Empowering Mobile Users: Applications in Mobile Data Collection

Arlindo F. Conceição
Institute of Science and Technology
Federal University of São Paulo (UNIFESP)
São J. dos Campos, Brazil
Email: arlindo.conceicao@unifesp.br

Dario Vieira
French School of Electronics and Computer Science (Efrei)
Paris, France
Email: dario.vieira@efrei.fr

***Abstract*—This paper presents an architecture for collecting and analyzing mobile data. The system offers simple and intuitive interfaces to create mobile applications (Apps). It allows the collection of conventional data, such as numbers and text, and also non-conventional data, such as multimedia files, location information, and barcodes. The collected data can be shared among users on a social network. In addition, we propose a pipeline architecture to data analyzing.**

Keywords–Mobile services; Smartphones; Data Collection.

I. INTRODUCTION

The mobile communication market has evolved fast. This evolution is mainly characterized by three factors: reduced smartphone prices, launch of mobile devices with high processing capability, and emergence of new technologies for the development of Mobile Applications (Apps). These factors have created conditions for the large-scale usage of Apps.

However, despite the advances in hardware and software, the creation of mobile applications continues to demand programming efforts and involvement of programmers and IT professionals. In our opinion, this is the main limitation to wider usage of mobile solutions and applications. The resources (money or/and programmers) to develop these mobile applications are not always available. In general, the end user cannot pay for the development of customized applications.

In order to mitigate this problem, we are developing an open cloud infrastructure for data collection and automatic creation of mobile Apps [1]. The platform allows the user to create and customize their own mobile applications using simple interfaces. The user does not need to know how to program.

By providing new tools to the users, we are opening opportunities for new applications, services and usage of mobile devices. We refer to this concept as Mobile User Empowering, which includes the following goals:

- To allow customization of mobile software requirements using simple interfaces.
- To host data and applications, transparently, in the cloud.
- The service must be free and the data must be that of the user.

To create a proof of concept of Mobile User Empowering, we focused on applications for Mobile Data Collection (MDC) and mobile surveys. These applications usually have the format of a questionnaire and contain a pre-established number of objective questions [2]. There are several good reasons to use mobile applications instead of traditional methods. First, we can reduce or even eliminate the usage of paper. Second, mobile applications may enhance the reliability of the collected data by implementing validation procedures. Finally, if we collect data using electronic devices, the information does not need to be manually moved from paper to an information system.

The project was called **Maritaca** (*MARitaca Is a Tool to creAte Cellular phone Applications*). Furthermore, *Maritaca* is also the name of a bird in Brazil. The project is open source and was designed to be highly scalable. The tool is available for evaluation [3].

The remainder of this paper is structured as follows: Section II presents the related works, Section III describes the distributed architecture, Section IV briefly shows the interfaces and features of the platform and integration model of the project and Section V proposes a pipeline architecture to data analysis. Section VI explores applications of the platform. Finally, we present future work and our final considerations.

II. RELATED WORK

There are several projects with similar purposes to Maritaca. For example, *App Inventor* [4] allows to build applications for Android visually. It focuses on the drawing interface components, step-by-step, connecting their respective events. The advantage of Maritaca over *App Inventor* is that it allows a simpler and more intuitive design of interfaces. This is possible because it focuses on MDC applications.

Nokia Data Gathering [5] is a system that allows questionnaires to be built which can be accessed by mobile devices with connection to the Internet. Data is gathered and stored in the mobile devices and can be transmitted to a server. However, it is a proprietary solution.

Another tool is Open Data Kit (ODK) [6], which consists of a set of open source tools that help to create and manage mobile data collection. It is composed of three tools: *Build*, *Collect* and *Aggregate*. *Build* is used for modeling the forms. *Collect* is used to start data collection on mobile devices that run the Android operating system, and also for sending the data to the server. Finally, *Aggregate* gathers the collected data on the server and converts to standard formats. It should be noted that the tool *Collect* generates the interfaces of the mobile application from a file format XForm, a standard formatting of forms, specified in XML. However, ODK does not offer an infrastructure, but sets standards and APIs.

The product DoForms [7] allows the creation of multiplatform and mobile questionnaires. It is similar to Maritaca, however, it is not open source.

