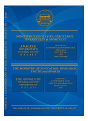
Annals of "Dunarea de Jos" University of Galati
Fascicle I. Economics and Applied Informatics
Years XXIII – nº1/2017
ISSN-L 1584-0409 ISSN-Online 2344-441X
www.eia.feaa.ugal.ro



Cross-Platform Technologies

Maria Cristina ENACHE[*]

ARTICLE INFO

Article history:
Accepted March 2017
Available online April 2017
JEL Classification
K22, M21

Keywords:
IT, Cross platform technologies, Technologies, Mobile application

ABSTRACT

Cross-platform - a concept becoming increasingly used in recent years especially in the development of mobile apps, but this consistently over time and in the development of conventional desktop applications. The notion of cross-platform software (multi-platform or platform-independent) refers to a software application that can run on more than one operating system or computing architecture. Thus, a cross-platform application can operate independent of software or hardware platform on which it is execute. As a generic definition presents a wide range of meanings for purposes of this paper we individualize this definition as follows: we will reduce the horizon of meaning and we use functionally following definition: a cross-platform application is a software application that can run on more than one operating system (desktop or mobile) identical or in a similar way.

1. Introduction

Due to problems imposed by each particular operating system (APIs different specific libraries, low-level operations available only on certain OS functions individual software interpretative different commands, etc.) have developed two main ways to make applications compatible with multiple operating systems.

The first method is to create and maintain applications give different (different code bases, individualized) for cation operating system side, but behaves functionally identical or similar level.

The second method refers to abstract the specifics of each OS separately using a specific technology. The Java platform is most famous instance of this case.

2. Cross platform technologies

In the cross-platform mobile development most influential technologies are: Apache Cordova, Xamarin and Unity [1].

Xamarin - Xamarin enables the development of cross-platform native applications for iOS, Android and Windows Phone using C # and the Mono framework.

Mono is an open source version framework-ului.NET based on ECMA sandardele.NET. The framework itself is very portable Mono (in addition to Windows, it works on Linux, Unix, FreeBSD, Mac OS X) and has a long history: there are almost the same time as framework-ul.NET.

Xamarin works differently for IOS and Android. IOS Xamarin has a compiler Ahead-of-Time (AOT), which builds applications directly into native code ARM for Android, Xamarin compiles code into Intermediate Language (IL), which is then compiled the Just-in-Time (JIT).

Runtime applications use Xamarin automates many tasks, such as memory allocation, garbage collection and others. As a final result, after Xamarin applications are compiled, the result is either an .apk file or .app to IOS or Android. These files are identical in function to the results of commonly used IDEs for native application development. [2]

Unity - Unity is a gaming engine that gives top priority portability, so he uses the following APIs: Direct3D on Windows and Xbox 360; OpenGL on Mac and Windows, OpenGL ES for Android and iOS, and APIs own video game consoles. [3] Developing games is either C # or Javascript, the language being used to access the native engine Unity (the wrappers), being written in C / C ++ for various reasons, the most important being related to improved performance and significant performance and manual memory management for games. [4]

Developers can export the games created a lot of platforms: Android, Apple TV, BlackBerry 10, iOS, Linux, Nitendo 3DS line, OS X, PlayStation 3, PlayStation 4, PlayStation Vita, Unity Web Player (Browser) Wii, Wii U, Windows Phone 8, Windows, Xbox 360 and Xbox One. [5] [6]

[*] Dunarea de Jos University of Galati, Romania. E-mail address: mpodoleanu@ugal.ro (M. C. Enache)

Unity statistics:

- Global, Unity account for 45% of market share game engine sites, about three times more than its nearest competitor.
- Unity is the most popular gaming software development among software developers (47% of them use it). The proportion of those who use it primarily grows from year to year (29%), as well as total dissolved pit.
- Unity clients include Cartoon Network, Coca-Cola, Disney, EA, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square, Ubisoft and Warner Bros.
- Unity reach 600 million players globally, compared Facebook has 829M users daily.
- Unity is growing rapidly and continuously, especially in the furniture market. [7]
- In the cross-platform desktop application development, it is important to distinguish between technologies that require compilation for each operating system separately (eg languages C and C ++) and technologies "Write Once, Run Anywhere" (eg Java)

None of the methods themselves do not guarantee application portability, portability is determined by the actual implementation, but can be significantly increased by using APIs specific to a particular operating system, use of libraries specific operations low level only available on select systems operating command interpreter's native operating system.

