, supported web standards and tools available for app developers.

5.2.1.1 Architecture

Figure 32[36] illustrates the architecture of the Opera TV platform. In the figure in blue are presented the contents and applications, in red (and black) the Opera components and in green the platform and third party components.

[36] Image courtesy of [64]

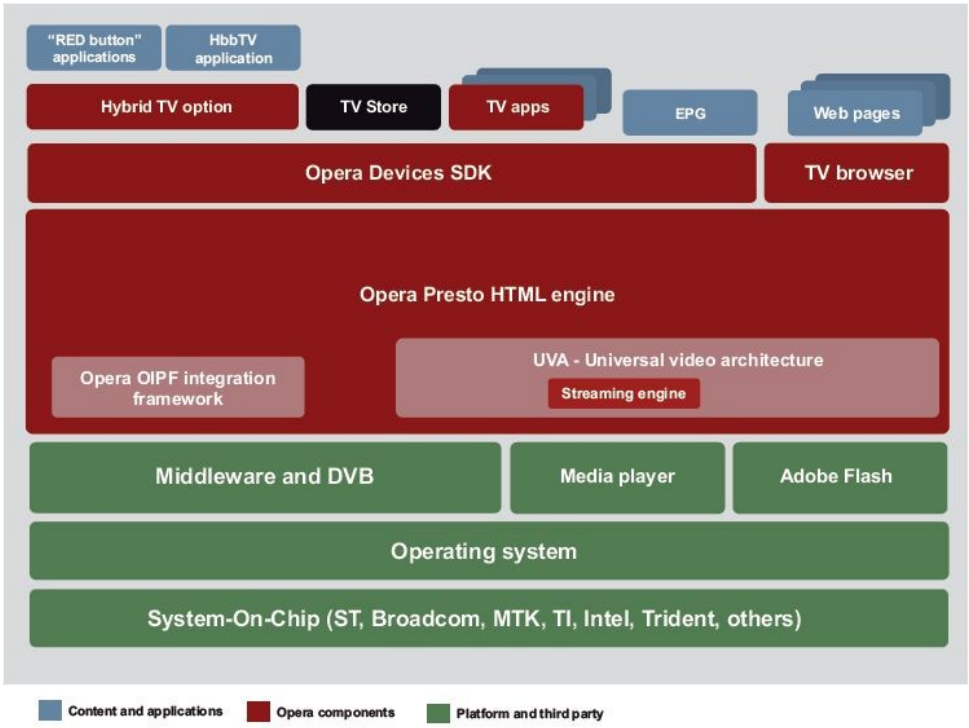


Figure 32 Opera TV architecture

Besides the Opera Presto HTML engine and the Opera Devices SDK, in the figure further can be seen the other Opera's products as follows:

- Hybrid TV option – a solution to run and display "red button" applications and other Hybrid Broadcast Broadband TV (HbbTV) applications;
- TV browser – an Opera web browser for connected TVs;
- TV Store – an HTML5-based storefront described in more detail in Section 5.2.2;
- TV apps – web-based applications optimized to run on TV with different screen sizes and resolutions. More details are presented in Section 5.2.3.

5.2.1.2 Supported Web Standards

Some of the web standards that are supported in the latest Opera Device SDK version 3.4 are as follows [61]:

- HTML 4.01, 5 (draft)
- HTML5 video
- Encrypted Media Extensions v0.1

- Media Source Extensions v0.5
- <track> subtitles/captions for HTML <video>
- WebSocket 2.0
- DOM fullscreen API
- XHTML Basic, 1.0, 1.1
- Web Forms 2.0
- XML
- CSS Level 1, 2, CSS3
- DOM 2, 3
- <canvas>
- HTML5 Forms
- HTTP 1.0, 1.1
- SSL 3 and TLS 1.0, 1.1, 1.2
- Unicode and legacy encodings
- SVG 1.1 Basic and 1.2 Tiny, CSS TV, WebGL, HbbTV (option)

The complete specification can be found on [92].

5.2.1.3 Tools

For testing and debugging purposes, Opera provides the following tools:

- Opera TV Emulator – enables to test HTML5 and CE-HTML contents developed to run on the Opera Device SDK as well as web applications for the Opera TV Store. It is packaged as Oracle VirtualBox and currently there are two versions available: Opera TV Emulator 3.3, which is compatible with Opera Device SDK 3.1 to 3.3; and Opera TV Emulator 3.4 compatible with Opera Device SDK 3.4.
- Opera Dragonfly– the built-in suite of developer tools of the Opera desktop browser, which also allows to connect to the Opera browser running on an actual device or on the Opera TV Emulator and perform remote JavaScript debugging, HTTP logs, RAM analysis, among others.

5.2.2 Opera TV Store

The Opera TV Store, presented in **Figure 33**, is an HTML5-based storefront that shows a catalogue of TV-oriented web applications. It allows users to browse, search and interact with the applications using a standard remote controller. The Opera TV Store is based on the Opera Device SDK and runs on a device that has the SDK integrated. Currently, it is available on Sony Bravia TV 2012 models (EX, HX, NX series); Sony Bravia TV 2013 models; Sony Blu-ray Disc Players 2012 and 2013 models. The complete list of retail devices can be found on [67].

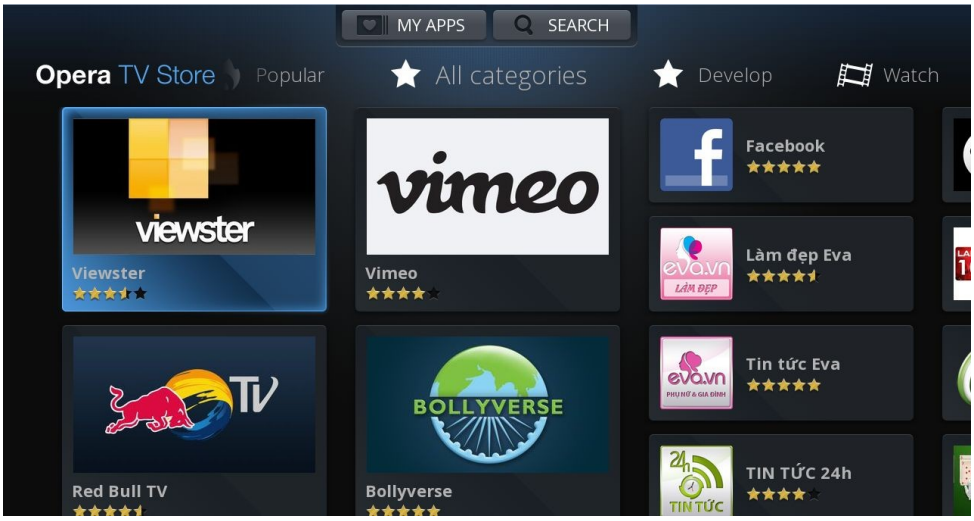


Figure 33 Opera TV Store

It should be noted that the Develop category illustrated in **Figure 33** is only available after the TV device or the Opera TV Emulator is paired with the developer's Opera TV Store Submission portal account. The exact steps that have to be followed for the pairing are described in [82]. The Develop category provides a URL-entry application, called URL Loader, which allows the URL of an application to be entered in order to test it before it is submitted for review. The apps that are already submitted are also available in the Develop category.

In **Figure 34** is presented the My Apps tab, which is a home screen that displays the installed applications in a grid layout. The user can navigate the apps using the four directional keys of the remote controller.

