POLITECNICO DI MILANO

School of Industrial and Information Engineering

Computer Science and Engineering

POLITECNICO
MILANO 1863

TRACKME DD
Design Document

Software Engineering 2 Project

The project was made by

Luca Alessandrelli 846260
Andrea Caraffa 919970
Andrea Bionda 921082

Version 1.0 - 2018/2019

Deliverable:	DD
Title:	Design Document
Authors:	Luca Alessandrelli, Andrea Caraffa, Andrea Bionda
Version:	1.0
Date:	10-December-2018
Download page:	https://github.com/lucaalessandrelli/AlessandrelliCaraffaBionda.git
Copyright:	

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The following *Design Document* aims to give more technical details than *RASD* about TrackMe's application system. While the *RASD* presented a general view of the system and which functions the system is supposed to provide, this document presents the implementation of the system including components and the interaction with one another, the run-time processes and the deployment design. This document also includes the implementation, integration and test plan. For these reasons both narrative and graphical documentations about the system design are present.

This document, that is mainly addressed to the development team, covers the topics below:

- Overview of the high level architecture.
- Main components and their interfaces provided one for another.
- Runtime behavior.
- Design patterns.
- User Interface design.
- Design constraints and restrictions.
- Implementation plan.
- Integration plan.
- Test plan.

1.2 Scope

This document describes the implementation details of TrackMe's system, which is composed by three different software-based services:

- Data4Help.
- AutomatedSOS.
- Track4Run.

Data4Help is a location-based health information service-to-be that allows third parties to monitor the location and health status of individuals and it is the main service among the three. This system, allowing the registration of third parties and individuals, supports third parties' requests in order to access the data of some specific individual (individual monitoring request) or to access anonymized data of groups of individuals (group monitoring request). The other two software-based services, AutomatedSOS and Track4Run, exploit the features offered by Data4Help.

AutomatedSOS is a service-to-be thought to help elderly people. Constantly monitoring the health status of the subscribed customers, this service sends to the user's location an ambulance as soon as the recorded values are anomalous, for example when some health parameters are below certain thresholds.

Finally, Track4Run is a service-to-be that tracks athletes participating in a run. The service, allows organizers to define the path for the run, participants to enroll to the run and spectators to see on a map the position of all runners during the run.

1.3 Definitions, Acronyms, Abbreviations

- Definitions

  (a) Individual monitoring request: request to access to the data of some specific individuals.

(b) Group monitoring request: request to access to anonymized data of groups of individuals.

(c) Live/real-time acquisition: third parties can access to data as soon they are ready, through service updates.

(d) On demand acquisition: third parties can access to data when they request them.

(e) User credentials: information that an individual has to provide to become a registered user: name, surname, date of birth, address, email, telephone. number, job, marital status and fiscal code.

(f) Third parties' credentials: information that a company has to provide to become a registered one: company name, p.iva.

(g) Run information: all the information about the run such as name, date, promoters, maximum number of participants and race path.

(h) Partner Application: Application installed on users' device, not necessarily developed by TrackMe, that is in charge with retrieve location and health status.

(i) Security Number / Fiscal Code: a Social Security Number (SSN) is a nine-digit number issued to U.S. citizens and its primary purpose is to track individuals. Fiscal Code is the equivalent of Social Security Number in Italy.

1.4 Revision History

This is a report about all the document's versions along with the reason of the updates/changes.

Version	Changes	Motivation
1.0	/	/

Table 1: Revision History

1.5 Document Structure

This document is composed by seven sections:

1. Introduction	This section introduces the *Design Document*, defining the main use of the document and which topics are covered.
2. Architectural Design	This section, first gives an overview of the high level architecture of our system, then describes the main components in more detail showing how they interact between each other. Then, this section focuses on the entity relationship diagram about the data of the model and on the *Deployment View* where the main nodes and the communication between them are illustrated. The interaction among the components is again explained in more detail through several Sequence Diagrams in the *Runtime View* and in the *Component Interfaces* subsection. Finally, in the last part of the section, the selected architectural styles and patterns are described.
3. User Interface Design	This section presents the user interface design, focusing on the interaction between the user and the system.
4. Requirements Traceability	This section describes how the requirements defined in the *RASD* map to the design components previously defined in the document.
5. Implementation, Integration and Test Plan	This section describes the order in which the several components of the system are implemented, integrated and tested. The reasons of the selected order are explained in detail as well as the different kinds of tests the system is subject to.
6. Effort Spent	This section include information about the number of hours each group member has worked for this document.
7. Reference Documents	This section contains the list of reference documents.

Table 2: Document Structure

2 Architectural Design

2.1 Overview

The TrackMe services are built on a Client-Server structure, this way the system is organized through abstraction levels. We chose to adopt a 3-tier architecture.

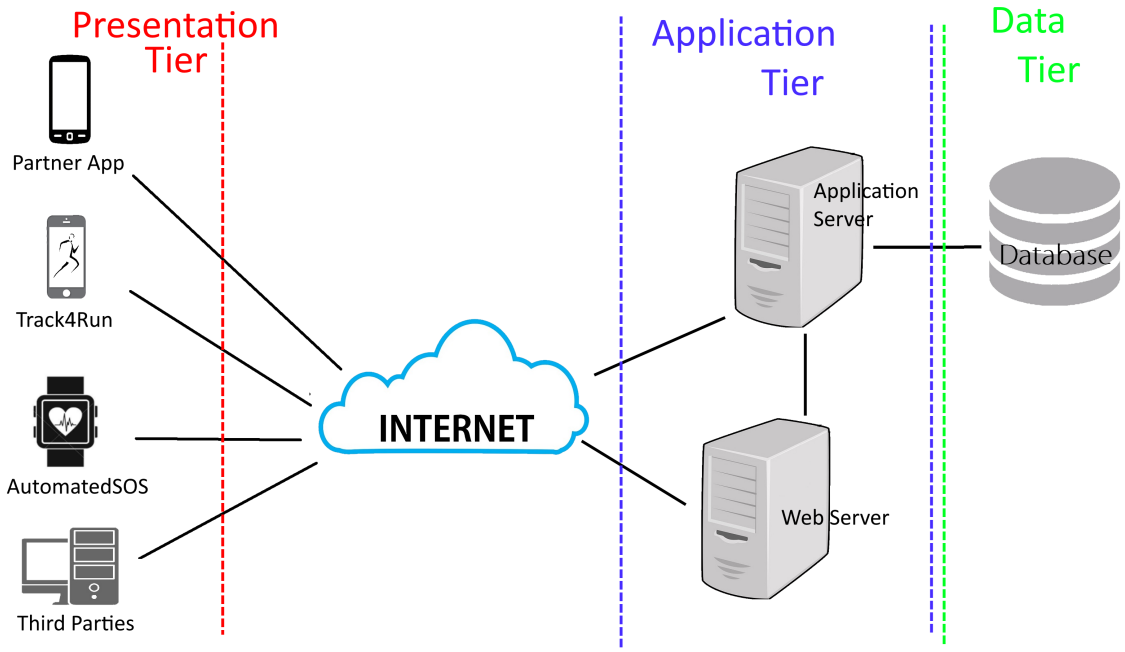

Figure 1: Overview Architecture

- **Presentation Tier**
  This layer makes the interaction possible between the user and the system. Here, the user sees all the information provided by the system in an easy and understandable way. This layer regards both the interaction with third parties and individuals.

- **Application Tier**
  This layer, managed almost totally by Data4Help service, is in charge of:

  - storing into the database the user's data;
  - collecting data from the database in order to answer to third party's requests;
  - generating statistics on collected data;
  - sending requested data to third parties;
  - managing run events, creating them and enrolling athletes to them.

  Moreover, even AutomatedSOS has a part of the logic application in order to continuously monitor user's health status.

- **Data Tier**
  In this layer all the user's data (location and health status) are stored into the database and are retrieved by the application tier in order to do statistics and answer third parties' requests. In addition, information about the run events, third parties' and individuals' credentials are stored.

More precisely, Data4Help manages the data and the core system logic while AutomatedSOS and Track4Run manage the presentation section. Actually, as already said, a small part of application tier is also present in AutomatedSOS, this is due to the fact that the Health Monitoring process requires to be executed as fast as possible.

2.2 Component View

2.2.1 Component Diagram

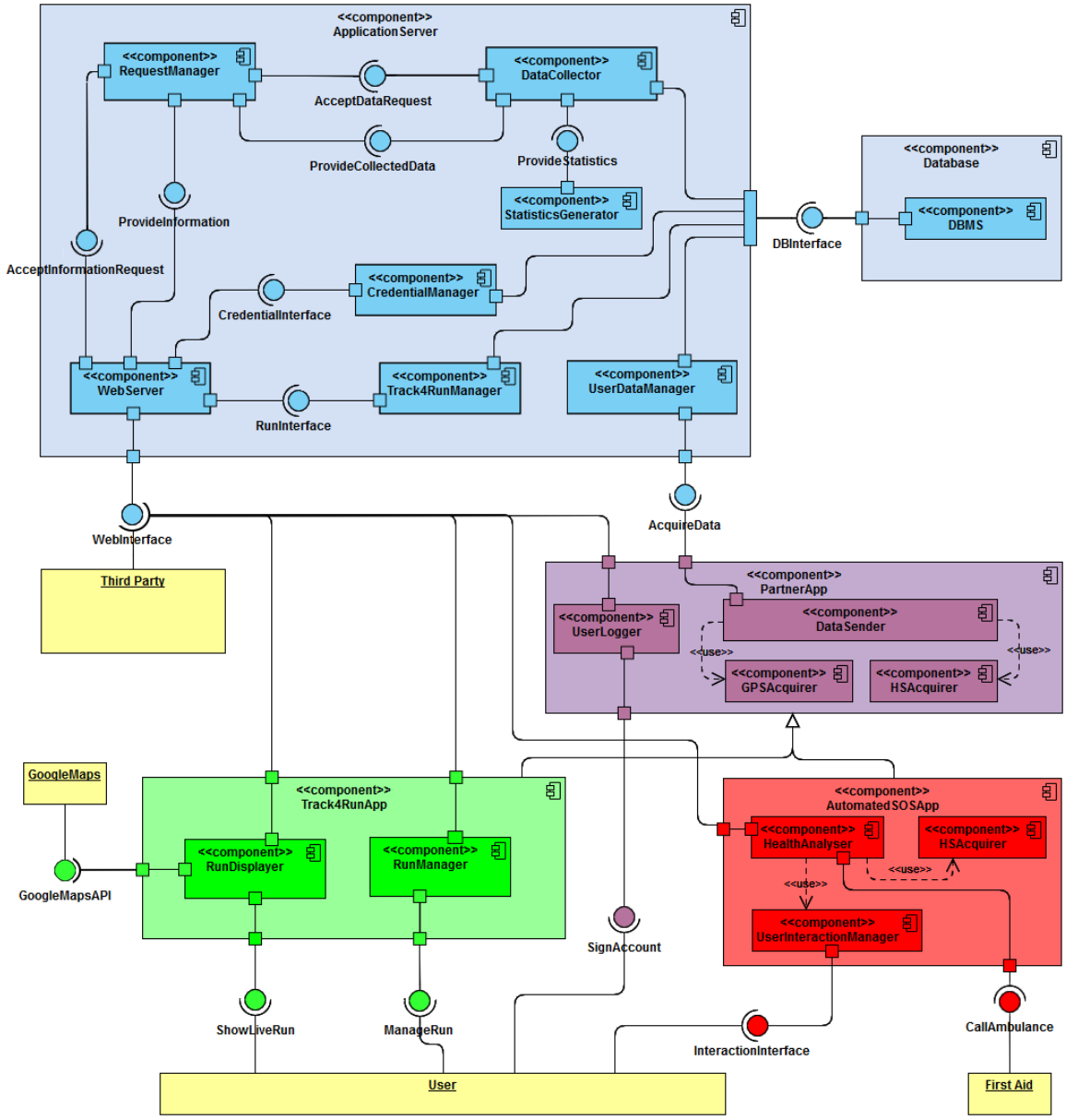

Figure 2: Component Diagram

Component diagram description

1 **ApplicationServer** This large component is in charge of managing Data4Help services like storing and providing, for interested companies, users' data such as GPS location and daily Health Status. Moreover, it manages run events information for Track4Run application.

1.1 **WebServer:** In order to accept requests and supply information to whoever needs them, this component offer a friendly web interface to simplify these operations.

1.2 **RequestManager:** In order to support the web server in its job, this component manages all the incoming requests: it sorts them per urgency, it wraps the requests in a smart data structure and sends it to the DataCollector. Also it unwraps the answers from the DataCollector and it continuously generates data exchange if a live acquisition is active.

1.3 **DataCollector:** This component communicates with the Database in order to retrieve information and supply them to the RequestManager. In order to perform these operations whenever the DataCollector receives requests from the RequestManger, it unwraps them, creates a query and sends it to the Database; once the Database answered the query, the DataCollector should wrap the answer and provide it to the RequestManager. Moreover if statistics on data are needed the DataCollector also sends data to the StatisticsGenerator which will answer with the statistics of interest.

1.4 **StatisticsGenerator:** This component generates statistics on data provided by the DataCollector such as arithmetic mean, variance from average, standard deviation and median.

1.5 **UserManager:** This component communicates with user's device in order to store into the Database data automatically collected by applications.

1.6 **Track4RunManager:** This component communicates with the Track4Run application in order to store into the database all the information about promoted run events, all the lists of athletes participating to runs and so on.

1.7 **CredentialsManager:** This component communicates with the WebServer in order sign up/in users to the system and to check whether the credentials inserted by a user that wants to log in are correct or not.

2 **Database** This component is in charge of physically storing data and organize them in a smart way according to DBMS rule, moreover it allows the access to those data.

2.1 **DBMS:** This component is in charge of storing all the data involving the system like users' parameters, third parties' requests and run events' information generated by the Track4Run application.

3 **PartnerApp:** This component describes how the partner applications of TrackMe are structured, from the components in charge of retrieving raw data from the device's API, to the components in charge of communicating with the ApplicationServer. The PartnerApp component is extended by all the partner applications that want to exploit Data4Halp service, so even by AutomatedSOS and Track4Run.

