April 30
International Journal of Advanced Studies in Computer Science and Engineering
IJASCSE Volume 3, Issue 4, 2014

An Application for the Management Movements of Via Verde

Miguel Martin
Centre of Innovation and Development, ISPGaya
V.N.Gaia, Portugal

Fernando Almeida and José Monteiro
Faculty of Engineering, Univ. of Porto & INESC TEC
Porto, Portugal

***Abstract*—This paper aims to introduce and present an web and mobile application that were developed to manage the movements from the vehicles that uses Via Verde. Since these applications were developed with different frameworks, two architectures of each application (web and mobile) will be introduced. The built prototypes will be presented and a short description of the frameworks involved will be analyzed too, as well all the other technologies involved. The results of the project were achieved by the construction of two applications that manage those movements.**

Keywords-Web Applications; Mobile Applications; Frameworks; Movements; IT Architectures

I. INTRODUCTION

Technology advances has facilitated many improvements in vehicles industry. Some of the benefits of those advances can be experienced directly on vehicles utilization, and other benefits can be experienced on services supplied to the vehicles' users. The case presented on this paper is a service named Via Verde, which is implemented on Portuguese highways, in some main itineraries, in park auto facilities, and in some gas stations. The concept, based on postpaid service associated to a debit card, consists on a previous acquisition of a small electronic device to install in front window of the vehicles to identify the vehicle when the driver uses the referred services. A small example is, a driver that has Via Verde service associated to its vehicle and decides to travel through a highway. When arrives to the portage post, it is not necessary the usual procedures: stop the vehicle and get the ticket. Also, on exit, is similar. It is not necessary to stop the vehicle to present ticket and to pay the service. The system registers the entry and the exit, and calculates automatically the value to pay. While vehicle is on movement, it is presented to the driver, the value to debt, in a small static panel.

Recently, the company responsible by Via Verde service has implemented a Website to make more easily to users manage is own account, anytime and anywhere. Some of the most relevant operations that can be done are: check invoices; manage the assignment of the electronic identifiers of each vehicle, get or check the resume of the use of the service. The last one allows users to obtain an electronic resume in different file format (XML, PDF, HTML, CSV). Nevertheless, the system is not enough friendly to allow users obtain an analysis of the reports.

Considering extremely useful to any user of Via Verde (private or even a company) to have an analysis of the use of each vehicle registered on Via Verde (mainly companies which have several vehicles), we decided to develop a prototype solution to process the information obtained from the files available on Website, by making the process of the analysis of the information more intuitive. The developed prototype is based on two complimentary applications (one mobile and the other web). Mobile application uses preprocessed information from the web application to reduce the efforts of processing of the mobile devices. In both applications, users can do multiple searches by vehicle, as well obtain a graphical analysis of the results.

In both applications, the used frameworks were chosen in order to fill four criteria: security, usability, efficiency and fast results. Mobile application development was based on Phonegap [1] framework and Web application development was supported by Cakephp [2] framework. This paper is structured as follows: (i) at section is presented a literature review about the foundations that support project development. (ii) at section three is presented a brief description about similar projects that currently exist in the market. (iii) at section four where documented the project requirements as well as the architecture of each application. (iv) at the section five we discussed the results obtained and, (v) in the final section, we present some conclusions.

II. LITERATURE REVIEW

A. *Framework*

A framework, in the context of computer science, is a software platform that is used to develop applications, that includes libraries, their own compilers, tool sets, API's (application programming interfaces) that enable the user to develop their own project or solution. Typically frameworks offer common features and have implicit some principles:

- To be responsible for the control flow of the computer program;

[1] www.phonegap.com
[2] www.cake.php.org

- It can be extended by the user using specialized functions or code to provide a new functionality, like a plugin for instance;
- Besides a predicted extensibility, mentioned above, the framework is not supposed to be modified.

The adoption of frameworks typically brings important benefits for application development, namely in terms of reusability, maintainability, security, predictability and extensibility [1]. At the same time, frameworks reduce development time while improving delivered software quality, which means that developers can spend more time focus on the business-specific problems, rather than on the code behind the scenes. As refer Blechar apud Lloyd and Cann [2], the future of application development lies with an open architecture that facilitates component choice, assembly and integration.

B. Framework Cakephp

The Cakephp framework is based on PHP language. This scripting language is widely used for web application development, having as main advantages its easy learning process, full support to the MySQL database engine and to be a cross-platform technology.