Without certain abstractions, the aforesaid difficulties can make it difficult, inefficient and costly cross-platform application development. Thus the main cross-platform technologies and classical languages are:

- First type (which requires compiling for each OS separately) languages C / C ++ and their related technologies, especially toolchain and compilers sites for each OS separately
- Second type (Write Once Run Anywhere) language and Java platform.

Knowledge about C / C ++ / Java (the language and platform) are wide-spread and fundamental, so we will not discuss in this paper, but we will nevertheless review the most relevant things about the platform and Java to provide a specific application context in this paper.

The problem addressed in the paper referred to the need for documentation applied in stages involved in the development of cross-platform software, but also to highlight the limitations and advantages or disadvantages related to both the desktop and mobile environment.

The proposed research work we wanted to present the most important and influential current and cross-platform technology to document, explore the applied stages of developing cross-platform applications based on these technologies. We also wanted to highlight the main limitations and advantages or disadvantages occurring in the development process for both the mobile and the desktop environment.

Specific objectives:

1. Document stages of developing cross-platform applications using Java SE technology for operating systems to desktop and mobile operating systems Cordova.
2. Documentation of the limitations and advantages or disadvantages occurring in the process of development.

Apache Cordova - Cordova is a framework for developing hybrid mobile apps, in contrast to the web or native denoting term hybrid in this case a combination thereof.

This framework allows developers to use HTML, CSS and JavaScript for mobile application development in a cross-platform. Using Javascript application logic, Cordova allows developers to access native functions and hardware devices through a set of APIs general abstracting differences between operating systems different programmers can avoid specifics APIs native and knowledge of different languages and Objective-C, Java, for example.

The main advantages of using this framework is limited to lower costs, greater availability of human resources (for example, there are undoubtedly many connoisseurs of front-end web technologies than people with experience in Objective-C or Swift), keeping into a less expensive and less complex projects and code base and a portable GUI for all mobile operating systems.

The main disadvantages can be linked for the most part of a lower performance compared to the native and a possible experience (GUI, the response rate, animations, native widgets, etc.) different from the native user. Other major disadvantages occur as follows: lack of libraries written statements to be "reinvented the wheel" substantially increase, thereby decreasing efficiency development. Another drawback may relate to lower efficiency for complex applications, there are no standard solutions for many of the important issues of development: in this case decreases development time and resources to grow, the need to find novel solutions.

As typology, Apache Cordova is an open-source framework for mobile application development. Like I said, allows the use of web languages: HTML5, CSS3 and Javascript to develop cross-platform applications. Applications results are neither native applications (written using native languages and technologies such mobile platforms), but neither are Web applications. The name used is that of hybrid application. Application hybrid because they are actually web application is a container for WebViews native to each platform and use a set of APIs universal allowing access to various phone functions, something that only application native who used technology they could do in the past. [8]

As an overview of the influence of Cordova, PhoneGap (distribution Cordova) it has been downloaded over 1 million times and is used by more than 400,000 developers / programmers (data available for 2016). [9] [10]

Cordova has a total market share of 5.88% of global applications and 0.84% of the overall installation. It has a market share of 4.7% of new applications launched and 0.61% of the newly launched app installs. [11]

The entire application is a container / wrapper native (native wrapper) containing three conceptual elements that communicate with each other, and with the operating system: web application engine for rendering HTML (WebView) and Cordova plugins.

The application is often called a hybrid, because all they are in a wrapper native (native wrapper) that can communicate with the mobile operating system in the same way as a native, is in fact a native application in which function engine for rendering HTML web files themselves (which are shown) and Cordova plugins. Further we will address each one and we also present how they interact.