The Mafuta Go project [8] is a specific mobile application designed to find the nearest gas stations in Uganda (located in

 ISBN: 978-1-61208-366-7

East Africa). The App also compares the prices of gas. The Maritaca could be used to create the same application.

Fulcrum [9] allows the creation of Android and iOS applications for data collection. It also collects location information, so that the collected data can be viewed on maps. It is quite similar to Maritaca. On the site of the project it is possible to see several Apps created using the system; the Apps are organized in several categories, such as tourism, utilities, financial tools, etc. However, Fulcrum is not a free platform.

All these products have similarities to Maritaca, but none of them implements all proposed features, and none of them allows flexible data analysis.

III. System Architecture

The Maritaca project was developed as a cloud application [10]. The data collected using Android devices are stored in the cloud and can be visualized using standard web browsers.

There are two main components:

Mobile component: this is an Android application that interprets the XML file (questionnaire descriptor) and generates the interfaces automatically. In fact, the mobile component is an engine, based on the design pattern Interpreter [11]. The mobile component design was the key factor for allowing to create mobile applications automatically.

Server component: the server side was written in Java, using the application server *JBoss* [12] and the framework *Spring* [13]. All web services were implemented based on the *RESTful* approach [14]. The server also integrates the following products: Form Editor, Analytics Editor, Cassandra database, Hadoop file system, Solr search engine, and MongoDB.

- **Form Editor**: this is an independent Web application, written using HTML5 and Ajax. It allows the quick and intuitive development of questionnaires by implementing drag-and-drop interfaces. As a result, this component generates a questionnaire descriptor, which is persisted in XML format, and is parsed by the Mobile component.
- **Analytics Editor**: it is also an independent Web application used to create queries about the collected data.
- **Cassandra database**: it is used for scalable storage of information. It is based on the paradigm *NoSQL* [15], [16].
- **Hadoop file system**: this is a distributed file system [17], [18] used to store non structured data, such as Apps and multimedia files.
- **Solr Engine**: it is a distributed search engine [19], [20] used to enable searching of Apps. Each App has a description; we used Solr to index the keywords of this description, so that it is possible to search for specific Apps.
- **MongoDB**: this is a scalable distributed database [21] used to analyze collected data. MongoDB is a NoSQL data repository that implements data queries using a semantic similar to SQL [22].

The Figure 1 illustrates the system architecture and the relation among server components.

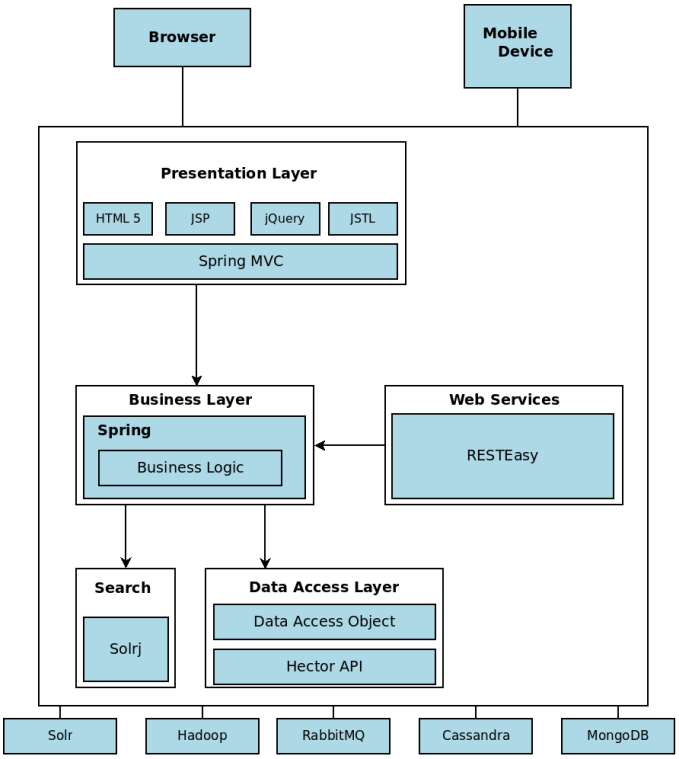

Figure 1. System architecture and main components of the server side.

