European Project Nº: **FP7-614154**
Brazilian Project Nº: **CNPq-490084/2013-3**
Project Acronym: **RESCUER**

Project Title: **Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management**

Instrument: **Collaborative Project**
European Call Identifier: **FP7-ICT-2013-EU-Brazil**
Brazilian Call Identifier: **MCTI/CNPq 13/2012**

Deliverable D2.5.2
Development of the Mobile Solution 2

Due date of deliverable: PM20
Actual submission date: October 12, 2015

Start date of the project: **October 1, 2013 (Europe)** | **February 1, 2014 (Brazil)**
Duration: **30 months**
Organization name of lead contractor for this deliverable: MTM Tecnologia

Dissemination level		
PU	Public	✓
PP	Restricted to other program participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

RESCUER

Executive Summary

Development of the Mobile Solution 2

This document presents deliverable D2.5.2 (Development of the Mobile Solution 2) of project FP7-614154 | CNPq-490084/2013-3 (RESCUER), a Collaborative Project supported by the European Commission and MCTI/CNPq (Brazil). Full information on this project is available online at http://www.rescuer-project.org.

Deliverable D2.5.2 provides the results of the second iteration of Task 2.5 (Development of the Mobile Solution). In this task, we develop the RESCUER component called Mobile Crowdsourcing Solution taking into consideration the specifications received from the other tasks of Work Package 2 (Mobile Crowdsourcing Solution) and the most feasible mobile technologies for the RESCUER Project. The main result of this task in the first project iteration was a version of the Mobile Crowdsourcing Solution with crowdsourcing information gathering features, which was evaluated during the FIFA World Cup 2014 (evaluation reported in D5.2.1: Evaluation of the Mobile Crowdsourcing Solution 1). In the second project iteration, this task delivered an operational version of the Mobile Crowdsourcing Solution for crowdsourcing information gathering to be evaluated by employees of the Industrial Park of Camaçari in November 2015 (to be reported in D5.2.2: Evaluation of the Mobile Crowdsourcing Solution 2). The prototype to be evaluated at this occasion might include the follow-up interaction features as well.

List of Authors

Paulo Soares – MTM

Gustavo Perez – MTM

List of Internal Reviewers

Paulo Jr. – UFBA

Kai Breiner – Fraunhofer (1st version)

Tobias Franke – DFKI (2nd version)

Karina Villela – Fraunhofer

RESCUER

Contents

1. Introduction

The purpose of the Mobile Crowdsourcing Solution in the RESCUER Project is to provide an interface to enable crowds in large-scale events or industrial parks to send information about an incident to a command and control centre and to receive warnings and directions on how to proceed.

Deliverable D2.1.2 (Conceptual Model of Mobile User Interaction in Emergencies 2) updated the study of the human behaviour in emergency situations in order to tune the design of the Mobile Crowdsourcing Solution with regards to the gathering of information provided by the crowd spontaneously or as a result of a known process (Deliverable D2.2.2: Crowdsourcing Information Gathering 2). Furthermore, D2.3.1 (Group-targeted Follow-up Interaction 1) provided the specification of the Mobile Crowdsourcing Solution with regards to the follow-up interaction trigged by the command and control centre. Due to some development issues faced in the first project iteration and some infrastructure work done in the second iteration, features specified in D2.3.1 have been assigned to the next version of the demonstrator.

1.1. Purpose

The purpose of this deliverable is to describe the main decisions and results of the development of the Mobile Crowdsourcing Solution as by the end of the second project iteration, which is still only concerned with information gathering but it is completely functional (persistence and upload of images and video), allows easy adaptation of the application to new incident types, integrates the DFKI library for crowd sensing and properly communicates to the ERTK. The presented results are available since the due date of this deliverable and are going to be evaluated in the second iteration of Task 5.2 (Evaluation of the Mobile Crowdsourcing Solution). This document focuses in the mobile application and makes no reference to the RESCUER's backend.