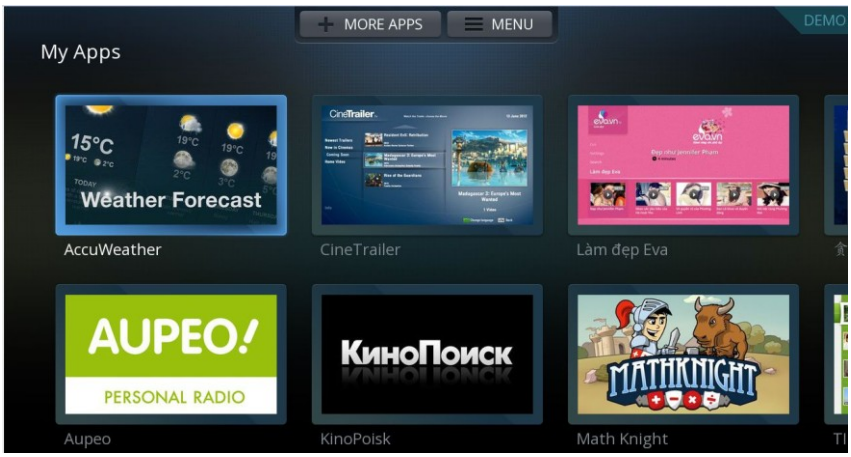


Figure 34 My Apps tab

The Opera TV Store is fully Cloud-based and therefore the applications are not actually installed on the device. This is illustrated in **Figure 35**[37] which presents the architecture of the Opera TV Store.

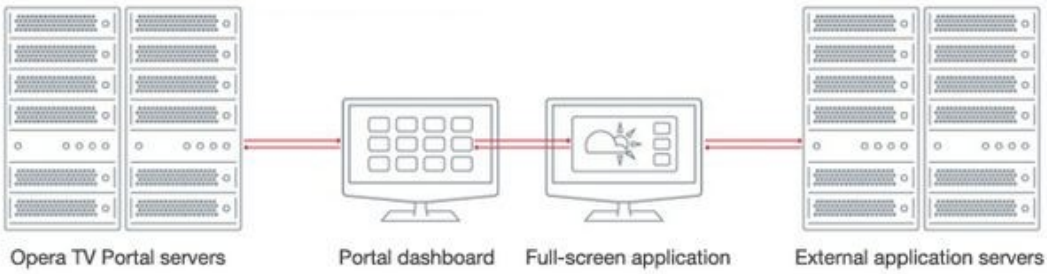


Figure 35 Opera TV Store architecture

When a user browses the categories of the Opera TV Store and "installs" an application, the application is added to the Portal dashboard (the My App tab) where it is displayed as a thumbnail image. Thereafter, when the application is launched it is displayed in a full-screen mode. The full-screen application is not hosted directly on the Opera's servers; instead the Opera TV Store contains the reference to the actual URL of the application, which is hosted on external servers.

Some of the new features of the Opera TV Store are as follows [64]:

[37] Image courtesy of [13]

- Companion SDK – allows developers to create companion apps that integrate Android or iOS phones/tablets with the TV, for example, to use them as a remote controller;
- Side-by-side applications – enables the application to be run side by side with the broadcast stream;
- Pre-roll advertisement – allows pre-roll video ads to be run before the application starts.

A demonstration Opera TV Store is available from the Opera TV Emulator at the following URL: https://demo.tvstore.opera.com.

5.2.3 Opera TV Store Applications

The Opera TV Store applications are web applications written in HTML5, CSS3, JavaScript and other web technologies. They run from the Cloud and are optimized to be displayed on a TV screen and controlled by a standard remote controller.

Existing HTML5 applications for desktop and mobile devices can be repurposed to run on TV following the recommendations for creating web content for TV and the design considerations for Opera TV Store that are accessible on [23] and [27], respectively. Currently, Opera TV Store provides two app templates namely Video player and RSS reader that developers can freely use and customize. These templates are available as zip files on [66].

Opera TV Store runs on Opera Device SDK, which uses the same rendering engine as the Opera desktop browser; nevertheless, there are still some differences due to platform-specific APIs, remote controller keys handling, available RAM and others. Hence, it is recommended to test the apps on an actual device and/or Opera TV Emulator and in the Opera TV Store environment in particular.

The apps can be submitted to the Opera TV Store Submission portal and then they are distributed to different OEMs; therefore, there is no need of separate submissions. The information and assets that have to be provided during submission are as follows [65]:

- Company information – name, address, email;
- Application name;

- Description of the application;
- Author name;
- Support email;
- Thumbnail (JPG, 480x270 or larger, with 16:9 aspect ratio, 1MB max);
- Application icons (JPG or PNG, 128x128px and 512x512px, 500kB max);
- Screenshots of the application (JPG, 1280x720px or larger, with 16:9 aspect ratio, 1MB max);
- URL to full-screen version of the application hosted at the external server;
- Languages supported by the TV application;
- Type of the TV the application was designed for (e.g. '44" Full-HD TV').

When the app is submitted, the developer does not have access to the metadata; therefore, in order to make any changes a new version of the app has to be submitted. After the submission the app is evaluated according to the acceptance criteria available on [65]. If the app satisfies the criteria, it is additionally tested on reference devices, which may also include review process by device OEMs. If the app fails on any of the criteria or the tests, it is rejected and the developer is allowed to fix the problem and resubmit it. Once the app satisfies all criteria it is published on the Opera TV Store.

5.2.4 Opera TV Case Study

We performed two case studies as part of the Opera TV study. For the first case study the Phune Gaming client application had to be adapted to run on Opera TV and it was required to register any issue that we would encounter. This was very valuable information for PTECH, because Phune Gaming does not support officially the Opera web browser. During this case study we solved most of the problems by using workarounds; nevertheless, our findings will help the company to find better working solutions. The second case study included implementation of the remote controller's key events handling in order to enable the navigation on the platform and Tic Tac Toe game via the remote controller. Since PTECH considers the two case studies confidential, they are presented in Appendix 9.6.

5.2.5 Opera TV Conclusions

The Opera TV platform is aimed at end users, OEMs and developers and to each one of them it provides the respective benefits. To end-users Opera TV and the Opera TV Store in particular mean a comfortable “lean-back” web experience allowing them to browse web content and use applications on their connected TV devices. To OEMs Opera TV gives the opportunity to provide this TV web experience, which makes their devices more attractive to consumers. Finally, it allows developers to create HTML5-based applications that run across all devices that integrate the Opera Device SDK. Opera provides the developers with developer portal and tools, such as the Opera TV Emulator and Opera Dragonfly, thus they can create and test applications even without access to an actual device.

Opera TV Store is still a new platform that is currently running only on Sony Bravia TV 2013 and Sony Blu-ray Disc Players. Nevertheless, Opera is already established in the Smart TV market. Because of the support for web standards and performance improvements many OEMs choose to integrate the Opera Device SDK on their devices. As it was announced [59], even Samsung which develops its own Smart TV platform will launch Blu-ray players powered by Opera Device SDK.

In our practical experience the implementation of the remote controller functionality was very simple compared to the one that we implemented for SSTV. The problems with Opera TV came from the fact that Phune Gaming does not officially support Opera, yet after the problems were solved it showed a proper behavior on the Emulator. Regarding the available documentation, at this time SSTV provides a more comprehensive documentation compared to Opera TV. The Opera TV documentation comprises articles about developing applications for TV; however, currently there is not much information about the platform’s specific features and how they can be implemented in applications.

6. Conclusions

The area of the mobile web applications is a very dynamic field where new platforms and frameworks are constantly emerging. Thus, since PTECH is a company whose mission is to research and develop innovative services with new and state-of-the-art technologies, it had the need to conduct the studies of the two new mobile OS's that run web applications, Tizen and Firefox OS, in order to prove whether it is advantageous to start developing applications for these platforms. PTECH is satisfied with the results that we obtained for Tizen and already started planning applications for Tizen. Although the Firefox OS study was not that profound compared to the one of Tizen, the Mozilla's new OS is a potential target for Phune Gaming.

The goal of the web applications packaging tools can be explained to some extent with the slogan *"write once, run everywhere"*. In the context of PTECH objectives this means to use PhoneGap or Titanium in order to make its existing web applications and the Phune Gaming client in particular target more platforms with less development costs. After performing the case studies we concluded that PhoneGap satisfies better this requirement, especially if PTECH decides to use PhoneGap Build it will decrease significantly the development time.