A. System integration

The interaction between mobile devices and the system is always done with RESTFul services. The web layer and the RESTFul services interact with the business layer using the framework Spring.

The architecture predicts the usage of many instances from the system components (JBoss, Cassandra, MongoDB, and Hadoop) in one computational *cluster*. The load balance of the requisitions will be implemented using nginx [23].

The project also includes one additional component, the RabbitMQ [24], that is used to enqueue messages, mainly to send email messages. For example, it is used to send new passwords (if they have been forgotten), send invitations to new users, etc.

B. Funcionalities of Mobile component

The mobile component is an engine that translates the descriptor of the questionnaire (represented in XML format) into a hierarchy of instantiated objects. These objects are responsible for rendering the interfaces and implementing data validation. The computational model used to represent the questionnaires is sophisticated, and it was this technological innovation that makes the solution possible.

The technique of mapping XML into a list of objects was based on the design pattern *Interpreter* [11], where we can use a hierarchy of classes to simplify message protocol and data interpretation. In our case, see Figure 2, the mobile application is a Context Manager, which always points to the object currently in use (question being answered). Each subclass of type *Question* [25] can implement its own policies, such as, field validation, interfaces, and data storage methods.

 ISBN: 978-1-61208-366-7

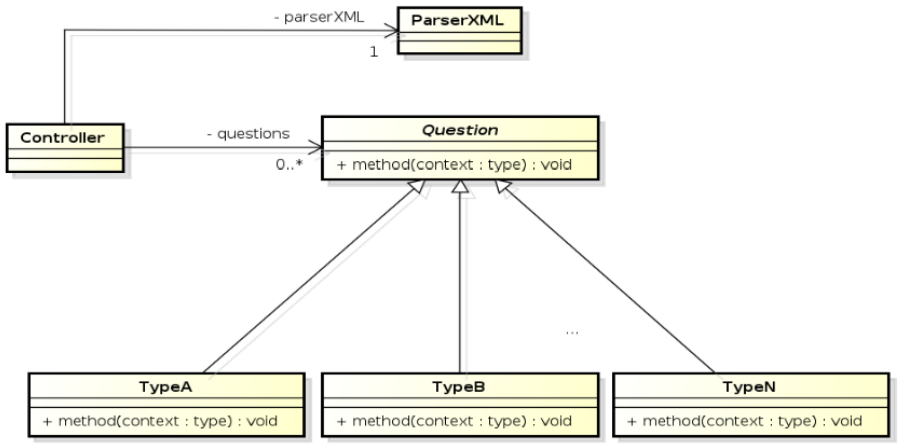

Figure 2. Application controller and XML mapping into objects using *Interpreter* design pattern.

1) Hierarchy of objects: We create a hierarchy of classes in which the questions are mapped. The interfaces for manipulating objects allows, for example, to validate the value collected (using method *validate*), to render the interface for each type of question (method *getLayout*) and to control navigation between each of the questions (methods *getNext* and *getPrevious*).

The *validate* method allows, for example, to validate minimum and maximum limits for the inputs of numeric data. Thus, if a form contains a question about the interviewee's age, the minimum and maximum value for the answers can be defined, for example, respectively as 0 and 105. Data validation is implemented in the class that defines the type of question and is a very effective method to prevent incorrect data collection.

2) Techniques for XML interpretation: The project uses the Simple framework [26] for the serialization and deserialization of the XML files. That is, the framework directly converts XML files into objects and vice versa. This technique is simpler to implement than a XML parsing, which simplifies code maintenance and extension.

3) XML file format: The mobile component interprets the XML file generated by the form editor. In this file, each question is represented as a XML tag with the following basic attributes: *id*, *next*, *previous*, *required*, *label*, *help*, *type* and *default*.

The following fields: *id*, *next* and *previous* are numeric type; *id* identifies the number of the current question, *next* points to the id of the next question and *previous* points to the previous question.

The field *required* determines whether the question is mandatory; its value is defined as: *true* or *false*. The field *label* must contain a question text to be shown in the questionnaire. The attribute *help* is not mandatory and contains a clarification about the question. The attribute *type* defines the type of the data, and can assume, for example, the following values: *text*, *number*, *radiobutton*, *combobox*, *video*, *gps*, etc. The attribute *default* contains a default value of the current question.

