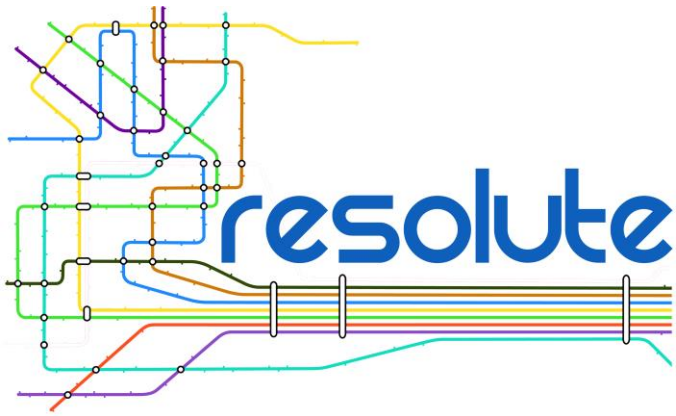

MOBILE EMERGENCY SUPPORT APP

Project Title	RESOLUTE
Project number	653460
Deliverable number	D5.4
Version	6.1
State	Final
Confidentially Level	PU
WP contributing to the Deliverable	WP5
Contractual Date of Delivery	M24 (30/04/2017)
Finally approved by coordinator	06-06-2017
Actual Date of Delivery	06/06/2017
Authors	T. Theodorou, A. Drosou, A. Zamichos, I. Kalamaras, A. Tryferidis, T. Tsirelis,
Email	{theodorou, drosou, zamihos, kalamar, thanasic, tsirelis}@iti.gr
Affiliation	CERTH
Contributors	**FRAUNHOFER:** Jan-Paul Leuteritz (Jan-aul.Leuteritz@iao.fraunhofer.de), Harald Widlroither (harald.widlroither@iao.fraunhofer.de) **UNIFI:** E. Bellini

Funded by the Horizon 2020
Framework Programme of the European Union

Executive Summary

During this deliverable will see in details the works that was done in the Task 5.3 of WP5 within the RESOLUTE project. This task is dedicated to the development of ESSMA application, a mobile emergency support app that aims to assist the resilience of a community.

Project Context

Work package	WP5: Platform front-end and end user application
Task	T5.3: Emergency Support mobile app
Dependencies	Input for tasks 5.1, 5.2, and 5.3

Contributors and Reviewers

Contributors	Reviewers
Traianos-Ioannis. Theodorou (theodorou@iti.gr)	Emanuele Bellini (emanuele.bellini@unifi.it)
Anastasios Drosou (drosou@iti.gr)	Roberto Di Vincenzo (roberto.divincenzo@swarco.com)
Alexandros Zamichos (zamihos@iti.gr)	
Ilias Kalamaras (kalamar@iti.gr)	
Athanasios Tryferidis (thanasic@iti.gr)	
Triantafyllos Tsirelis (tsirelis@iti.gr)	
Jan-Paul Leuteritz (Jan-Paul.Leuteritz@iao.fraunhofer.de)	
Harald Widlroither (harald.widlroither@iao.fraunhofer.de)	

Version History

Version	Date	Authors	Sections Affected
01	25/01/2017	T. Theodorou, A. Drosou	ALL

02	12/02/2017	T. Theodorou, A. Zamichos	ALL
03	10/03/2017	T. Theodorou, I. Kalamaras	ALL
04	07/04/2017	T. Theodorou, A. Tryferidis	ALL
05	15/05/2017	T. Theodorou,, A. Drosou, A. Zamichos, T. Tsirelis, I. Kalamaras, A. Tryferidis,	ALL
06	31/05/2017	T. Theodorou,, A. Drosou, A. Zamichos, T. Tsirelis, I. Kalamaras, A. Tryferidis	ALL
06.1	06-06-2017	P. Nesi, E. Bellini	All

Abbreviations

Abbreviation	Full term
CRAMSS	Collaborative Resilience Assessment and Management Support System
ESSMA	Emergency Support smart mobile app
eDSS	Evacuation Decision Support System
GUI	Graphical User Interface
UI	User Interface
UTS	Urban Transport System(s)
OS	Operating System
API	Application Programming Interface
SMS	Short Message Service
REST	Representational State Transfer
CRUD	Create Read Update Delete
MVC	Model View Controller
URI	Uniform Resource Identifier
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol

JSON	JavaScript Object Notation

Copyright Statement – Restricted Content

Table of Content

List of Figures

List of Tables

1 INTRODUCTION

1.1 Scope of this Deliverable

The task 5.3 “Emergency Support smart mobile app” is a part of WP5 that is devoted to implement the CRAMSS front end, the Emergency Support app and the Game based Training App according to the user experience design. The specific task was focused on the development of the ESSMA application, a mobile application that could add some assistance to its users during an emergency as well as an accurate source of information. In this deliverable, we will describe this application from different perspectives both the User Interface and the Technical point of view. Additionally, in this deliverable a number of use cases will be described, in order to depict the value of the application in these simulating scenarios.

1.2 Relation to other Deliverables

The Human Computer Interaction analysis of ESSMA application was done during *Task 5.1*. This task, has elaborated user requirements, developed and tested user interaction concepts in the shape of storybooks and wireframes. These have been provided by FHG project partner for the development of the front-end applications. User requirements were collected separately for ESSMA application, based on the specific usability / user experience engineering approach chosen. Apart from additional literature research, which resulted in the collection of specific design guidelines or general standards, this approach is mainly based on focus-group discussions and interviews. The development was based on traditional Human Factors (HF) and user centred design methods, enriched by special literature on similar systems (e.g. control room software).

One of the core components of CRAMSS is the eDSS that is part of *Task 5.2*. The eDSS is the responsible module for providing evacuation planning to the evacuation responsible (eDSS operator) in critical situations, facilitating them to take critical decisions. In order to provide optimal evacuation plans, considering the number of the involved ones and the critical situation, the eDSS co-processes and fuses all the available information from all the existing sources. Thus, the eDSS considers information retrieved from the Data Management Layer, the rest CRAMSS’s components, the eDSS’s front-end, as well as data retrieved from the ESSMA. Except from the evacuation plans the eDSS is also responsible for identifying possible individuals or groups of individuals as rescuers or to-be-rescued, assigning the appropriate task to each and providing the corresponding guidance. This guidance are used by ESSMA application in order to navigate the users to safe point as well as to guide helpers to point that need help.

1.3 Deliverable Structure

This deliverable is organized as follows. In the Section 2, we will illustrate the Application Overview of ESSMA that contains a description of the involved users. Following that, the objectives of the application will be analyzed and the features that covers these objectives will be referenced in an extended manner. In the section 3 of D5.4, the implementation of the app will be described, together with the technologies that was used we will describe the architecture of ESSMA as well as the web service API that was developed. The section 4 of this deliverable is dedicated to the user interface of the application, where every aspect of the application will be described and screenshots of the application will give a better overview of this task. Furthermore, in the section 5 a sample of uses cases will be described, illustrating the behavior of the application in different conditions.

2 APPLICATION OVERVIEW

2.1 Introduction

Emergency Support Smart Mobile App (ESSMA) main purpose to be used by professionals, such as rescue teams, and civilians. It turns its users into sensors and active agents of the resilient urban transport system. Thereby, it turns these actors into resources to be managed by the operator of the eDSS. It follows two main purposes: one is to track user movement and behaviour and thus provide the eDSS with data on a level of detail that could not be achieved otherwise. The other is to provide each user with individualized information, aiming to support self-rescue or to divert passenger flow in the UTS in case of a disruption, or to provide guidance to other citizens in need of help.

2.2 Involved Users

The target groups meant to make use of the ESSMA are civilian users and rescue professionals; in order to avoid cumbersome registration processes, which would probably hinder the widespread usage of the app, the app only distinguishes between helpers and non-helpers. ESSMA is a cross platform application that is addressed to every single citizen of the community that owns a smartphone, ether android or iOS. Downloading and installing the application users are able to select between the two aforementioned roles.

A key User Roles is the operator role. A Users that has this role does not user the ESSMA app, but the eDSS interface, however operator is responsible for the information and guidance of the ESSMA users. In the following three subsections, we will describe further each of these user roles in more details.

Although rescue professionals can belong to different organizations, yet they share common characteristics: A professional training with respect to rescue activities and a more restricted age span (from 18 years of age to retirement age).

2.2.1 Non-Helpers

Non-helpers will only receive information relevant for their self-rescue, whilst helpers receive additional information on where they can go to help civilians. Civilian users of the ESSMA and consequently of the GBTA as well, may vary in age from 14 years to old age. Younger users would supposedly need a separate app, which is not foreseen within the context of this project. The civilian helpers represents the “ad-hoc volunteers [18].

2.2.2 Helpers

Helpers are professionals, such as police, fire brigades, and emergency services. Voluntary helpers, for example civilians with first aid training, also belong to this category. All civilians that are not willing or able to help others in cases of emergency bid on to the category of non-helpers. Each user can decide on this category in the user profile. The rationale behind this decision is that once an emergency, such as a flooding, has occurred, publishing the positions of people in help does not compose a security threat.

2.2.3 Operator

Despite the fact that operator is not an ESSMA user, their role is very significant in the proper functioning of the system itself. They are responsible for the accurate information of either the helpers or the non-helpers using sending notifications for areas in danger or emergencies, updating the status of same conditions that are necessary for the users to be aware. Additionally, their main responsibility is the management of the evacuation because they actually are the eDSS manager.