3.1 **GPSAcquirer:** This component acquires the user's GPS location at constant time periods.

3.2 **HSAcquirer:** This component acquires hearth rate, blood pressure and calories consumed from user's device at constant time periods.

3.3 **UserLogger:** This component offers to the user the possibility to sign up/in. For this reason it is in charge of showing the data policy and acquiring all the credentials inserted during registration. Moreover it has to communicate with the WebServer in order to forward the sign up/in request.

3.3 **DataSender:** This component exploits GPSAcquirer and HSAcquirer in order to provide to the UserManager component all the retrieved data.

4 **AutomatedSOSApp:** This component extends what is already specified in the PartnerApp component to exploit all its features. AutomatedSOS component has to use the HSAcquirer in order

to check health status parameters and it also has to use GPSAcquirer to specify the user's location whenever a First Aid call is required.

4.1 **HealthAnalyser:** This component exploits HSAcquirer features in order to continuously analyze health parameters, compare the last acquired data with historical ones to improve the analyzing process, and last, it checks data in order to prevent user's diseases and finally calls an ambulance whenever such parameters are below certain thresholds, compiling and providing a special report to First Aid.

4.2 **HSAcquirer:** This component acquires hearth rate, blood pressure and calories consumed from user's device at constant time periods.

4.3 **UserInteractionManager:** This component gives to the user the possibilities to see his/her current or historical health status, to set preferences, and to show a warning message whenever an ambulance has been sent to his/her location.

5 **Track4RunApp:** This component extends what is already specified in the PartnerApp component to exploit all its features. Track4Run component also provides to users the possibility to spectate a run, allows promoters to create and manage a run and, last, allows athletes to enroll to a given run.

5.1 **RunDisplayer:** This component is in charge of providing a human interface to the user in order to specify the run the user wants to spectate and then it shows the position of all the athletes in that particular run exploiting GoogleMapsAPI.

5.2 **RunManager:** This component provides to the user the possibility to promote a run specifying fields like date, path, name, maximum number of participants and description. Moreover, a promoter can invite specific athletes providing their security number/fiscal code. This component also allows athletes to enroll to a run.

2.2.2 Entity Relationship Diagram

The following figure shows how data are stored inside the database using an Entity Relationship Diagram.

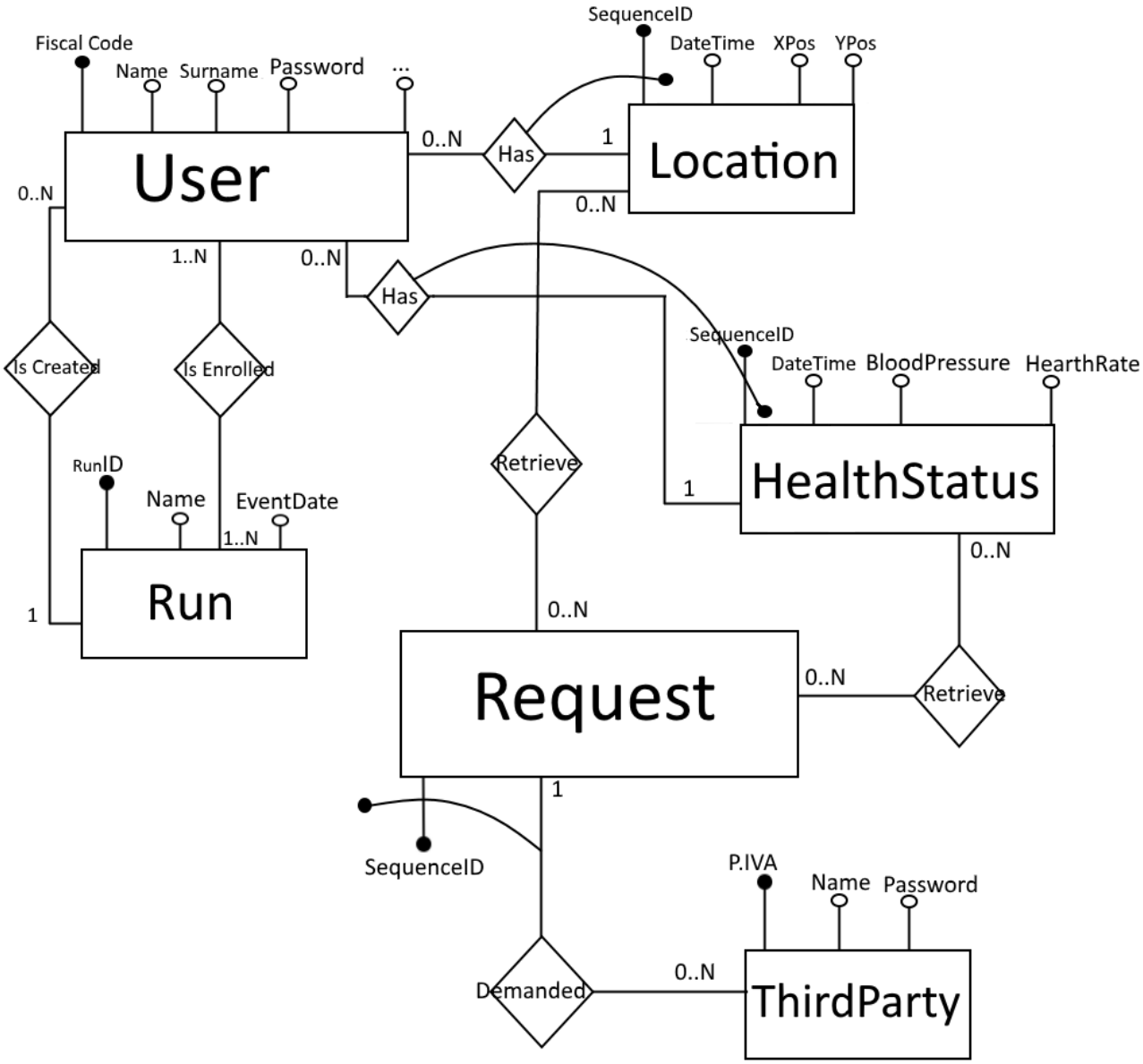

Figure 3: Entity Relationship Diagram

This diagram shows how every users' data are stored in order to create an history of their movements (Location) and health parameters (Health Status). In the database all the credentials of third parties and also all the requests are stored. Moreover the DB contains all the information about the runs for the Track4Run application.

2.3 Deployment View

The following Deployment Diagram captures the topology of the system's hardware and software distribution.

The SmartphoneApp, the SmartwatchApp and WebBrowser communicate to the WebServer through HTTP protocol due to the fact that the applications are based on browsers in order to improve portability among different Operative Systems. The two applications also communicate directly to the Application Server in order to exchange automatically retrieved data like the GPS Location and the Health Status.

The WebServer runs on a RedHat system on which Java Servlets and JSPs are deployed, also it communicates to the Application Server through RMI.

The Application Server runs on an Ubuntu system on which the business logic of the system is deployed, also it communicates to the DB Server through JDBC.

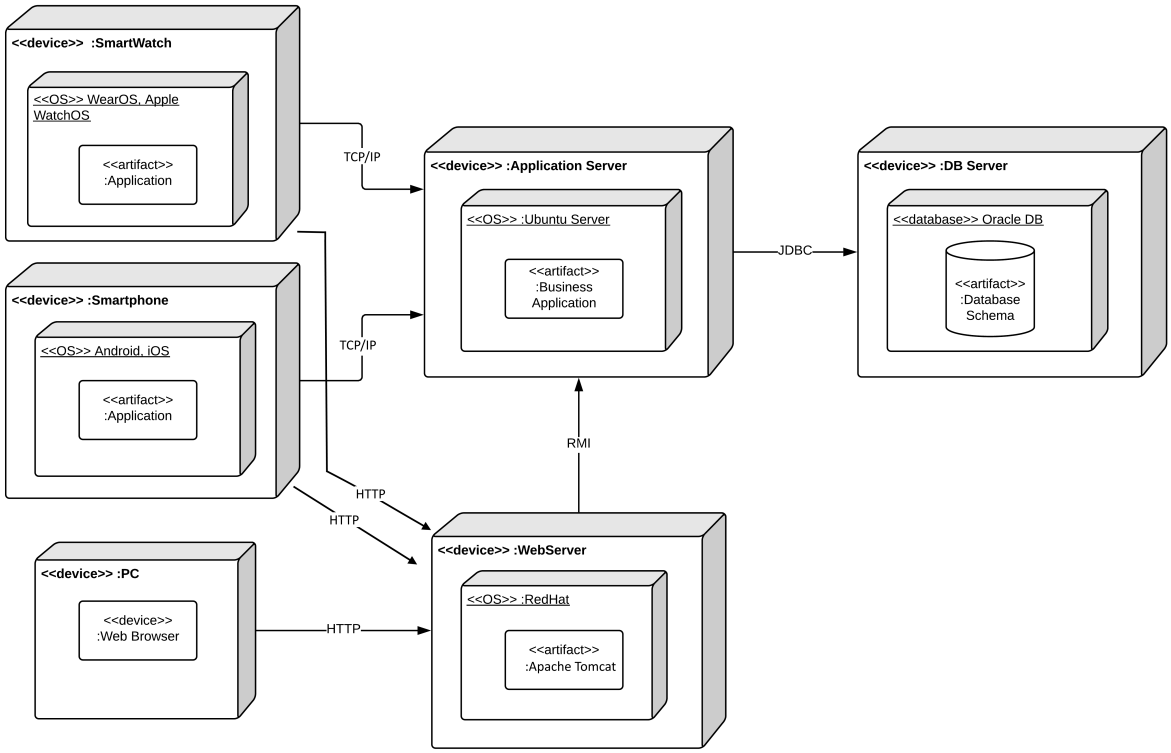

Figure 4: Deployment Diagram

2.4 Runtime View

In this section several Sequence Diagrams are shown in order to point up the interaction among components and their behavior in particular scenarios.

2.4.1 Individual Information Request

The following Sequence Diagram shows the interaction between components when a third party makes a request to obtain individual information. Since the individual's security number provided by the third party in the request form could not be associated to any real user (incorrect security number) or could be actually associated to a user that has not accepted the individual request privacy policy, the third party can get three different answers from the system.

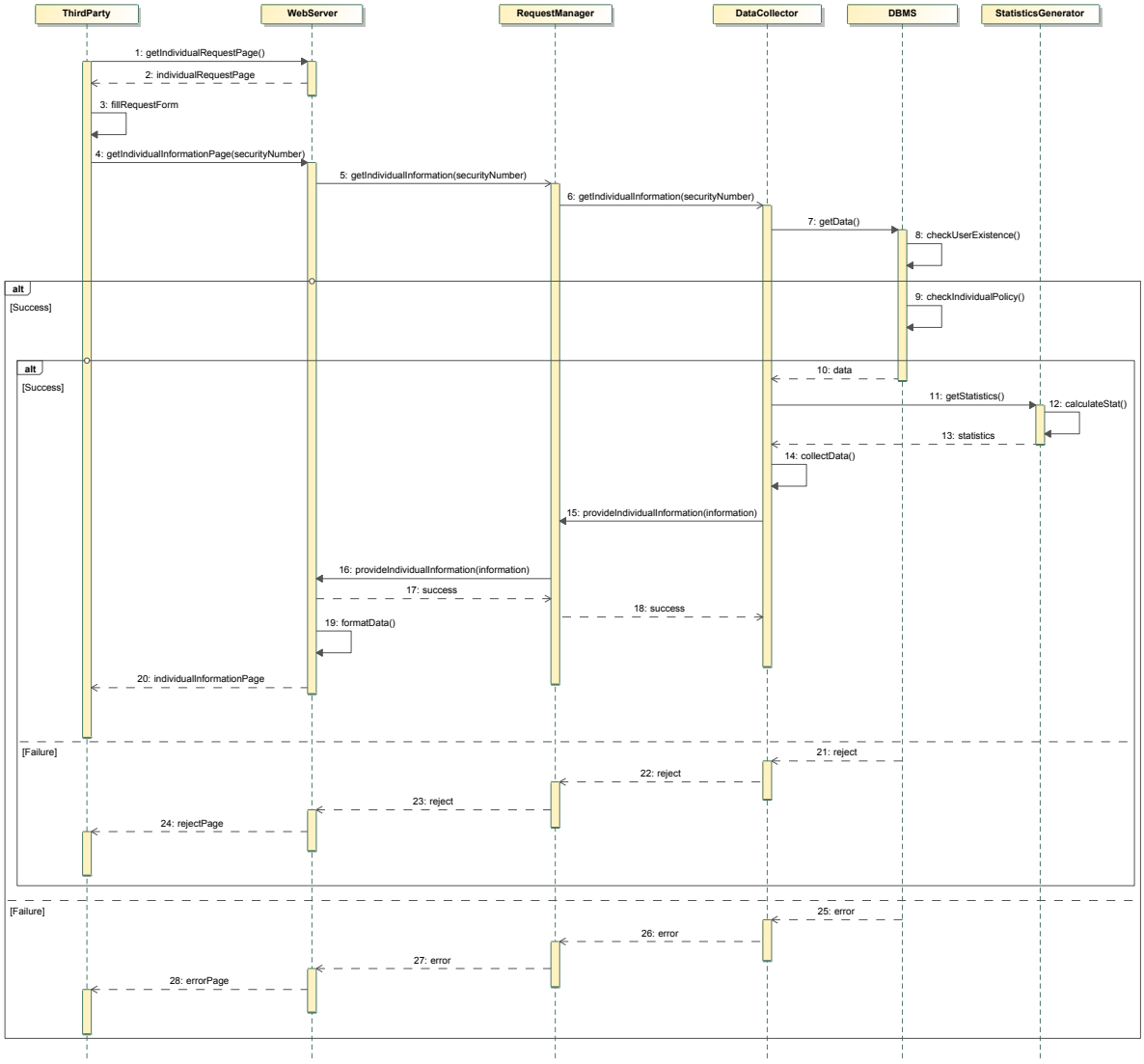

Figure 5: Individual Information Request Sequence Diagram

2.4.2 Send Ambulance Request

In the diagram below, the interaction between components to send an ambulance request to the first aid is shown. The data retrieved by the GPS Acquirer and HealthStatus Acquirer are both sent to Data4Help and to the HealthAnalyser.