Since PTECH intends to target the Smart TV area, during the course of the internship emerged the need to perform the studies of the two Smart TV technologies – SSTV and Opera TV. SSTV is more mature platform compared to Opera TV thus it provides more comprehensive documentation and tutorials for the developers. For the SSTV Convergence Application case study we did not find a way to perform the discovery in accordance with the company's needs, nevertheless the problem was identified.

Finally, what all technologies that were used have in common is that they all support HTML5, therefore the ultimate conclusion is that HTML5 is a technology with a great potential that allows applications to be developed for diverse platforms such as desktop, mobile and TV.

6.1 Achievements

All of the objectives defined in Section 1.2 were achieved. We performed the studies for Tizen, Firefox OS, PhoneGap, Titanium, SSTV and Opera TV. With these studies

PTECH gained better understanding and knowledge of the platforms it intends to use. For each one of the platforms we conducted a case study in order to prove its feasibility for the PTECH's needs in practice. Thus, the company has working prototypes and findings that can use in its projects. Furthermore, for the case studies of PhoneGap, Titanium and Opera TV the Phune Gaming client application was used, therefore PTECH has concrete results which will consider for the implementation of Phune Gaming. In addition, for each one of the studies was elaborated a report to serve as internal reference material and the company's wiki pages were updated, thus the information is easily accessible for the PTECH's employees whenever it is needed.

Personally, the internship gave me the opportunity to be in a company that is established in the area and working with its professionals helped me understand its needs and the decisions that have to be made in real situations. I further improved my research skills and the ability to apply the theory to practical work. Moreover, performing the internship in a company that works with new technologies arouse my interest in new technologies, to be aware of them and possibly to use them in my work. Finally, I gained knowledge of the app development for mobile and TV which I find to be two areas with very good career perspectives and I would like to specialize in them in the future.

Therefore, considering the positive outcome for the company and my own professional growth in the area I can conclude that the internship was a success.

6.2 Limitations and Difficulties

The main limitation of the case studies that were conducted during the internship is caused by the fact that we did not have some of the actual devices for the testing. Tizen, Firefox OS, Opera TV and BlackBerry 10 are new platforms and thus PTECH could not acquire their devices by the end of the internship. For SSTV the company has an actual TV, yet its model does not support the functionality that PTECH intends to use. Hence, for these platforms we could test only on emulators, nevertheless we are aware that the applications may have different behavior when running on actual devices.

Besides the hardware limitations, for the PhoneGap and Titanium case studies we faced some software limitations since Cordova supports older versions of Tizen and Bada, and the Titanium support for Tizen was recent. In addition, some of the target OS's require the development to be only on specific platforms, such as iOS apps can be developed

only on Mac OS and Windows Phone 8 apps only on Windows 8. However, these problems were overcome with the help of PTECH, whenever it was possible they found an alternative solution that allowed me to fulfill the tasks.

The main difficulty of this internship was that when I started it I had no background of the mobile application development as well as for TV. Moreover, I used existing applications whose source code I did not know, thus initially it was difficult to understand it. For the tasks of the packaging tools I needed to become familiar with the different platforms and the tools they use for the development process. I further consider difficulty the fact that the technologies that were used in the internship are new and rapidly changing platforms, thus the main source of information were the documentation and posts published on the platforms' official websites. Nevertheless, these difficulties were overcome with time and now I can say with certainty that they greatly contributed to my learning.

6.3 Future Work

Referring to the limitations presented in the previous subsection the future work that would follow as a natural consequence from them is to repeat the tests on actual devices. PTECH already acquired a Tizen device and will perform the tests on it. In addition, the company plans to acquire an actual device running Firefox OS and a SSTV model 2013. This will allow the tests to be repeated on the respective actual devices and thus PTECH will gain more confidence in the behavior of the applications.

The prototype for the Tizen Facebook Chat application performs the authentication using the DIGEST-MD5 authentication mechanism. However, Facebook provides one more authentication mechanism called X-FACEBOOK-PLATFORM, which permits to connect to Facebook Chat using the Facebook authentication. This mechanism is more desirable, because it provides a better user experience and integration with the Facebook Platform. Therefore, the prototype can be improved by implementing the X-FACEBOOK-PLATFORM mechanism for the authentication.

PTECH already started the creation of UI for Phune Gaming that is customized for TV in order to target SSTV and Opera TV Store. Regarding SSTV it will explore in depth how to perform the discovery for the Convergence Application in a more efficient way. The Phune Gaming problems on Opera TV are also part of the future work.

In the meantime the hybrid packaging was made available in Tizen. It should be explored since it can provide more flexibility for the PTECH's applications.

These are some of the points that the company will realize in the foreseeable future as a result of the work performed during the internship.

7. Bibliography

[1] *"About Apache Cordova"*, Retrieved [Online] May 6th, 2013, [Address] http://cordova.apache.org/

[2] *"About the Project"*, Retrieved [Online] May 13th, 2013, [Address]

http://phonegap.com/about/

[3] *"Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap"*, Retrieved [Online] May 6th, 2013, [Address] http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html

[4] *"Adobe PhoneGap Build"*, Retrieved [Online] May 6th, 2013, [Address] https://build.phonegap.com/

[5] *"An Introduction to Tizen"*, Retrieved [Online] January 27th, 2013, [Address] http://www.youtube.com/watch?v=TkPqPz4Y03M

[6] *"Android & iOS generated 90% of smartphone shipments in Q1- IDC"*, Retrieved [Online] June 5th, 2013, [Address] http://www.mobileworldlive.com/android-ios-dominated-90-of-shipments-in-q1-idc

[7] *"Apache Cordova Documentation"*, Retrieved [Online] May 15th, 2013, [Address] http://cordova.apache.org/docs/en/2.6.0/

[8] *"App development API reference"*, Retrieved [Online] March 8th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Apps/Reference

[9] *"Application does not Launch on Samsung Platform"*, Retrieved [Online] March 5th, 2013, [Address] http://www.samsungdforum.com/Guide/tec00127/index.html

[10] *"Appcelerator Raises $4.1 Million for Open Source RIA Platform"*, Retrieved [Online] May 30th, 2013, [Address] http://techcrunch.com/2008/12/09/appcelerator-raises-41-million-for-open-source-ria-platform/

[11] *"App manifest"*, Retrieved [Online] March 8th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Apps/Manifest

[12] *"Application Fundamentals Developer Guide"*, Retrieved [Online] January 21st, 2013, [Address] https://developer.tizen.org/documentation/application-fundamentals-developer-guide

[13] *"Building Applications for the Opera TV Store"*, Retrieved [Online] June 20th, 2013, [Address] http://dev.opera.com/articles/view/building-applications-for-the-opera-tv-store/

[14] *"Can I use CSS pointer-events?"*, Retrieved [Online] June 20th, 2013, [Address] http://caniuse.com/pointer-events

[15] *"Client (HHP) to TV Application Communication"*, Retrieved [Online] March 4th, 2013, [Address] http://www.samsungdforum.com/Guide/ref00003/convergence_app_clienttotvappcomm.html

[16] *"Coding Your JavaScript Application"*, Retrieved [Online] February 26th, 2013, [Address] http://www.samsungdforum.com/Guide/art00011/index.html

[17] *"Common Modules API"*, Retrieved [Online] March 5th, 2013, [Address] http://www.samsungdforum.com/Guide/ref00006/index.html

[18] *"Comparing Titanium and PhoneGap"*, Retrieved [Online] May 3rd, 2013, [Address] http://developer.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap.html

[19] *"Convergence App"*, Retrieved [Online] March 1st, 2013, [Address] http://www.samsungdforum.com/Guide/art00027/index.html

[20] *"Cordova Contributors: Who's who"*, Retrieved [Online] May 6th, 2013, [Address] http://wiki.apache.org/cordova/who

[21] *"Cordova Tizen",* Retrieved [Online] May 13th, 2013, [Address] https://github.com/apache/cordova-tizen

[22] *"Creating a Convergence Application"*, Retrieved [Online] March 4th, 2013, [Address] http://www.samsungdforum.com/Guide/tut00024/index.html

