Multiplatform Application Development with HTML5 for Smartphones

Luis Vivas, Mauro Cambarieri, Nicolás García Martínez,
Horacio Muñoz Abbate, Marcelo Petroff

Laboratorio de Informática Aplicada - Universidad Nacional de Río Negro
{lvivas, mcambarieri, ngarcia, hmunoz, mpetroff}@unrn.edu.ar

Abstract. This paper presents an architecture for the development of software for smartphones. It explains the advantages of web application development with respect to native applications - those installed on mobile devices that were developed using a programming language compatible with the operating system used by the device. It also describes frameworks and software development kits currently available under the free software paradigm. Alternatives are presented for developing the user interface with support for HTML5 and cloud computing through which we can expose the data service and application. It explains the contributions of HTML5 for mobile applications and platforms that currently Android, iPhone OS and BlackBerry - as a distinguished group based on the number of devices and network traffic.

Keywords: Smartphone, multiplatform, Free software, Web applications, HTML5

1. Introduction

Smart phones are fundamental characteristics a computing capability and greater than the conventional [1] mobile connectivity. These make the phone in some cases replace the personal computer and lead the consumption of Internet data [2].

Currently the dominant devices are: Android, iPhone and BlackBerry. They have several functional features, such as: camera, high-definition touch screens, also called capacitive screens, Wi-Fi and 3G connectivity of long-range communication of short range (Near Field Communication - NFC) that is used for payments for services and is known under the name of digital wallet. The devices also have global positioning (GPS) for the geolocation system; several types of sensors such as gyroscope, accelerometer,

light sensor and proximity. The heart of the hardware consists of a machine Advanced RISC (ARM) and graphics processing unit (GPU) powerful and low power consumption, thanks to the advances of current manufacturing processes, by means of which can support the range of characteristics mentioned [3].

Counting with the described functionality, hardware is not sufficient for the proper management of the resources and capabilities of smart phones. Specially designed operating systems are needed. Currently, the operating systems most relevant are Google's Android, Apple iOS and BlackBerry RIM OS [2]. Android is based on Linux is open source software, while iOS is based on Mac OS X, which is proprietary software, as well as BlackBerry OS.

Android, iOS and BlackBerry 10 without QWERTY keyboard (keyboard layout that refers to the first six letters), are based on a concept of direct manipulation, using multi-touch gestures. The controls are based on: components sliding, switches and buttons. Interaction with the OS includes gestures such as lapses, touches and, which have different functions depending on the context of the interface. This new way of interacting with the operating system is mainly due to capacitive screens - those who have better picture quality, response time and allow the use of several fingers at once.

The increased use of these devices, their features and multiple existing platforms, poses a problem: how to build software for multi-platform smart phones? Looking for a solution, this paper proposes a software architecture based on HTML5 standards, discusses the advantages of the development of web applications, and explains frameworks for the development of applications for these devices. The work is structured in the following way. Section 2 presents the related concepts that allow you to understand the proposed solution. Section 3 explains the proposed methodology to develop cross-platform applications. Finally, section 4 summarizes conclusions, lessons learned and lines of research for future work.

2. General concepts

2.1. Web and native applications

Development for mobile devices has two distinct paradigms: native and web [4]. A mobile web application is what you need from a web browser or browser to run. The application data can be hosted or consume from a remote server, as well as also obtained from the mobile device. Native applications are those that are installed on mobile devices and are developed using the programming language compatible with the operating system, embedded software development kits (software development kit - SDK). The most prominent are Android SDK, iOS SDK and BlackBerry Java SDK.

Both the SDK of Android and BlackBerry using Java as a programming language, instead of iOS using the Objective-C language (programming language used in Mac OS X and iOS) [5] [6] [7]. Each of these development platforms has its peculiarities, not only for the programming language but also for development framework.

Basically, web applications are delivered as markup language hypertext (HTML) interpreted by a browser, which provides a range of features that increase the performance of applications.

Mobile web for the development of applications exist today a number of frameworks that make use of the advantages of HTML5 and WebKit platform [8], incorporated in all major Web browsers, such as for example: Safari, Chrome, and the BlackBerry browser; achieving very fluid applications using, among other features, touch screens and GPS for the georeferencing. The most important features of HTML5 are explained in the following section.Among the most outstanding frameworks include: Sencha Touch [9], [10] jQuery and Gwt Mobile [11]. Sencha Touch and jQuery are composed of a series of libraries written in JavaScript and styles Cascading Style Sheets (CSS). Some of the advantages of Sencha Touch with jQuery is a new Model View Controller (MVC) architecture. MVC is a design pattern that separates data from an application, the user interface, and the logic of business in three components [12]. Another advantage of Sencha Touch is the very complete documentation with many examples to download and very accomplished components both functional and visually. In contrast, JQuery has a faster learning curve. GWT Mobile, unlike the above-mentioned frameworks, is programmed in Java.