2.3 ESSMA Application Objectives

ESSMA aims to assist the resilience of a community keeping civilians updated and guided for their reactions under danger situations. The main objective of the ESSMA application is to give the opportunity to civilians to be aware of emergencies and to know the most appropriate way to reach in order to be safe serving the ERMG function "*Manage Awareness & User Behavior*" of the ANTICIPATE cornerstone of Figure 1. Additionally, ESSMA assists both the "*Coordinate Emergency Actions*" and the "*Restore/Repair Operations*" functions of RESPOND cornerstones of ERMG, because it provides to citizens necessary information about danger events, location of areas that needs to be avoided, evacuation paths that reduces the risk of citizens and the communication between ESSMA and eDSS. Finally, the usability test of Task 5.1 that was described in D5.2 [20] increases the user friendliness of the application. [19]

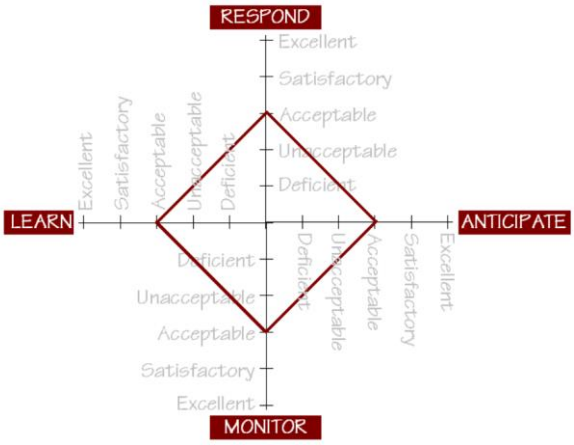

Figure 1. Resilience Analysis Grid [17]

2.3.1 Avoid Dangerous Situation

One of the main objectives of ESSMA is the prevention. Providing all necessary information through different ways to the end users, they can be informed about emergencies and danger areas, by the eDSS operator. This information can be used by users in order to avoid their involvement in that kind of conditions. This is achieved by application features like the news feeds, the navigation map or the technology of push notifications. For instance, if a number of roads are closed because of a flood, then users near to these roads, are able to be informed through push notification and learn more information by the news feeds or the map about these specific road and the can avoid them, instead of encountering unexpectedly.

2.3.2 Personal-Evacuation in Emergencies

Another objective of ESSMA is to provide guidance to its users that are involved in an emergency. It is vital for the resilience of a community and the safety of citizens to know how to reach safe point during an emergency. Additionally, this guidance needs to be personalized because each individual is different with special need and different disabilities. For this reason, it is a necessary requirement of ESSMA application the personalized guidance of the population. In addition to the aforementioned, users need to acquire navigation route to safe area when they are in danger without the necessity of an emergency.

2.3.3 Ad-hoc Assistance in Emergencies

In many cases, citizens are trapped and they cannot escape, this could happened either individually or during an emergency. It is very important for the user as well as for the authorities to know their exact location and the conditions of the event. This accurate propagation of this knowledge is crucial for the resilience a community in dangerous situations. Additionally to that, the system of ESSMA needs to provide ad-hoc assistance to these

citizens providing their position and guidance to helpers either voluntary or not. This objective is significant; in order to ensure the safety of the population as well as their sense of security.

2.3.4 Content Upload and Communication

Additionally to the aforementioned, the system aim to give the opportunity to the users to send media content to the operator as well as to communicate with them. This is an important objective in order to make easier the collaboration among the individual that are involved in an event. Additionally, it is crucial for both citizens and operator to have the ability to send media content each other such as photos or videos. The media transfer could assist the accurate information as well as the better overview of a condition by the operator. Moreover, this objective is significant from the perspective of the helpers because the media content could provide better information about the condition of an emergency or a user that is in danger, advising beforehand helpers, for the situation that they are going to face.

2.4 Features of the application

In order to fulfil the aforementioned objectives the ESSMA application was developed. A cross-platform mobile application that tries to address this communication and guidance problems during the emergencies. After the usability testing that was done in Task 5.1, it is concluded that the application will be unique for both helper and non-helper. In the following subsection, we will describe the six basic feature of ESSMA that cover the objectives that was analyzed in section 2.3.

2.4.1 Account Registration

In order to use the ESSMA, users are obliged to create a new account, during the procedure the system acquire the basic information about a user as well as their willing to be helpers. Additionally to the registration, users are able to declare more information about the physical condition or special requirement that they might have. This feature is of great importance because provide to the system and more specifically to the CRAMSS component (eDSS) all the necessary information for the production of personalized guidance during an emergency.

2.4.2 User Profiling

ESSMA acts as an input device to the user-profiling module that is described in D4.3 [21]. More specifically, it collects the information about the movement of the users, and save them into their database. The application watch the position of users and generates a list of coordinates that represents the paths that each ESSMA user followed. This information is accessible to the eDSS through web services either for a specific user or for all the ESSMA users. The eDSS uses this stored historical data for the user's paths and creates a movement user profile for each of them.

2.4.3 Data Sharing

Another feature of ESSMA id the data sharing, users and operators are able to upload media content and exchange messages. More specifically, operator is able to upload either photos or videos as well as to post text to the system. This content is visible from the ESSMA user in a view of the application and will be depicted in section 4.2. This feature could assist the accurate information of the crowd and the prevention of a danger, because this content is managed exclusively by the operator of CRAMSS. Additionally, users and operators are able to exchange media content and messages through the chat component of the ESSMA.

2.4.4 User Notifications

Users of ESSMA will get notifications for every event is crucial to known. ESSMA support push notifications. An important advantage of push notifications in mobile computing is that the technology does not require specific applications on a mobile device to be open in order for a message to be received. This allows a smartphone to

receive and display social media or text message alerts even when the device's screen is locked and the social media application that is pushing the notification is closed. These push notification will inform user about evacuation path that is sent to them and need to be followed as well as important alert that the operator consider necessary to be known by the users

2.4.5 Route Guidance

The most important feature of the ESSMA application is the route guidance for both helpers and non-helpers. As we presented in section 2.2, there are two kind of ESSMA users, helpers and non-helpers. For the helpers we have two subcategories professional and voluntary helper, however they are treated by the system with the same way, the only difference is that a voluntary helper have the choice to opt out a rescue plan. More specifically, during an emergency the system understand which users are in the affected area and provide to them personalized evacuation paths that they need to follow in order to be safe. For those who cannot escape, system provide route guidance to nearby helpers (voluntary or not) in order to assist the trapped citizen. The evacuation plan is produced by the evacuation decision support system (eDSS), this plan is sent to the back end of ESSMA and then it is distributed to each ESSMA user separately. At this point, we must not fail to mention that ESSMA is not able to guarantee the fact that users will follow the proposed paths, as a result the guarantee their security. However, in our use cases that will be described in the section 5, we assume that all the ESSMA users follow the instructions of the system.

2.4.6 Emergency Signal Propagation

One of the most important goals of RESOLUTE project is the accurate information of the whole system about emergencies, in order to maintain the resilience of a community. The contribution of ESSMA in this goal if the signal propagation feature that is done the SOS call. The UI of this feature in depicted in section 3.4 and it is a button that user is able to use in order to inform the system about their location and other details about the danger situation that they participate. By doing this, this information is received by the eDSS and therefore CRAMSS is responsible for the reaction of the necessary personnel in order to restore the situation to normal levels.

The SOS call is used by ESSMA users, in order to inform the system that they are injured, they cannot escape, other citizens are injured or other citizens cannot escape together with the location of the event. This information is propagated to all the voluntary helper that are located near to the event. Thereafter they are able to request guidance to those citizens that need assistance using the feature that was described in 2.4.5.

2.4.7 User Communication Toolset

The communication toolset is developed in order to cover the objective 2.3.4. A chat application part of ESSMA was developed in order to give the ability to user to communicate with operator and other local bodies. Additionally to the chat, communication toolset contains the features of phone calls and SMS that use the native applications of each device. The chat tool is not always available to the user, in order to avoid the spamming of messages to the operator and local bodies. When a user has been used either the SOS call or are involved in an emergency then the chat tool is available with predefined contacts. On the other hand, users are always able to communicate with local bodies using the phone call and the SMS tools.

3 IMPLEMENTATION OF ESSMA

3.1 Used Technologies

The ESSMA is a mobile application that follows the server-client architectural paradigm and therefore utilizes web development frameworks that enable this approach. The server-side of the platform is built with the Liferay v6.2 content management framework, which allows data storage, application logic implementation and web service deployment. The client-side of the platform, which provides a Graphical User Interface to the end-user, is developed with the Ionic framework, an open source framework used for developing mobile applications. Ionic provides tools and services for building mobile UI with native look and feel. Ionic framework needs native wrapper to be able to run on mobile devices. Since Ionic is built on top of AngularJS and Apache Cordova, we used previous experience of CERTH in these technologies.

3.1.1 Liferay

Liferay Portal is a free and open source, enterprise software product. [4] The distribution that we use runs under the GNU Lesser General Public License. It is primarily used to power corporate intranets and extranets. The Liferay content management framework offers developers the ability to develop applications, rapidly, using popular technologies such as the Java programming language, the MySQL database server and RESTful web services via the Tomcat application server. Java is a high-level, object-oriented; cross-platform programming language that features a large standard library and the ability to build web applications that can be deployed by Java application servers. Developers use Java to build portlets, which can be thought of as Java web applications, containing application logic and access to database data that is exposed via RESTful web services when the portlet is deployed on Tomcat. While Liferay may also be used to develop the front-end of a web application with the use of Java server pages, in the case of the Social Interaction Platform the portlet only contained web services that enabled access to the back-end. Liferay provides a mechanism where the developer describes the application entities in XML format and subsequently the corresponding tables are auto-generated in the database and Java classes that provide basic CRUD (Create, Read, Update, and Delete) operations on these tables are auto-generated. By utilizing the aforementioned classes, the developer is able to write the application logic in the web services without having to deal with the database directly. This mechanism also allows Liferay to employ a cache of database data that significantly increases the access speed to the data, especially in heavy usage scenarios. Liferay also features premade entities, such as users, user groups, categories, social relationships, calendar events, entities for messaging and forum implementation, etc., which are common in web applications and thus expedites the development effort.