[23] *"Creating web content for TV"*, Retrieved [Online] June 21st, 2013, [Address]

http://dev.opera.com/articles/view/creating-web-content-for-tv/

[24] Cromar S. (2010), *"Smartphones in the U.S.: Market Analysis"*

[25] *"Debugger"*, Retrieved [Online] March 14th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Tools/Debugger

[26] *"Debugging and Testing Applications"*, Retrieved [Online] March 8th, 2013, [Address] http://www.samsungdforum.com/Guide/art00012/index.html

[27] *"Design considerations for Opera TV Store applications"*, Retrieved [Online] June 21st, 2013, [Address]

http://dev.opera.com/articles/view/design-considerations-for-opera-tv-store-applications/

[28] *"Dev Guide"*, Retrieved [Online] January 8th, 2013, [Address] https://developer.tizen.org/documentation/dev-guide

[29] *"Development Guide"*, Retrieved [Online] March 5th, 2013, [Address] http://www.samsungdforum.com/Guide/

[30] *"Device Discovery, Authentication, and Pairing"*, Retrieved [Online] March 1st, 2013, [Address] http://www.samsungdforum.com/Guide/art00030/index.html

[31] *"Devices"*, Retrieved [Online] January 7th, 2013, [Address] https://www.tizen.org/about/devices

[32] *"Early Thoughts on New Operating Systems – Ubuntu, Sailfish (Jolla/MeeGo), Tizen (Samsung's update to MeeGo), Firefox; and some updates to classics BB10, WP8"*, Retrieved [Online] February 10th, 2013, [Address] http://communities-dominate.blogs.com/brands/2013/01/early-thoughts-on-new-operating-systems-ubuntu-sailfish-jollameego-tizen-samsungs-update-to-meego-fi/comments/page/1/

[33] Espial Group (2012), *"HTML5 Technologies for the Smart TV Experience"*

[34] *"Experience the Wonder of Samsung's New Smart TVs"*, Retrieved [Online] March 14th, 2013, [Address]

http://www.samsung.com/us/2012-smart-tv/smart-content.html

[35] *"Experimental Firefox OS software for Xperia™ E available for developers [ROM]"*, Retrieved [Online] March 8th, 2013, [Address] http://developer.sonymobile.com/2013/02/27/experimental-firefox-os-software-for-xperia-e-available-for-developers-rom/

[36] *"Facebook Chat API"*, Retrieved [Online] July 11th, 2013, [Address] https://developers.facebook.com/docs/chat/

[37] *"File Properties"*, Retrieved [Online] July 11th, 2013, [Address] http://msdn.microsoft.com/en-us/library/0c6xyb66(v=vs.80).aspx

[38] *"Finding a new name that isn't PhoneGap"*, Retrieved [Online] May 6th, 2013, [Address] http://markmail.org/message/vcrw2swiwiwcojsd

[39] *"Firefox OS architecture"*, Retrieved [Online] March 11th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture

[40] *"Firefox OS Simulator"*, Retrieved [Online] March 11th, 2013, [Address] https://marketplace.firefox.com/developers/docs/firefox_os_simulator

[41] *"From MeeGo to Tizen: The Making of Another Software Bubble"*, Retrieved [Online] January 29th, 2013, [Address] http://www.visionmobile.com/blog/2011/10/from-meego-to-tizen-the-making-of-another-software-bubble/

[42] *"Functional key handling in Opera TV Store applications"*, Retrieved [Online] June 20th, 2013, [Address] http://dev.opera.com/articles/view/functional-key-handling-in-opera-tv-store-applications/

[43] *"Functional Buttons"*, Retrieved [Online] June 20th, 2013, [Address] https://publish.tvstore.opera.com/developer/guidelines/functional-keys/

[44] *"Getting started with app development"*, Retrieved [Online] March 12th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Apps/Getting_Started

[45] *"Getting Started with PhoneGap and PhoneGap Build"*, Retrieved [Online] May 6th, 2013, [Address] http://phonegap.com/blog/2013/02/18/getting-started-with-phonegap-and-phonegap-build/

[46] *"Getting Started with Windows Phone 8"*, Retrieved [Online] May 14th, 2013, [Address] http://cordova.apache.org/docs/en/2.6.0/guide_getting-started_windows-phone-8_index.md.html#Getting%20Started%20with%20Windows%20Phone%208

[47] *"Hydration"*, Retrieved [Online] May 6th, 2013, [Address] https://build.phonegap.com/docs/hydration

[48] IBM Software Group (2012), *"Native, web or hybrid mobile-app development"*

[49] *"Introduction to Firefox OS"*, Retrieved [Online] March 11th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Introduction

[50] *"Intro to Open Web apps"*, Retrieved [Online] March 19th, 2013, [Address] https://marketplace.firefox.com/developers/docs/intro_apps

[51] *"Jay Sullivan at the Mozilla MWC Press Conference"*, Retrieved [Online] June 5th, 2013, [Address] https://air.mozilla.org/jay-sullivan-mwc/

[52] Jewett F. (2011), *"Why Smart TV is the Next Big Thing"*, Mobile Connect, 2, 3-5

[53] *"Making Elements Focusable with Tabindex"*, Retrieved [Online] June 20th, 2013, [Address] http://snook.ca/archives/accessibility_and_usability/elements_focusable_with_tabindex/

[54] *"MeeGo"*, Retrieved [Online] January 27th, 2013, [Address] http://en.wikipedia.org/wiki/MeeGo

[55] *"Mozilla Announces Global Expansion for Firefox OS"*, Retrieved [Online] March 8th, 2013, [Address] http://blog.mozilla.org/press/2013/02/firefox-os-expansion/

[56] *"Mozilla's Mobile Firefox OS Raises Security Questions"*, Retrieved [Online] June 5th, 2013, [Address] http://www.technologyreview.com/news/511706/mozillas-mobile-firefox-os-raises-security-questions/

[57] *"Mozilla VP talks Firefox OS, the threat from Tizen and low-quality HTML5 apps"*, Retrieved [Online] June 5th, 2013, [Address] http://thenextweb.com/mobile/2013/02/27/mozilla-vp-talks-firefox-os-the-threat-from-tizen-and-poor-quality-html5-apps/

[58] *"Namespace List"*, Retrieved [Online] April 17th, 2013, [Address] https://developer.tizen.org/help/topic/org.tizen.native.apireference/namespaces.html

[59] *"New eye candy from Opera-powered Samsung Blu-ray players"*, Retrieved [Online] June 24th, 2013, [Address] http://business.opera.com/press/releases/devices/2013-04-30

[60] Nosrati M., Karimi R. & Hasanvand H. (2012), *"Mobile Computing: Principles, Devices and Operating Systems"*, World Applied Programming, 2(7), 399-408

[61] Opera Software, *"Opera Devices SDK 3.5 Connected TV within your reach"*, Product Sheet

[62] Opera Software, *"Opera TV browser Full internet browsing for TVs"*, Product Sheet

[63] *"Opera TV Emulator build and test HTML5 and CE-HTML content for TVs"*, Retrieved [Online] June 24th, 2013, [Address] http://business.opera.com/solutions/tv/emulator

[64] *"Opera TV Store: Applications made for TV"*, Product Sheet Retrieved [Online] June 20th, 2013, [Address] http://business.opera.com/solutions/tv

[65] *"Opera TV Store Application Publishing Guidelines"*, Retrieved [Online] June 20th, 2013, [Address] https://publish.tvstore.opera.com/guidelines/

[66] *"Opera TV Store app templates"*, Retrieved [Online] June 21st, 2013, [Address] http://dev.opera.com/articles/view/opera-tv-store-app-templates/

[67] *"Opera TV Store at retail devices"*, Retrieved [Online] June 21st, 2013, [Address] http://my.opera.com/community/forums/topic.dml?id=1428782

[68] *"Packaged apps"*, Retrieved [Online] March 12th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Apps/Packaged_apps#Types_of_packaged_apps

[69] Park J., *"Smart TV Technology and Service"*

[70] *"PhoneGap, Cordova, and what's in a name?"*, Retrieved [Online] May 6th, 2013, [Address] http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/