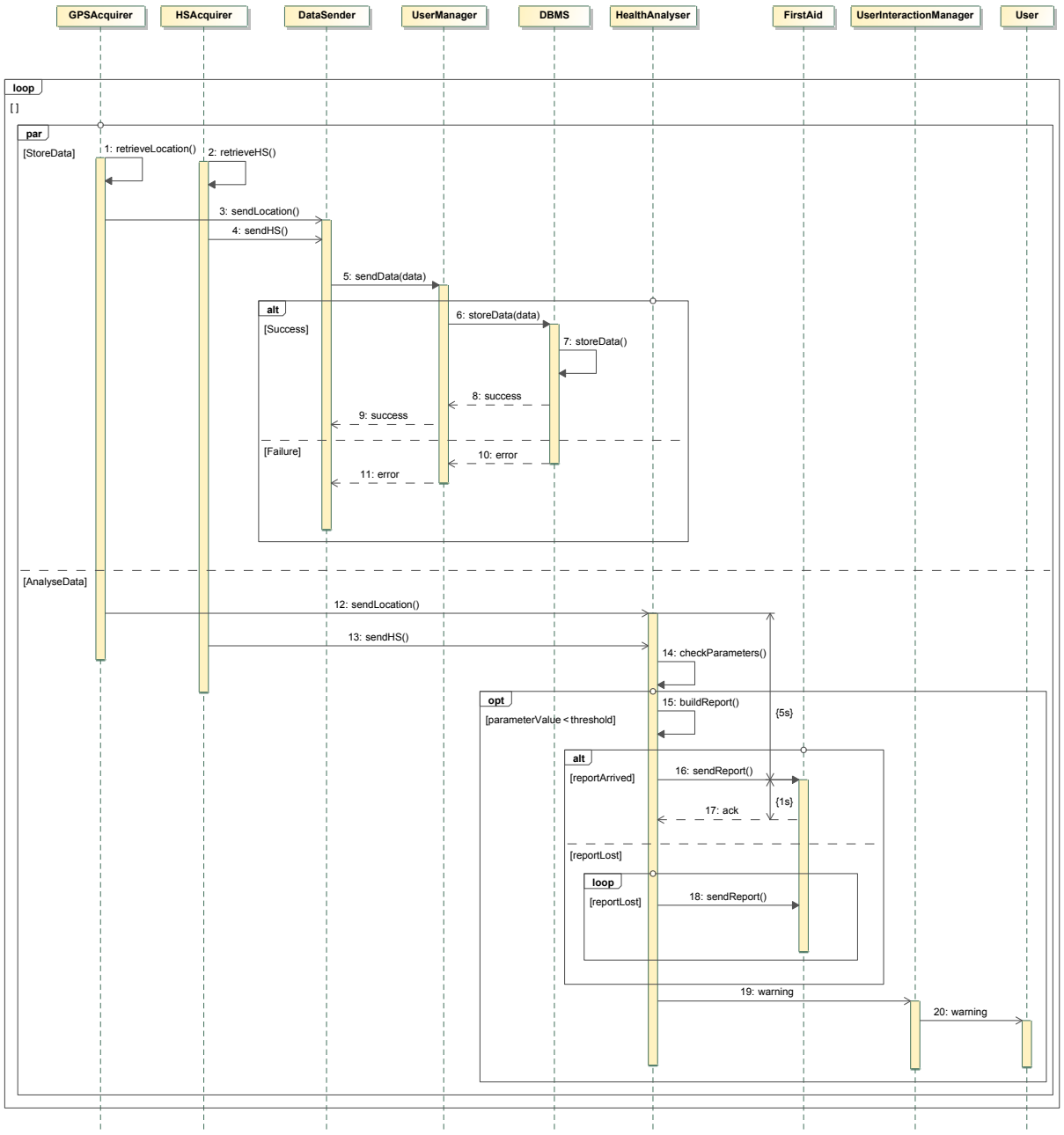

Figure 6: Send Ambulance Request Sequence Diagram

2.4.3 Spectate A Run

The next Sequence Diagram represents the interaction between components to show the live positions during a run event. The chart starts at the moment when the user, already logged in, select a specific run to spectate.

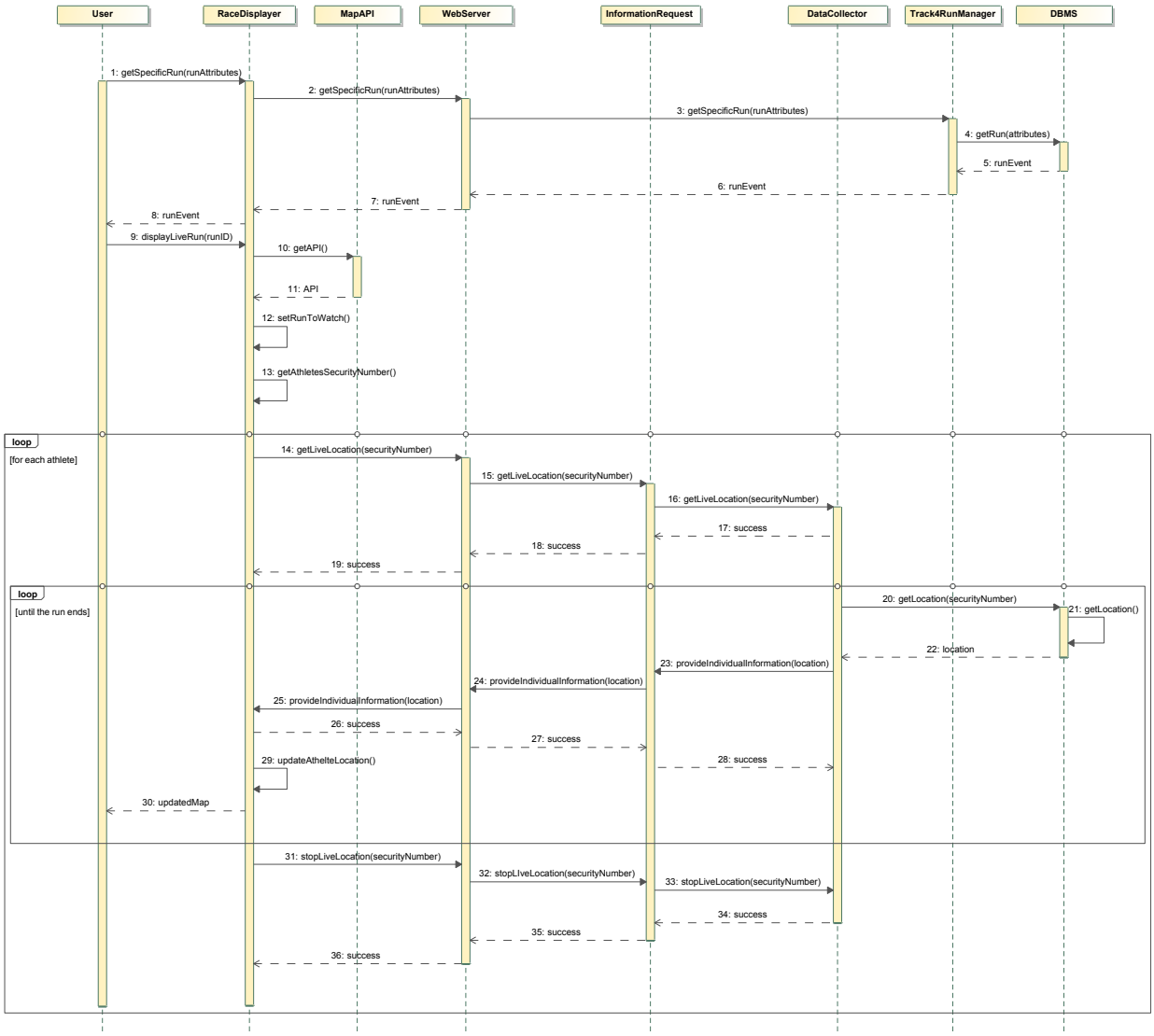

Figure 7: Spectate A Run Sequence Diagram

2.5 Component Interfaces

The following Diagram shows the dependencies between the main interfaces offered by the components of the system. In addition, for each interface it is possible to see the primary methods.

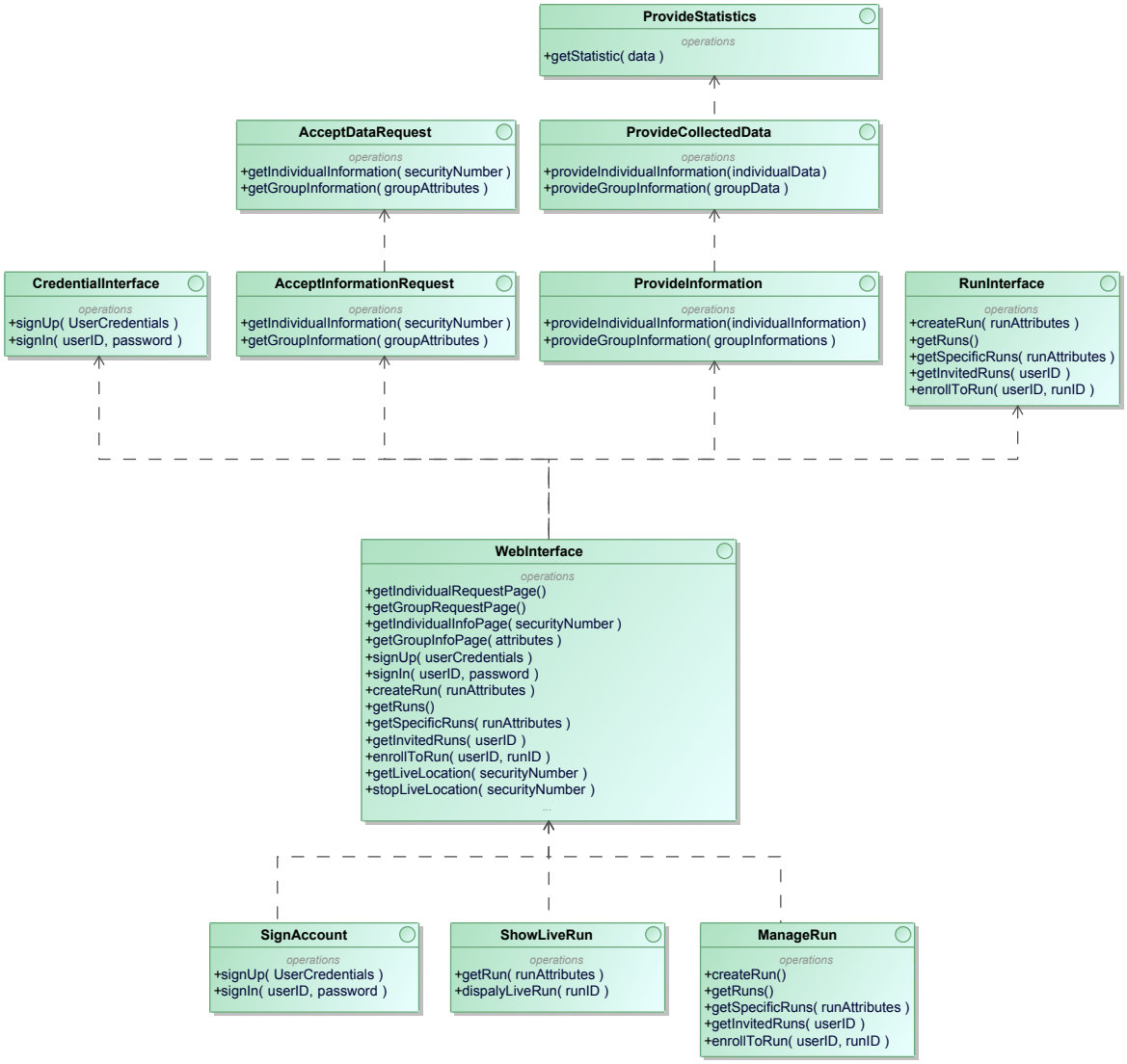

Figure 8: Component Interfaces Diagram

Below, a description for each interface offered by every component is given.

1.1 **WebServer**

1.1.1 **WebInterface:** Interface for third parties that want to interact with the system in order to obtain information through Data4Help service. This interface is user also by Track4Run and AutomatedSOS in order to perform a various number of tasks.

1.2 **RequestManager**

1.2.1 **AcceptInformationRequest:** Interface that allows the WebServer to submit requests to be evaluated.

1.2.2 **ProvideInformation:** Interface that provides to WebServer the information answers that match previous requests.

1.3 **DataCollector**

1.3.1 **AcceptDataRequest:** Interface that accepts requests from the RequestManager formatted in the proper way.

1.3.2 **ProvideInformation:** Interface that provide to RequestManager the information answers formatted in the proper way.

1.4 **StatisticsGenerator**

1.4.1 **ProvideStatistics:** Interface that provides statistics to the DataCollector.

1.5 **UserDataManager**

1.5.1 **AcquireData:** Interface that allows partner application to send to Data4Help the acquired data.

1.6 **Track4RunManager**

1.6.1 **RunInterface:** Interface that allows Track4Run application to create run events and to enroll a user to a specific run.

1.7 **CredentialsManager**

1.7.1 **CredentialsInterface:** Interface that saves (sign in) or checks (sign up) the credentials both from third party and individuals provided by the WebServer. Actually these operations are not performed by this interface that is only in charge of send credentials to the Credentials-Manager component.

2.1 **DBMS**

2.1.1 **DBInterface:** Interface that allows to store users' data into the database and to get users' data already collected in the database. This interface also allows to store and get the information about run events which are available in the database.

3.1 **UserLogger**

3.1.1 **SignAccount:** Interface that manages sign in and sign up operations.

4.2 **UserInteractionManager**

4.2.1 **InteractionInterface:** Interface that allows the user to interact with the App, such us set the preferences, see live health status parameter values and see the (past) acquired info. This interface is also in charge of displaying the warning message on the smartwatch device.