HTML rendering engine (WebView): This component acts as a web browser, the de facto web browser without user interface elements. This WebView allows playback of web pages containing HTML, CSS and Javascript. Novelty comes in the fact that WebView, because it is a native wrapper can communicate with both the operating system and with Cordova plugins that are within the shell. Of course, as with a normal browser, this WebView can communicate with HTML APIs.

In figure 1 we can see the whole conceptual architecture and operation Cordova.

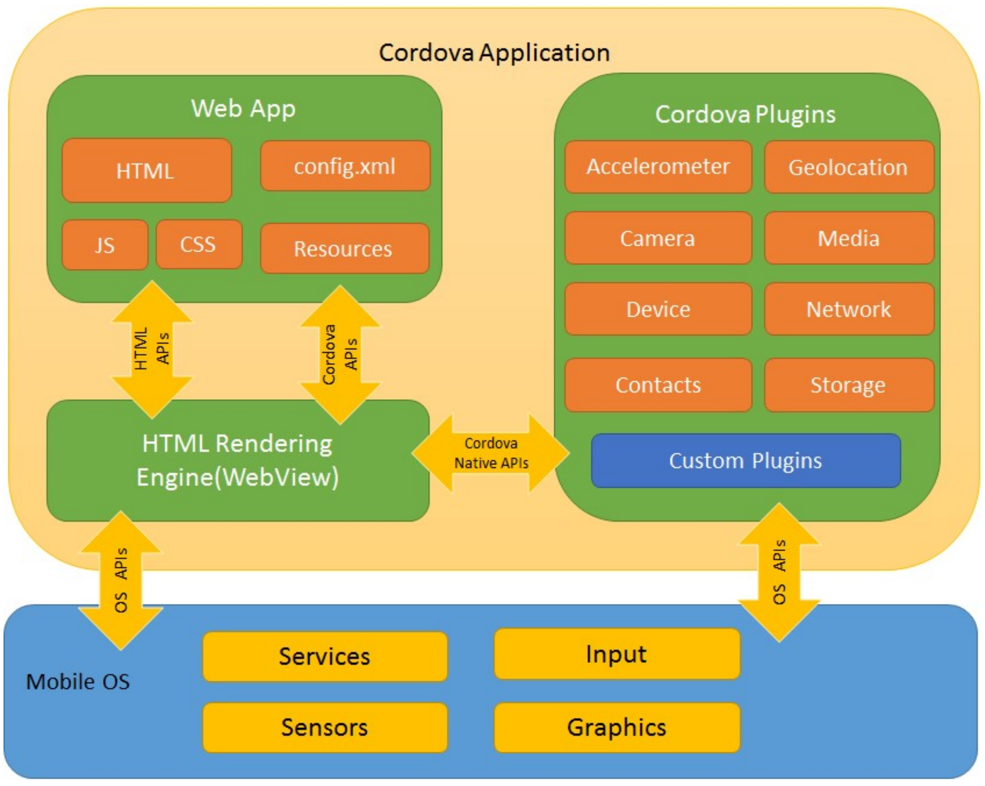


Figure1. Architecture of Cordova Applications

Web application can be viewed as a web page or collection of Web pages that can only HTML, CSS and Javascript (only those languages are processed by WebView). They are played by the HTML rendering engine above (WebView).

The resources config.xml files contain metadata about different aspects of the application and the plugins used. These files are used to provide information about setting up applications as follows: for each platform file information are automatically config.xml configuration files native to each platform that builds the application.

Config.xml file is automatically updated by various instruments during development to highlight changes in the plugins used, permissions and access levels, different meta-information. Resources folder contains native resources for each platform (icons and splash screens).

Cordova plugins: they communicate with the operating system and WebView and make possible the native capabilities of an application Cordova (storage access, room sensors, geo-location, media, contacts, device details s.a.s.m.d.). For Cordova, a plugin is a package containing the injected code that allows to communicate with WebView's native operating system. In fact, a Cordova sets plugins Javascript access interface to a native function (written in native code). [12]

Java SE and Java - Java was originally developed by James Gosling during Sun Micro (acquired by Oracle Corporation) and released in 1995 as a core component of the Java platform.