Cakephp framework uses a programming model called MVC (Model-View-Controller), which is a model of software architecture that separates the representation of the information from the user [3].

MVC is composed of three components illustrated in Fig. 1.

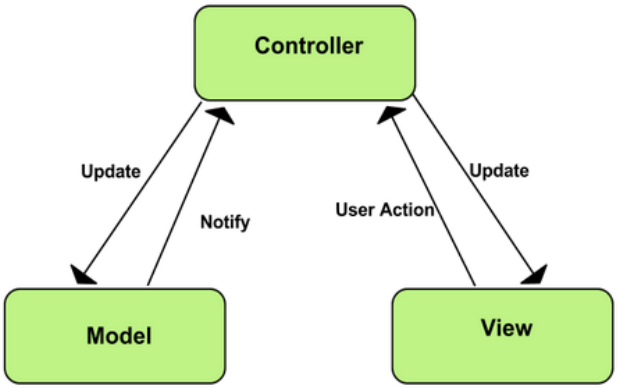

Figure 1. MVC architecture [4]

The model consists in the separation of the data objects from the application, functions. The view is a representation of the data, and it is possible to have multiple views of the same data structure. Finally, we have a controller, which is responsible for convert instruction sets to a model or a view.

The control flow through MVC generally works as follows [5]:

1) The user interacts with the user interface in some way (e.g., user presses a button);

2) A controller handles the input event from the user interface, often via a registered handler or callback;

3) The controller accesses the model, possibly updating it in a way appropriate to the user's action (e.g., controller updates user's shopping cart). Complex controllers are often structured using the command patter to encapsulate actions and simplify extensions;

4) A view uses the model to generate an appropriate user interface (e.g., view produces a screen listing the shopping cart contents). The view gets its own data from the model. The model has no direct knowledge of the view;

5) The user interface waits for further user interactions, which begins the cycle again.

The main conceptual steps for creating a Cakephp application are the following:

- Creating the database;
- Defining the model;
- Defining a controller;
- Defining the views.

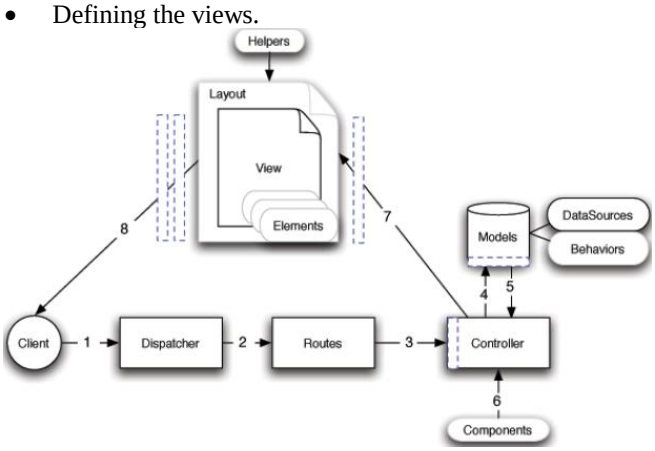

Figure 2. Flow diagram of a typical Cakephp request [6]

In figure 2 required elements are represented with black color, option elements use gray scale, and callback is highlighted in blue.

The flow process can be described as follows [6]:

1) The user accesses the URL and his browser makes a request to the web server;

2) The router parses the URL in order to extract the parameters for this request: the controller, action, and any other arguments that will affect the business logic during this request;

3) Using routes, a request URL is mapped to a controller action (a method in a specific controller class);

4) The controller may use models to gain access to the application's data. While model usage is not required, all Cakephp controllers initially require at least one model;

5) After the model has retrieved the data, it is returned to the controller;

6) The controller may use components to further refine the data or perform other operations (session manipulation, authentication, or sending emails, for example);

7) Once the controller has used models and components to prepare the data sufficiently, that data is handed to the view using the controller's set() method.

C. HTML5, CSS and Javascript

HTML is a markup language for creating webpages, based on a tag system. CSS is a style language used to create

different presentations of HTML pages, having advantages like not being necessary to be defined in the HTML page, making the webpage more faster to reload and more easier to code. JavaScript is a script language used to construct objects, using dynamic variables, and its invention and inclusion in webpages became valuable to improve HTML.