5.1 **RunDisplayer**

5.1.1 **ShowLiveRun:** Interface that allows to show to the user the map in which all the athletes involved in the run are displayed.

5.2 **RunManager**

5.2.1 **ManageRun:** Interface that allows a user to create and manage a run providing all the useful information. This interface also allows a user to enroll to a run.

2.6 Selected Architectural Styles and Patterns

This system is designed to be a Client-Server application with three tiers that well separate the different components as shown in *Section 2.1 Overview* in order to implement the MVC software design pattern.

- **Architectural Design:**

  - Client-Server architectural pattern is chosen because it is the most used and easily implementable communication pattern, in fact in our application it is the best one to supply the third parties information requests and to acquire data from users via Internet. Regarding the acquisition of users' data, when data are successfully collected from the device, the smartphone decides to call the Data4Help server in order to provide and store data. Furthermore, even third parties have to call Data4Help in order to exchange information. Since these two operations are the main core of the service the Client-Server architecture is the right choice.
  - 3-tier architecture should be implemented to assure the reusability and maintainability of the system, in fact this division permits to change some parts of the system without the need to reconfigure the entire solution.
    * **Presentation Tier**, as described above, is in charge of displaying on final user's screen a human interface that allows the interaction with our system. This tier is present only on the user's device. Regarding third parties the software in charge to display human interface will be the web browser, so the exchanged information will be HTML pages exchanged through HTTP connection. Instead, regarding users, the software will be the application that runs on the device that works on HTML pages as well; so for both the type of users the interface that connects them to the Application Tier is the WebServer, which allocates the right page to the right target. In order to supply on this aspect the partner application (AutomatedSOS and Track4Run as well) will be developed in HTML (with the support of JS and CSS) that are web languages, this aspect make also the application multi-platform so they can be launched on either iOS or Android devices. HTTP communication protocol will be used for both connection.
    * **Application Tier**, as described above, is in charge of acquiring all the data incoming from users, of storing them into the Data Tier and of handling the requests on viewing data. This Tier is present on the largest part on Application server but even AutomatedSOS has inside his software a part of the application service. All the component inside this tier will be developed in Java, in particular the Web Server runs Tomcat Java servlet that generates the HTML pages for the Presentation Tier, acquires data that users submit and communicates through RMI with the Business Application that runs on the Application Server. The Business Application is the main core of Data4Help service, and it is composed by all the other components inside the ApplicationServer (as Request Manager,Data Collector,Statistic Generator...); It is required that it can interact with the Web Server, in order to acquire the external requests, and then using the components described above supply the information required; furthermore it is in charge of managing all the information of users, like logins or registrations, and managing Track4Run races. Another very important aspect is that this application has to retrieve data from users implementing a daemon on a specific port and listening for a TCP/IP connection incoming from the partner application in order to acquire the users' data. As mentioned before even AutomatedSOS implements a small part of the application tier inside his software which is in charge of monitoring continuously the health status acquired, of comparing it with old data and of calling an ambulance via web if there is a necessity filling out a form in which are indicated all the critical parameters detected and all the useful information about the user.

    * **Data Tier**, as described above, is in charge of storing and providing information. This tier is present only on the Database Server that is an Oracle machine using SQL Schemas. The database is divided in two parts, one regarding Data4Help service that stores and manages users' data: historical users' location and health parameters; the other one regarding Track4Run stores races' information. This Tier offers an SQL interface that can be exploited by Business application using JDBC libraries that supports SQL functions.

- **Software Pattern:**

  - The MVC (Model-View-Controller) is the obvious software pattern that can be applied on 3-tier architecture because it can split the entire software solution in the three main region: the Model implements all what is described in the data tier, the controller all what is described on the Application tier and View all what is described in the Presentation tier. All these three groups communicates with each other through the so called Adapter pattern.
  - The Adapter design pattern is developed to create an interface that permits the MVC components to easily communicate with each other using different languages or different operating system that is mandatory in our system. This pattern is the representation of what is described in the Component Interface section.
  - Strategy design pattern can be very useful in the implementation of the RequestManager component because it can dynamically change the way it handles the request queue in order to supply efficiently an heavy request load.
  - Proxy design pattern is an obvious consequence of the Web Server, because this component stays in the middle between the two tier and creates a sort mask that obscures what is behind it and simplify a lot the Presentation Tier software that has only to display what the Web Server indicates.

3 User Interface Design

In this Section the user interface design, already presented in *Section 3.1.1 User Interfaces* of *Requirements and Analysis Specification Document* with several mockups, is explained in more detail. A special attention is focused on the interaction between the user and the system, and on how the mockups are correlated to each other.

- Data4Help

  Third parties can interact with Data4Help system through a website from which they can make the several kind of requests.

Figure 9: Data4Help's Homepage Mockup

This is the homepage of Data4Help's website, reachable from the user's browser by the link `www.data4help.com`. In the homepage it is possible to read a brief overview of the service offered by Track4Me. Several actions are possible from here clicking on the different buttons present in the header **HOME**, **ABOUT**, **SERVICES**, **MAKE NEW REQUEST** and, since the user is already logged into the system, also the options **MY REQUESTS** and **LOG OUT** are available. Besides the options that can be selected in the header, scrolling down the page it is possible to select a button to directly make a monitoring request.

Clicking on **MY REQUESTS** a list of all the historical requests will appear, both the approved and the denied ones and also the requests waiting for the approval. For the approved ones it is also possible to see the relative answers containing all the request information.

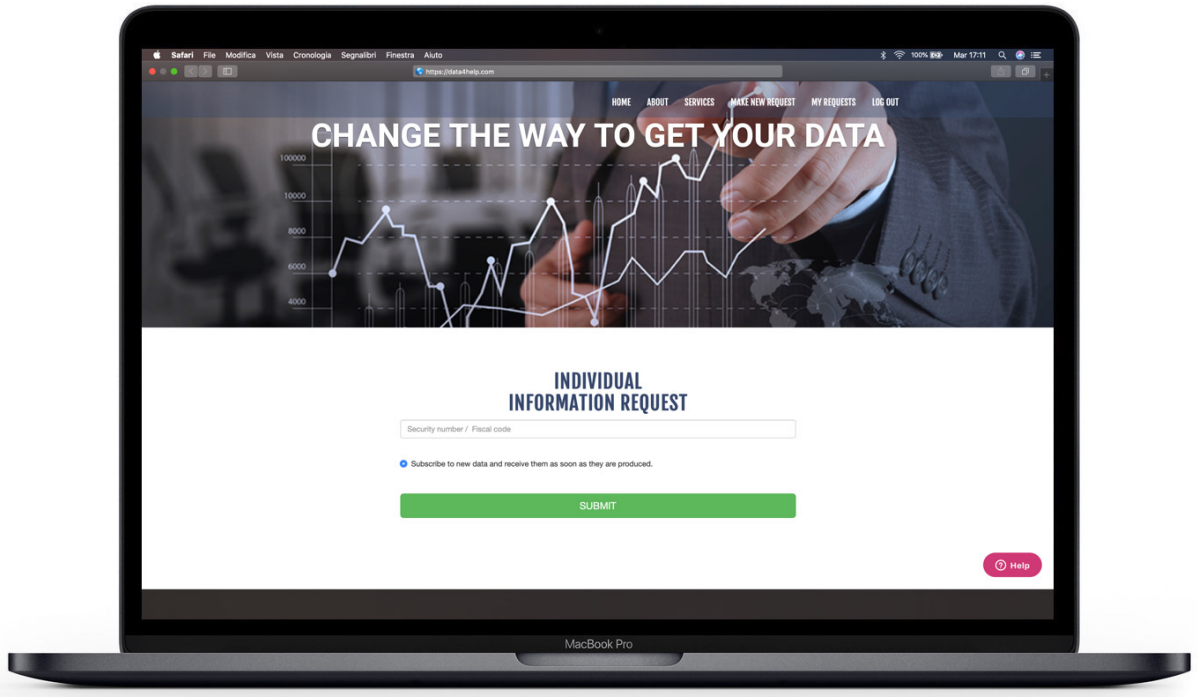

Figure 10: Individual Request Form Mockup

Clicking on **MAKE NEW REQUEST**, the website loads another page where the third party is asked to select an option between **INDIVIDUAL MONITORING REQUEST** and **GROUP MONITORING REQUEST**. Clicking on the first option the system loads a new page containing the form to make an individual monitoring request. Once filled the security number's field it is possible to submit the request pressing the **SUBMIT** button.

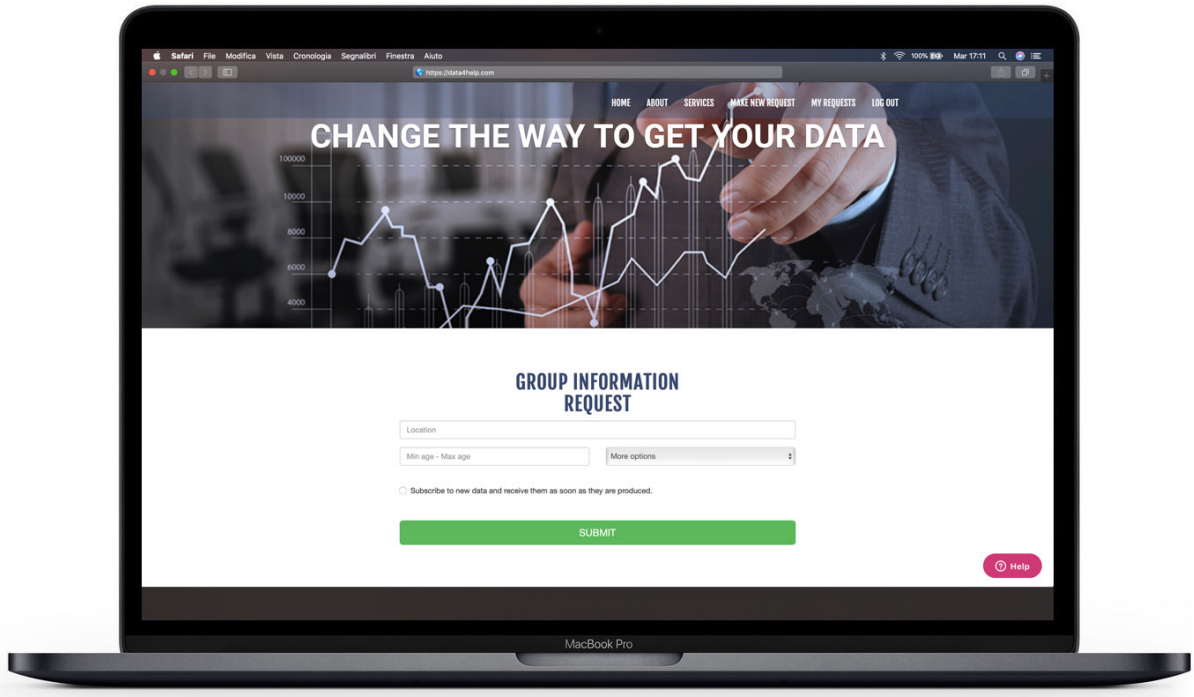

Figure 11: Group Request Form Mockup

Otherwise, clicking on **GROUP MONITORING REQUEST** button the system loads a new page containing the form to make a group monitoring request. In the form it is possible to limit the request to a group living in a specific zone, writing the geographical area on the **Location** field. It is then possible to select a minimum and a maximum age. Clicking on the **More** button it is possible to furthermore limit the request adding other filters to the group of individuals. Once the user has filled all the fields, clicking on the **SUBMIT** button the request is forwarded to Data4Help system.

- AutomatedSOS

TrackMe offers to AutomatedSOS users an App for smartwatches, with which the users can see their location and health status information.

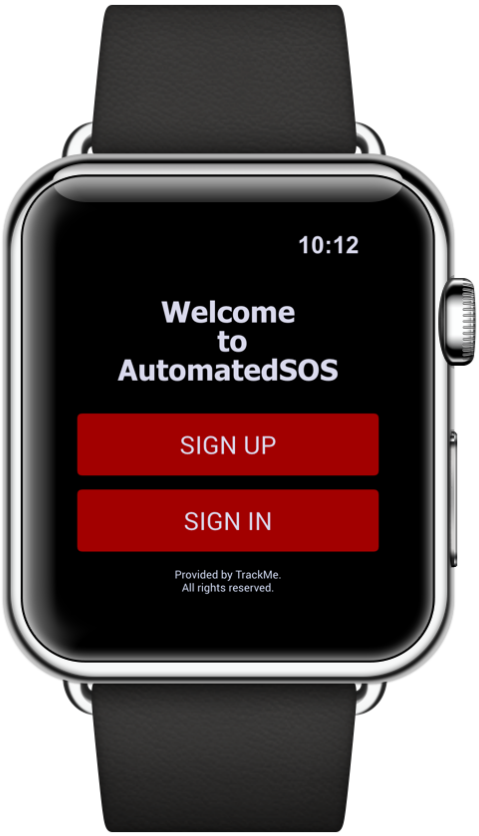

Figure 12: Welcome Page Mockup

Figure 13: Registration Form Mockup

In the first access to AutomatedSOS App the user is asked to sign up or to sign in (Figure 12). Based on the fact that the user could already have a TrackMe account, he/she will choose the right option between the two. In future accesses to the App the user do not need to select each time one of these two possible options because the App automatically remembers the account which is logged in. In case of sign in (Figure 13), the user has to fill all the mandatory fields in order to complete the registration form. Scrolling down the screen other fields will appear. Once every field is correctly filled, it will be possible to select the **NEXT PAGE** button.