[71] *"PhoneGap Explained Visually"*, Retrieved [Online] May 13th, 2013, [Address] http://phonegap.com/2012/05/02/phonegap-explained-visually/

[72] *"PhoneGap Support"*, Retrieved [Online] May 6th, 2013, [Address] http://phonegap.com/support/

[73] *"Push Messaging"*, Retrieved [Online] April 16th, 2013, [Address] https://developer.tizen.org/help/topic/org.tizen.native.appprogramming/html/guide/messaging/push_messaging.htm

[74] *"Quietly, Opera is working on becoming a Smart TV powerhouse"*, Retrieved [Online] June 21st, 2013, [Address] http://gigaom.com/2013/03/27/opera-smart-tv/

[75] *"Samsung to Merge Bada with Tizen"*, Retrieved [Online] January 30th, 2013, [Address] http://www.sammobile.com/2012/10/09/samsung-to-merge-bada-with-tizen/

[76] Samsung Smart TV (2011), *"Application Development Guide for Samsung Smart TV"*

[77] *"Samsung SmartTV SDK 4.0 Overview"*, Retrieved [Online] February 26th, 2013, [Address] http://www.samsungdforum.com/Guide/d08/index.html

[78] *"Samsung unveils Evolution Kit at CES 2013 to complete an evolving Smart TV"*, Retrieved [Online] March 8th, 2013, [Address] http://www.samsung.com/us/aboutsamsung/news/newsIrRead.do?news_ctgry=irnewsrelease&news_seq=20329

[79] *"Setting up to debug on Firefox OS using Firefox developer tools"*, Retrieved [Online] March 14th, 2013, [Address] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Debugging/Setting_up

[80] *"SDK Release Note"*, Retrieved [Online] March 5th, 2013, [Address] http://samsungdforum.com/Devtools/SdkReleaseNote#4b2d3a2e448e2bbe

[81] *"Spec & Features"*, Retrieved [Online] March 6th, 2013, [Address] http://samsungdforum.com/Devtools/Spec

[82] *"Testing your app inside the Opera TV Store"*, Retrieved [Online] June 21st, 2013, [Address] http://dev.opera.com/articles/view/testing-your-app-inside-the-opera-tv-store/

[83] *"Titanium Documentation"*, Retrieved [Online] May 23th, 2013, [Address] http://docs.appcelerator.com/titanium/latest/

[84] *"Titanium Mobile Development Environment"*, Retrieved [Online] May 30th, 2013, [Address] http://www.appcelerator.com/platform/titanium-platform/

[85] *"Titanium Mobile Tizen now available"*, Retrieved [Online] May 28th, 2013, [Address] http://developer.appcelerator.com/blog/2013/03/titanium-mobile-tizen-now-available.html

[86] *"Tizen 1.0 Larkspur"*, Retrieved [Online] January 28th, 2013, [Address] https://source.tizen.org/release/tizen-1.0-larkspur

[87] *"Tizen Architecture Overview"*, Retrieved [Online] January 28th, 2013, [Address] https://www.tizen.org/sites/default/files/tizen-architecture-linuxcollab.pdf

[88] *"Tizen SDK 2.0 Alpha Release Notes"*, Retrieved [Online] January 28th, 2013, [Address] https://developer.tizen.org/downloads/sdk/2.0-alpha-release-notes

[89] Vision Mobile (2006), *"Mobile Operating Systems: The New Generation"*

[90] Vision Mobile (2011), *"Open Governance Index Measuring the True Openness of Open Source Projects from Android to WebKit"*

[91] *"WebAPI"*, Retrieved [Online] March 18th, 2013, [Address] https://wiki.mozilla.org/WebAPI

[92] *"Web specifications support in Opera"*, Retrieved [Online] June 24th, 2013, [Address] http://www.opera.com/docs/specs/

[93] *"Why you should stay away from Appcelerator's Titanium"*, Retrieved [Online] May 24th, 2013, [Address] http://usingimho.wordpress.com/2011/06/14/why-you-should-stay-away-from-appcelerators-titanium/

8. Appendices

This chapter presents the publicly available appendices of the report.

8.1 Internship Proposal

PROPOSTA DE ESTÁGIO

Ano Lectivo de 2012 / 2013

Mestrado em Informática e Sistemas (Desenvolvimento de Software ou Tecnologias da Informação e do Conhecimento)

TEMA

Mobile Web Applications

SUMÁRIO

Este estágio tem como objectivo o estudo de várias ferramentas e frameworks orientadas ao desenvolvimento de aplicações web mobile. Pretende-se que no final do estágio a Present Technologies passe a ter conhecimento na área do desenvolvimento e packaging de aplicações web em diferentes plataformas mobile.

- **ÂMBITO**

A área mobile web tem vindo a crescer exponencialmente nos últimos anos, sendo que esta é uma área muito dinâmica onde estão constantemente a surgir novas plataformas e frameworks de desenvolvimento. Este estágio irá centrar-se no estudo e avaliação em diferentes áreas:

- Sistemas operativos mobile
- Ferramentas de packaging
- Frameworks para desenvolvimento

- **OBJECTIVOS**

O presente projecto pretende atingir os seguintes objectivos:

- Estudo do sistema operativo Tizen
- Estudo e avaliação de ferramentas de packaging para web applications

- Estudo e avaliação de mobile web frameworks

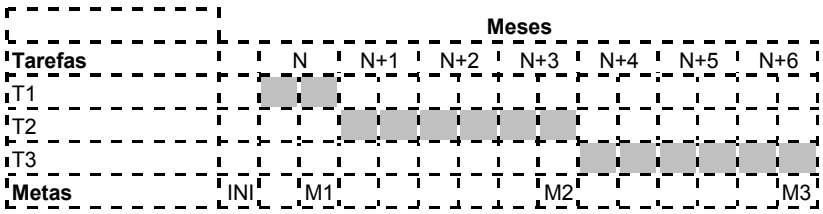
• PROGRAMA DE TRABALHOS

O estágio consistirá nas seguintes actividades e respectivas tarefas:

- T1 – Mobile Operating Systems:
  - Estudo da plataforma Tizen
- T2 – Web applications packaging tools:
  - Estudo e avaliação da ferramenta PhoneGap
  - Estudo e avaliação da ferramenta Appcelerator
  - Estudo e avaliação da ferramenta Sencha Touch
  - Estudo e avaliação da ferramenta Qt
  - Estudo e avaliação da ferramenta Rhodes
  - Comparação das ferramentas estudadas
- T3 – Mobile Web Frameworks:
  - Estudo e avaliação da framework jQuery Mobile
  - Estudo e avaliação da framework jQTouch
  - Estudo e avaliação da framework Sencha Touch
  - Estudo e avaliação da framework Jo
  - Estudo e avaliação da framework Yiibu
  - Comparação das frameworks estudadas

• CALENDARIZAÇÃO DAS TAREFAS

O plano de escalonamento dos trabalhos é apresentado em seguida:



INI		Início dos trabalhos
M1	(INI + 4 Semanas)	Tarefa T1 terminada
M2	(INI + 16 Semanas)	Tarefa T2 terminada
M3	(INI + 28 Semanas)	Tarefa T3 terminada

• RESULTADOS

Os resultados do estágio serão consubstanciados num conjunto de documentos e deliverables a elaborar pelo estagiário de acordo com o seguinte plano:

M1:

R1.1: Protótipo de uma aplicação web para o sistema operativo Tizen.

R1.2: Documento que descreva o sistema operativo Tizen, a sua arquitectura e a forma de desenvolver aplicações para este.

M2:

R2.1: Packaging de uma aplicação recorrendo a cada uma das ferramentas estudadas. A aplicação deverá ser testada nos diferentes sistemas operativos suportados. A aplicação a testar será disponibilizada pela Present Technologies.

R2.2: Documento com detalhes de cada ferramenta de packaging estudada e com um comparativo entre as mesmas.

M3:

R3.1: Desenvolvimento de um protótipo recorrendo a cada uma das frameworks estudadas.