HTML5 (last version of HTML language) brings some new features [7]:

- New syntactic features, like audio, video and canvas elements, that were designed to make it easier to the developer to add graphical and multimedia content into the websites and applications, without using external plugins;
- HTML5 introduces specific API's that can be used with Javascript, like drag-and-drop and web storage;
- This new version of HTML is designed to be compatible with the old browsers, in order to cause lower impact during period of the transition from a deprecated technology to an updated technology, causing a minimum error effects;

But, although these new features, it has some disadvantages:

- The actual browsers still need to provide support to HTML5, because some issues with the new elements may arise [8];
- There are some performance issues when the app is one that needs to be deliver speed or a familiar app-like experience [9];
- With the constant changes in the HTML elements and tags, it is a little difficult to monitor the development of HTML.

D. *JSON*

JSON is a subset of JavaScript but do not requires necessarily JavaScript. It serves as an exchange of computer data and it emerges as an alternative, less verbose, to XML. It supports some basic data types, for example: Number, String, Boolean, Array, Object.

JSON has JSON Schema too (by analogy with XML schema), that specifies a format of a JSON document, its validation, documentation and the control of the document flow [10].

A good feature oft JSON is that can be often used in Ajax technology, to parsing objects from a URL specified by a developer [11].

E. *Angular.js*

Angular.js is an open source JavaScript framework, based on MVC model, to improve the experience on the development code using JavaScript.

The library angular.js [12] reads the HTML file that contains the tags and executes the code belonging to the same tag, making connection with the model that is represented by variables of JavaScript.

F. *Framework Phonegap*

This framework is designed to build mobile applications using HTML, CSS and JavaScript, instead of native languages for Android (that uses java) or iOS (Objective-C). Phonegap, works with SQLite databases, allowing the mobile devices to have support to a local database. Additionally, provides a development environment easier to developers without experience with native languages for mobile devices [13]. Although these advantages, Wargo [14] considers that the fact of working only with HTML, CSS and JavaScript reduces the capability of producing very powerful applications, due to the limitations of HTML, CSS and Javascript.

G. *Slim Framework*

Slim [15] is a PHP micro framework used to write little web services (API's). This is used nowadays to connect some applications that can't share the same resources, and in this project it was used to connect the mobile application to the web application.

III. SIMILAR PROJECTS

There are currently two applications in Portugal, in addition to the Via Verde website, that do some analysis, but not statistically. Start by an application called ″m.Portagem″ [16], it was developed for android and iOS, and allows the users to view amounts debt associated to vehicle license plates. This application offers the following main features:

- View debts related to vehicle license plate;
- History of license plates of vehicles that user have owned;
- Add alerts for payments of the debts;
- Simulation cost of e-tolls.

However, this project has some limitations too that are:

- The login system is flawed and contains authentication issues;
- The list of transactions is faulty and there is a differentiation against the presentation of movements in enrollment associated with a particular user, emerging every movement simultaneously, regardless of whether we want to select the registration search.

Figure 3. Interface of m.Portagem application

The other application is "aiScuts" [17], which is linked to the Portuguese mail company, and it retrieves the values in debt for a specific license plate, but does not allow the user to make any payment of the debts.

Figure 4. Interface of aiScuts application

The last similar project that exists is the official Via Verde website [18], that allows every user to access their account and manage everything regarding vehicles, contracts and even to download extracts of their movements, but don't allow any statistical analysis of the movements.

Figure 5. Interface of Via Verde website

IV. IMPLEMENTATION

A. *Architecture of the Application*

The application follows a traditional 3-tier application composed by a presentation interface, business logic and database.

The web application contains the following elements:

- The front-end application was designed with HTML, CSS and Javascript;
- The core of the application was designed using Cakephp framework, which works with PHP language;
- The database of the application was implemented with MySQL.

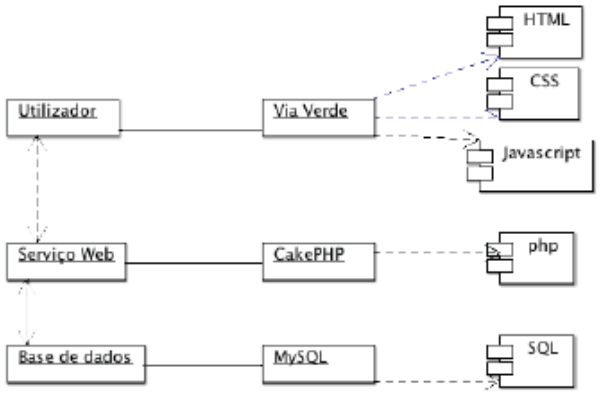

Figure 6. Architecture of web application

The mobile application has the following architecture:

- The user interface of the application was designed in HTML and CSS;
- The core of the application was designed using the Phonegap framework and javascript.