Figure 14: Privacy Policy Conditions Mockup

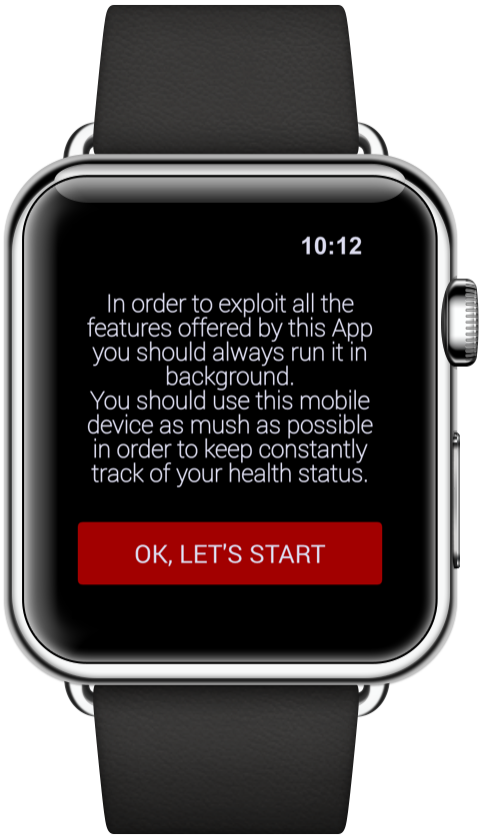

Figure 15: Usage Conditions Mockup

During the first access to the App, the next step after sign up/sign in is to agree to the Privacy Policy Conditions (Figure 14). In order to use this App the user has to agree to the treatment of Location and Health Status data by Data4Help. In addition, these data could be used by third parties for the group monitoring request. Without agreeing the Privacy Policy Conditions the user cannot go to the next page, therefore he/she will not be able to use this App. Once clicked on **I AGREE** the last step before starting using the App is to take note of the importance of wearing the Smartwatch as much as possible and to let the App runs in background (Figure 15).

Figure 16: Main Menu Mockup

Figure 17: Warning Message Mockup

The main menu of the App, which is immediately accessible by selecting the AutomatedSOS App on the Smartwatch home, is composed by three parts (Figure 16). Selecting **Monitor Health** the user can see live health information, like the current Heart Rate, Blood Pressure and so on. Choosing the **Acquired Info** button the user can see all the historical data stored since the first access to the App. Finally, **Preferences** option allows the user to set his/her own threshold parameters according to the particular illnesses he/she is affected, own age and so on. As soon as an ambulance request is sent and acknowledged due to the user's critical health status a warning message (Figure 17) appears on the screen of the smartwatch notified by an alert sound.

- Track4Run

Track4Run users can use an App for smartphone and another one for smartwatches. The first one could be used by everyone, while the second one is made only for the athletes.

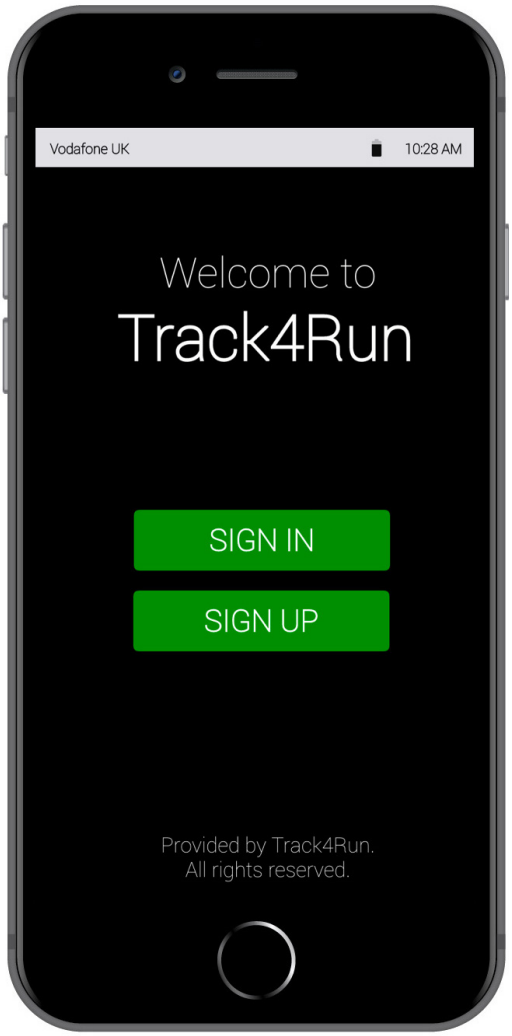

Figure 18: Welcome Page Mockup

Figure 19: Registration Form Mockup

In the first access to Track4Run App the user is asked to sign up or to sign in (Figure 18). Based on the fact that the user could already have a TrackMe account he/she will choose the right option between the two. In future accesses to the App the user do not need to select each time one of these two possible options because the App automatically remembers the account which is logged in. In case of sign in (Figure 19), the user has to fill all the mandatory fields in order to complete the registration form. Once every field is correctly filled, it will be possible to complete the registration selecting the **Register Now** button.

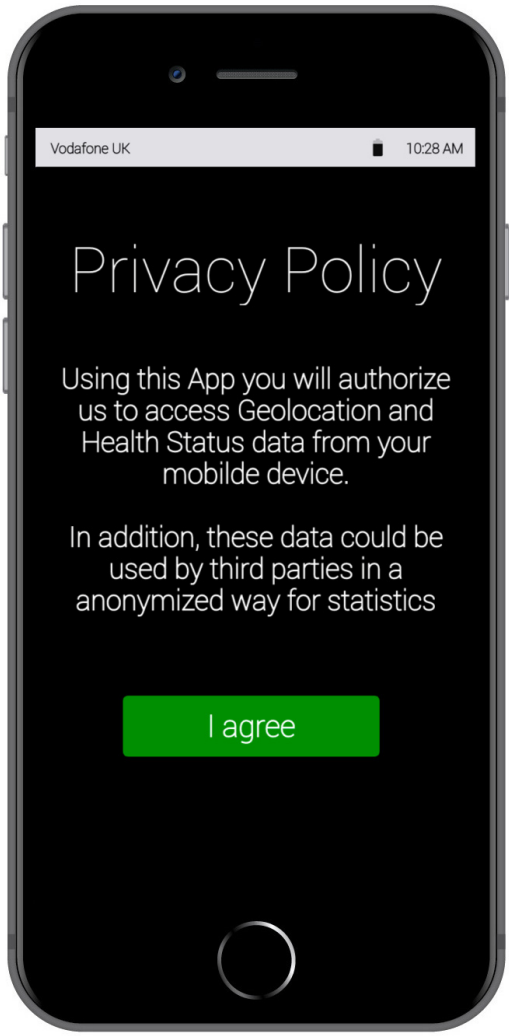

Figure 20: Privacy Policy Conditions pt.1 Mockup

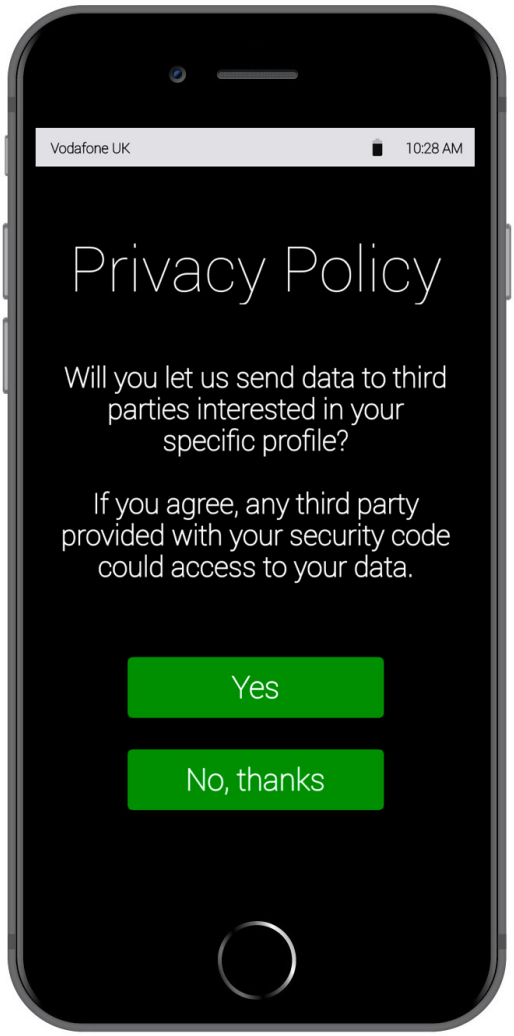

Figure 21: Privacy Policy Conditions pt.2 Mockup

During the first access to the App, the next step after sign up/sign in is to agree the Privacy Policy Conditions (Figure 20). In order to use this App the user has to agree the treatment of Location and Health Status data by Data4Help. In addition, these data could be used by third parties for the group monitoring request. Without agreeing the Privacy Policy Conditions the user can not go to the next page, therefore he/she will not be able to use this App. All the three first steps presented so far are substantially the same of AutomatedSOS. The next step regards the second part of the Privacy Policy Conditions (which was not presented for the previous system) about individual monitoring request (Figure 21). In this case it is not strictly necessary to accept it, the user simply can select the **No, thanks** option.

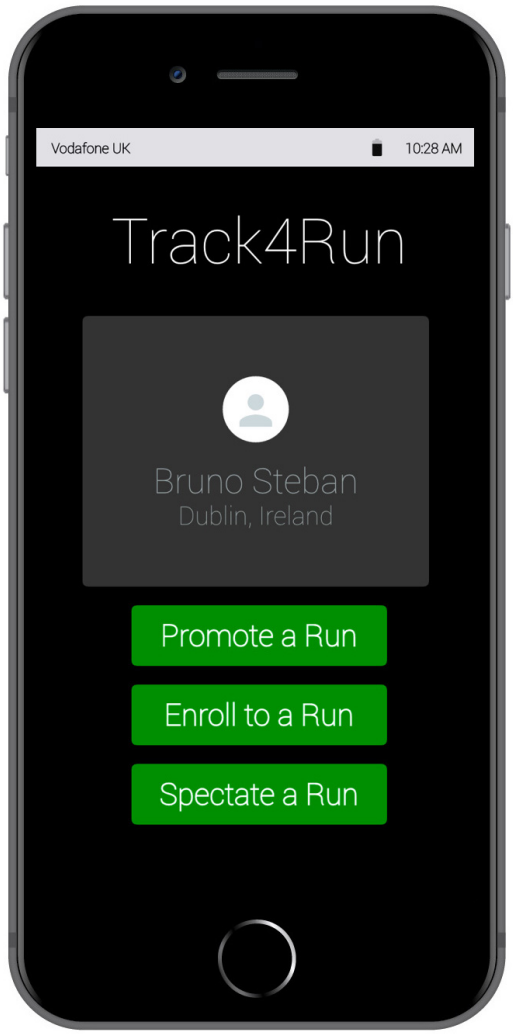

Figure 22: Main Menu Mockup

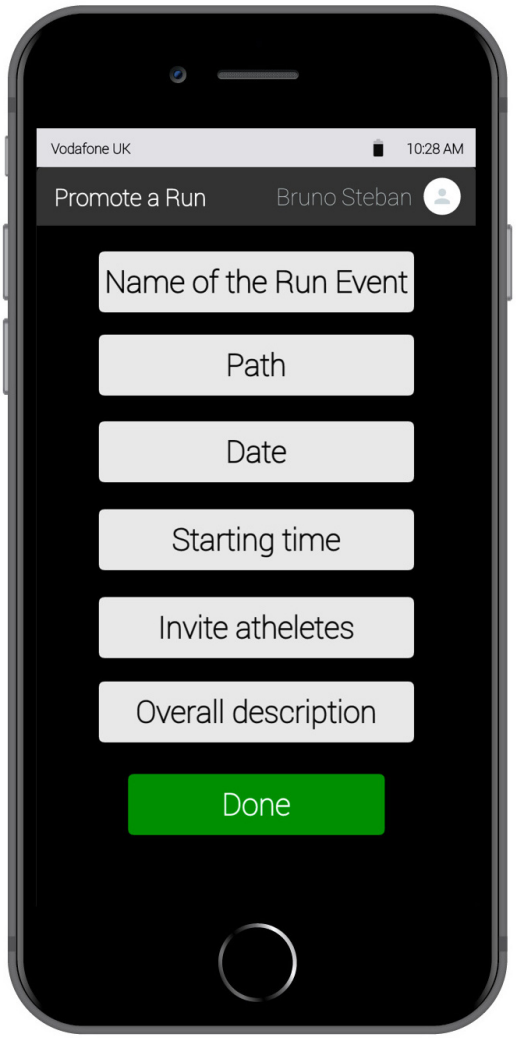

Figure 23: Promote a Run View Mockup

In Track4Run Home three options are possible (Figure 22). Selecting the first one, **Promote a Run**, a user can create a run event and promote it inviting athletes. The second option, which is **Enroll to a Run** allows the user to enroll to a run. Finally, choosing **Spectate a Run** the user can see on a map the position of all athletes during the run. Now, we analyze these three possibilities in detail. Choosing the first one the App leads the user to a new page in order to manage the event (Figure 23). Here, it is possible to set all the necessary features of the run, like defining the path on a map, setting the date, the starting time and so on. It is also possible to invite athletes to the run in order to promote the event.