Java is a programming language for general-purpose with the following properties: a competitor (the possibility of executing instructions in an asynchronous mode), based on class and object oriented.

Intended purpose is to allow developers to approach such as "write once, run anywhere", meaning: Java code can run on any platform that supports Java without the need for recompilation specifically for that platform. Java applications are typically compiled to bytecode in, and it can run on any Java Virtual Machine, independent computing architecture.

Java is one of the most popular language, especially for web applications client-server, Java with an estimated 9 million developers who use it. Language borrow much of the C and C ++ syntax, but has fewer low-level facilities than these.

Java is a widely used platform for developing and packaging portable code (cross-platform) for desktop and server environments. Java uses the Java programming language and is part of the Java software platform. Java Standard Edition defines a broad set of APIs such as the Java Class Library (includes Java Language Specification and Java Virtual Machine Specification).

One of the most popular implementations of these platforms is the JDK (Java Development Kit). [13] One of the most important components of the platform refers to APIs made available by this (these give specific functionality Java). We talk here about various types of specific basic exceptions, especially class and high-level interfaces that are used for network, security, database access, GUI and parsing XML or JSON. In addition to APIs, Java SE platform also features the following: a virtual machine, development tools, technologies, packaging applications and other class libraries and tools used in Java. [14] Central Java platform is the concept of "virtual machine" - executing Java bytecode.

Bytecode is the same regardless of the hardware specifications or operating system used. Java Virtual Machine contains a compiler JIT (Just in Time) that "translates" java bytecode into native CPU instructions to execution and during execution of native code holds the remaining unused cache memory to spores performance.

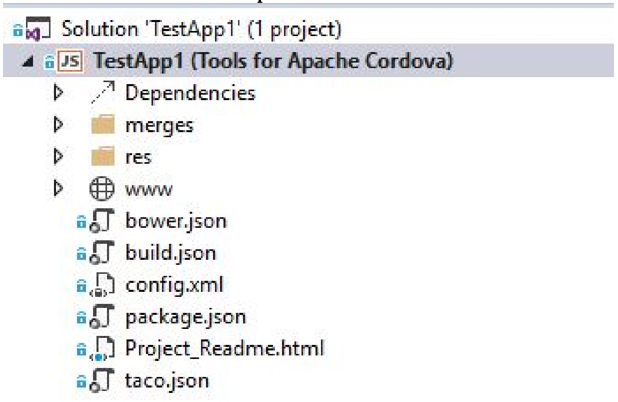
Because Java uses an intermediate language bytecode that a Java program can run in any environment software / hardware that has a Java Virtual Machine. Application performance is close to that of native applications, just-in-time compilation be without a significant deficit for performance. [15]

Further summarizes some data about usage and influence of the Java platform:

- 97% of corporate computers using Java
- 89% of US computers using Java
- There are nine million Java programmers
- chose # 1 developers
- chose # 1 development platform
- 3 billion mobile phones using Java
- 100% of Blu-ray using Java
- 5 billion Java cards in use
- 125 million TVs using Java
- 5 of the top 5 OEMs using Java ME [16]

3. Standard file structure in Cordova

To attaining an overview of the development using Cordova must do a brief overview of the standard file structure. We will refer only to the most important and used folders / files are automatically managed entirely by development environment. In the next picture we see the standard structure of files at the root:



```
Solution 'TestApp1' (1 project)
  TestApp1 (Tools for Apache Cordova)
    Dependencies
    merges
    res
    www
    bower.json
    build.json
    config.xml
    package.json
    Project_Readme.html
    taco.json
```

Figure 2. Standard file structure in Cordova

- es - this folder contains all application resources Cordova by category: Android, WindowsPhone, Windows and iOS. By default contains image files to launch icons and screens (for each platform).
- config.xml - a very important file that contains all the configuration information of the application - permissions, filters and plugins used for each operating system separately.
- build.json - enter the required information in the file signature of Android application step required for publication in Google Play store
- www - the most important project folder