2.2 HTML5 [13]

On any web site, the user interface and data reside on a specific server, and they must return to download each time it is requested, this creates a performance problem as well as an insurmountable obstacle when a user connection is weak. HTML5 is intended to solve this kind of problem, and may keep the data and/or user interface online.

HTML5 provides cache application that allows you to store the HTML, CSS and javascript code on the mobile device in the first charge, as if it were a facility, allowing us to access the application over the line as if it were a native application.

One of the new features is the call: local storage, which allows you to store any variable that you want to in a manner similar to cookies, persistent (data sent from a web site and are stored in a user's web browser while the user is browsing a web site). This ease of local storage is easier and more flexible to manage.

HTML5 also provides database SQLite (open source database) relevant characteristic .as, HTML5 includes integration with an application programming interface (API)

geolocation, allowing access to the position of the mobile through the available devices GPS data. This tool allows you to do a wide range of applications where it is required to know the position of the user, for example: calculate routes, nearby landmarks, etc.

HTML5 incorporates new types of input form fields with a property named type (type), which allows you to differentiate between different types of data entry as text, email, phone, number, colors, numbers and date range. The virtual keyboard changes its behavior depending on the type of data. Example: If the type is numeric, the keyboard features the digits to enter.

2.3 Cloud Computing

Computing in the cloud (internet metaphor), is a paradigm that can offer computing services through Internet development of applications for mobile devices, web or native, need to consume data that feed it.

We can mention an application that will give us the climate in the city where the user is currently located. In this type of service would be impossible to have the information stored in the user's device because such information is dynamically created with climatic variables. In these cases the cloud computing allows us to create services and applications available all the time in any place where you have access to Internet to carry out an architecture using Cloud Computing we can use appEngine of Google, which gives us a SDK for software development, being able to choose some choices of programming languages such as Python, Ruby or Java.

Working with this architecture leave the common infrastructure problems that have an application or service of high availability, connectivity to power. It is also an option for SMEs that do not have an infrastructure to provide a service with the scalability and performance which gives us appEngine.

El hardware that is used using appEngine is the datacenter of Google, i.e., thousands and thousands of computers located throughout the world. In addition to the hardware, google provides a virtualization system based on Linux, modified with a system of its own file distributed (Google File System - GFS). For data, will have access to BigTable (database engine created by Google with the characteristics of being: distributed and high efficiency), the google storage system. This system is not a database, is a distributed system that has the particularity that nothing is deleted. With this system, google gives us all his knowledge in searches to access data quickly and efficiently [14] [15].For the management of data use Objectify [16], mapping object relational (ORM) which allows us to work with BigTable, which provides us with simple and elegant handling of data, such as operations: get, put, delete and query.

The following code defines the persistent entity Car, as initiates a transaction using begin, creates, obtains, and deletes a Car.

```
class Car {
       @Id String vin;
       String color;}

Objectify ofy = ObjectifyService.begin();
ofy.put(new Car("206", "azul"));
Car c = ofy.get(Car.class, "206");
ofy.delete(c);
```

3. Proposed methodology to develop cross-platform applications

To run a native application on all platforms, we must build an application for each platform. All platforms have your SDK development and programming language. Then we can say that web applications are the alternative for cross-platform development through the standard.

For the construction of web software development process is based on the MVC pattern, which is a design pattern very studied and used in software engineering, which separates into three layers model, the controller and the view. For the view, we opted for using Sencha Touch 2 and the java SDK's appEngine with Spring (framework of open source in developing applications for the Java platform) for the model and controller.

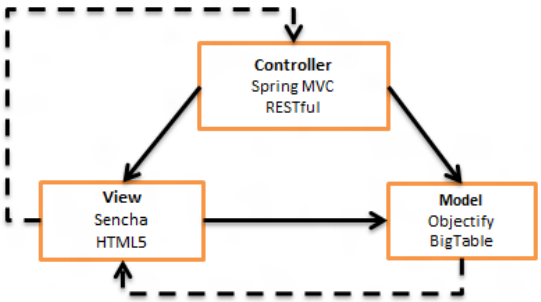

Fig. 1. Pattern Model View Controller - MVC

Communication between controller and view is through objects serialized using the json format (javascript object notation, is a lightweight format for data exchange) through Representational State transfer services (REST or RESTful) [16].