1.2. Change Log

Table 1 summarizes the changes performed in this document with regards to its previous version (D2.5.1: Development of the Mobile Solution 1), submitted in January 21, 2015.

Table 1: Change log with regard to D2.5.1 (January 21, 2015)

Change Location	Change Description
Section 2	• Inclusion of Section 2.3 Dynamic Generation Screen
Section 3	• Update of the Introduction text • Update of the implementation status of the specified features • Inclusion of the new specified features in Table 2
Subsection 3.1	• Update of the Mobile App Screens and corresponding text
Section 4	• Update of the conclusion

1.3. Partners' Roles and Contributions

The partner responsible for this deliverable is MTM. Fraunhofer contributed to this deliverable by conducting a one-week workshop for specification and prototyping of the Mobile Crowdsourcing Solution for the first project iteration and by conducting regular project monitoring meetings in the first and second project iterations. It is in the scope of MTM's work the research of the most appropriate technologies to be applied in the project development and in the distribution of the mobile applications, taking into consideration the strategy and guidelines defined in D1.4.2 (Portability & Variation Management Strategy 2).

1.4. Document Overview

The remainder of this document is structured as follows:

- Chapter 2 describes project decisions and selected technologies, which includes the dynamic generation of screens introduced in the second project iteration.
- Chapter 3 presents the implementation status of the specified features (D2.2.2 and D2.3.1) and the screenshots of the second iteration of the Mobile Crowdsourcing Solution.
- Chapter 4 presents the conclusion of this document.

2. Project Decisions

An important decision when developing a mobile app is the approach to be used. There are mainly three kinds of approaches for mobile application development: Native, Hybrid and Web App.

The native approach requires the same system to be developed independently for each mobile platform to be supported. It consumes a lot of effort to develop and maintain those different source codes. The Web App approach uses pure web technology to develop the system. Despite the fact that it uses one source code for all platforms, the technology behind it does not give access to core features of the devices that are important to the RESCUER Project, like GPS, camera, accelerometer, notification, background threads, among others. The hybrid approach is able to access the core features of the device and share major parts of the source code between different platforms. Hence, this was the approach chosen for the development of the Mobile Crowdsourcing Solution.

Section 6.3 (Development Approaches) of deliverable D1.4.2 (Portability & Variation Management Strategy 2) presents a study about the three approaches for mobile development.

2.1. Hybrid Frameworks

The next step after choosing the development approach was to identify the best frameworks to use for hybrid development. Three frameworks were evaluated for use in the project: Phonegap [1], Xamarin [2] and Titanium [3].

Xamarin has different versions available. One of them is free, but is limited and has some constraints like the size of the compiled application [10]. Xamarian is not open source and this deliverable opted for an open source framework that provides the maximum level of reuse.

Titanium's source code is not entirely reusable between platforms [11]. Approximately 70% - 80% of the code can be reused between platforms. This results in the need for native code for each platform, reducing the advantage of using a hybrid approach.

Phonegap supports access to all main core features of the devices, as presented in Figure 1, and the entire code can be reused between platforms, except when the solution requires the use of native libraries.

	iPhone / iPhone 3G	iPhone 3GS and newer	Android
Accelerometer	✓	✓	✓
Camera	✓	✓	✓
Compass	x	✓	✓
Contacts	✓	✓	✓
File	✓	✓	✓
Geolocation	✓	✓	✓
Media	✓	✓	✓
Network	✓	✓	✓
Notification (Alert)	✓	✓	✓
Notification (Sound)	✓	✓	✓
Notification (Vibration)	✓	✓	✓
Storage	✓	✓	✓

✓ - supported feature
x - unsupported feature due to hardware or software restrictions

Figure 1: Core features supported by Phonegap [12]

In addition, Phonegap has a big community and good documentation and online resources. The Phonegap development uses web technologies like HTML5, CSS and JavaScript. Some frameworks to create mobile application components with HTML5, CSS and JavaScript were analyzed including Ionic [4] with AngularJS [5], Sencha Touch [6], jQuery Mobile [7] with Backbone [8], and Kendo[9]. These frameworks have the web components adapted for the use in mobile devices, and reduce the work and time of implementation of mobile applications.