3.1.2 Ionic

Ionic is a complete open-source SDK for hybrid mobile app development. Built on top of AngularJS [2] and Apache Cordova [3], Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass. Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by leveraging Cordova. Ionic provides all the functionality that can be found in native mobile development SDKs by using the Cordova plugins (3.1.3). Users can build their apps, customize them for Android or iOS, and deploy through Cordova. Ionic includes mobile components, typography, interactive paradigms, and an extensible base theme. Using Angular, Ionic provides custom components and methods for interacting with them. One such component, collection repeat, allows users to scroll through a list of thousands of items without any performance hits. Another component, scroll-view, creates a scrollable container with which users can interact using a native-influenced delegate system. [1]

3.1.3 Cordova Plugins

The ESSMA app is a cross-platform application, where both Android and iOS are supported. We have used the Ionic which is a framework built on top of AngularJS and Cordova. Using AngularJS MVC architecture, we were able to develop a single page application optimized for mobile devices. Together with the CSS components, all elements that a mobile application needs were offered. Additionally, using JavaScript we were able to extend these components enhancing the functionalities of the app. Moreover, a number of Cordova plugins were used in order to use the native device functions with JavaScript code. The plugins are illustrated in Table 1.

Table 1. List of Cordova Plugins

Name	Version	Description
cordova-plugin-device	~1.1.3	This plugin defines a global device object, which describes the device's hardware and software. Although the object is in the global scope, it is not available until after the device ready event. This plugin is necessary in order to known when we can use the other Cordova plugins. [5]
cordova-plugin-console	~1.0.4	This plugin is meant to ensure that *console log* is as useful as it can be. We used this plugin for debug purpose during the developing stage. [6]
cordova-plugin-whitelist	~1.3.0	Domain whitelisting is a security model that controls access to external domains over which your application has no control. Cordova provides a configurable security policy to define which external sites may be accessed. This plugin gives the opportunity to define the cross origin policy of the application. [7]
cordova-plugin-splashscreen	~4.0.0	This plugin is required to work with splash screens. This plugin displays and hides a splash screen during application launch. [8]
cordova-plugin-statusbar	~2.2.0	Using this plugin, Cordova provides an object, the StatusBar object, which contains some functions to customize the iOS and Android StatusBar. It is used for styling purpose. [9]
ionic-plugin-keyboard	~2.2.1	Using this plugin, Cordova provides an object, the Keyboard object, which contains some functions make interacting with the keyboard easier, and fires events to indicate that the keyboard will hide/show. [10]
cordova-plugin-camera	~2.3.0	Using this plugin, Cordova provides an object, the Keyboard object, which contains an API for taking pictures and for choosing images from the system's image library. This plugin is used in order to capture photos that will be sent to operators through chat. [11]

cordova-plugin-file-transfer	~1.6.1	This plugin allows you to upload and download files. It is used with the coordination of the previous plugin in order to send photos to operators through chat. [12]
cordova-plugin-geolocation	~2.4.1	This plugin provides information about the device's location, such as latitude and longitude. This information is necessary for the evacuation and the rescue path that will be provided by the EDSS that is developed in the Task 5.2. [13]
cordova-sms-plugin	~0.1.11	This plugins is used by ESSMA in order to send pre-defined SMS using the native device application for SMS. [14]
phonegap-plugin-push	~1.9.2	This plugin offers support to receive and handle native push notifications with a single unified API. We use the Ionic push notification system in order to send to the ESSMA users when either a new task is available and needs to be done or to send a number of alerts by the system to the users. [15]

3.2 ESSMA Architecture

ESSMA is one of the apps that is contained in the actuation channels module of the RESOLUTE architecture. A web application; with its own backend and front-end communicates directly with the Evacuation DSS.

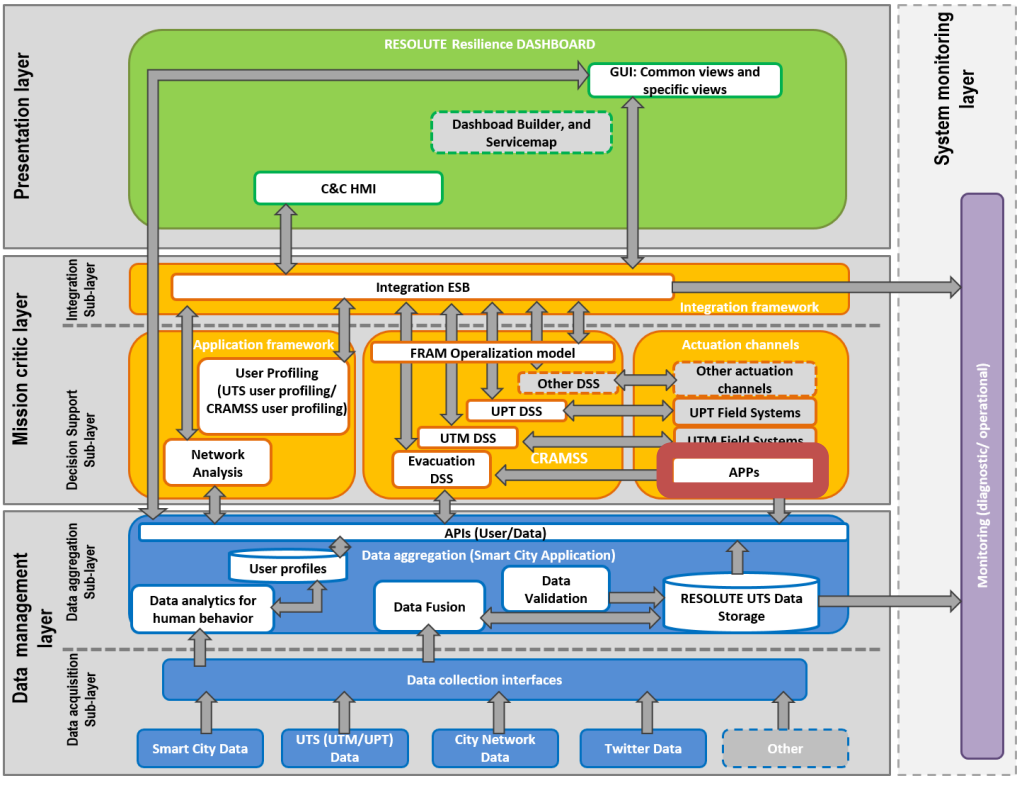

Figure 2. RESOLUTE Architecture

The ESSMA architecture follows the client-server model, where the front-end part (client-side) contains the User Interface and the back-end part (server-side) is responsible for the business logic, data storage and data manipulation. The general Architecture of the back end is shown in Figure 3.

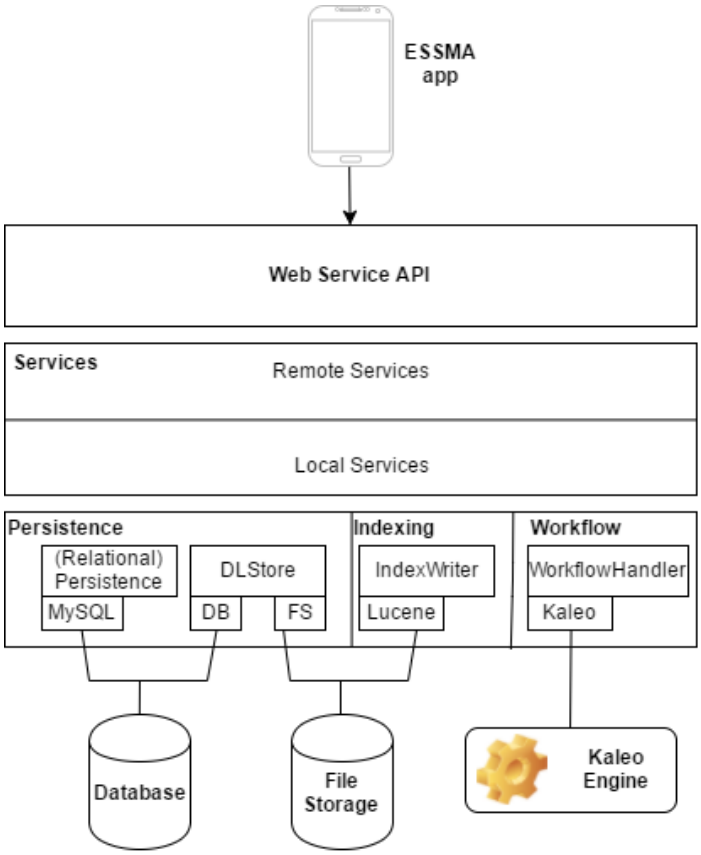

Figure 3. Back-End Architecture

The front end consumes web services, exposed by the back end, which allow it to retrieve the data that are presented to the user and to store the data that the user wants to upload to the platform. The back end consists of three layers. The top layer is the Remote API layer where all the web services are defined by declaring name, arguments, return type, authorization policy and the specific function call that implements the business logic. Below the Remote API layer, is the Business layer that contains all the code written by the developers consisting of the functions that implement the web services as well as any other helper functions and utility classes. Below the Business layer, the Entities layer is found. The Entities layer contains an XML description of all the developer-defined entities and the auto-generated code that the Liferay platform produces based on the aforementioned XML description. The developer may define in XML an entity such as (e.g.) the "User" entity that describes a user of the Social Interaction Platform. Afterwards the Liferay platform can be used to generate a table for the "User" entity in the system's database along with Java classes that contain pre-made functions that handle CRUD operations for the "User". This way the developer does not have to deal directly with the system's database, but he can call the aforementioned functions from the Business layer in order to manipulate the various entities in the code. The Entities layer handles most of the communication with the system's database and the file system where multimedia files are stored. This information is processed and forwarded to the client so it can be displayed to the end user. Liferay Portal is Java-based and runs on any computing platform capable of running the Java Runtime Environment and an application server. Liferay is available bundled with a servlet container such as Apache Tomcat. Additionally, for the storing of the data our back end contains a MySQL database and a File Storage System.