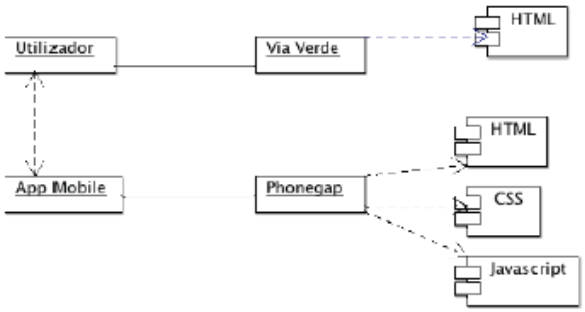

Figure 7. Architecture of mobile application

The database structured is despicted in figure 8.

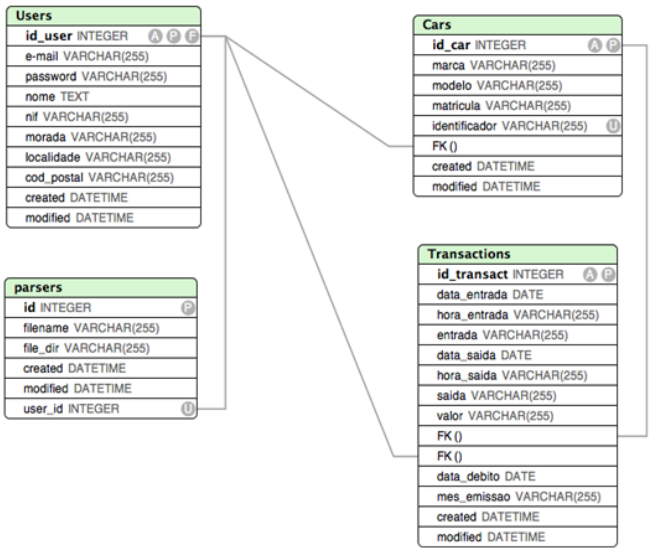

Figure 8. Database structure of appication

The table *Users* is responsible to store information about the user of aplication. Each user can have several cars, which details are saved in table *Cars*. Each user can make along the time several *Transactions*, and for that it is relevant to store the date and time of entrance and exit. Finally the *Parsers* table is used to import information in a XML file from Via Verde website.

B. *Functional Requirements*

The functional requirements of the project are:

- Register user – the user can make its registration in the application;
- Edit profile of user – here the user can edit his own profile directly from the web application;
- Import XML file – this is the main part of the application that will import a XML that the user chooses;
- Add license plate – this module is used to add a vehicle with license plate;
- Search for movements – module created in order to implement the search of movements with a license plate and to view graphics.

In order to implement these functional requirements we implemented various modules in the web application. In the user registration module, a system of registration and authentication based on the conventions of Cakephp was created which allows the user to register their data and submit it to finish the registration. Subsequently the password is encrypted using the encryption defined in Cakephp that uses an encryption algorithm and a key called security salt [19]. Then, it was finally drawn the desired view to place the form visible to the user registry by using a CSS layout and HTML.

The web application also allows each user profile to be updated. It was used an identical form of other modules and various functions were made in PHP to edit the profile. To do this, it is necessary that the web application only allows editing the profile that is connected to the system, and it was necessary to make a PHP function for that purpose. This function also included a query to the database that populates the data in the user profile that were previously inserted when the user was created. Noteworthy, for security reasons, it is not possible to change the previously entered password.

The module that imports data from the XML was done by using the MVC model of Cakephp, due to its simplicity. The first step was to search localy a XML file that was downloaded from the Via Verde website in order to make a PHP function that, using the parser, could extract the desired data from XML file and fill the tables of the database with the desired data. The next step was to create one view in PHP to present the table of movements that had been entered. To complete this step it was necessary to create a controller that deals with the parser. The end result was the complete data parser of the XML, as well as recording the file in a folder, for future analysis.

The module makes the registration of new vehicle and license plate was made using the same method of the user registration, but only with the difference that the data was from the vehicle and not the user. In this module, it was necessary to make a function in PHP so that the car was entered in accordance with the user that was connected, to belong to it. In the vehicle registration, the identifier of the equipment of Via Verde may be filled or not, since when the XML file is imported, the identifier is parsed and placed in the database, associated with the corresponding vehicle.