2.2. Phonegap

Phonegap works like a wrapper. It encapsulates a web page renderer into a native application; therefore it can access the features of the mobile device and still ensure reuse of code. The graphical interface of the application is developed with HTML5, CSS and JavaScript, and the resulting code can be interpreted in different platforms that have a web renderer.

Figure 2 shows the operation of Phonegap and the components used to generate the executable code for the various supported platforms.

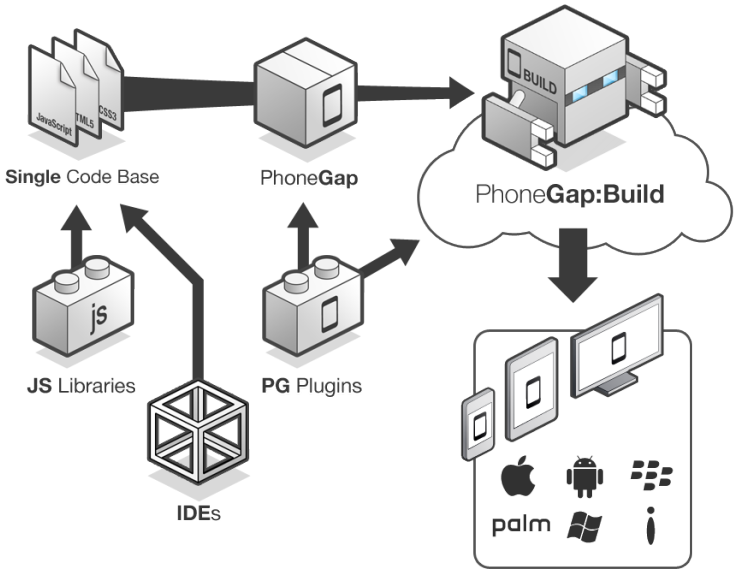

Figure 2: Phonegap development components [13]

2.3. Dynamic Generation Screens

Another important project decision in the scope of the Mobile Crowdsourcing Solution was the introduction of dynamic generated screens during the second project iteration in order to deal with variation in incident types, user profiles, and languages. To define the dynamic data to be shown in the application, a JSON (JavaScript Object Notation) file was created which is embedded on the application and can be updated from an online server – with this file the Quick Report and Standard Report Screens are built.

The JSON structure was defined as:

```
{
  "version": 0,
  "personRoles": {
    "1": {
      "name": {
```

```
      "en": "Civilian",
      ...
    },
    "auth": 0,
    "pass": "",
    "forms": [
      "1",
      ...
    ]
  },
  ...
},
"languages": [
  {
    "name": "English",
    "mn": "en"
  },
  …
],
"forms": {
  "1": {
    "event": "FIRE",
    "name": {
      "en": "Fire",
      ...
    },
    "color": "#f4b04b",
    "icon": "I=",
    "sections": [
      {
        "name": "Fire Report",
        "fields": [
          {
            "order": 1,
            "id": "1",
            "label": {
              "en": "Are you seeing people injured?",
               ...
            },
            "type": "LIST",
            "mandatory": 0,
            "min-lenght": "",
            "max-lenght": "",
            "values": {
              "component": "radio",
              "textlist": [
                {
                  "en": "injured people",
                  ...
                },
                …
```

```
                ]
              }
            }
          ]
        }
      ]
    }
  }
}
```

With the following semantics:

- *version*: the version of the JSON file
- *personRoles*: the user profiles to be chosen in the application
    - *name*: the name of the user profile
    - *auth*: if the user profile needs a password to login. (0 = it doesn't need password, 1 = it needs password)
    - *pass*: the password to access using the user profile
    - *forms*: the ids of the forms to be shown
- *languages*: the languages of the texts to be shown to the user
- *forms*: all the forms of each person role
    - *event*: incident type
    - *name*: name of the incident type
    - *color*: color In RGB of the custom graphic interface
    - *icon*: the icon of the incident type in base64
    - *sections*: the form content
        - *name*: incident report title
        - *fields*: fields for the user to provide information about the incident
            - *order*: order of the appearance in the form
            - *id*: field id
            - *label*: the text the appears before the component
            - *type*: field type
            - *mandatory*: if the field needs to be filled
            - *min-length*: minimum amount of characters in text fields
            - *max-length*: maximum amount of characters in text fields
            - *values*: values of the component
                - *component*: type of the component (values should be: combo, check, radio or list)
                - *textlist*: text to be shown in each value field

Using the form detailed above, the application builds the following screens: Language, Profile, Quick Report and Standard Report. For more details about each screen see Section 3.8 of D2.2.2 (Crowdsourcing Information Gathering 2).

3. Specifications vs Implementation

This chapter presents the features implemented in the first as well as in the second project iteration. The objective of the first iteration was to have a first version of the Mobile Crowdsourcing Solution to be evaluated during the FIFA World Cup 2014, in Brazil. Figure 3 presents the interaction diagram with the features implemented in this previous version of the Mobile Crowdsourcing Solution. The focus of the second iteration was to allow for easily adaptation of the application to new incident types and to integrate the DFKI library for crowd sensing.