3.2.1 RESTful web services

For the communication of the application with the back end a number of RESTful we services are developed. RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, which enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture. It is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

3.2.1.1 Login and Register Services

In order to use the ESSMA app it is necessary to have an account. The following services are used for the creation and the authentication of the accounts. The first time that a user install the app in their mobile, they need to create a new account; this process is being done using the service with title *Add User* in Table 2. This service send to the back end of the application a model structure with the input values that the users has filled in the registration form.

Table 2. Add User Service

Title:	**Add User**
URL:	/cramss-portlet.cramss/add-user
Method:	POST
URL Params:	---
Data Params:	{userJsonStr: [string] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

On the other hand if a user has already an account they need to login first to the application in order to use it. The login of a user is done using the service with the title *Authenticate User* in Table 3. As input in this service, user need to provide their email and password. This data is sent to the back end of the application that is responsible for the authentication of the users.

Table 3. Authenticate User Service

Title:	**Authenticate User**
URL:	/cramss-portlet.cramss/authenticate-user
Method:	POST
URL Params:	---
Data Params:	{email: [string], password: [string]}
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

3.2.1.2 News Feed Services

One of the most significant parts of the application that is responsible for the accurate information of citizens is the News Feed view of the application. This feature contains all the posts that the operator has upload for the updated condition of an emergency or in general, post that are necessary to distributed to the users. In order to get all these posts a service was developed with the title *Get All Activities* in Table 4, which communicates with the back end of

the application and retrieves the posts of the operator. This service is inextricably linked with the eDSS because these posts are generated by the operator that uses this module of CRAMSS and their main purpose is to spread accurate information to the final users of ESSMA.

Table 4. Get All Activities Service

Title:	**Get All Activities**
URL:	/cramss-portlet.cramss/get-all-activities
Method:	GET
URL Params:	---
Data Params:	{From: [Long], To: [Long], lastIndex: [int] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Furthermore, the citizens that are the actual users of ESSMA are able to interact with the operator's posts writing comments on that. In order to do that ESSMA uses another service with title *Add Activity Comment* in Table 5, that needs as input the ID of the users, the ID of post, the entry ID and finally the a text that is the actual comment of the user.

Table 5. Add Activity Comment Service

Title:	**Add Activity Comment**
URL:	/cramss-portlet.cramss/add-post-comment
Method:	POST
URL Params:	---
Data Params:	{userId: [Long], entryId: [Long], postId: [Long], Comment: [String], type: [String] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

In addition, users are able to edit a comment that is made by them using the service *Edit Activity Comment* or remove it by the service *Remove Activity Comment*, which are described in the Table 6 and Table 7, respectively.

Table 6. Edit Activity Comment Service

Title:	**Edit Activity Comment**
URL:	/cramss-portlet.cramss/modify-post-comment
Method:	POST
URL Params:	---
Data Params:	{commentId: [Long], newComment: [String] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Table 7. Remove Activity Comment Service

Title:	**Remove Activity Comment**
URL:	/cramss-portlet.cramss/remove-post-comment
Method:	POST

URL Params:	---
Data Params:	{commentId: [Long] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Moreover, Users of ESSMA are able to interact to a post or a comment in the News Feed by liking this. This action can be done using the *Add Activity Like* service and *Add Activity Comment Like* service. These two services are described in Table 8 and Table 10, respectively. Additionally, Users can remove their likes using the *Remove Activity Like* service and *Remove Activity Comment Like* service that is shown in the Table 9 and Table 11, respectively.

Table 8. Add Activity Like Service

Title:	**Add Activity Like**
URL:	/cramss-portlet.cramss/add-post-like
Method:	POST
URL Params:	---
Data Params:	{userId: [Long], entryId: [Long], postId: [Long], type: [String] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Table 9. Remove Activity Like Service

Title:	**Remove Activity Like**
URL:	/cramss-portlet.cramss/remove-post-like
Method:	POST
URL Params:	---
Data Params:	{userId: [Long], entryId: [Long] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Table 10. Add Activity Comment Like Service

Title:	**Add Activity Comment Like**
URL:	/cramss-portlet.cramss/add-post-comment-like
Method:	POST
URL Params:	---
Data Params:	{userId: [Long], entryId: [Long], commentId: [Long] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Table 11. Remove Activity Comment Like Service

Title:	**Remove Activity Comment Like**
URL:	/cramss-portlet.cramss/remove-post-comment-like
Method:	POST

URL Params:	---
Data Params:	{userId: [Long], entryId: [Long] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

3.2.1.3 SOS-call services

Users that are in danger are able to inform the system for their situation. This action can be done using the service with title *Send SOS Alert* in Table 12. Using this service users are sent the most necessary information to the system and therefore to the operator. This information contains their ID, their current coordinates and details about the emergency. This service call is done directly to eDSS that triggers a number of events that are needed in order to spread this information to all the components of RESOLUTE.

Table 12. Send SOS Alert Service

Title:	**Send SOS Alert**
URL:	/cramss-portlet.cramss/add-person-in-danger
Method:	POST
URL Params:	---
Data Params:	{userId: [Long], longitude: [String], latitude: [String], detail: [String }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

3.2.1.4 User Profile Services

One of the main tabs in the application is the account of user. In this tab, users are able to watch and edit their personal information. In order to get all the information of the user to the application a service, with title *Get User* in Table 13, was developed that is called with identifier the ID of the user. This service is calls initially, each time that the application starts

Table 13. Get User Service

Title:	**Get User**
URL:	/cramss-portlet.cramss/get-user
Method:	GET
URL Params:	Required: user-id: [Long]
Data Params:	---
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Additionally, users are able to edit their personal information by calling the service *Edit User* in Table 14. Using this service, users can change a number of personal information such as their full name, phone number and email. Moreover, through this service users are able to declare the willing to be voluntary helpers or their special requirements.

Table 14. Edit User Service

Title:	**Edit User**
URL:	/cramss-portlet.cramss/edit-user

Method:	POST
URL Params:	---
Data Params:	{userId: [Long], userJsonStr: [String] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Furthermore, in the Profile tab, users are able to add or change their profile portrait using the service with tittle *Change User Portrait* in Table 15. This service requires access to the gallery of the users' phone and uploads a picture to the back end of the application.

Table 15. Change User Portrait Service

Title:	**Change User Portrait**
URL:	/cramss-portlet.cramss/change-user-portrait
Method:	POST
URL Params:	Required: user-id: [Long], filename: [String]
Data Params:	{File: [File] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Finally, in order to get the current position of the users we created the service with title *Update Current Position* in Table 16. This service needs as input the id of the user and their type, as well as, the longitude and latitude at the time that the service is called. This service is called by a WebSocket message that is sent every one-minute using CRON scheduler.

Table 16. Update Current Position Service

Title:	**Update Current Position**
URL:	/cramss-portlet.cramss/update-last-location
Method:	POST
URL Params:	---
Data Params:	{userId: [Long], userType: [String], longitude: [String], latitude: [String]}
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

3.2.1.5 Communication Services

The chat functionalities of the ESSMA application are based on a number of web services in order to achieve the communication between users that are located in different locations. The first service that is called when the user opens the chat view as it is shown in Figure 8 has as title *Get User Threads* in

Table 17. Using this service the application retrieves all the chat threads that the users has participated from the first time that they started using the application. In order to call the specific service, the application uses only the user-id of the user; the response of the server is a JSON array that contains all the conversations of the user; However this service does not retrieve the whole content of all threads but only a description of them in order to keep the application fast.

Table 17. Get User Threads Service

Title:	Get User Threads
URL:	/cramss-portlet.cramss/get-user-threads
Method:	GET
URL Params:	Required: user-id: [Long]
Data Params:	---
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

Additionally, Users are able to search among their chat threads using the service with title *Search User Threads* in Table 18. This search is being done by typing the query in an input field, this query is used as identifier among the user's threads. This service is called automatically when the user start typing in the input field.

Table 18. Search User Threads Service

Title:	Search User Threads
URL:	/cramss-portlet.cramss/entity.chatthread
Method:	GET
URL Params:	Required: user-id: [Long], query: [String], from: [int], to: [int]
Data Params:	
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

As we have already mentioned in 2.4.7 user is able to use the chat tool only in two cases. Either when they are under emergency or when they are in danger. For this reason, the application uses the service with title *Hide User Threads* in Table 19 where the chat threads is being hide when the user is safe.