Furthermore, some types of questions may have a *conditions* structure, which is used to define the conditional navigation between questions. By using this tag, the response to the current question is used to determine which question will be displayed. For example, consider the following question: "*How old are you?*". If the answer is a value under 18 years, the next question might be: "*What is the name of your parent?*"; otherwise, this question could be omitted.

4) Authentication: Before the first data collection, the mobile application user must authenticate their identity on the server; this guarantees that the user has permission to collect data for that form. This process is done using the OAuth authorization framework [27] that enables third-party applications to obtain limited access to an HTTP service. All data transfer is implemented using RESTFul services and JSON message format.

C. Capturing unusual data

In addition to collecting usual data, such as texts and numbers, the solution also allows collection of unusual data, such as multimedia (audio, video and images) [28], geolocation [29], drawings, barcodes, etc. In summary, the questionnaire can include questions such as: *What is your current location? Take a picture! Record an audio message!*

The implementation of new types of data captured can be easily performed. To do this, simply extend the class *Question* and make the appropriate changes in the parsing class.

D. Automatic generation of Apps

Every time a form is saved in the Form Editor, the system generates a new Android App (executable APK file format) and stores it in the Hadoop distributed file system. This was not the first approach adopted. Initially, we planned to create a single Android application, where the XML descriptors would be loaded.

However, it was difficult to maintain several versions of Apps, thus we implemented this new process of form publication. The process of compilation takes a few seconds, but it is done in the background, and thus does not affect the perceived usability of the server system.

IV. SYSTEM USAGE

This section briefly describes how the system can be used [3], [30].

 ISBN: 978-1-61208-366-7

First, the user can access the system and create an App (mobile application) for data collection; the user must be registred in the platform. The App can be installed onto any compatible mobile device that runs Android 2.2, or above. The installation of the mobile App is simple and straightforward.

On the mobile device, the application allows data to be collected using user-friendly interfaces. Figure 3(a) shows the main interface of the application. Figure 3(b) shows a screenshot of a data collection interface. By default, it uses a Wizard interface design pattern, i.e. each question is shown on a screen. The user can also choose to see a complete list of questions, it is useful in large surveys.

(a) Screenshot of the home interface of the Mobile component.

(b) A data collection interface for a question of type Text.

Figure 3. Interfaces automatically created.

To carry out the data collection, it is not necessary to be connected to the Internet. After collection, data is stored on the mobile device. An Internet connection is needed only for authentication and data upload.

The user can use the web interface to visualize and manage forms, see Figure 4. The list of forms is organized in two panels, forms created by the user (top) and shared forms (bottom).

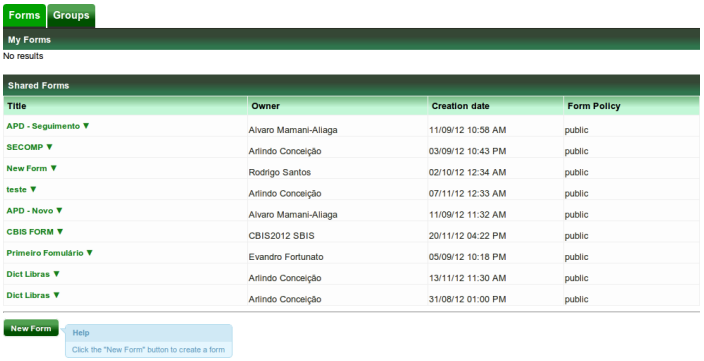

Figure 4. Form management interface.

Currently, the system allows the creation of three types of forms: private, public and shared. In Table I, these policies are summarized. As the names indicate, the **private** form can be only used by its owner and the **public** form can be viewed by any user of the platform.

For **shared** forms, the owner can invite other users, and the owner can see all data collected. Furthermore, the shared forms can be divided into two subtypes: hierarchical and social. For **shared-hierarchical** forms, the owner can invite users, for example, users A and B, but the data gathered by user A is not visible by user B, and vice versa. In turn, for **shared-social** forms, all invited users can visualize the data collected by one another.

TABLE I. DATA SHARING POLICIES.