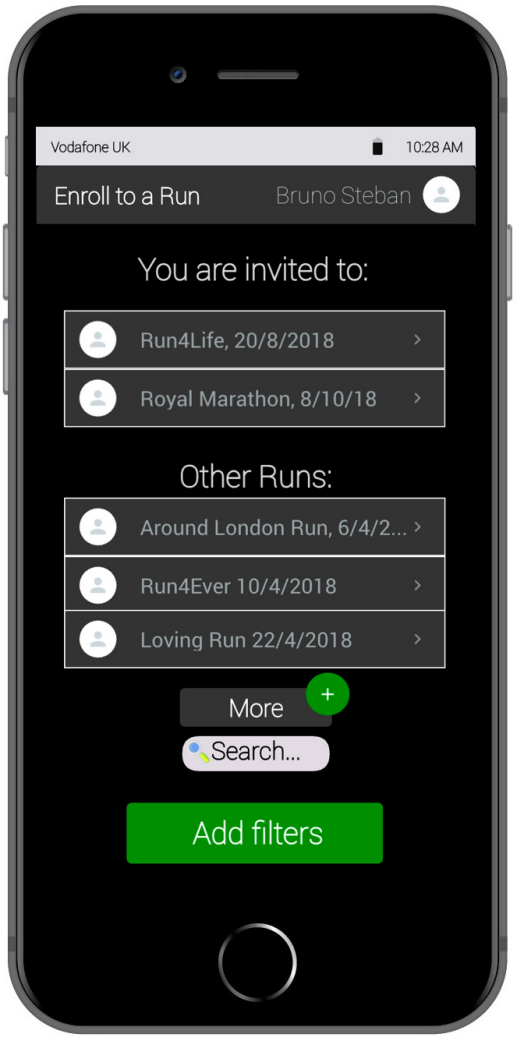

Figure 24: Enroll to a Run View Mockup

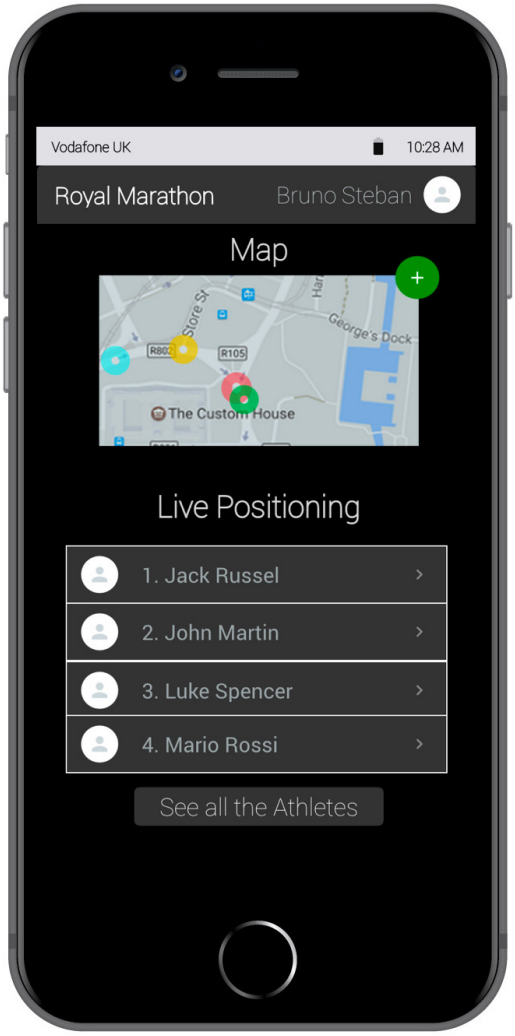

Figure 25: Spectate a Run View Mockup

In the section Enroll to a Run (Figure 24) a user can see a list of all the runs in which he/she has been invited to. The user can select one of these runs to see furthermore details about it and at the end to enroll to it. A list of all the runs that will be soon held is showed immediately below. Since the large number of all the future runs, only a little portion of them is showed, selecting the **More +** button the user can see another set of run events. In addition, through **Add filters** it is possible to limit the list of the all runs to a specific date, location and similar options. To easily access to a specific run, the **Search** button allows the user to immediately find the run he/she was looking for. In the last section, Spectate a Run (Figure 25), it is possible to spectate a specific run in an interactive way. A map shows all the runners enrolled in it, each one represented by a different colored circle. Selecting the **+** button is possible to zoom in the map that will appear on full-screen mode. Below the map the live positioning shows the first four athletes, anyway selecting **See all the Athletes** option it is possible to see the live positions of all the participants.

4 Requirements Traceability

This section shows how the requirements can be supplied by the designed components.

G.1 Acquire user's position and health status.

R.1 Retrieve user credentials inserted into partner application as group attributes.

- UserLogger: Receives user's credentials from SignAccount interface and then stores them locally. After this operation it waits for the privacy policy acceptance.

R.2 Allow users already registered in Data4Help world to sign in with their account without providing user credentials again.

- UserLogger: This component automatically inform the WebServer whenever a user starts the application.
- WebServer: Receives users' log requests from WebInterface and sends them to the CredentialManager.
- CredentialManager: Checks the users' availability and communicate it in reverse order allowing the UserLogger to decide if a user is admissible or not.

R.3 Allow individuals to agree the privacy policy (first part) so that they can be tracked in group mode through installed application.

- UserLogger: Receives from SignAccount interface that the user has accepted the first part of the privacy policy. Then if the user decided to create the account, it passes all these information to the WebInterface.
- WebServer: Receives from WebInterface the credentials and provides them to the CredentialManager through CredentialInterface.
- CredentialManager: Checks the inserted credentials, if they are valid it stores them into the DB using DBInterface.
- DBMS: Stores data incoming from DBInterface.

R.4 During the registration allow individuals to specify if they are also interested to be tracked in single mode (agree the second part of privacy policy) through installed application.

- UserLogger: Receives from SignAccount interface that user has accepted the second part of the privacy policy. Then if the user decides to create the account, it passes all these information to the WebInterface.
- WebServer: Receives from WebInterface the credentials and provides them to CredentialManager through CredentialInterface.
- CredentialManager: Checks the inserted credentials, if they are valid it stores them into the DB using DBInterface.
- DBMS: Store data incoming from DBInterface.

R.5 For each user registered the system has to automatically retrieve and store data from partner applications with a resolution of 10 minutes; independently from the requests reached.

- DataSender: This component, exploiting the data acquired by GPSAcquirer and HSAcquirer, sends every ten minutes all the information acquired since the last message to the UserDataManager using the AcquireData interface.
- UserDataManager: When a data update is arrived through the AcquireData interface, this component generates and sends an SQL query that allows the insertion of the new data into the DB.
- DBMS: Stores data incoming from the DBInterface.

G.2 Provide to third parties, the user's position and health status.

R.6 Allows third parties to register to Data4Help service specifying all their credentials.

- WebServer: using the WebInterface, it provides to third parties a web page that allows them to insert all the necessary credentials; then it retrieves these information. Moreover it sends these data to the CredentialManager using CredentialInterface.
- CredentialManager: Checks the inserted credentials and if they are valid it stores them into the DB using DBInterface.
- DBMS: Stores data incoming from DBInterface.

R.7 The system should allow third parties to send information requests.

- WebServer: Using the WebInterface, it provides to third parties a web page that allows them to make a request inserting all the necessary information. Then it passes the requests using the AcceptInformationRequest interface.
- RequestManager: Receives requests from AcceptInformationRequest interface and stacks them in a queue sorted by urgency criteria.

G.2.1 Provide data on demand to non-subscribed third parties.

R.8 The system has to collect all the useful data that match the request.

- RequestManager: This component periodically checks the queue in order to select the most urgent request. Once an on-demand request is picked up it wraps the request and send it to the DataCollector.
- DataCollector: Once a data request is received, this component builds a DB query that matches what third party wants and sends it to the DB using DBInterface.
- DBMS: When the query is arrived, this component computes it and provides data to the DataCollector using DBInterface.

R.9 The system has to generate a statistic on data selected.

- DataCollector: Once data are arrived from the DB, this component asks to compute some kind of statistic on the data through the ProvideStatistic interface.
- StatisticGenerator: This component receives data from ProvideStatistic interface, computes statistics and sends them back to the DataCollector.

R.10 The system has to send to the third party all the raw data collected until the moment of the request.

R.11 The system has to send all the statistics already produced.

- DataCollector: Once data are arrived from the DB and statistics are generated, this component wraps them and sends this package to the RequestCollector using the ProvideCollectedData interface.
- RequestCollector: This component receives information from DataCollector, unwraps and sends them using ProvideInfromation interface to the WebServer.
- WebServer: Using WebInterface, it provides to third parties a web page that allows them to see statistics and download retrieved information.

G.2.2 Provide data in real-time to subscribed third parties.

R.12 Allow third parties to subscribe to groups or individuals in order to receive live data.

- RequestManager: Once a real-time request is acquired this component stores all the attributes selected in order to make the subscription. Using a parallel thread it automatically adds into the priority queue the data request to supply the live acquisition.
- DataCollector: Once a data request is received, this component builds a DB query in order to retrieve only new data that matches what the third party wants and it sends it to DB using the DBInterface.

- DBMS: It computes the received query and provides data to the DataCollector using the DBInterface.

R.13 Provide to subscribed third parties raw data as soon as they are available by the system.

- DataCollector: Once data are arrived from DB, this component wraps them and sends this package to the RequestCollector using ProvideCollectedData interface.
- RequestCollector: This component receive information from DataCollector, unwraps and sends them using the ProvideInfromation interface to the WebServer.
- WebServer: Using WebInterface, it provides to third parties a web page that allows them to download retrieved information.

G.3.1 Allow third parties two different ways to get users' data.

R.8 The system has to collect all the useful data that match the request.

R.10 The system has to send to the third party all the raw data collected until the moment of the request.

R.14 Allow third parties to insert the fiscal code of the user he wants to track.

- WebServer: In this specific case this component has to offer to third parties a web page that allows them to fill in the fiscal code that they want to track. Then the request formulated has to be forwarded to the RequestManager through the AcceptInformationRequest interface.

R.15 Deny third parties to receive single mode information about users that have not accepted the second part of the privacy policy.

- RequestManager: This component periodically checks the queue in order to select the most urgent request. Once a single mode request is picked up it wraps the request and sends it to the DataCollector.
- DataCollector: Once a data request is received, this component builds a DB query that matches what third parties wants and it sends it to the DB using the DBInterface, requiring also all the information of the user involved.
- DBMS: Compute the received query and provides data to the DataCollector using DBInterface.
- DataCollector: Once data are retrieved this component checks, before it returns an answer to the RequestManager, if the user involved has accepted the second part of the privacy policy: if he/she did it wraps the information package and proceeds as always, otherwise it communicates to the RequestManager that the request cannot be accepted.

G.3.2 Allow third parties to get data of a group of people.

R.8 The system has to collect all the useful data that match the request.

R.10 The system has to send to the third party all the raw data collected until the moment of the request.

R.16 Allow third parties to insert search area and attributes in which they are interested to restrict their field of search.

- WebServer: In this specific case this component has to offer to third parties a web page that allows them to fill in all the attributes that the users to track must have. Then the request formulated has to be forwarded to the RequestManager through the AcceptInformationRequest interface.

R.17 Deny third parties to receive information if the provided information can hurt users' privacy, for this purpose group request under 1000 users involved are rejected.

- RequestManager: This component periodically checks the queue in order to select the most urgent request. Once a group mode request is picked up it wraps the request and sends it to the DataCollector.
- DataCollector: Once a data request is received, this component builds a DB query that matches what the third party wants and it sends it to the DB using DBInterface, requiring also all the number of users involved in the specific query.
- DBMS: This component computes the received query and provides data to the DataCollector using DBInterface.
- DataCollector: Once data are finally retrieved this component checks, before it returns an answer to RequestManager, if the number of users involved are less then 1000; if so it wraps the information package and proceeds as always, otherwise it communicates to the RequestManager that the request cannot be accepted.

G.4 Provide data in an anonymous way, to protect users' privacy.

R.15 Deny third parties to receive single mode information about users that have not accepted the second part of the privacy policy.

R.17 Deny third parties to receive information if the provided information can hurt users' privacy, for this purpose group request under 1000 users involved are rejected.

G.5 Retrieve user's position and health status.

R.18 Allow users to be tracked from AutomatedSOS filling up the registration and agreeing only to first part of privacy policy.

- UserLogger: Receives from the SignAccount interface that user has filled all the fields on the registration page and has accepted the first part of privacy policy (the only one presented) then, if the user decides to create an account, it passes all these information to the WebInterface.
- WebServer: Receives the credentials from the WebInterface and provides them to the CredentialManager through CredentialInterface.
- CredentialManager: Checks the credentials inserted and if they are valid it stores them into the DB using DBInterface.
- DBMS: Stores data incoming from the DBInterface.

R.19 The application has to retrieve users' health status every 2 seconds in order to guarantee a reaction time of 5 seconds.

- HealthAnalyser: Exploiting the data acquired by GPSAcquirer and HSAcquirer, this component acquires health status parameters every 5 seconds and the GPS location every ten minutes like before.

G.6 Monitor user's health parameters.

R.19 The application has to retrieve users' health status every 2 seconds in order to guarantee a reaction time of 5 seconds.

- HealthAnalyser: Exploiting the data acquired by GPSAcquirer and HSAcquirer, this component acquires health status parameters every 5 seconds and the GPS location every ten minutes like before.

R.20 The application sends to Data4Help service all the data retrieved in live acquisition.

- DataSender: Exploiting the data acquired by GPSAcquirer and HSAcquirer, this component sends every ten minutes all the information acquired since the last message to the UserDataManager using AcquireData interface.

- UserDataManager: When a data update is arrived through the AcquireData interface, this component generates and sends an SQL query that allows the insertion of the new data into the DB.
- DBMS: Stores data incoming from the DBInterface.

R.21 The application gets from Data4Help service all the historical data about the user.

- HealthAnalyser: When the user requires it or when it is necessary to check the health parameters, this component sends a request to the WebServer in order to retrieve all the historical data about the user using WebInterface. From this point the sequence of action takes the standard procedure of managing requests (R.8 , R.10) until the WebServer.
- WebServer: Once the retrieved information is arrived to the WebServer, this component sends the package to the HealthAnalyser component.

R.22 Allow the user to set personal threshold values.

- UserInteractionManager: This component has to show to the users a human interface that allows them to personalize the threshold parameters of their health status.
- HealthAnalyser: Exploiting the service offered by UserInteractionManager, this component has to store (and to use them when necessary) all the settings generated by the user.

G.7 Send an ambulance to user's location whenever certain parameters are below the threshold.

R.23 The application has to control health status with data retrieved in local to immediately realize whether certain parameters are critical.

- HealthAnalyser: This component periodically checks the last parameters acquired and compares them to the historical ones and then decides with a special algorithm if the user needs an ambulance.

R.24 The application sends an ambulance request to the nearest hospital whenever parameters are critical.

R.25 Supply to the hospital the user's location and all the useful information to provide efficient first aid.

- HealthAnalyser: When the analysis finds something bad, this component sends an ambulance request to the nearest hospital specifying the user's location and health status.

R.26 In the case no answer arrives from the hospital the software must repeat another time the request until an answer in reached.