Table 19. Hide User Threads Service

Title:	Hide User Threads
URL:	/cramss-portlet.cramss/entity.chatthread
Method:	GET
URL Params:	Required: user-id: [Long], hide: [String]
Data Params:	{File: [File] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

The following service with title *Set Thread Read* in Table 20 is called every time that the user opens a specific thread that has unread messages. This is used because the application creates a notification every time that a new message received. Using this service, we make the system aware that the user has read a message. Nevertheless, this service is not called if the user has the specific thread open because we assume that the user can see the new message without any notification.

Table 20. Set Thread Read Service

Title:	Set Thread Read

URL:	/cramss-portlet.cramss/set-thread-read
Method:	POST
URL Params:	Required: user-id: [Long], thread-id: [Long], thread: [boolean]
Data Params:	---
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

The following thread with title *Get Unread Tread Count* in Table 21 is used in order to get the number of unread threads that will be shown the notification that we mentioned previously.

Table 21. Get Unread Thread Count Service

Title:	**Get Unread Thread Count**
URL:	/cramss-portlet.cramss/get-received-unread-thread
Method:	GET
URL Params:	Required: user-id: [Long]
Data Params:	---
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

In order to open a chat thread and read all the messages, ESSMA uses the service with title *Get User Thread* in Table 22. The call of this service has as input parameters the user-id as well as the thread-id. The response of the service contains a JSON array with all the messages in this thread between the user and the other participant of the thread.

Table 22. Get User Thread Service

Title:	**Get User Thread**
URL:	/cramss-portlet.cramss/get-user-thread
Method:	GET
URL Params:	Required: user-id: [Long], thread-id: [Long]
Data Params:	---
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

The following service with title *Send Message* in Table 23 is used in order to send a new message. As input of this service, we use the user-id of the sender, the user-id of the receiver. Additionally, the subject of the message that in the most cases is empty, the main content of the message and finally the thread id that this message is part.

Table 23. Send Message Service

Title:	**Send Message**
URL:	/cramss-portlet.cramss/send-message
Method:	POST
URL Params:	---
Data Params:	{fromUserId: [Long], toUserIds: [Long], subject: [String], body:[String], threadId: [Long]}

Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

In order to send a message with an attachment like a photo, the ESSMA application uses the following service with title *Send Message With Attachment* in Table 24. This service is almost the same with the above service in Table 23 however, an extra field is necessary in order to call it, which is the filename. Additionally, this service contains as Data parameter a File object that is uploaded to the server using a Cordova plugin, the *cordova-plugin-file-transfer*, which will be described in section 3.1.3.Cordova Plugins

Table 24. Send Message With Attachment Service

Title:	**Send Message With Attachment**
URL:	/cramss-portlet.cramss/send-message
Method:	POST
URL Params:	Required: from-user-id: [Long], to-user-id: [Long], subject: [String], body: [int], thread-id: [Long], filename: [String]
Data Params:	{file: [File]}
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

3.2.1.6 User Profiling Services

As we mentioned in Section 2.4.2 the ESSMA collects the paths that the user has followed, in order to save these paths the service with title “Add User Path” in Table 25. As input of this service, we use the user-id of the user that followed this path and the path itself.

Table 25. Add User Path Service

Title:	**Add User Path**
URL:	/cramss-portlet.cramss/add-user-path
Method:	POST
URL Params:	---
Data Params:	{user-id: [Long]], userPath: [String] }
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

The eDSS uses the following service with title “Get All Users Paths” in order to acquire this historical data. The response of this service contains all the paths of all users and this data acts as input to the user-profiling module that will cluster the users based of the paths that they use to follow.

Table 26. Get All Users Paths Service

Title:	**Get All Users Paths**
URL:	/cramss-portlet.cramss/get-all-users-paths
Method:	GET
URL Params:	---

Data Params:	---
Response Code:	Success (200 OK), Bad Request (400), Unauthorized (401)

3.2.2 Database and File Storage

The data of ESSMA application is stored in a SQL database that is generated by the Liferay portlet. All the tables and attributes of the database are declared in a XML file with their specifications, thereafter, by building the Liferay application, automatically, the tables are generated. The whole database is created by using custom and auto generated entities of Liferay. Additionally, Liferay gives the ability to developers to create their custom finder inside a table. For instance, in the aforementioned entity we have declared a new finder column, the *userId*. By doing this, Liferay generates all the necessary methods in order to retrieve the user's coordinates by the column *userId.*

In the Table 27. Example of Database Entity, we can illustrate an example of the XML file that creates the table *'UserCoordinates'* from the respective entity.

Table 27. Example of Database Entity

```
<entity name='UserCoordinates' local-service='true' table='UserCoordinates'>
        <column name='id' type="long" primary="true" id type="increment"></column>
        <column name='userId' type='long'></column>
        <column name='userType' type='String'></column>
        <column name='longitude' type='String'></column>
        <column name='latitude' type='String'></column>
        <column name="timestamp" type="long"></column>
        <finder name="UserFinder" return-type="UserCoordinates">
                <finder-column name="userId"></finder-column>
        </finder>
</entity>
```

Additionally, Liferay contains its own file system that is used in order to save the media content that is uploaded to the system. Liferay's Documents and Media library provides a mechanism for storing files online using the same type of structure that you use to store files locally.

3.2.3 WebSockets

The ESSMA uses WebSockets technology in order to send messages and push notification to users even if they do not have open the application. WebSockets is an advanced technology that makes it possible to open an interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply. [16]

The communication of ESSMA with the server is based on WebSockets in the case of the Map tab that was described in 4.7. More specifically, all the ESSMA users get a message with WebSockets when the operator has pointed an area that is in danger. With this way, the map of the application always show the areas that need to be avoided. Additionally, WebSocket messages, that contains the personalized evacuation path, are received by ESSMA when the operator has triggered an evacuation plan in the area that the user of ESSMA is located. Furthermore, for those that are helpers, the system sends WebSocket messages with information about citizens that need help. This information is shown in the map of helpers.

4 USER INTERFACE OF ESSMA

4.1 Flow of Application

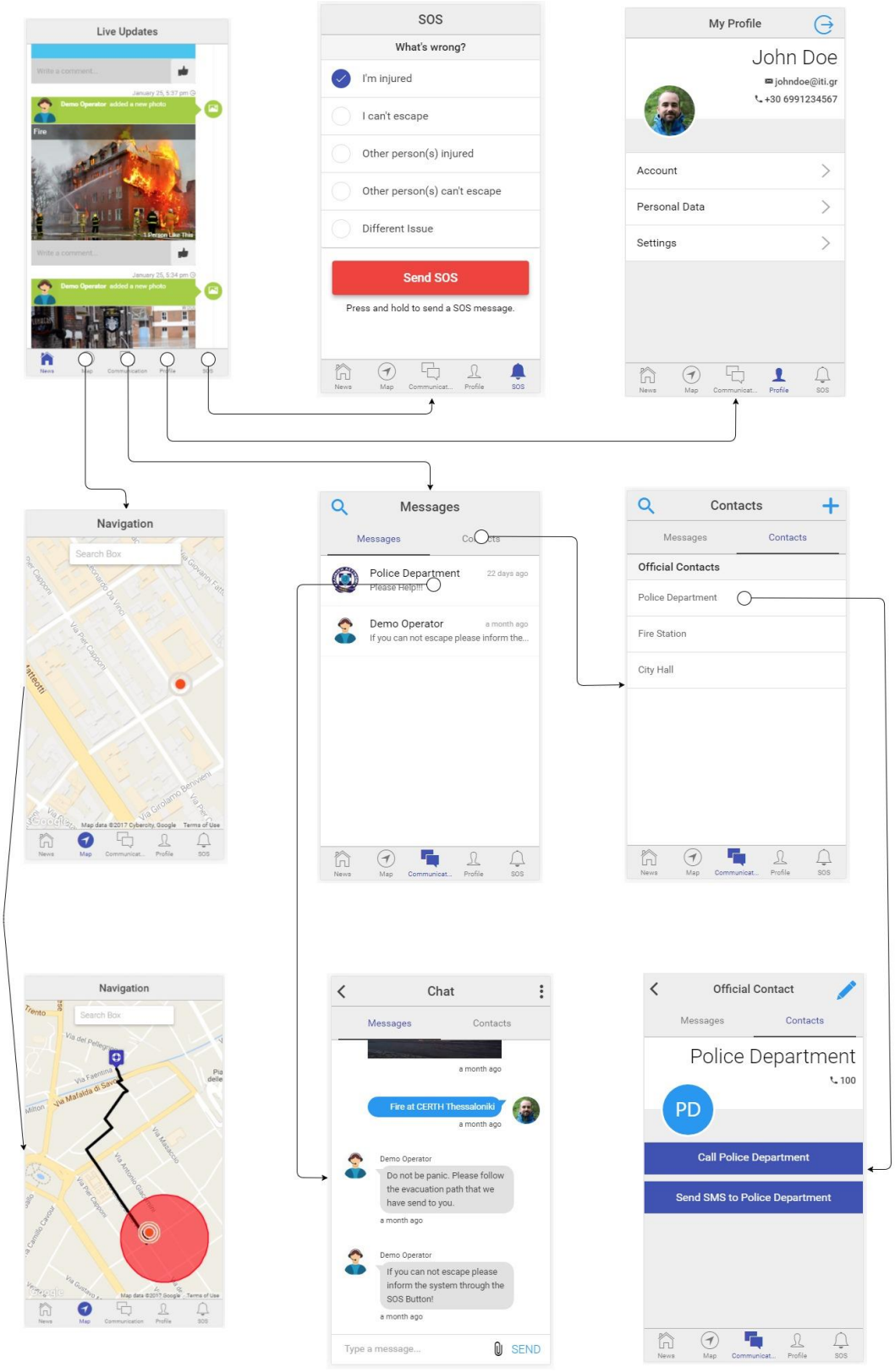

Figure 4. Flow of the application

4.2 News

In this tab, users are able to monitor any post of the operator about events that needs to be aware. More specifically, posts that shown warnings or the condition of an emergency will be depicted as in Figure 5. In this tab, the user can scroll down or up to see, more posts. Additionally, they are able to interact with each post adding a comment or liking the post. The specific tab is updated either automatically or by user when they scroll down. The most significant fact of this tab is that every time that the user opens it, the application always shows the latest update of an emergency status. With this way, users are always aware and up to date about an emergency. The content of this tab contains pictures, videos or plain text that the operator uploads. This part of the application is the first view, that a user confronts when they opens the application. For this reason, the interaction of the user with the app is very simple just scrolling for accurate information that is provided by the operator.