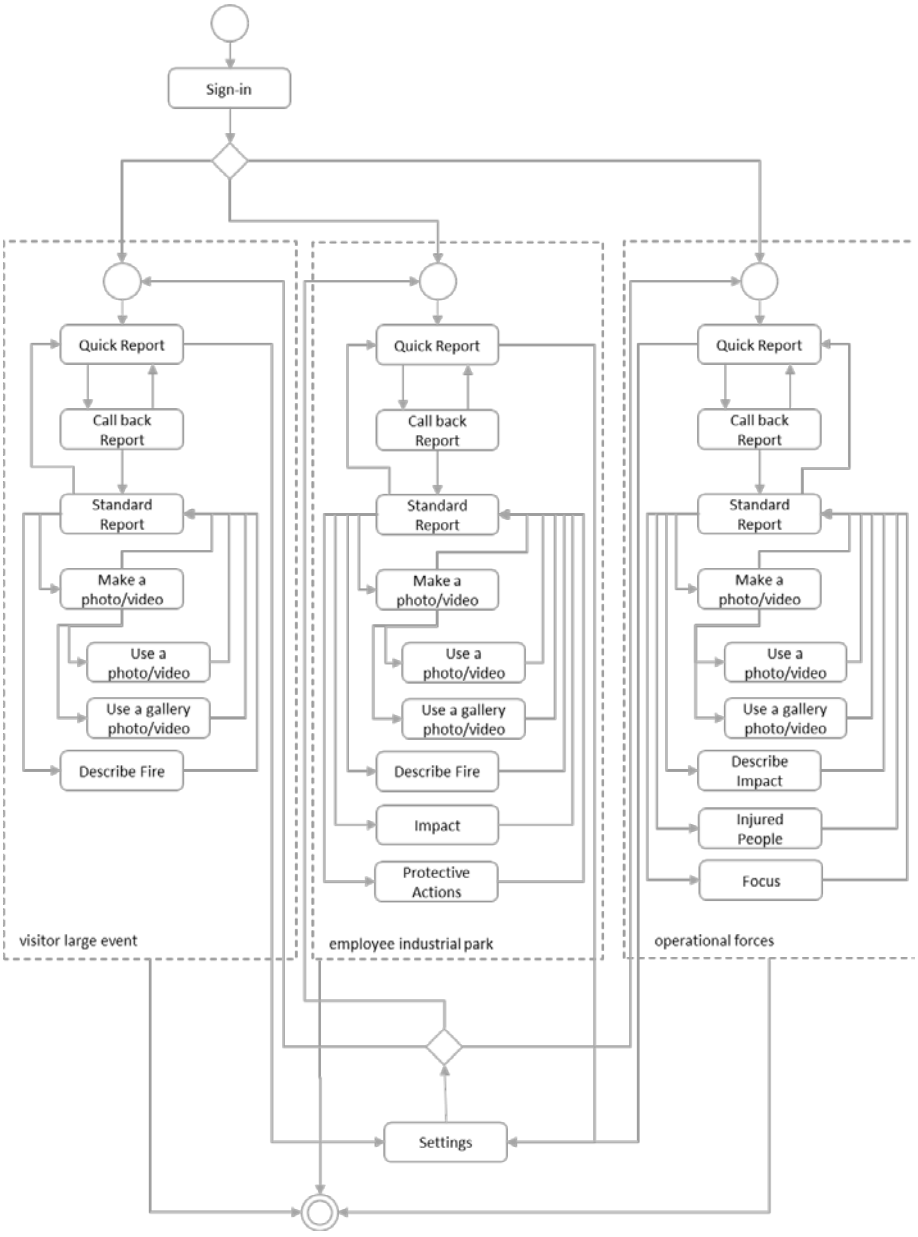

Figure 3: Interaction case diagram

From the features listed in Figure 3, the feature “Use a gallery photo/video” was removed in the second iteration because the Data Analysis Components need sensor data information collected at the moment that the pictures or videos are produced. Table 2 describes the features defined for the mobile app and their status.

Table 2: Implementation status of the specified features

Features	Description	Implemented	Partially Implemented	Not Implemented	Comments
Sign-In Screen	The user set his profile to get profile based information	x			
Quick report	The user can send a very quick report, which is just a notification of an emergency situation of a specific type (e.g., fire). This action generates a new incident report.	x			
Call back report	The user can take back the incident notification (Quick Report) in case of a false alarm.	x			
Defining the incident spot on the map	Map component that allows the users to change the location of the incident.		x		An open-source JavaScript library for mobile-friendly interactive maps, called Leaflet (http://leafletjs.com/) was used in the app, but it stopped to work well in the new version of the Iconic Framework.
Describing the incident	The user can describe the incident filling the standard form.	x			The following forms were implemented: “fire”, “gas leakage”, ‘explosion” and “environmental”.
Taking a photo or recording a video	Camera component that allows the users to take photos or videos of the situation. The following sensor data will be recorded automatically, when taking a photo or video: • Position (GPS) • Timestamp • Gyroscope	x			

Features	Description	Implemented	Partially Implemented	Not Implemented	Comments
User profile and permissions	The user can edit his profile information as well as permissions related to the information monitoring and recording done by the app.			x	The server side of this feature has not been implemented yet.
Upload of reports to RabbitMQ	The screen information should be send to the server. The following information is kept as a single report: • Report ID • GPS position • Timestamp • Report information (informed by the user through the interaction with the user interface)	x			
Upload of data to Amazon S3	The photos and videos should be send to the Amazon S3 account. • Photo file • Video file	x			
No save button	People do not need to save or to explicitly send information to the RESCUER backend. Any report will be automatically sent after 5 seconds without interaction.			x	Currently the information is send just when the user taps on a send button.
Constant interaction	During the upload of a picture/video the user can continue interacting with the mobile application.	x			
Dynamic Generation of Screens	The Quick Report Screen and the Standard Report Screen should be created based on the incident type and forms defined on the server side	x			

Features	Description	Implemented	Partially Implemented	Not Implemented	Comments
Online/Offline capabilities	The system has to work in both online and offline mode. Furthermore, the system requires a well-defined synchronization concept that works in both online and offline mode.			x	The app is just loading when the smartphone is online.
Persistence	As long as a report is open, the information has to persist on the screen. After leaving the screen, when starting a new report the report screen is reset.	x			
DFKI library integration	The MCS has to use the DFKI Sensing Library to: • Send message data to RabbitMQ • Subscribe to RabbitMQ messages • Retrieve sensor data for data annotations • Activate the Crowd Sensing		x		The subscription to the RabbitMQ messages is integrated but the features that use it have not implemented yet. The Crowd Sensing has not been integrated yet.