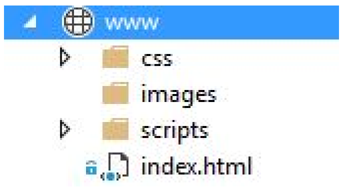


Figure 3. www Folder

- CSS - this sub-folder contains all CSS files related to the project
- images - here are found resources used in the application image
- scripts - this sub-folder contains all scripts used by application
- index.html - perhaps most importantly, the app home file structuring all other resources and play in the first instance user

The bottom line shows including only the components necessary for Kendo UI Mobile applications in libraries, including jQuery library and file cordova.js (including the latter is compulsory and his generation is done automatically during the build site)

```
<script type="text/javascript" src="cordova.js"></script>
<script src="js/jquery-2.1.0.min.js"></script>
<script src="js/kendo.ui.core.min.js"></script>
<script src="js/kendo.mobile.listview.js"></script>

<link href="css/kendo.dataviz.mobile.min.css" rel="stylesheet" />
<link href="css/kendo.mobile.all.min.css" rel="stylesheet" />
```

Using the development environment Visual Studio Community Edition, plugins can be added easily by selecting the file config.xml. After this selection they can be easily added to the GUI. After selecting a plug-in, it is automatically downloaded and copied project config.xml is automatically populated with the information needed to use and permissions required for running it are also automatically added by Visual Studio.

```
function playAudio(src) {
  if (device.platform == 'Android') {
    src = '/android_asset/www/' + src;
  }

  var media = new Media(src, success, error_error);

  media.play();
}
```

Using native file structure is possible using the plugin InAppBrowser permitting a browser independent as an alternative to the system, and behavior change that interface and allows you to change the standard behavior when accessing links. If our application allows targeting each platform's native browser to enter in good condition some selected links. If not using this plugin, links would be opened in WebView web engine and the user could not interact in any way with it.

```
$('a[class=external]').on('click', function (e) {
        e.preventDefault();
        window.open($(this).attr('href'), '_system');
        return false;
         });
```

Resources are used in an application Cordova usually have standard storage folders. If we need a folder it will be found in root www.

The entire basic structure of an application reside in a single HTML file, de facto a single HTML page, it is encouraged to ensure an even more important application performance and is made possible framework's Kendo UI Mobile concealing the continuity of the page web involved.

Web page is divided into different elements markup type <div> using certain special attributes conferred by the framework of the Kendo UI Mobile, these elements can be regarded each <div> as a separate activity (a "screen" separately) even if the structure once the application is actually the one HTML page with common properties normally.

```
<div data-role="view" data-title="main-menu" id="main-menu" data-persist="true" class="body" data-
transition="overlay:up">
    <h1>Alegere1.</h1>
    <br /><br />
    <ul id="routine-activities" class="menu">
      <li><a href="#menu-d" data-transition="overlay:up">O alegere din meniu</a></li>
      <li><a id="rain-menu-button" href="#scrollview-home2" data-transition="overlay:up" onclick="if
(first_time == 1)
{ playAudioMain('folder/suntet1.mp3'); first_time++; }
else btnClickPlay('folder/suntet1.mp3');"> Alegere2</a></li>
      <li><a href="tehnici.html" data-rel="external" data-transition="overlay:up">Tehnici </a></li>
      <li><a href="#menu-c" data-transition="overlay:up"> Alegere2</a></li>
      <li><a href="fisier.html" data-rel="external" data-transition="overlay:up"> Alegere3</a></li>
      <li><a href="#scrollview-home" data-transition="overlay:up">Imagini</a></li>
    </ul>
          </div>
```

As we can see from the example above, use data-role attribute with the "view" in an item <div> application instructs it to be played as a page / separate activity. We can see other attributes such as date-title (assign the job title), data-persist (instructs application that the state activity to persist from one activity to another), data-transition (can take many values and provide transition effect from a "screen" to another).