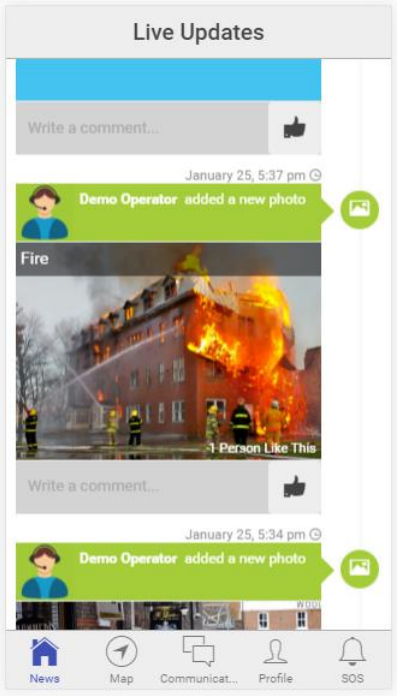

Figure 5. Live Updates Tab

4.3 Communication

The communication tab consists of two parts the Messages (Figure 7) that are the chat functionality of the application and the Contacts (Figure 6), where users are able to communicate with local authorities using the native applications of their smartphone, sending predefined SMS or making calls. The contact tab of the application consists of the local authorities with all the necessary information about them.

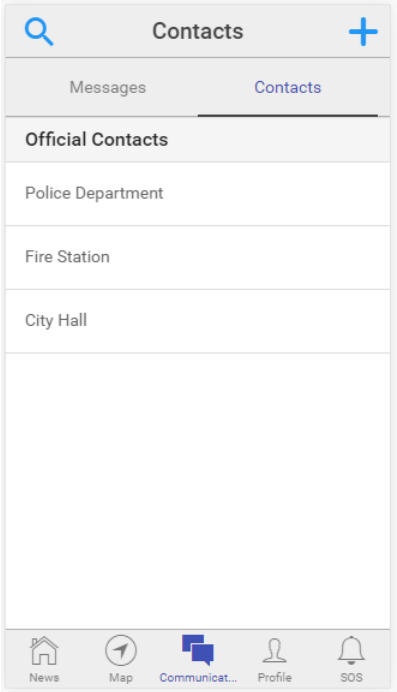

Figure 6. Contacts of Communication Tab

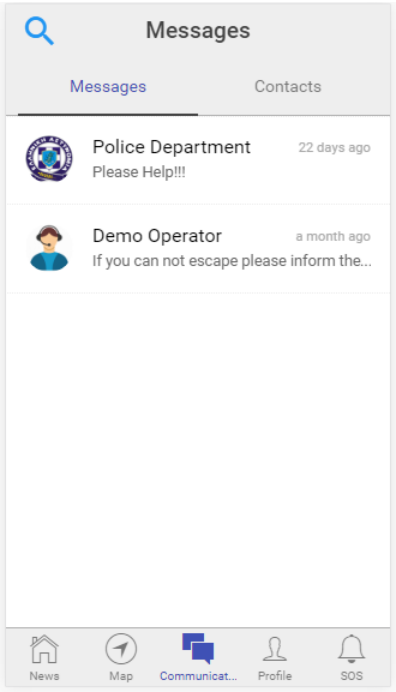

Figure 7. Messages of Communication Tab

4.4 Chat

One of the most significant way to communicate with the operator and other local authorities is the chat functionalities. When a user selects to chat with one of the predefined users as they shown in Figure 7, the view of the Figure 8 is shown. As it is shown, the chat functionalities are very simple like all the other chat applications.

Users type their message in the input field with the placeholder *"Type a message"* and send this message by tapping the button SEND. In addition to a message, users are able to attach content to their messages like photos. With this way user has the opportunity to send to the operator or to the other local authorities content that could be vital for the stockholders in order to understand the critical level of a situation.

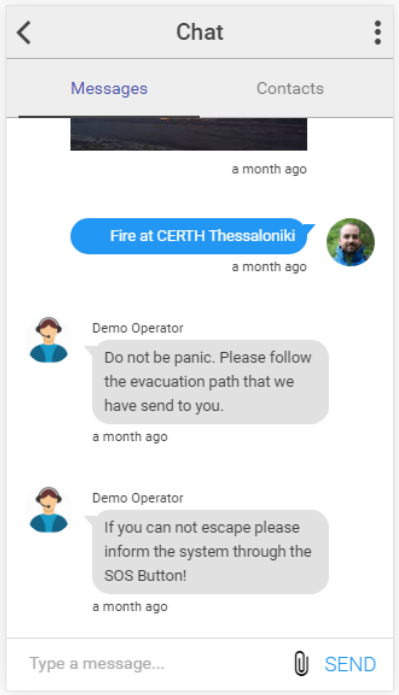

Figure 8. A Chat Tab

Thereafter, operators are able to filter this content in order to upload it to the live updates that is described previously. With this cyclic flow of the information, all the ESSMA users could be aware of an event that was detected by some civilians; however, this event could be dangerous for a number of other civilians.

Additionally, users that are in danger could get live guidance, not only for their evacuation (that will be analyzed in the following subsections), but also on how to react in unknown conditions to them. Operators and personnel that uses the chat, for instance using the eDSS front-end, could reassure citizens who are blocked informing them about the arrival of the rescue team or other advices that would be helpful for the life of the citizen.

4.4.1 Contact

The other way that is available for communication through the ESSMA is the contact tab. The content of the Contacts Tab is shown in Figure 9 and it is consisted by a number of Official Contacts such as, the Police Department, the Fire Station or other local authorities. Despite the fact that all the contacts are predefined users are able to edit these contacts in the case that they want to change or add extra information about them.

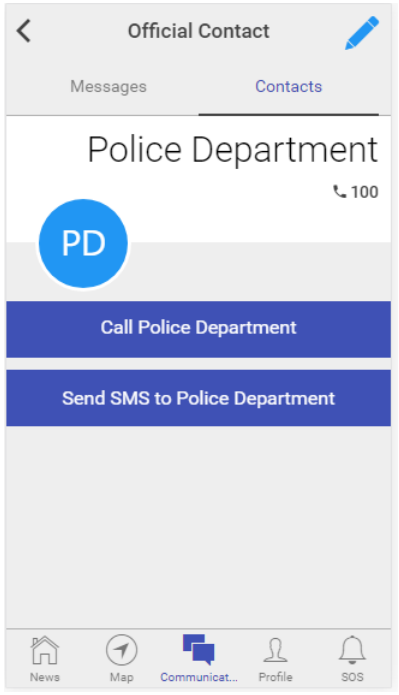

Figure 9. A Contact Tab

Using this kind of communication users are able to select between two different ways to communicate with others. The first one is to make a phone call by pushing the button *"Call Police Department"* as it is shown in Figure 9. Respectively, the same button exists to the other contacts, like the Fire Station or the City Hall.

The other way that the users can select to communicate with others is the SMS functionality. The button *"Send SMS to Police Department"* as it is shown in Figure 9 need to be pushed in order to send a predefined message to local bodies.

In these two cases, the ESSMA application uses the native applications of the device for calls and SMS, opening them and changing the user's view to them.

4.5 Profile

Another main tab of the ESSMA application is the *"My Profile"* tab where users are able to control a number of different features of the application as well as to watch their personal details. An overview of this tab is shown in the Figure 10.

The initial view of this tab consists of three main option the *"Account"*, the *"Personal Data"* and the *"Setting"*. Additionally, in the main view of this tab users are able to change their personal portrait and monitor their basic personal contact information such as their email and their phone number.

Moreover, from this tab, users are able to log out from the application. When a user logged out from the application, every information about them is lost and the application is initialized again when they logged in.

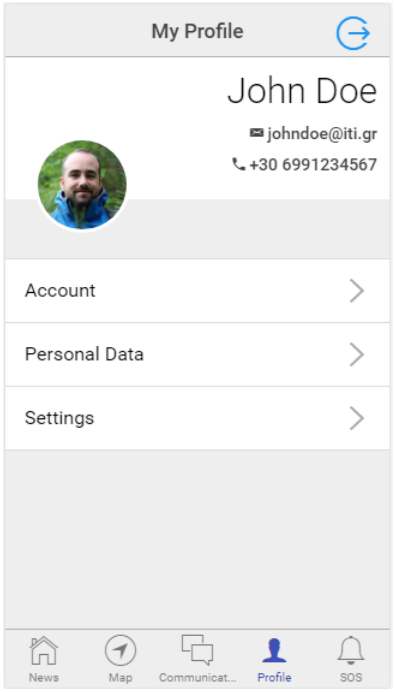

Figure 10. The My Profile Tab

4.5.1 Account

When a user tap on the *"Account"* layer of the Figure 10, the view of the account is opened as it is shown in Figure 11 There, users are able to view and edit there information such as their, full name, phone number, and email. In order to do this, the upper right button needs to be pressed and a new pop-up window opens that allows to the user these modifications (Figure 12).