R2.2: Documento com detalhes de cada framework de desenvolvimento estudada e com um comparativo entre as mesmas.

- ## LOCAL DE TRABALHO

O estágio decorrerá nas instalações da Present Technologies, em Coimbra em regime de full-time.

- ## METODOLOGIA

A metodologia de desenvolvimento de software seguirá o processo de desenvolvimento interno da Present Technologies, baseado no modelo em cascata/waterfall.

O acompanhamento do estágio será efectuado através de reunião regulares entre o orientador e o estagiário.

- ## ORIENTAÇÃO

ISEC:

Nome (nome@isec.pt)
Categoria

Entidade de Acolhimento:

Aurélio Santos (aurelio.santos@present-technologies.com)
Software Engineer

- **CARACTERIZAÇÃO E REMUNERAÇÃO**

  - Data de início: 02/01/2013
  - Data de fim: 31/07/2013
  - Horário: Será praticado o horário em vigor na Present Technologies

Remuneração: O estágio será não remunerado

8.2 Tizen

This appendix contains the additional material that was elaborated for the Tizen study.

8.2.1 Tizen Core Components

The description of the Tizen Core subsystem's components is as follows [86], [87]:

- Application Framework – provides functionality for application management, including launching other applications using the package name, Uniform Resource Identifier (URI), or Multipurpose Internet Mail Extensions (MIME) type. In addition it notifies applications for common events, such as low memory, low battery, changes in screen orientation and push notifications.
- Base – contains of Linux-based system libraries that provide features for database support, internationalization, and XML parsing. The Base is defined as self-sufficient and by using the packages in Base the system is able to boot itself to console/login.
- Connectivity – provides functionalities for networking and connectivity, such as 3G, Wi-Fi, Bluetooth, HTTP, and NFC. The Connection Manager is based on ConnMan[38].
- Graphics and UI – consist of the system graphic and UI stacks, which include Enlightenment Foundation Library[39] (EFL), an X11-based window management system, Input Service Framework (ISF), and OpenGL ES[40].

[38] https://connman.net/
[39] http://www.enlightenment.org/?p=about/efl
[40] http://www.khronos.org/opengles/

- Location – provides Location-Based Services (LBS), which contain information for position, geocoding, satellite, and GPS status. It is based on GeoClue[41], which delivers location information from GPS, Wi-Fi Positioning System (WPS), Cell ID, and sensors.
- Messaging – supports functionality for sending and receiving SMS, MMS, and email messages;
- Multimedia – it is based on GStreamer[42] and provides functionality for playing and manipulation of audio, video, images, and VoIP. The Audio server functionality is based on PulseAudio[43].
- PIM – enables the managements of user data on the device, such as calendar, contacts, and tasks, and also enables retrieving data about the device context (e.g. device position).
- Security – it is responsible for the security deployment across the system and consists of platform security enablers, such as: access control; certificate management, and secure application distribution. Security is based on Simplified Mandatory Access Control Kernel[44] (SMACK).
- System – provides system and device management functionality, including interfaces for accessing devices, such as: sensors; display; vibrator; monitoring devices and handling events (e.g. USB, MMC, charger, and ear jack events); power management.
- Telephony – provides functionality for cellular and VoIP calls, such as: managing call-related and non-call-related information and services for Universal Mobile Telecommunications System (UMTS) and Code Division Multiple Access (CDMA).
- Web – provides a complete implementation of Tizen Web API optimized for mobile devices. It includes the WebKit rendering engine.

[41] http://www.freedesktop.org/wiki/Software/GeoClue/
[42] http://gstreamer.freedesktop.org/
[43] http://www.freedesktop.org/wiki/Software/PulseAudio/
[44] http://schaufler-ca.com/

8.2.2 Tizen Web Device APIs

Table 4 provides the description for the APIs part of the Tizen Web Device API set [28].

Table 4 Tizen Web Device APIs

API	Description
Tizen	Provides basic definitions that are used in all other Tizen Web Device APIs, such as generic callback for success and error, WebAPIError and WebAPIException interfaces, and different types of filters.
Alarm	Provides functionality for setting and unsetting alarms. Each client application has its own alarm storage, thus it cannot see an alarm that is set by another application.
Application	Provides functionality for launching other applications and thus they can process a task and return the result to the caller application.
Bluetooth	Provides access to diverse Bluetooth functionalities.
Calendar	Provides functionality for creating, deleting, reading and updating items in specific calendars.
Callhistory	Provides access to call history for cellular and VoIP calls.
Contact	Provides functionality for creating, deleting, reading and updating contacts in specific address books.
Content	Provides functionality to discover multimedia contents on a device.
Download	Provides functionality for downloading remote objects by HTTP requests.
Filesystem	Provides access to the file system of a device.
Messaging	Provides functionality for sending and receiving SMS, MMS, and Email messages.
NFC	Provides access to NFC devices.
Notification	Provides functionality to notify the user for events that happen in the application.
Power	Provides functionality for requesting resource states related to the power management (e.g. display brightness).
System Information	Provides information for the hardware of a device, such as device's display, storage, network and other capabilities.

API	Description
System Setting	Provides functionality for system settings.
Time	Provides information about date, time and time zones.

8.2.3 W3C/HTML5 API

In **Table 5** are presented the W3C APIs divided into categories based on their functionality [28]. Some of the APIs are stable while others are draft specifications and thus they are subject to change.

Table 5 W3C APIs

Category	Specifications
Communication	• The WebSocket API • HTML5 Web Messaging • XMLHttpRequest Level 2 (Partial) • HTML5 The session history of browsing contexts (Partial) • Server-Sent Events
Device	• Touch Events (Partial) • Device Orientation Event Specification (Partial) • Battery Status API • Vibration API • HTML5 Browser state • The Screen Orientation API • The Network Information API
DOM, Forms and Styles	• HTML5 Forms (Partial) • Selectors API Level 1 • Selectors API Level 2 (Partial) • Media Queries (Partial) • CSS 2D Transforms • CSS 3D Transforms Module Level 3 (Partial) • CSS Animations Module Level 3 • CSS Transitions Module Level 3

Category	Specifications
	• CSS Colors Module Level 3 • CSS Backgrounds and Borders Module Level 3 (Partial) • CSS Flexible Box Layout Module (Partial) • CSS Multi-column Layout Module (Partial) • CSS Text Module Level 3 (Partial) • CSS Basic User Interface Module Level 3 (CSS3 UI) (Partial) • CSS Fonts Module Level 3 (Partial) • Web Open Font Format (WOFF) File Format 1.0 • HTML5
Graphics	• HTML5 The canvas element (Partial) • HTML Canvas 2D Context • HTML5 SVG
Location	• Geolocation API Specification
Media	• HTML5 The video element (Partial) • HTML5 The audio element (Partal) • getUserMedia (Partial) • Web Audio API • HTML Media Capture
Performance and Optimization	• Web Workers (Partial) • Page Visibility API • Timing control for script-based animation
Security	• Cross-Origin Resource Sharing • HTML5 iframe element
Storage	• Web Storage (Partial) • File API • File API: Directories and System (Partial) • File API: Writer (Partial) • HTML5 Application Cache • Web SQL Database • Indexed Database API (Partial)

Category	Specifications
UI	• Clipboard API and events • HTML5 Drag and drop
Widget	• Widget Packaging and XML Configuration • Widget Interface • XML Digital Signatures for Widgets • Widget Access Request Policy

8.2.4 Supplementary API

Table 6 presents the non-W3C specifications supported by Tizen.

Table 6 Supplementary specifications description

Specification	Description
WebGL (Khronos Spec)	Describes an additional rendering context and support objects for the HTML 5 canvas element.
Typed Arrays (Khronos Spec)	Provides an API for interoperability with native binary data.
FullScreen API (Mozilla Spec)	Allows elements to be displayed in full screen mode programmatically.
viewport MetaTag (Apple Spec)	Allows the control of the viewport's size and scale.

The supplementary specifications are partially supported in the current release. The list of the supported functionalities can be seen in [28].