3.1. Mobile App Screens

The screens of the Mobile Crowdsourcing Solution for Android are provided below. All screens are detailed in D2.2.2: Crowdsourcing Information Gathering 2, subsection 3.8.

The “Quick Report” screen (Figure 4) is where the user can inform that the incident occurred by selecting the type of incident. Pressing one of the incident type buttons, a set of information related to the incident is sent to the server, this set of information is composed by:

- unique report id
- user profile
- the report position
- timestamp
- incident type

Figure 4: “Quick Report” screen

Figure 5 shows the Call Back Report Screen which informs the user that the emergency has been reported and that he can continue giving more information about the incident, if s/he is in condition for doing so. S/He can take back the report in case it was sent by mistake.

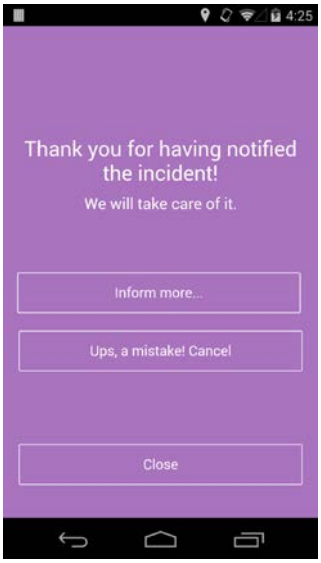

Figure 5: "Call Back Report" screen

Figure 6 shows the screen where the user provides additional information about the incident. The user can also take pictures and send the report. The screenshot shows an explosion incident, other screens can be found in D2.2.2: Crowdsourcing Information Gathering 2.

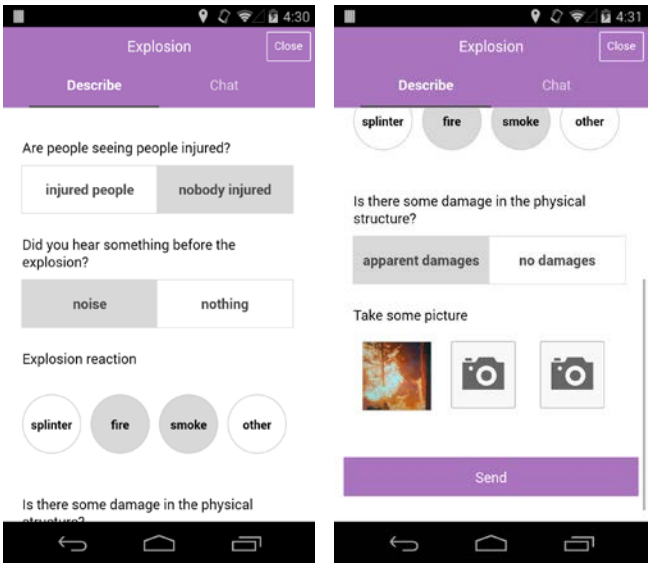

Figure 6(a) and 6(b): "Standard Report" screens

In the Free Report (Chat) screen, shown in the Figure 7, the user can send text, photos and videos, as many as s/he desires.