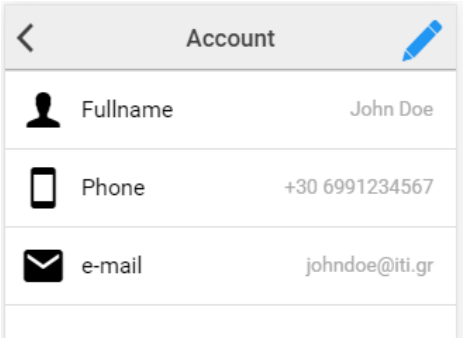

Figure 11. Account Tab

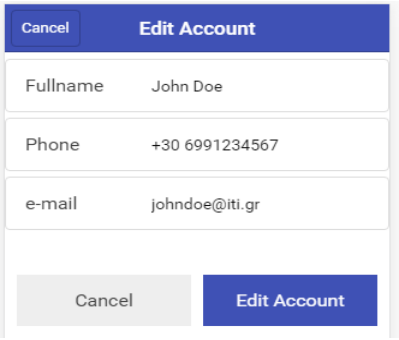

Figure 12. Edit Account

4.5.2 Personal Data

In the Personal Data view, users are able to control the personal preferences about different features of the application. From this view the can view the voluntary Helper option, to declare their special Requirement and to monitor the Demographic data. The interaction between user and application in this view (Figure 13) is the same with the one that is shown in Figure 10, by tapping on the layout that they want to open. In the following three subsection, we will see in details the three features of Personal Data View.

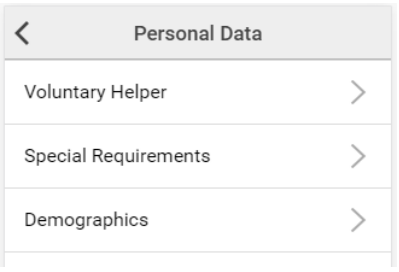

Figure 13. Personal Data Tab

4.5.2.1 Voluntary Helpers

By tapping in one of these two options, users are able to select their role. Users that has the physical capability and the willing to help other citizens during an emergency can select "*YES"* as it is shown in Figure 14By selecting to be voluntary helper, a user will have extra features in the Map Tab that will be described in a following section.

At this point, we must not fail to mention, that the option is not an obligatory one and citizens that have selected, need first to care about their own safety.

Every time the user decides to change their choice, they can do it simply by tapping *"NO"* in this view. Thereafter the system is updated and the users will be treated by the system as a simple citizen and the points that need help will not be shown in their map.

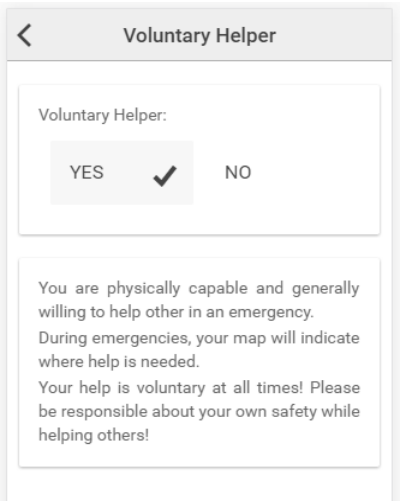

Figure 14. Voluntary Helpers Tab

4.5.2.2 Special Requirements

In the special requirements view (Figure 15) users will special abilies are able to declare these disabilities. At the current status the application and general the whole system together with the eDSS status, three different kind of special requirements are supported. For wheelchairs, for general mobility problems and for vision problems.

Initially, these three requirement are disabled and the users themselve need to declare their disabilities. This information is very crusial because the accurate profiling of the users, results to more accurete and personalized guidance during the emergenies.

This information is dissiminated to the eDSS that is responseble for the provision of the personalized guidance and evacuation paths to the ESSMA users.

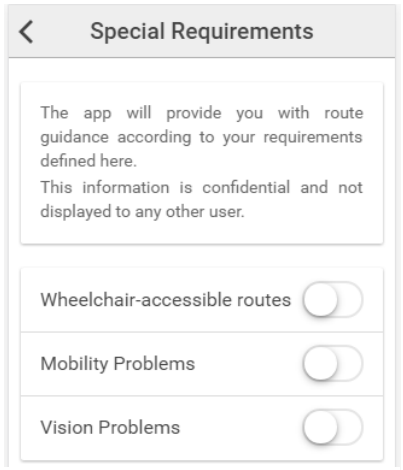

Figure 15. Special Requirements Tab

4.5.2.3 Demographics

The demographics view (Figure 16) shows the demographic data as it is provided by the user during the registration. It is important information for the profiling of user.

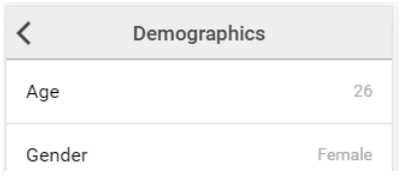

Figure 16. Demographics Tab

4.5.3 Settings

In the Settings View, users are able to control some features of the application. These features could be the *"Notifications"* of the application (Figure 18), where users can control the push notification and other notifications. Moreover, users can select the *"Language"* of the application as it is shown in Figure 19. Additionally, in this section users can see general information about the app (Figure 20).

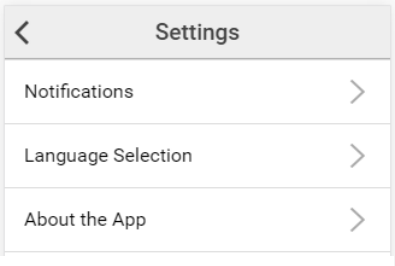

Figure 17. Settings Tab

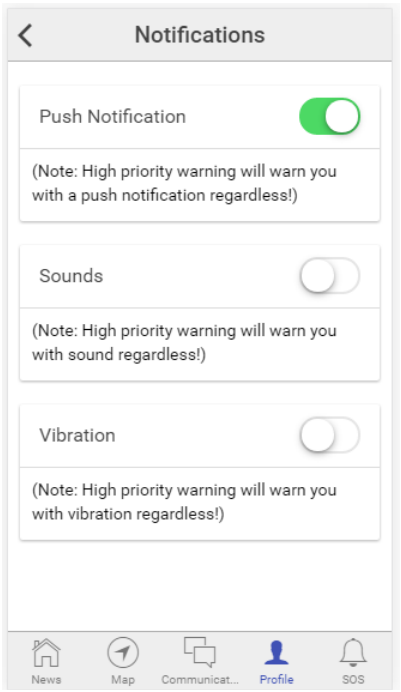

Figure 20. Notifications Settings

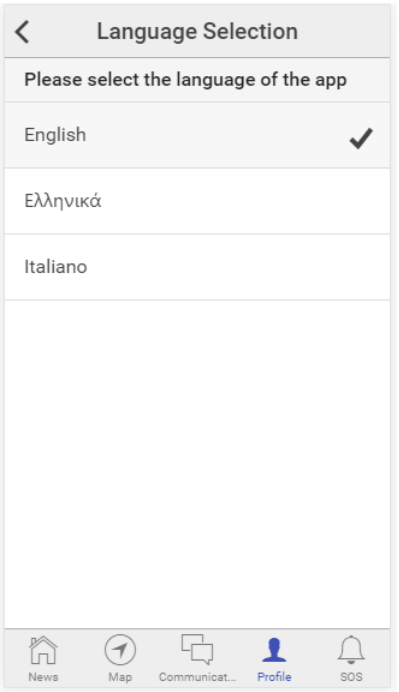

Figure 19. Language Settings

Figure 18. About The App

4.6 SOS

This view of the Application contains the SOS button that triggers the emergency call. As it is shown in the Figure 21. SOS-call Tab, there is a red button with the title *"Send SOS"*. When the user tap and hold this button for 500ms, an emergency signal is sent to the system that contains the current location of the user in terms of latitude and longitude, together with details about what is wrong. The user has the ability to select one or more of choices that are predefined that will be the details of the emergency signal. When the signal has been propagated success fully a pop-up window is shown that suggests the user to make a phone call to the Police and by tapping *OK* the view of the application goes to the tab of Figure 9.