		Private	Shared Hierarchical	Shared Social	Public
Forms	Read	Owner	Owner and List	Owner and List	All
	Update	Owner	Owner	Owner	Owner
Answer	Read	Owner	Owner	Owner and List	All
	Collect	Owner	Owner and List	Owner and List	All

V. PIPELINE DESIGN FOR DATA ANALYSIS

In addition, we recently created a solution that allows to analyze the data. The user can configure a pipeline for data processing, as illustrated in Figure 5. The user can: filter the data (time, specific fields, or geographical region), apply data transformations similar to SQL commands (*order by, sum, average*, etc.), and, finally, choose a data visualization mode, such as table or map.

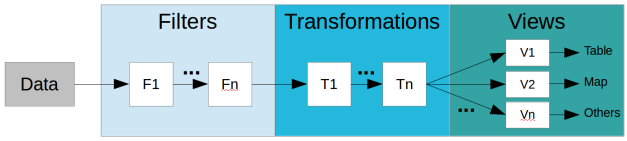

Figure 5. Analytics module, based in a pipeline of information

For example, suppose that a user creates a App to collect the price and location of restaurants. Then, the user can define a query using the pipeline architecture. A filter can be applied to restrict the results to restaurants near the user. A transformation can be applied to order the restaurants by price. Finally, the user can choose to see the results as a map. Figure 6 shows the result of a query created using the *Analytics* module. In fact, every time a user makes a query request, the data is imported from the Cassandra database to the MongoDB, where the filters and transformations are applied using a map reduce strategy [22].

VI. APPLICATIONS

The project allows:

- **The creation of your own mobile data application.** The user can create and modify their own app for data collection. A salesman can coordinate a customer's orders, students can share pictures of a party, and parents can visualize where their children are. There are no costs and no need for programming skills.
- **Your own Social Network.** The user can create an application for data collection and define three different models of data sharing. Thus, it is possible to create social networks for specific interests.

 ISBN: 978-1-61208-366-7

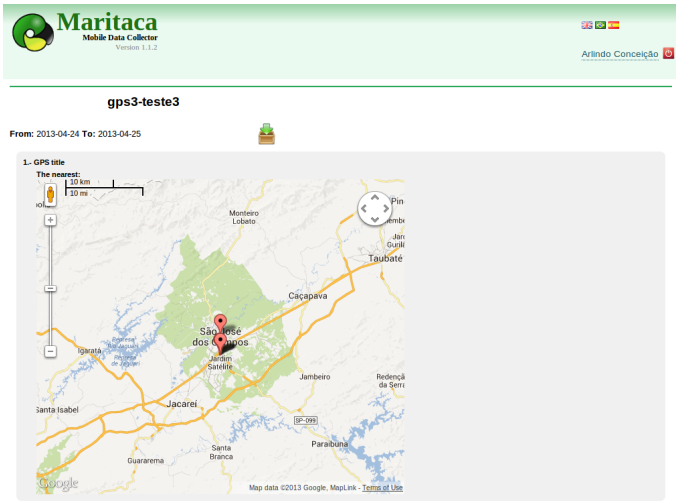

Figure 6. A result of *Analytics* module.

- **Extracting Collective Intelligence.** By using the *Analytics* module, the user can define special views of the data, offering services with high aggregated value.

Therefore, the user can create an effective and complete mobile solution using simple interfaces. There are many possibilities. Currently, we are using the system to collect data in homecare health services and nutritional monitoring.

VII. Conclusion and Future Work

We explored the concept of Mobile User Empowering, that provides to the user the power to create, modify and use his own mobile applications. The project offers tools to collect, share and analyze mobile data, allowing users total customization of software requirements using simple interfaces, without needing knowledge of programming languages or IT infrastructure. The architecture has been developed to cover most mobile applications based in questionnaires, storing both conventional (number, text, etc.) and non-conventional data (video, pictures).

In addition, we proposed a pipeline architecture and its cloud implementation that can be used to data analysis.

Currently, the project can create mobile applications on Android platform. We are developing a multi-platform version using the Phonegap technology. In addition, we are deploying the solution in a private cloud with high processing power.

The latest version of the project is available for evaluation [3]. The source code and additional documentation can be found at the code repository [30].

Acknowledgment

We thank FINEP [31] by funding this research project.