This new form of communication has meant a new option to web services. REST consists of using the HTTP specification (HyperText Transfer Protocol) [17]. This gained widespread adoption on the web as an alternative simpler to SOAP (Simple Object Access Protocol, is a standard protocol that defines how two objects in different processes can communicate through XML data exchange) [18] and WSDL (Web Services Description

Language or web services based on the web services description language), describes the public interface to web services) [19]. Large companies in the web 2.0 (Yahoo, Google and Facebook, etc.) are using REST, who marked obsolete to your SOAP and WSDL services and went on to use an easier to use, resource-oriented model.

The last implementation of Spring and other alternatives such as Java Server Faces (framework for Java based applications web that simplifies the development of user interfaces in Java EE - JSF2 applications), have full support with Restful in addition to an elegant and easy implementation.

Below we describe how to integrate Spring and Objectify [20]. The example searches for a list of Car objects. In principle we have to import the library to our project objectify.jar, which has the class com.googlecode.objectify.util.DAOBase [21], which will be the basis of our data access objects (DAO, it is a software component that provides a common interface between the application and one or more data storage devices). This class provides access to the data connection through the instance of the ofy() method.

```
public class ObjectifyDao<T> extends DAOBase
{ public Query<T> listAll()
        {Query<T> q = ofy().query(clazz);
          return q;
```

The second step is to create the CarDao extending from ObjectifyDao to handle the persistent class Car. With the annotation @Repository can put the services that we use in the context of Spring.

```
@Repository
public class CarDao extends ObjectifyDao<Car> {
       static {     ObjectifyService.register(Car.class);}
       public CarDao() {
              super(Car.class);
```

The next step is to generate the CarService service using the annotation @ Service, in the services of Spring inject all of the Dao that we need, in our case only CarDao. The annotation @Autowired is responsible for making the injection of controls in our service.

```
@Service
public class CarService {
       @Autowired
       private CarDao carDao;
       public List<Car> findAll() {
              return this.carDao.listAll().list();
```

The last layer that we use with Spring is the controller, this layer communicates directly with the user interface. With this controller we could provide services to web technologies but also native like Android or iOS technologies:

```
@Controller
@RequestMapping(value = "/car")
public class CarController {
	@Autowired
	private CarService carService;
	@RequestMapping(method = RequestMethod.GET,
	value = "/list", produces = "application/json")
	public @ResponseBody
	List<Car> carList() {
		return carService.findAll();
```

This code fragment highlights the RequestMapping annotation, and the /car and/list of the carList method values. Using HTTP accessing the url http://server:8888/car/list the customer can access the resource without having to configure or Exchange a communication contract. The parameter produce = "application/json" indicates that the result is transformed to json format.The outcome of appeal:

```
[{"vin":"206","color":"amarillo"},
{"vin":"ka","color":"rojo"},
{"vin":"mondeo","color":"blanco"}]
```

The last layer is the view that will consume the list of Car in json format. The only project we have to add libraries of javascripts, images and styles that are available from the site of Sencha. This framework has a number of components to consume local or remote data. The component that will be used in this case will be an Ext.data.Store [22], which allows us to configure the url of the resource previously created.

```
Ext.create('Ext.data.Store', {
		fields: ['vin', 'color'],
		sorters: 'vin',
		groupField: 'color',
		 autoLoad: true,
		proxy:{type: 'ajax',
			url : '/car/list',
			reader: { rootProperty: 'data',
				totalProperty: 'total',
				successProperty: 'success',
				type: 'json'}
```

Code shows the storage type (Store), the attributes that we consume in the fields property defined, that attribute I want to sort and group information, I also define the url of the resource, in this case as the view and the controller are on the same server the resource is consumed with a relative path /car/list, but only the case, and this application would be on another server or view this packaged with a mobile device, the url would be liaproyect.appspot.com/car/list. Finally it is explicit that our store expects a json with the

type property: 'json'. With the configured store we can only define a list component with can draw the Car list. Sencha has a component Ext.dataview.List [23].

```
xtype: 'list',
itemTpl: '<div class="contact">{vin}
<strong>{color}</strong></div>',
store: this.getStore(),
grouped: true,
emptyText: '<div>No tiene elementos</div>',
```

The list that is defined with the property xtype: 'list' with the itemTpl property defines the fields that you want to display and store property define you the instance of the store that you created earlier.

The result of the above is available in http://liaproyect.appspot.com and the sources of the project in https://liamobileweb.googlecode.com/svn/liaProyect.