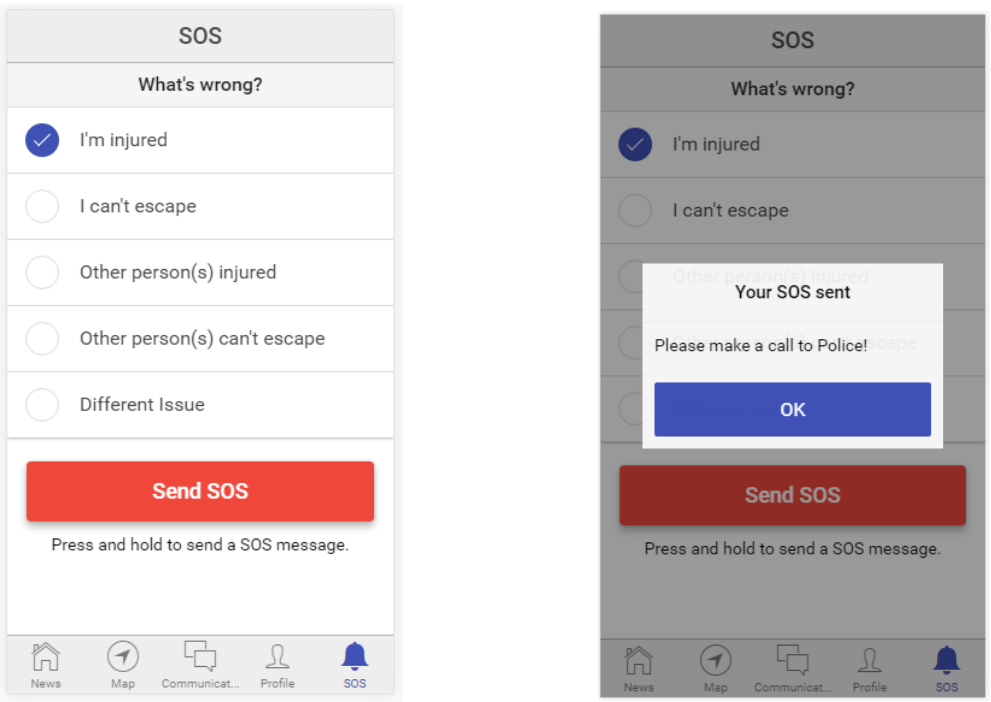

Figure 21. SOS-call Tab

4.7 Map

This view is one of the most significant parts of the application that provides useful information about the location of danger events. As you can see in Figure 22, initially the map only shows the current location of the ESSMA user. Additionally, ESSMA map gives the ability to users to search places in the search box like hospitals or police station etc. As it is shown in Figure 23, when a user is located inside an area that needs to be evacuated, the ESSMA shows to the UI of the application the proposed route to a Safe Point. This information about the evacuation path is provided by eDSS through WebSockets and it is shown to the map automatically when the operator starts an evacuation plan.

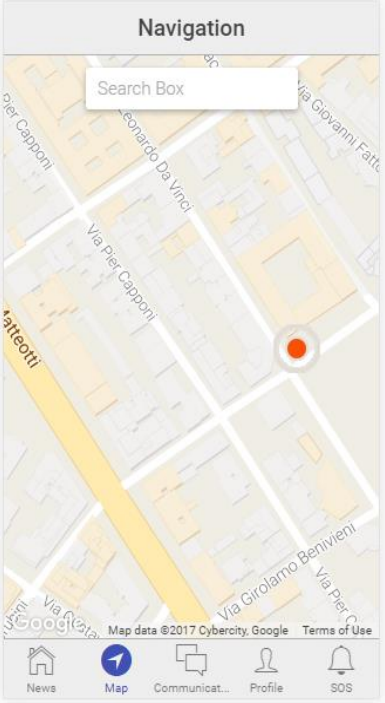

Figure 22. Map Tab

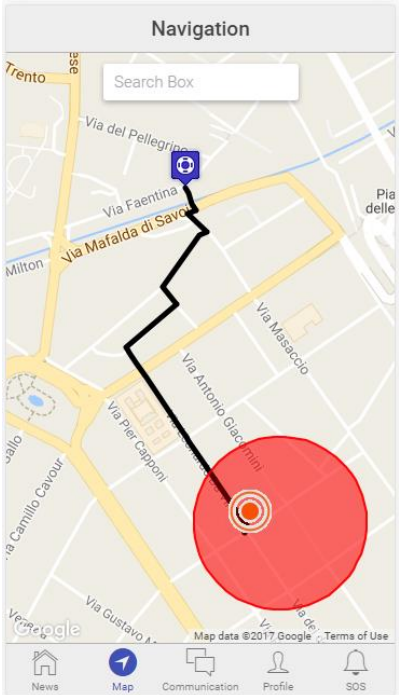

Figure 23. Map Tab during Evacuation

In general, the map tab of ESSMA application is responsible to show to the end user the paths that they need to follow as well as other information such as the areas in danger, the safe points, and the citizens or places that need assistance. The last feature of Map tab (citizens or places that need assistance) is only available to those users that are helpers either professional or volunteer. A more detailed example of the specific tab will be illustrated in the section 5 where the basic use cases will be described.

5 SAMPLE OF USE CASES

5.1 User Evacuation Use Case

The first scenario that will be presented is the most common use of ESSMA application, which is an evacuation scenario of a non-helper user. As it is shown in Figure 24, the evacuation of ESSMA users are triggered by the eDSS operator, who has set the safe point and the eDSS has generate all the paths that need to be followed by the ESSMA users. When the evacuation plan begins, the whole plan is uploaded to the ESSMA back-end, the server that is responsible for the correct distribution of the personalized paths to the ESSMA users. Following that, the server sends push notification to the ESSMA users that an evacuation plan is ongoing state in their location and thereafter each path is sent to the respective users using the WebSocket technology of the back-end.

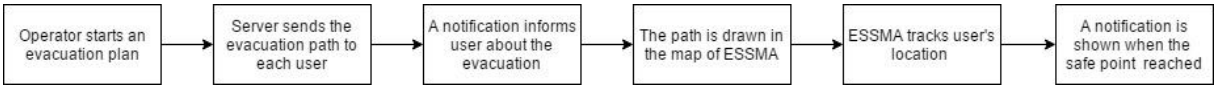

Figure 24. User Evacuation events flow

When the mobile application receives a new socket message with the identification of *evacuation path,* in the map of the application is drawn the path that the user needs to follow. The evacuation message that is received by the app contains a list of coordinate that describes the generated evacuation path as well as the safe point location that is the final destination of the rout. After that, user can depict in the Map tab their location, the evacuation path and the safe destination; we assume that the user follows the provided path in order to keep themselves safe. During this, the location of the user is updated in order to get a better feeling of their position compared with the destination. Finally, when the ESSMA user arrived to a safe point, a notification is shown by the application that informs the user for their safety.

5.2 User Rescue Use Case

The second scenario that will be described is a rescue of a citizen by a voluntary helper, in this scenario, we assume that the user cannot escape and want a rescuer to help them as it is shown in Figure 25. The whole procedure is initialized by the citizen themselves by tapping in the SOS button in the ESSMA application. When the SOS button is pressed a message, that contains the location of the user and other details of the event, is uploaded to the ESSMA back-end. Following that the back end of the application forward this message through WebSockets to the operator front end and to the other ESSMA users that are voluntary helper. From now, the user that needs help waits for the rescuer and they cannot do nothing else except from calling the police or chatting with the operator for updated information.

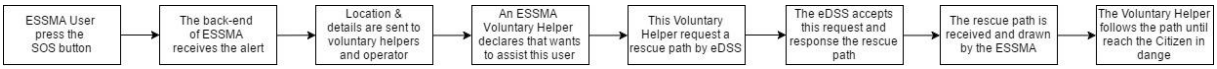

Figure 25. User Rescue events flow

On the other side, a voluntary helper that receives this socket message, is able to illustrate in the ESSMA application map that a citizen near to them needs help and as a Voluntary Helper declares their willing to assist in this situation. By requesting this intention to the operator, the operator decides whether or not accepts this rescue by the specific helper. In our scenario, the request is accepted and a rescue path is sent to the helper. This path is drawn in the map of the application and is up to the helper to follow the proposed path. During the follow of the path, the location of the helper is tracked, by reaching the citizen that needs help; the application provides the ability to helper to get a new path for the nearest safe point.

6 CONCLUSIONS

This deliverable presents the outcome of the task 3 in work package 5 (T5.3) that is the Mobile Emergency Supporting App (ESSMA). The main aim of this task was the development of a cross-platform application that was based in the human computer interaction concepts that was the results of Task 5.1. Additionally, ESSMA cooperates with a component of Task 5.2 the eDSS that provides the evacuation routes to the application.

The ESSMA application is one of the key outcomes of the RESOLUTE project because it serves one of the main purposes of a project on resilience. ESSMA provides information, assistance and guidance to the end users of the application that are actually the citizens of a community. With this way, citizens are always aware of a dangerous situation with accurate and updated information as well as the get assistant and guidance in this situation that is very important improvement for the resilience of a community as well as the whole country.

7 REFERENCES

[1] Ionic Framework: https://ionicframework.com

[2] AngularJS: https://angularjs.org

[3] Apache Cordova: https://cordova.apache.org

[4] Liferay: https://www.liferay.com/

[5] Cordova Plugin Device: https://github.com/apache/cordova-plugin-device

[6] Cordova Plugin Console: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-console/

[7] Cordova Plugin Whitelist: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-whitelist/

[8] Cordova Plugin SplashScreen: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-splashscreen/

[9] Cordova Plugin Statusbar: https://github.com/apache/cordova-plugin-statusbar

[10] Cordova Plugin Keyboard: https://github.com/cjpearson/cordova-plugin-keyboard

[11] Cordova Plugin Camera: https://github.com/apache/cordova-plugin-camera

[12] Cordova Plugin File Transfer: https://github.com/apache/cordova-plugin-file-transfer

[13] Cordova Plugin Geolocation: https://github.com/apache/cordova-plugin-geolocation

[14] Cordova SMS Plugin: https://github.com/cordova-sms/cordova-sms-plugin

[15] Phonegap Plugin Push: https://github.com/phonegap/phonegap-plugin-push

[16] WebSockets: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[17] RAG – Resilience Analysis Grid Technical Document prepared by the Industrial Safety Chair, January 2009

[18] Grant Agreement of RESOLUTE project, 653460, p131

[19] Deliverable 3.5 “European resilience management guidelines” of RESOLUTE Project

[20] Deliverable 5.2 “Blueprint of user experience and interaction for CRAMSS and apps” of RESOLUTE Project

[21] Deliverable 4.3 “Application Framework” of RESOLUTE Project