8.2.5 Tizen Native API

A list of all documented Tizen C++ namespaces along with brief descriptions is provided in **Table 7** [58]. More information for each namespace in the list can be found in the Tizen Native API Reference on [28].

Table 7 Namespace list

Namespace	Description
Tizen	The root namespace of the Tizen native framework
Tizen::App	Contains classes for application development
Tizen::App::Package	Contains classes and interfaces for a package
Tizen::Base	Contains classes and interfaces for basic features
Tizen::Base::Collection	Contains classes and interfaces for various collections
Tizen::Base::Runtime	Contains classes for running applications
Tizen::Base::Utility	Contains classes for various utilities
Tizen::Content	Contains classes and interfaces for content management and search services
Tizen::Graphics	Contains classes for drawing-related functionalities
Tizen::Graphics::Opengl	Contains interfaces for OpenGL
Tizen::Io	Contains classes and interfaces for performing basic I/O operations
Tizen::Locales	Contains classes that define culture-related information
Tizen::Locations	Contains classes and interfaces for location-related information and services
Tizen::Media	Contains classes and interfaces for media processing services
Tizen::Messaging	Contains classes and interfaces for messaging services
Tizen::Net	Contains classes and interfaces for network account, connection, and addressing utilities
Tizen::Net::Bluetooth	Contains classes and interfaces for Bluetooth services
Tizen::Net::Http	Contains classes and interfaces for HTTP 1.1 client programming
Tizen::Net::Nfc	Contains classes and interfaces for NFC services
Tizen::Net::Sockets	Contains classes and interfaces for Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) socket programming
Tizen::Net::Wifi	Contains classes and interfaces for Wi-Fi management and Wi-Fi Direct functionalities
Tizen::Security	Contains classes and interfaces for security services
Tizen::Security::Cert	Contains classes and interfaces for managing the X.509

Namespace	Description
	digital certificate
Tizen::Security::Crypto	Contains classes and interfaces for the cryptographic primitives
Tizen::Shell	Contains classes for phone shell management
Tizen::Social	Contains classes and interfaces for managing the user's social information
Tizen::System	Contains classes and interfaces for System
Tizen::Telephony	Contains classes and interfaces of the Telephony service
Tizen::Text	Contains classes that encode and decode characters
Tizen::Ui	Contains classes and interfaces that act as the UI foundation for the applications
Tizen::Ui::Animations	Contains classes for animation-related functionalities
Tizen::Ui::Controls	Contains classes and interfaces for creating rich user interface components for the applications
Tizen::Ui::Effects	Contains classes and interfaces for effect-related functionalities
Tizen::Ui::Scenes	Contains the classes for the scene management and its related functions
Tizen::Uix	Contains the Ui extension classes and the Tizen interfaces
Tizen::Uix::Sensor	Contains Sensor classes and Tizen interfaces
Tizen::Uix::Speech	Contains classes for speech-related functions
Tizen::Uix::Vision	Contains the classes for face-related functions
Tizen::Web	Contains classes to manage the history data
Tizen::Web::Controls	Contains classes and interfaces to interact with the browser engine
Tizen::Web::Json	Contains interfaces to manipulate JSON documents

8.2.6 Tizen Web App Configuration File

The Tizen IDE provides an editor for the config.xml file; thus it makes it easier to preserve the required XML schema. The only mandatory element of the configuration file is the widget element. All other elements and their respective attributes are optional. The optional elements that can be modified using the editor are described in **Table 8** [28].

Table 8 Configurations of the config.xml file

Type	Description
Identifier	Identifier for the widget.
Version	Widget version attribute, indicates the current version of the widget.
Name	Full human-readable name for a widget that is used, for example, in an application menu or in other contexts.
Content	Custom start file the user agent is expected to use when it instantiates the widget.
Icon	Custom icon for the widget.
Author	People or an organization associated with the creation of the widget.
E-mail	Email address associated with the author.
Web Site	IRI (a URL that contains characters from the Universal Character Set (UCS)) associated to author (e.g. a homepage, a profile on a social network, etc.).
License	Software license, which may include: a usage agreement, redistribution statement, and/or a copyright license terms under which the content of the widget package is provided.
License URL	Valid IRI or a valid path that points to a representation of a software and/or content license.
Description	Human-readable description of the widget.
Widget UI Width	Preferred viewport width of the instantiated custom start file.
Widget UI Height	Preferred viewport height of the instantiated custom start file.
View Modes	Author's preferred view mode (full screen, floating, windowed, maximized, and minimized).

The editor also allows the management of some Tizen specific information about the widget and the source code of the config.xml file, as well as the configuration of the elements: feature, access, preference and localization. The purpose of these elements is described below:

- feature – indicates the Tizen APIs that the widget needs to access at runtime;
- access – indicates the permissions that the widget needs to access network resources. All the URLs that the widget needs to access must be defined and for

each URL has to be indicated if the widget is allowed to access the URL sub-domains;

- preference – indicates the preferences that are associated with the widget the first time it is initiated;
- localization – defines the localization for elements of the config.xml file.

8.3 Firefox OS

This appendix presents the additional material that was elaborated for the Firefox OS study.

8.3.1 API Reference

This subsection presents the API support of Firefox OS published on [8] as of March 2013. Some of the Web APIs are still under development thus their current status can be consulted on [91].

8.3.1.1 Firefox Device APIs

The Firefox OS Device APIs expose access to device's features. These APIs are summarized in **Table 9**.

Table 9 Firefox OS Device APIs

API	Description
Alarm	Provides access to the alarm settings of the device in order to schedule a notification or an application to be started.
Audio Policy	Introduces the concept of a hierarchy of audio channels which gives priority to the sounds of the different channels.
Browser	Allows the app to implement a browser.
Contacts	Gives access to the contacts from the device's address book and from the SIM card.
Desktop Notification	Enables displaying notifications on the screen.
Device Storage	Allows manipulating picture, audio and video files stored on the device or on the SD card.
FM Radio	Provides access to the FM radio of the device to turn it on/off and

API	Description
	change the radio stations.
Geolocation	Enables the app to obtain the user's current location.
Storage	Allows utilizing storage without size limitation, for example for application caching or IndexedDB (an API that enables to store significant amounts of structured data).
SystemXHR	Permits anonymous cross-origin XMLHttpRequest [45]even when Cross-Origin Resource Sharing (CORS) is not enabled in the target site.
TCP Socket	Supports the creation of TCP sockets and the communication over them.

8.3.1.2 General Web APIs

The General Web API consists of the standard Web APIs supported by the Firefox browser. These are presented in **Table 10**.

Table 10 General Web APIs

API	Description
Audio	Uses the HTML5 <audio> tag to embed and manipulate audio content.
Device orientation	Detects changes in the device's current orientation using the orientation sensors on the device.
DOM events	Lists all the events that can be used to interact with DOM objects.
Geolocation	Allows the app to request and use the current location of the user.
History	Enables the access to the browser's history.
IndexedDB	Provides an interface for storing and retrieving large amounts of data on the device.
Network requests	Uses XMLHttpRequest to send and receive data via HTTP.
Online and offline events	Enables the app to respond to changes in the network connection.
Screen orientation	Detects changes in the screen orientation of the device.
Storage	Provides various ways to store small amounts of data on the

[45] http://www.w3.org/TR/XMLHttpRequest/

API	Description
	device.
Touch events	Provides functionality for supporting touch events.
Video	Uses the HTML5 <video> tag to embed and manipulate video content.
Web workers	Allows scripts to be run in background threads.

8.3.1.3 Firefox Marketplace Services

The APIs presented in **Table 11** support publishing and managing apps on the Firefox OS Marketplace.

Table 11 Firefox Marketplace APIs

API	Description
Marketplace	Makes available all documentation related to the Marketplace.
Payment	Allows information about the available pricing tiers to be obtained and also processing in-app purchases.
Submission	Supports the process of publishing an app which includes validation, creation, update, etc.

8.4 Samsung Smart TV

This appendix presents the additional material that was elaborated for the SSTV study.