References

[1] A. F. da Conceição, J. V. Sánchez, T. Barabasz, A. H. Mamani-Aliaga, B. G. dos Santos, and M. F. Mendonça, "Open architecture for mobile data collection using cloud computing," in International Workshop on Mobile Cloud Computing: Data, Management & Security (mCloud). In conjunction with 14th IEEE International Conference on Mobile Data Management (IEEE MDM). Milan, Italy., 2013.

[2] R. Ghiglione, B. Matalon, C. Pires, and A. de Saint-Maurice, O inquérito: teoria e prática. Lisboa: Editora Celta, 1997.

[3] A. F. da Conceição et. al, "Maritaca," http://maritaca.unifesp.br, accessed: 2014-06-07.

[4] MIT, "Mit App Inventor," http://appinventor.mit.edu, accessed: 2014-06-07.

[5] Nokia Corp., "Nokia Data Gathering," https://nokiadatagathering.net, accessed: 2014-06-07.

[6] ODK Community, "Open Data Kit," http://opendatakit.org, accessed: 2014-06-07.

[7] doForms Inc., "DoForms," http://www.doforms.com, accessed: 2014-06-07.

[8] MafutaGo, "Official Mafuta Go website," http://www.mafutago.com, accessed: 2014-06-07.

[9] Fulcrum, "Gather data anywhere, anytime with fulcrum," http://fulcrumapp.com, accessed: 2014-06-07.

[10] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, 2010, pp. 50–58.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns. Addison-Wesley Professional, 1994.

[12] S. Davis and T. Marrs, "JBoss at work: A practical guide," 2005.

[13] B. Tate and J. Gehtland, Spring: a developer's notebook. O'Reilly Media, Incorporated, 2005.

[14] L. Richardson and S. Ruby, RESTful web services. O'Reilly Media, Incorporated, 2007.

[15] E. Hewitt, Cassandra: the definitive guide. O'Reilly Media, Incorporated, 2010.

[16] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," SIGOPS Oper. Syst. Rev., vol. 44, no. 2, 2010, pp. 35–40.

[17] T. White, Hadoop: The definitive guide. O'Reilly Media, 2012.

[18] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–10.

[19] D. Smiley and E. Pugh, Solr 1. 4 Enterprise Search Server: Enhance Your Search with Faceted Navigation, Result Highlighting, Fuzzy Queries, Ranked Scoring, and More. Packt Publishing, 2009.

[20] O. Gospodnetic and E. Hatcher, Lucene. Manning, 2005.

[21] K. Chodorow, MongoDB: the definitive guide. O'Reilly, 2013.

[22] MongoDB Aggregation, http://docs.mongodb.org/manual/aggregation, accessed: 2014-06-09.

[23] nginx, http://nginx.org, accessed: 2014-06-07.

[24] "RabbitMQ Messaging," http://www.rabbitmq.com/, accessed: 2014-06-07.

[25] A. Durham, E. Sussumu, and A. da Conceição, "A framework for building language interpreters," in Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA). Educators Symposium. ACM, 2003, p. 196.

[26] Simple Framework, http://simple.sourceforge.net, accessed: 2014-06-07.

[27] A. Tassanaviboon and G. Gong, "OAuth and ABE based authorization in semi-trusted cloud computing: aauth," in Proceedings of the second international workshop on Data intensive computing in the clouds, ser. DataCloud-SC '11. New York, NY, USA: ACM, 2011, pp. 41–50.

[28] A. da Conceição, R. Pereira, J. Rezende, B. Silva, R. Correia, H. Domingues, R. Kon, and F. Kon, "Projeto Borboleta: Ferramentas Móveis e Multimídia para Atenção Básica Domiciliar," in Congresso Brasileiro de Informática em Saúde. Artigo curto, 2008.

[29] A. El-Rabbany, Introduction to GPS: The Global Positioning System. Artech House Publishers, 2002.

[30] Maritaca Team, "Maritaca Source Code," http://sourceforge.net/p/maritaca, accessed: 2014-06-07.

[31] Financiadora de Estudos e Projetos (FINEP), http://www.finep.gov.br, accessed: 2014-06-07.

 ISBN: 978-1-61208-366-7