The module that allows the users to make searches of movements it's the initial menu of the application. It presents a table for the entries that were imported from XML, as well as graphics for statistical analysis. This module was conducted using perfectly the concept of MVC Cakephp, because it was necessary to create the model of the transactions, the view that allows the user to view the tables and graphs, as it was necessary to create the controller that contains all functions that are needed for the visualization of movements of vehicles, following the Cakephp conventions. For viewing and searching movements on this module, was necessary to create search functions for registration of the vehicle , and when the registration issued for research and compared with the enrollment of user entered in the database and it exists, the table appears populated with the data from the database, followed by graphics of statistical analysis.

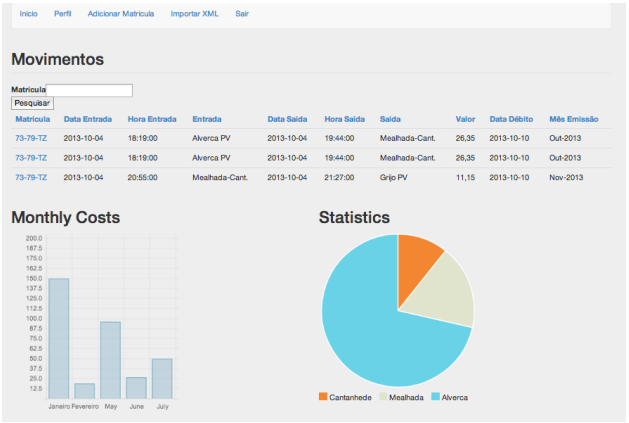

Figure 9. Screenshot of web application

Regarding the mobile application using the Phonegap, it was exclusively designed to bind to the web application and return the data to the movements of the vehicles present a table, so that the user has access, after performing login to these same data. The login module was conducted in Javascript functions, returning values entered by the user are compared with the same in javascript, to give permission to the user to enter the application.

Subsequently the module that allows you to see the movements of the vehicles has javascript functions and a table that was created using HTML and angular.js in order to create the same dynamic shape and fill it with the data that the application receives the database in JSON format.

Figure 10. Screenshot of mobile application

C. *Non-Functional Requirements*

The non-functional requirements of this project includes:

- Portability – the application should be run in several platforms and devices. Therefore, the application was tested in more than an operated system and three browsers. The mobile application was tested in two mobile devices with Android and IOS;
- Usability – the application should be easy to use, with an intuitive navigation, not very extensive, and essentially based in menus;
- Security – the web and mobile application must guarantee the authentication of users and not allow the access to the system by unauthorized users. The persistence of authenticated users should be made by adopting sessions;
- Performance – the application code must be optimized in order to allow its access by users that use a web or mobile interface access.

V. DISCUSSION

During the development of this project, and taking into consideration that similar projects does not have this statistical analysis, we intended to accomplish the functional requirements previously presented, as well as meet the non-functional requirements of the applications.

In terms of portability, the web application has been tested in several browsers and devices:

- Mozilla Firefox V. 25.0;
- Google Chrome V. 32.0.1700.102;
- Safari V.7.0.1;
- Ipad 2.64.

The tests were completed successfully in these browsers as well in the device, without having any issues of portability and the application being fully functional.

The mobile application, despite being developed for android using Phonegap framework, was also tested on mobile devices running iOS, without modifications. The result was satisfactory to our expectations about the level portability.

Talking about usability, first the objective was to provide the web application with a simple, intuitive and fast interface, with few options, to be easy for the user. The user only needs to do maximum two clicks to get a result (select menu option and, occasionally a sub-option to get more selective results).

A less successful operation was the attempt to validate each page with the W3C validator. Due Cakephp control over the application and the tags of HTML, because they are embedded inside a *.ctp* file, the validation returns a few errors. However, they have no influence on user view neither cause malfunctions on the application.

This application has another non-functional requirement that is very important, that is the security. Cakephp allows the developer to make an application with high security standards. In this application, it is possible to show the security in the user's password, because as mentioned earlier, the password is encrypted with an encryption algorithm and a key named security salt. Due to this security issue, the password was unable to decrypt in the mobile application.

In terms of application performance, it is important to focus on the data volume transacted in applications (Web and mobile). Depending on the use of web application it could have a considerable volume of data. Due to the need of constantly accessing the database to perform queries to return the user desired results and the need of load in database the data extracted to a XML file from Via Verde website, it was necessary to do stress tests to analyze the performance of the application. The tests were made locally. However, in future, we consider relevant to do this test in a Web server with several users to confirm or not the performance the application in a real context. Unfortunately, due to lack of physical resources, it was not possible to run this performance test using a server, and all tests were performed on localhost. Despite the large volume of data used (four months of registers of two vehicles), the whole application has a very satisfactory performance with a very quickly responses to user solicitations. The level of mobile application and taking into consideration that needs to communicate with an *api* as explained previously, the response time and presentation of the phone is slightly high, so it needs some time until the JSON data format appear in the correct table.