4. Conclusion and future work

This work presented an architecture platform for the development of software for smart phones with web technology. It was based on the standards of HTML5 that are widely supported by all major mobile platforms; this was proven in the http://liaproyect.appspot.com application. It was explained the difference between the development of web and native applications, this last is developing an application for each platform which increases costs and times, according to the results obtained in our lab through the realization of applications.

Another point of study has been consumption by data services REST alternative that can be used for web and native applications remain a fundamental part in the optimization of the agile performance of applications that consume data from Internet, based on the results obtained in the laboratory.

Opened us different paths of research the future possibility of access to other resources on smart phones, such as: the contacts in the phonebook, get pictures using the camera of photos, as the ringtone alert or vibration of the device, to provide other types of features related to the ubiquitous computing or augmented realities. Applications such as these are so-called hybrid [24] that would allow web applications access to the APIs system and digital stores.

References

1. Wikipedia. 2012. Teléfono inteligente. http://es.wikipedia.org/wiki/Teléfono_inteligente (accedido 10/07/2012).
2. Cisco Visual Networking index, Global mobile data traffic forecast update. 2011.
3. Proceso de Fabricación ARM Cortex 28nm. http://www.tsmc.com/tsmcdotcom/PRListingNewsAction.do?action=detail&newsid=6781&language=E (accedido 12/07/2012).
4. 2012. Mobile HTML5 java developer. http://www.jboss.org/webinars (accedido 20/06/2012).
5. JBossCommunitty. 2012. SDK BlackBerry. https://developer.BlackBerry.com /java/ (accedido 10/06/2012).
6. Appe.2012. SDK iPhone. https://developer.apple.com/ (accedido 11/05/2012).
7. Google. 2012. SDK Android. http://developer.android.com/sdk/index.html (accedido 02/04/2012).
8. Apple, Kde. 2006. http://www.webkit.org/ (accedido 19/07/2012)
9. The jQuery Foundation. 2012. Framework Jquery para Móviles. http://jquerymobile.com/ (accedido 01/06/2012).
10. Gwtmobile. 2012. Framework Gwt mobile (Google web Toolkit) http://code. google. com/p/gwtmobile/ (accedido 21/02/2012)
11. Sencha Inc. 2012. Framework Sencha para móviles http://www.sencha. com/products/touch/ (accedido 10/11/2011).
12. Wikipedia. 2012. Patrón MVC. http://es.wikipedia.org/wiki/Modelo_ Vista_Controlador (accedido 10/07/2012).
13. HTML5rocks.2012. HTML5 lo nuevo y características slider. HTML5rocks. com (accedido 13/03/2012).
14. Google Developers. 2012. AppEngine. http://code.google.com/intl/es-ES/appengine/ (accedido 2/05/2012)
15. Google España. 2012. AppEngine Campus Party. http://www.youtube.com/watch?v=9Ocjqxhh3RQ. (acedido 10/05/2012).
16. Google App Engine. 2012. Objectify. http://code.google.com/p/objectify-appengine/wiki/IntroductionToObjectify (accedido 28/05/2012).
17. Navarro Marset, R. 2006. Servicios REST http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf (1/07/2012).
18. W3C. 2010. Estandar HTML http://www.w3.org/MarkUp/ (accedido 10/07/2012)
19. W3C. 2007. SOAP http://www.w3.org/TR/soap/ (accedido 11/07/2012)
20. W3C. 2001 WSDL http://www.w3.org/TR/wsdl (accedido 11/07/2012)
21. Matías Molinas. 2012. Integración de Spring y Objectify http://fuse21.blogspot.com.ar/2012/04/spring-mvc-y-objectify-en-google-app.html (10/06/2012).

22. Google. 2011. Clase DAOBase http://objectify-appengine.googlecode.com/svn-history/r635/trunk/javadoc/com/googlecode/objectify/helper/DAOBase.html. (accedido 2/07/2012).
23. Sencha Inc. 2012. Ext.data.Store. http://docs.sencha.com/touch/2-0/#!/api/Ext.data.Store (accedido 10/07/2012).
24. Sencha Inc. 2012.Ext,dataview.list. http://docs.sencha.com/touch/2-0/#!/api/Ext.dataview.List (accedido 10/07/2012).
25. Kaiser, C. 2011. Aplicaciones móviles Hibridas. How to Develop Mobile Applications with Web-Technologies. Université the Fribourg Suisse. http://diuf.unifr.ch/main/is/sites/diuf.unifr.ch.main.is/files/documents/student-projects/eBiz_2011_Christian_Kaiser.pdf (accedido 21/04/2012).