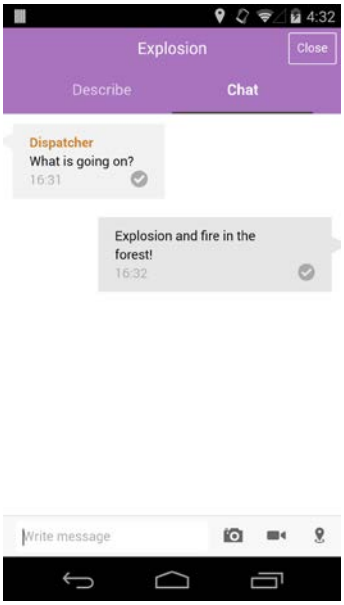

Figure 7: "Free Report" screen

4. Conclusion

This document and the current prototype of the Mobile Crowdsourcing Solution comprise D2.5.2 (Development of the Mobile Solution 2).

The features planned for first iteration were:

- Quick report of an incident;
- Taking back of a report that was sent;
- Standard report for large-scale events, which includes:
  - Location of the incident on the map;
  - Description of the incident;
  - Picture of the incident;
  - Video of the incident;
  - Selection of a picture or a video from the device gallery.

The selection of a picture or video from gallery was removed from the set of feature of the Mobile Crowdsourcing Solution in the second iteration because it is not possible to have the respective sensor data information, which is required for data analysis. In addition, the exclusion of this feature adds to the reliability of the information (images/videos) provided as part of the incident report.

The features planned for second iteration were:

- Data persistence;
- Integration with the ERTK;
- Integration with the DFKI Sensing Library;
- Dynamic generation of screens;
- Upload of images and videos to the Amazon S3.

The use of a hybrid approach to implement the Mobile Crowdsourcing Solution presented some challenges that required more implementation time than initially planned in the first iteration. We faced performance issues because of the number of plugins required by the application. We had an issue with one of the plugins, which required the app to be online to be used.

The issues faced during the development in the first iteration had an impact on the implementation of certain features (e.g. on the features for taking back reports, making video, and taking picture), which could only be finished in the second iteration. Regarding the goals for the second iteration, the dynamic generation of screens and the upload of images and videos to Amazon S3 are implemented, the integration with ERTK (which suffered changes) is finished, but the integration with the DFKI Sensing Library still needs to be tested.

The prototype of the first iteration was evaluated during the 2014 FIFA World Cup and by employees of the Industrial Park of Camaçari. The results of these evaluations were reported in D5.2.1 (Evaluation of the Mobile Crowdsourcing Solution 1). The current prototype is planned to be evaluated by employees of the Industrial Park of Camaçari in November 2015.

In the third iteration the prototype will include the features specified in D2.3.1 (Group-targeted Follow-up Interaction 1) as well as the features to be specified in D2.4.1 (Group-targeted Crowd Steering).

Bibliography

[1] Phonegap. Date of Access: 14.08.2014. http://phonegap.com/]
[2] Xamarin. Date of Access:14.08.2014. http://xamarin.com/]
[3] Titanium. Date of access:14.08.2014. http://www.appcelerator.com/titanium]
[4] Ionic Framework. Date of Access:14.08.2014. http://ionicframework.com/]
[5] AngularJS. Date of Access:14.08.2014. https://angularjs.org/]
[6] Sencha Touch. Date of Access:14.08.2014http://www.sencha.com/products/]
[7] JqueryMobile. Date of Access:14.08.2014. http://jquerymobile.com/]
[8] BackboneJs. Date of Access:14.08.2014. http://backbonejs.org/]
[9] Kendo UI. Date of Access:14.08.2014. http://www.telerik.com/kendo-ui]
[10] Xamarin Pricing. Date of Access: 11.09.2014. https://store.xamarin.com/]
[11] Titanium SDK. Date of Access: 11.09.2014. http://www.appcelerator.com/titanium/titanium-sdk
[12] Phonegap supported features. Date of Access: 11.09.2014. http://phonegap.com/about/feature
[13] Phonegap development. Date of Access: 11.09.2014. http://www.metaltoad.com/service/phonegap-application-development