Each sub-menu is divided into two parts: a header and the actual content. This ensures modularity and reuse application code. I used a header footer as adaptive changes at run-time depending on the operating system, so the header can become a footer for example for running on Android.

```
<header data-role="header">
      <div data-role="navbar">
        <a id="back-button" class="nav-button" data-align="left" href="#:back" data-
role="backbutton">Inapoi</a>

        <span data-role="view-title"></span>
        <a class="nav-button" data-align="right" href="#credits-images">Licenta</a>
      </div>
            </header>
```

We observe time-reel attribute values of "header", it is necessary to define the structure as a header Kendo Mobile. We see also a div data-role attribute value "navbar" similarly structured so that the division, just a navbar Kendo Mobile with its own characteristics. Observe a span attribute value "view-title", it works the same way. At the application level and the main menu, interactivity and logic program is provided using standard language and JavaScript library jQuery to provide some useful functions or to connect the programming logic to events initiated by the user, such as event click when the user operates a specific item on the screen.

```
$("#rain-menu-button").click(function () {
var timer = setInterval(function () {
 if (window.location.href.substr(window.location.href.length - 5) != 'home2'
                                                  &&
window.location.href.substr(window.location.href.length - 12) != 'credits-rain') {
   btnClickStop();
   clearInterval(timer);
 }},300);
    });
```

When the button identifier "rain-menu-button" is activated, start a timer that checks every 300 milliseconds if there has navigated to a different address than the one current or claims and if so, then the audio btnClickStop will stop the function (), and the timer will be removed.

```
$('a[class=external]').on('click', function (e) {
        e.preventDefault();
        window.open($(this).attr('href'), '_system');
        return false;
              });
```

With aid InAppBrowser plugin and jQuery library when the user selects conditioned open a link that has the class "external" to the browser's native operating system.

```
function playAudioMain(url) {
        try {
          if (device.platform == 'Android') {
            url = '/android_asset/www/' + url;
          }
          my_media = new Media(url,null,null,onStatus);

          // Play audio
          my_media.play();
        } catch (e) {
          alert(e.message);
        }
      }
```

4. Conclusion

Cordova and Java are two very different approaches on mobile and desktop compatibility issues that still surprisingly have several things in common.

Both technologies are compatible with an approach Write Once Run Anywhere and both sums adaptive component that changes depending on the operating system detected, so that the programming logic and on the graphics hardware. In this Java has native capabilities to adapt to OS and Cordova has Kendo Mobile UI application framework with the same role. Both also use certain components programmatic adaptive, two examples of this are the runtime context Java (CMD or Mac Terminal) or components and layouts adaptive interface graphics such as widgets Kendo UI Mobile if Cordova or Swing widgets if Java.

Also, both have native support for conditional execution of code at runtime, depending on your operating system. Both technologies represent a degree of influence and use very high in the desktop or mobile environment, and both technologies provide simple yet powerful facilities to tend to achieve the ideal of cross-platform software so if desktop and mobile platforms.

References

1. *http://www.visionmobile.com/product/cross-platform-tools-2015/*
2. *https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/#How_Does_Xamarin_Work*
3. *http://docs.unity3d.com/Manual/Shaders.html*
4. *http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html*
5. *https://en.wikipedia.org/wiki/Java_(software_platform)#Java_Virtual_Machine*
6. *https://en.wikipedia.org/wiki/Swing_(Java)*
7. *https://cordova.apache.org/docs/en/latest/guide/next/#ios-debugging*
8. *https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition*
9. *https://cordova.apache.org/docs/en/4.0.0/guide/hybrid/plugins/*
10. *http://www.appbrain.com/stats/libraries/details/phonegap/phonegap-apache-cordova*
11. *http://stackoverflow.com/questions/8916782/how-much-is-phonegap-used*
12. *http://phonegap.com/about/*
13. *https://cordova.apache.org/docs/en/latest/guide/overview/index.html*
14. *https://unity3d.com/public-relations*
15. *http://www.polygon.com/2013/8/20/4641786/unity-for-wii-u-opens-up-gamepad-hardware-and-more-to-developers*
16. *http://venturebeat.com/2012/11/02/game-developers-start-your-unity-3d-engines-interview/*
17. *https://sometimesicode.wordpress.com/2014/12/22/how-does-unity-work-under-the-hood*