8.4.1 Common Module's Objects

Table 12 [17] presents the objects provided by the Common Modules component.

Table 12 Common Modules' objects

Object	Description
TVKeyValue	Defines TV key code.
Widget	Provides functions needed for running an application.
Plugin	Allows some plugin functions to be used.
CimageViewer	Enables JPEG files to be displayed in Samsung DTV (Digital

Object	Description
Module	Television).
IME (Input Method Editor) Module	Enables text input in applications via the remote controller.
SSO Module	Enables SSO in applications.
Common popup IME	Provides functions for the popup IME.
IMECN Module	Chinese IME Module

8.4.2 Device APIs

Table 13 presents the APIs in the Device API set along with their description [29].

Table 13 Device APIs

API	Description
AppCommon	Deals with functions for key registration.
Audio	Controls audio related functions.
Common	Describes common functions of all plugins.
Download	Downloads file asynchronously to the DTV platform using HTTP or HTTPS protocol.
External Widget Interface	Provides functions for account management.
Filesystem	Controls the file system on the DTV platform.
FrontPanel	Displays the Blu-ray disc player.
ImageViewer	Displays JPEG image.
IME	Enables text input in applications.
Network	Controls and gets network relative information.
Nnavi	Controls SSTV specific functions.
Player	Plugin for multimedia playback.
Screen	Deals with 3D effect functions of TV screen.
TaskManager	Deals with inter-task action of TV.
Time	Deals with time functions of TV.
TV	Handles the EPG (Electronic Program Guide) functions of TV.

API	Description
TVMW	Controls various functionalities on the DTV platform, including country, language, input source, etc.
Video	Controls video related functions.
Window	Deals with channel and screen functions of TV.

8.4.3 SSTV API Summary

The APIs supported by the SSTV are summarized in **Table 14**. More details can be found on [29].

Table 14 SSTV API reference

API	Description
Advertisement Service	Provides advertisement functionality.
AllShare	Provides a set of interfaces that are used for developing convergence services such as contents sharing, device control, etc.
AppsFramework	Includes the following APIs: Core, Scene Manager, Util, Service, and UI Components.
Common Modules	Provides information about some general purpose objects that can be used in applications.
Convergence App	Provides a REST-based interface to allow devices that support the HTTP protocol to communicate with a SSTV.
Device	Provides access to some middleware DTV features.
File	Allows the I/O functionality of build-in flash memory to be used.
In-Application Purchase	Enables users to purchase items using their Samsung Apps accounts.
Interactive Remote	Allows a SSTV application to be controlled by using a remote application launched on a smartphone.
Interactive Mobile Device	Supports the interaction between a mobile device and the SSTV.
SEF (Service Extension	Provides the functionality to call native C++ middleware from JavaScript. It has the same functions as Device API.

API	Description
Framework) Plugin	
Web Device	Enables the utilization of SSTV middleware functions, such as access to the file system, smart interactions, audio and video control, etc. The Web Device API is an alternative to the Device APIs.

8.4.4 Configuration of SSTV config.xml

Table 15 presents a description and possible values for each element of the config.xml file [16].

Table 15 Config.xml elements description

Element	Description	Value
<widget>	The parent element for the application.	-
<ThumbIcon>	An icon image displayed in the Application Manager. It is used in case of no focus and its size is 106 x 86 pixels.	File path
<BigThumbIcon>	An icon image displayed in the Application Manager. It is used in case the focus is placed on an image and its size is 115 x 95 pixels.	File path
<ListIcon>	An icon image displayed in the Application Manager. The size is 85 x 70 pixels.	File path
<BigListIcon>	An icon image displayed in the Application Manager. The size is 95 x 78 pixels.	File path
<category>	The category of the application. The possible values are: video; sports; game; lifestyle; information; education.	String
<autoUpdate>	Defines whether to synchronize with the hub site.	y | n
<apptype>	The contents type of the application. 11: HTML + JavaScript + Flash Player Object 12: Adobe SWF (Ver. Flash Lite 3.1) 13: Adobe SWF (Ver. Flash 10.1) 14: Lua Script	Number
<contents>	File path and name to the initial execution of contents. This tag is required only for the following application	File Path

Element	Description	Value
	types: 12: Adobe SWF (Ver. Flash Lite 3.1) 13: Adobe SWF (Ver. Flash 10.1) 14: Lua Script	
<channelType>	Channel-bound Service Type (optional)	root | child
<channelRoot>	Defines the root-child relation where the root is application ID (optional, only used when the channel-bound service type is the child). When connected to more than one root, the roots are separated by '::'.	Application ID
<channelName>	Channel information to be executed for channel-bound service (optional, only used when the channel-bound service type is the root). Each channel is separated by using '::' (e.g.: AAA::BBB::CCC).	String
<channelDisplay>	Defines whether the installed channel-bound service is displayed on the first main screen or not.	y | n
<cpname>	Defines the application provider.	String
<cpauthjs>	Defines the name of the JavaScript file, which allows the account information of application providers to be confirmed. This file has to be written in a defined format.	String
<login>	Defines whether or not a service is available for login. If 'y' is selected, ID and password have to be entered in the Integrated Sign-in site of the Application Manager for login. Validity verification should be performed in the JavaScript file defined in the <cpauthjs> element.	y | n
<ver>	Defines the application version, which is needed for the application updates.	x.xxx
<mgrver>	Defines the version of the Application Manager, which is required to run the application.	x.xxx
<fullwidget>	Defines whether the application is full-screen or single-wide. The display type affects the audio policy of the application when it is run.	y | n
<srcctl>	If 'y' is selected, the TV source automatically switches from the current TV channel or external input to the	y | n

Element	Description	Value
	internal media player and goes back when the application is completed.	
<childlock>	Defines whether to use the childLock function. This function allows the user to lock an application.	y | n
<audiomute>	Turns on/off the audio. If 'y' is selected the TV broadcasting sound is muted when entering the application. The 'y' value is selected for full screen and 'n' for single-wide application.	y | n
<videomute>	Turns on /off the video. If 'y' is selected, TV broadcasting is not displayed on the screen when entering the application.	y | n
<dcont>	Sets the "Disable dynamic contrast" function, which adjusts TV contrast and brighten TV screen ratio by darkening the dark screen and lightening the light screen. Selecting 'y' turns off the Dynamic contrast, and selecting 'n' turns on the Dynamic contrast. For full screen application 'y' should be selected to remove the sparkling.	y | n
<movie>	Applications that play video files can cause problems as stated below: 1. If the video file is played on a device connected to the HDMI port, such as a DVD player, sounds can get mixed, when executing an application converting sources (e.g. YouTube). 2. Sparkling can happen at the entry of the application, due to the difference of frame rate between the TV image and video file. Such problems can be avoided by selecting 'y' – the HDMI device is stopped, or the frame rate is fixed.	y | n
<widgetname>	Defines the name of the application.	String
<description>	Provides a brief description of the application.	String
<width> <height>	The screen area that the application will occupy. It is recommended to define 960 * 540 pixels – the DTV specification.	Number
<author>	Defines the name of the author.	String
<network>	This tag is used to check the network while the application is running. If the value is 'y' and the network test result is	y | n

Element	Description	Value
	'fail', entry for the application can be blocked with a message that indicates the failure. If no value is selected, the default is 'y'.	
<hubsite>	This tag is used to define if the hub site has been authorized or not while the application is running. If the tag value is 'y', and the hub site has not been authorized, entry for the application can be blocked with a message that indicates the failure. If no value is selected, the default is 'n'.	y | n
<pushNotice>	Defines if the application provides Push Notification Service. If no value is selected, the default is 'n'.	y | n
<pushControl>	This tag is reserved for former Push Notification Service. If no value is selected, the default is 'n'.	y | n
<pushUerbinding>	Defines if Push Notification Service is provided for a specific user. If no value is selected, the default is 'n'.	y | n
<flashplayer>	The 'y' value is selected for applications that use embedded Flash player objects or a stand-alone Flash player.	y | n