- HealthAnalyser: When an ambulance request is sent, this component actives a trigger that resend the request if an answer from the hospital is not received in time.

R.27 As soon as the acknowledgement message is received a warning message is displayed on the user's smartwatch.

- HealthAnalyser: When an answer is finally arrived from the hospital this component, exploiting UserInteractionManager, displays the ETA of the ambulance.

G.5 Retrieve user's position and health status.

R.3 Allow individuals to agree the privacy policy (first part) so that they can be tracked in group mode through installed application.

R.4 During the registration allow individuals to specify if they are also interested to be tracked in single mode (agree the second part of privacy policy) through installed application.

R.28 The application has to interact with Smartwatch/Smartphone APIs in order to retrieve GPS location with a resolution of 10 seconds when the user is in the run.

- DataSender: Exploiting the data acquired by GPSAcquirer and HSAcquirer, this component sends every ten seconds all the information acquired since the last message to the UserDataManager using AcquireData interface when the run is active.
- UserDataManager: When a data update is arrived through the AcquireData interface, this component generates and sends an SQL query the allows the insertion of the new data in the DB.
- DBMS: Stores data incoming from the DBInterface.

G.8 Allow user to manage a run.

R.29 Allow users to create a run once all the general information are inserted.

- RunManager: Once all the information above are submitted and once the name and the description of the run are also specified, this component provides all these information to the WebServer.
- WebServer: Receives from WebInterface the information about the new race and forwards it to the Track4RunManager component.
- Track4RunManager: Checks all the information about the new run that the user wants to create and if they are good, the component creates and sends an SQL query in order to store these information to the DBMS.
- DBMS: Stores data incoming from the DBInterface.

R.30 Allow promoters to define a path for the run by selecting the routes inside a map.

- RunManager: This component allows the promoters to specify, using ManageRun interface, the path of the race, displaying a map in which the promoter has to select the track.

R.31 Allow promoters to send a participation request.

- RunManager: This component allows the promoters to specify, using ManageRun interface, the athletes that the promoter wants to invite to the run.

R.32 Allow promoters to specify maximum number of athletes that can participate.

- RunManager: This component allows the promoters to specify, using ManageRun interface, the number of allowed athletes.

G.9 Allow athlete to enroll on a specific run.

R.33 Allow the user to see all the runs generated (which he is invited or not).

- RunManager: When user wants to see all the runs in the system, this component sends a request to the WebServer.
- WebServer: Receives from the WebInterface the request to see all the runs and forwards it to the RequestManager using AcceptInformationRequest interface.
- RequestManager: This component periodically checks the queue in order to select the most urgent request. Once an on-demand request is picked up it wraps the request and sends it to the DataCollector.
- DataCollector: Once a data request is received, this component builds a DB query that matches what the user wants and sends it to the DB using DBInterface.
- DBMS: When the query is arrived, this component computes it and provides data to the DataCollector using DBInterface, then the information goes in the opposite direction to the RunManager.
- RunManager: When all the runs are finally arrived, this component shows to the user all the race in which he can participate.

R.34 Allow user to enroll to a run.

- RunManager: When user wants to enroll to a run, this component retrieves through ManageRun interface which run he/she is interested in; then it provides this information to the WebServer.
- WebServer: Receives from the WebInterface the request to enroll to a run and forwards it to the Track4RunManager using RunInterface.
- Track4RunManager: This component has to create a query that asks to the DBMS if the interested user can participate to the specific run.
- DBMS: This component computes the received query and provides an answer to the DataCollector using DBInterface
- Track4RunManager: If the answer from DBMS is positive the component creates and sends another query that asks the DBMS to enroll the user, then it communicates it to WebServer; In the other case it communicates to the WebServer that the user cannot be enrolled.
- DBMS: Stores data incoming from DBInterface.
- WebServer: Receives from Track4RunManager the answer about the enrolling and forwards it to the RunManager through WebInterface.
- RunManager: When the answer is arrived, this component displays to the user the result of the enrollment using ManagerRun interface.

R.35 Deny user to enroll to a run if maximum number of participants is already reached.

G.10 **Allow spectators to watch in real time the position of every athletes in a specific run.**

R.33 Allow the user to see all the runs generated (which he is invited or not).

R.36 Allow user to select a run to be viewed.

- RunDisplayer: When user wants to spectate to a run, this component has to retrieve through the ShowLiveRun interface which run he/she is interested in; then it creates and sends a live acquisition request for the specific user to the WebServer.

R.37 The application requests to Data4Help the position of all the other athletes involved.

- WebServer: Receives from WebInterface the live acquisition request and forwards it ton the RequestManager using AcceptInformationRequest interface.
- RequestManager: Once a real-time request is arrived, this component stores all the attributes selected in order to make the subscription. Using a parallel thread the component automatically add into the priority queue the data request to supply the live acquisition.
- DataCollector: Once a data request is received this component builds a DB query in order to retrieve only new data that matches what user wants and it sends it to the DB using DBInterface.
- DBMS: When the query is received this component computes it and provides data to the DataCollector using DBInterface.

R.38 The application receives and displays the position of all the other athletes involved.

- DataCollector: Once data are arrived from DB, this component wraps them and sends this package to the RequestCollector using ProvideCollectedData interface.
- RequestCollector: This component receive information from DataCollector, unwraps and sends them using ProvideInfromation interface to the WebServer.
- WebServer: Provides to the RunDisplayer all the positions of the athletes involved in the race using WebInterface.
- RunDisplayer: Using the GoogleMapsAPI interface to compute the live positions, this component offers to the user a live map that specifies all the athletes' position inside the race path.

5 Implementation, Integration and Test Plan

5.1 Implementation Plan

The TrackMe system will be implemented component by component, prioritizing the development of the most critical ones. Another key element on which the order of the implementation is decided is the level of dependency between components. This means that if multiple components depend on a single one then the development of the latter is prioritized upon the others, this way we can start integrating components as soon as they get developed and tested. Also it is considered good practice to concentrate first on the model part of the system, then to concentrate on the controller and at last on the view. For the previous reasons a Top-Down development approach has been selected.

The following tree contains components as nodes and it is useful for describing the order of implementation from top to bottom following this rule: a node is implemented before its children. Therefore, the plan starts implementing the Database and the WebServer components in parallel.

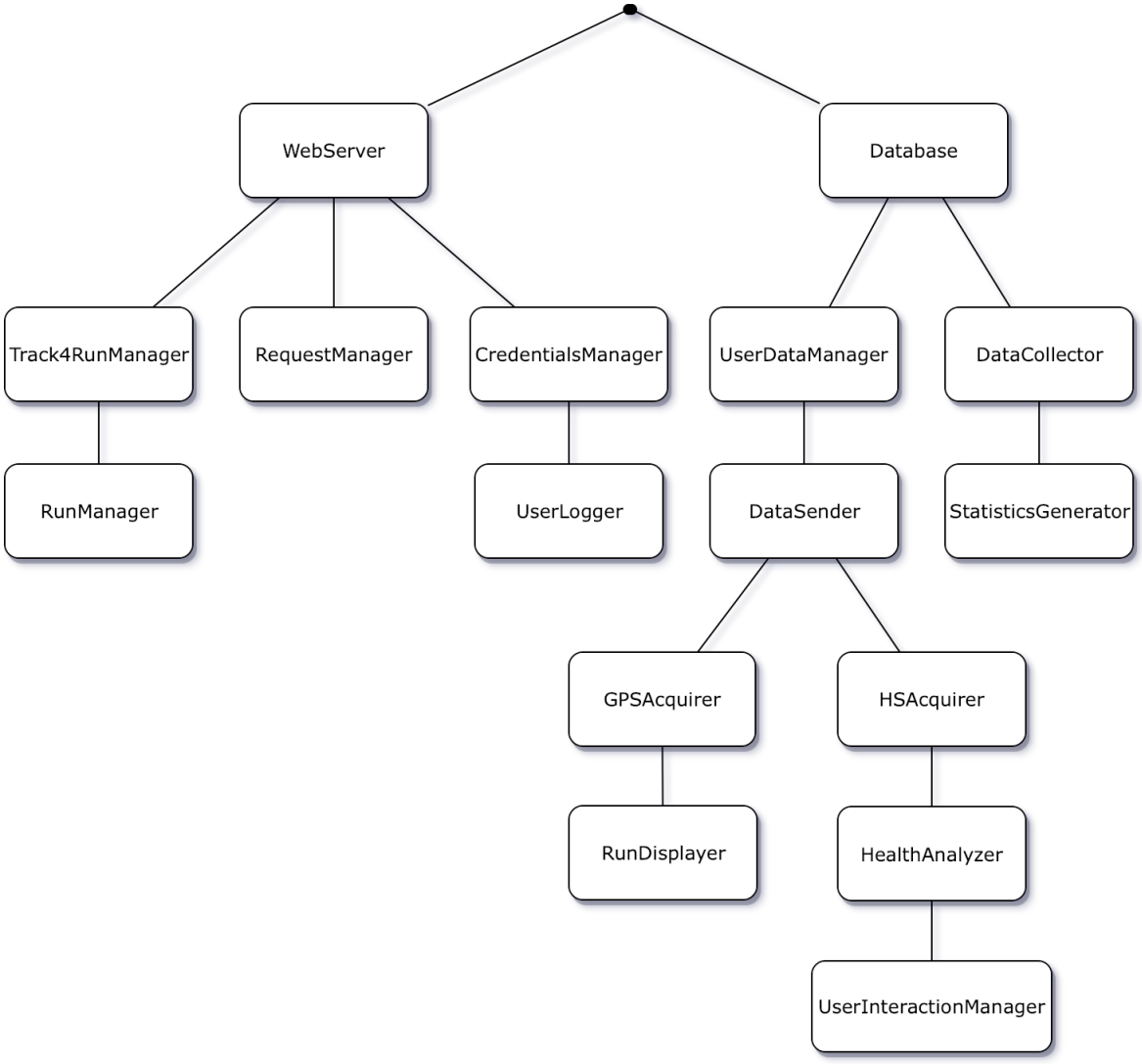

Figure 26: Implementation Tree

One of the first components that will be implemented is the Database due to the fact that almost every component of the system will exploit Database's features, also it is one of the most critical parts of the

system, if it fails then all our system will fail, and lastly, it forms the model of our system.

The other component that will be implemented in parallel to the Database is the WebServer: it is one of the most critical element of our system as well, due to the fact that the third parties and the users will exploit this component to carry out every functionality that TrackMe offers.

Then the components belonging to the second level of the tree will be developed since they form a big part of the business logic of our system and most of them manage information requests.

As soon as the DataSender is completed, GPSAcquirer and HSAcquirer will be developed, since they interact directly with DataSender in order to store the retrieved data into the Database (already developed in the previous step). Then, once HSAcquirer is completed, the last two components of AutomatedSOS will be developed, starting with HealthAnalyzer, since it contains part of the logic of our system (in order to have faster reaction time), moreover, this component performs a critical functionality: calling an ambulance whenever necessary. Finally, the last component of this branch to be developed will be UserIntercationManager.

5.2 Integration and Test Plan

Every component of the system must be subjected to Unit Testing during its development in order to identify as soon as possible any problem within the component. This way, before actually preceding to any Integration Test, we are pretty sure that all components work well in isolation.

As said in the previous subsection, the Integration Plan will also follow a Top-Down approach. More precisely, an *Incremental Integration and Testing* approach is taken, since it couples really well with the dependency priority implementation order. This way, as soon as components that depend on others are released, they can be integrated and tested immediately, allowing us to discover integration problems as soon as possible.

5.2.1 Integration Diagrams

The following diagrams show the components to which the element just developed and already subjected to Unit Test is integrated with. The element just created is represented by a blue square, the grey ones on the upper lever represent already developed and unit tested components, while the grey ones on the same level represent components that could or could not be already developed and unit tested. In these diagrams, if there are more than one grey square, it does not mean that all the showed elements are integrated all together, but it means that they will be integrated just one by one, coupled with the one just developed or that will be developed soon.

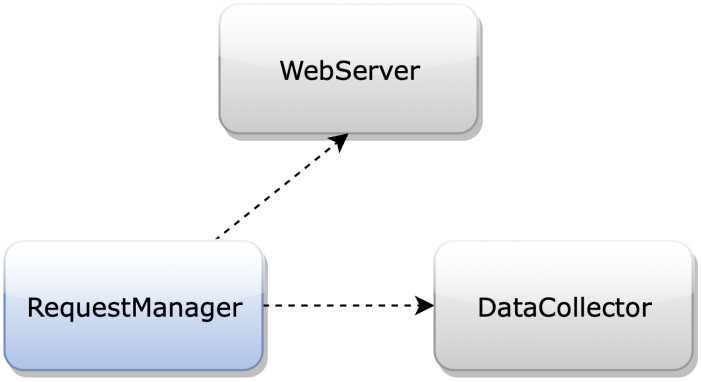

Figure 27: RequestManager Integration

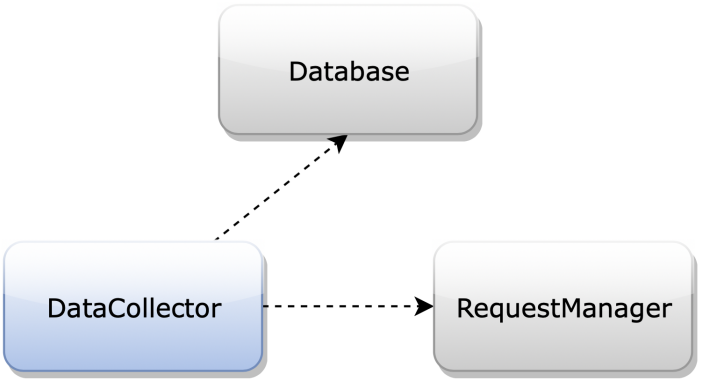

Figure 28: DataCollector Integration

In the two diagrams above, RequestManager and DataCollector components are coupled twice, this means that the last component actually developed among them two will be integrated with the other one.