VI. CONCLUSIONS

Most relevant conclusions about this work can be separate in two main contributions categories. First contribution category is about the purpose of the project. We built a prototype, consisting on a Web application and a mobile application to improve the experience of the users to process and analyze data of their own vehicles obtained from

Via Verde website. Beyond technical concept, our solution will influence future improvements on Via Verde website to offer users a more complete service. The second contribution category is about to prove how helpful can be framework solutions to fast code development. We built our project mainly based on frameworks to automatize as maximum as possible the operation of software development. Due to the fact that we start to the project without previous knowledge about each framework used, we proved that is possible to combine different framework technologies and obtain functional and usable prototype application (mobile or not mobile).

As final conclusions, many improvements need to be made in future in order to add more functionality, namely in the processes of load into local database data obtained from Via Verde website by automate these processes. In web application, graphics need to be more diversified, with more options, so user experience shift from text view to a graphical view. Regarding to the mobile application, it can be done natively, without the use of frameworks, which makes it automated and also be connected to the Via Verde API.

REFERENCES

[1] J. Lerman, and R. Miller, Programming Entity Framework: Code First, Sebastopol: O'Reilly Media, 2011.

[2] D. Lloyd, and S. Cann, Why Use An Architectural Framework?. Available on: http://www.jcorporate.com/expresso/doc/edg/edg_WhyUseFramework.html (accessed on 25 of February 2014).

[3] A. Bari, and A Syam, Cakephp Application Development, Birmingham: Packt Publishing, 2008.

[4] CHROME, MVC Architecture, Available on: http://developer.chrome.com/apps/app_frameworks (accessed on 27 February 2014).

[5] A. Zecevic, Model View Controller (MVC) architecture, Available on: http://poincare.matf.bg.ac.rs/~andjelkaz/pzv/cas4/mvc.pdf (accessed on 27 February 2014).

[6] CakePHP Foundation, CakePHP Cookbook Documentation, Available on: http://book.cakephp.org/2.0/_downloads/en/CakePHPCookbook.pdf (accessed on 17 April 2014).

[7] A.Bradford, and P. Haine, HTML5 Mastery, Semantics, Standards and Styling, New York: Apress, 2011.

[8] P. Irish et al, Front-end Code Standards & Best Practices, Available on: http://isobar-idev.github.io/code-standards/ (accessed on 14 April 2014).

[9] B. Smus, Improving HTML5 Canvas Perfomance. Available on: http://www.html5rocks.com/en/tutorials/canvas/performance/ (accessed on 12 January 2014).

[10] M. Droettboom et al, Understanding JSON Schema. Available on: http://spacetelescope.github.io/understanding-json-schema/UnderstandingJSONSchema.pdf (accessed on 14 April 2014).

[11] S. Fehrenbach, S. Erdweg, and K. Ostermann, Software Evolution to Domain-Specific Languages. Available on: http://www.student.informatik.tu-darmstadt.de/~xx00seba/publications/evolution-to-DSLs.pdf (accessed on 15 January 2014).

[12] AngularJS. HTML enhanced for web apps. Available on: http://angularjs.org/ (accessed on 15 January 2014).

[13] R. Gathol, and Y. Patel, Beginning Phonegap Mobile Web Frameworks for Javascript and HTML5, New York: Apress, 2012.

[14] J. Wargo, Phonegap Essentials: Building Cross-Plataform Mobile Apps, Boston: Addison-Wesley Professional, 2012.

[15] Slim, PHP Micro Framework. Available on: http://www.slimframework.com/ (accessed on 21 January 2014).

[16] M.Portagem. Available on: https://play.google.com/store/apps/details?id=com.bzp.app.portagens (accessed on 21 January 2014).

[17] AiScuts. Available on: https://play.google.com/store/apps/details?id=com.fm.cttscuts&hl=pt_PT (accessed on 21 January 2014).

[18] ViaVerde. Available on: http://www.viaverde.pt/Website/ (accessed on 21 January 2014).

[19] Cakephp. Cakephp Tutorial. Available on http://book.Cakephp.org/2.0/en/index.html (accessed on 10 January 2014).