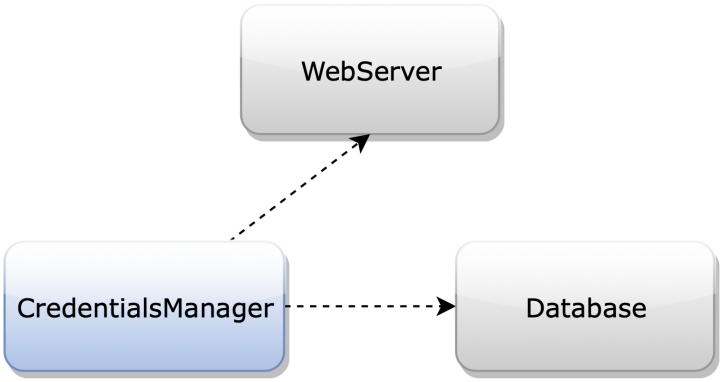

Figure 29: CredentialManager Integration

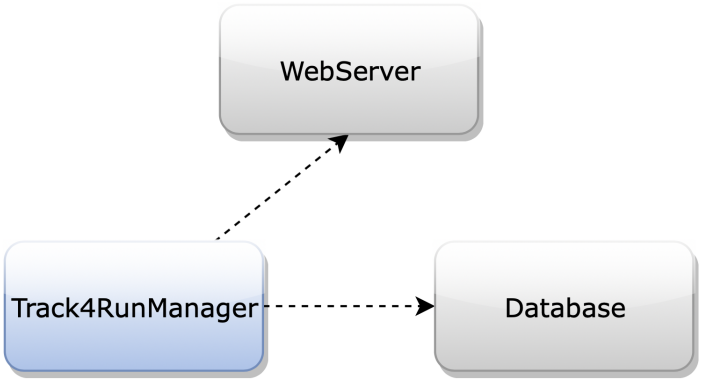

Figure 30: Track4RunManager Integration

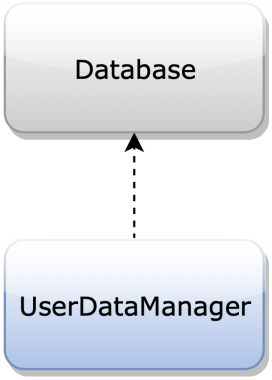

Figure 31: UserDataManager Integration

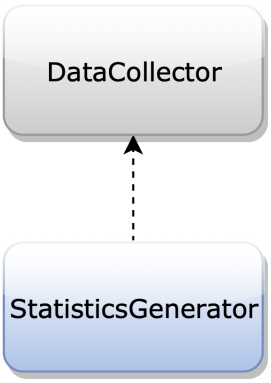

Figure 32: StatisticsGenerator Integration

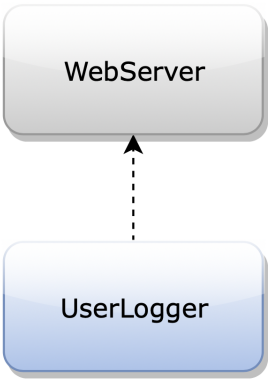

Figure 33: UserLogger Integration

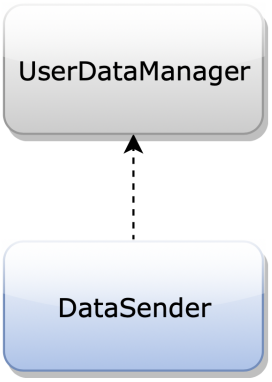

Figure 34: DataSender Integration

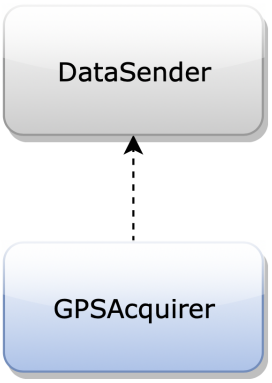

Figure 35: GPSAcquirer Integration

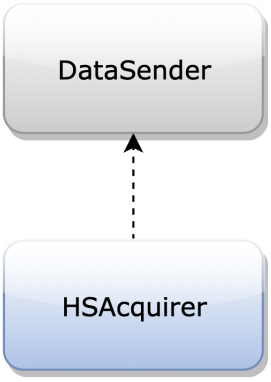

Figure 36: HSAcquirer Integration

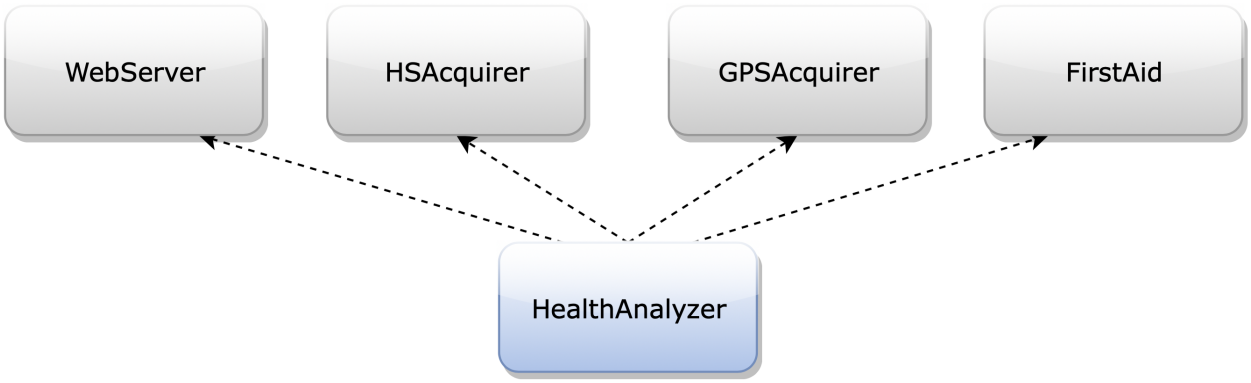

Figure 37: HealthAnalyzer Integration

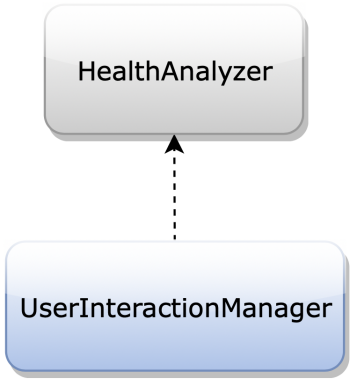

Figure 38: UserInterfaceManager Integration

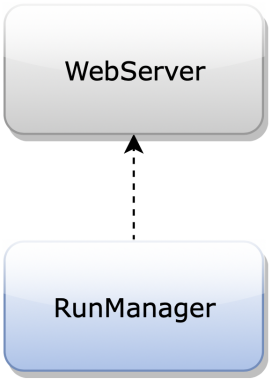

Figure 39: RunManager Integration

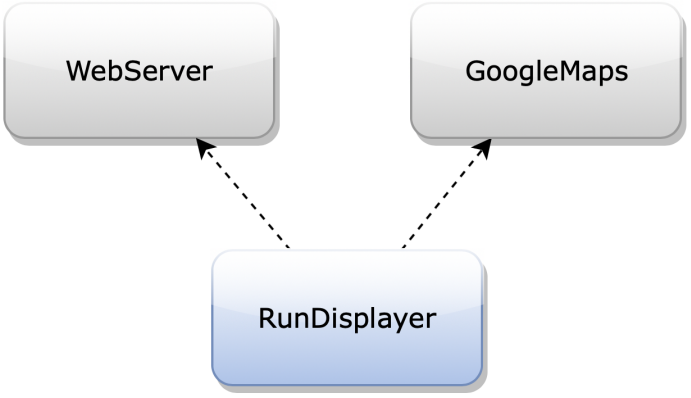

Figure 40: RunDisplayer Integration

5.2.2 Third Party Request Integration

In the following two figures, the integration including more than two components at the same time are shown. More precisely, they explain the order in which components are integrated together in order to test the complete functionality of third parties' requests. As we stated before we adopted an *Incremental Integration and Testing* approach and, therefore, we add to the integration one component at a time until we got the full chain of components which should perform the complete functionality.

Figure 41: ThirdPartyRequest semi-chain integration

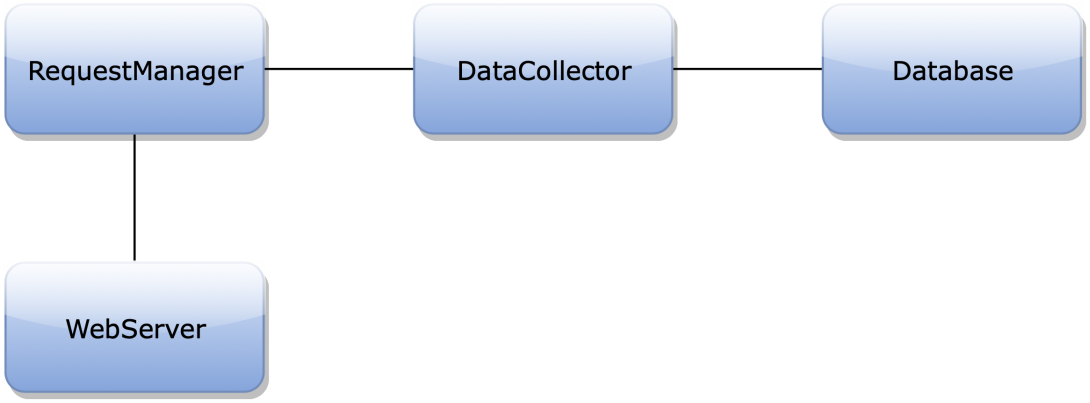

Figure 42: ThirdPartyRequest full-chain integration

5.2.3 Run Event Storing Integration

The next two figures shows integration including more than two components at the same time. More precisely, they explain the order in which components are integrated together in order to test the complete functionality of storing into the Database all the information regarding run events.

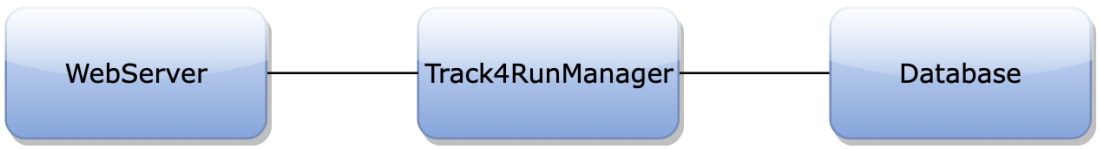

Figure 43: Run event storing semi-chain integration

Figure 44: Run event storing full-chain integration

5.3 System Testing

Once the system is integrated completely and once it has an acceptable performance it will be tested as a whole. This due to the fact that in the future the system could possibly be subject to heavy workloads. System Testing will identify the following problems:

- Inefficient algorithms: it's really important to have efficient algorithms implemented in the system, especially for the RequestManager, that manages the requests deciding which one is more urgent, and especially for the HealthAnalyzer which needs to have very small reaction time in order to call the ambulance as soon as possible.
- Query optimization problems: taking into account that the system's functionality heavily relies on queries, testing on DB query optimization is for sure a good test to implement.
- Hardware issues: taking into account that the system relies on some piece of hardware that our system cannot directly test, such as the GPS and HS hardware retrievers, it can be important to check if the data retrieved are subjected to relevant noise.
- Network issues: it's really important to check how much request load the Web Server can withstand and if necessary to expand the connection broadband.

More precisely two types of System Testing will be done:

Load Testing $\rightarrow$ increasing the load of the system for a long period of time in order to find and solve memory leaks, buffer overflows, bad memory management and also identify upper limits of components.

Stress Testing $\rightarrow$ trying to break the system by overwhelming its resources or by taking resources away from it will make sure that the system recovers gracefully after failure, this is very important for our system especially for AutomatedSOS components which are in charge of giving real time assistance to users that suffer from health issues.

6 Effort Spent

This section contains information about how many hours each group member has spent in working at this document.

6.0.1 Luca Alessandrelli

Date	Task	Hours
23/11/18	Overview	1
24/11/18	Deployment View	3
26/11/18	Deployment View	1
27/11/18	Runtime View	6
29/11/18	Component Interface	3
30/11/18	Runtime View	2.5
1/12/18	Component Diagram	2
1/12/18	Runtime View	1.5
2/12/18	Deployment View	0.5
2/12/18	Component Diagram	1
3/12/18	Implementation,Integration and Test Plan	5
4/12/18	Implementation,Integration and Test Plan	7.5
7/12/18	Document Revision	3
10/12/18	Document Revision	5
Overview		1
Deployment View		4.5
Runtime View		10
Component Interface		3
Component Diagram		3
Implementation,Integration and Test Plan		12.5
Document Revision		8
Total		42

Table 3: Effort Spent Luca Alessandrelli

6.0.2 Andrea Caraffa

Date	Task	Hours
26/11/18	User Interface Design	4
27/11/18	Runtime View	3
1/12/18	Runtime View	3
1/12/18	Component Interfaces	2
2/12/18	Runtime View	2
2/12/18	Component Interfaces	2
3/12/18	Runtime View	1
3/12/18	Component Interfaces	1
4/12/18	User Interface Design	4
7/12/18	Implementation Plan	1
7/12/18	Integration Plan	1
7/12/18	Introduction Section	3
8/12/18	User Interface Design	2
8/12/18	Document Revision	6
9/12/18	Document Revision	4
10/12/18	Document Revision	4
User Interface Design		10
Runtime View		8
Component Interfaces		5
Implementation Plan		1
Integration Plan		1
Introduction Section		3
Document Revision		14
Total		42

Table 4: Effort Spent Andrea Caraffa

6.0.3 Andrea Bionda

/	Task	Hours
Overview		3
Component Diagram and description		15
ER Diagram and description		3
Deployment View		1
Component interface description		1
Architectural styles and patterns		5
Requirements traceability		10
System Testing		1
Document revision		1
Total		40

Table 5: Effort Spent Andrea Bionda

7 Reference Documents

- Specification Document "Mandatory Project Assignment AY 2018-2019".
- Slides about "Design".
- Slides about "Implementation, Integration and Test Plan".
- Sequence Diagrams created with https://www.nomagic.com/products/magicdraw
- Mockups created with https://www.fluidui.com and https://www.simbla.com
- Component Diagram created with https://online.visual-paradigm.com
- Other Diagrams created with https://www.draw.io