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:	23-November-2018
Download page:	https://github.com/lucaalessandrelli/AlessandrelliCaraffaBionda.git
Copyright:	

Contents

Table of Contents **3**

1 Introduction **4**
1.1 Purpose 4
1.2 Scope 4
1.3 Definitions, Acronyms, Abbreviations 4
1.4 Revision History 4
1.5 Document Structure 4

2 Architectural Design **5**
2.1 Overview 5
2.2 Component View 6
2.2.1 Component Diagram 6
2.2.2 Entity Relationship Diagram 9
2.3 Deployment View 10
2.4 Runtime View 11
2.4.1 Individual Information Request 11
2.4.2 Send Ambulance Request 12
2.4.3 Spectate to a Run 13
2.5 Component Interfaces 14
2.6 Selected Architectural Styles and Patterns 16
2.7 Other Design Decisions 17

3 User Interface Design **18**

4 Requirements Traceability **28**

5 Implementation, Integration and Test Plan **34**
5.1 Implementation Plan 34
5.2 Integration and Test Plan 35
5.2.1 Integration Diagrams 35
5.2.2 Third Party Request Integration 38
5.2.3 Run Event Storing Integration 39
5.2.4 Non so cosa mettere 40
5.3 System Testing 40

6 Effort Spent **42**
6.0.1 Luca Alessandrelli 42
6.0.2 Andrea Caraffa 43
6.0.3 Andrea Bionda 44

7 Reference Documents **45**

1 Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, Abbreviations

1.4 Revision History

1.5 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.

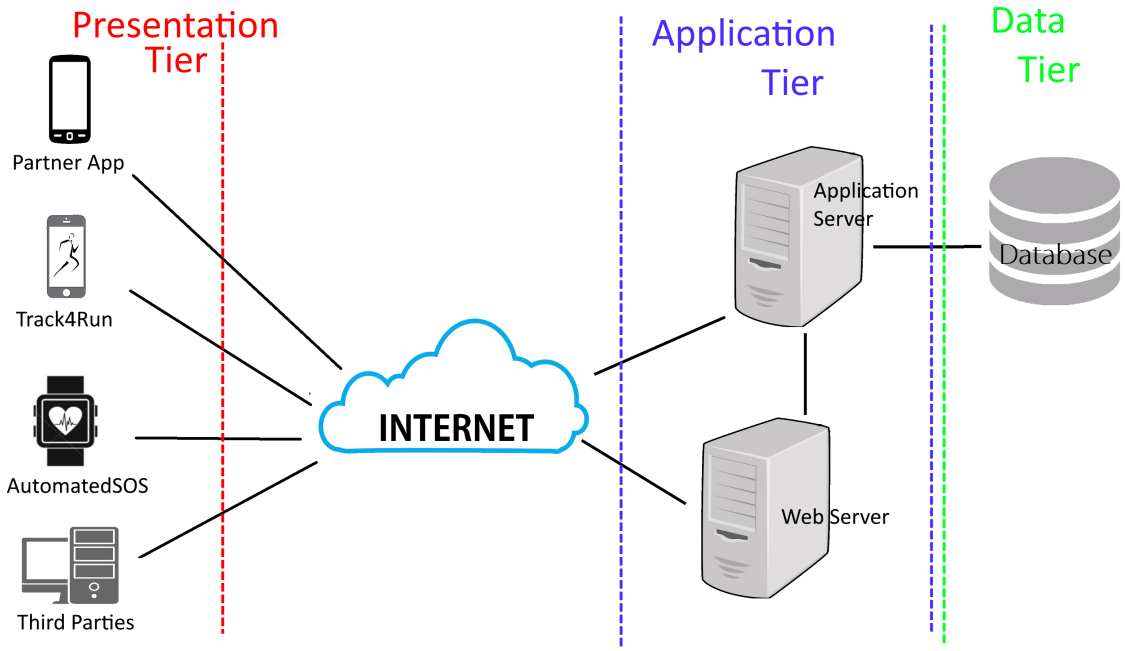

Figure 1: Overview architecture

- **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 a easily way to understand them. This layer regards both the interaction with third tarties and with individuals.

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

  - storing data incoming from the external;
  - collecting data information from database in order to execute third parties' requests;
  - generating data statistics on collected data;
  - sending to third parties requested data.
  - managing run events, creating them and enrolling athletes to them.

  Moreover, even AutomatedSOS has logic application in order to continuously monitor users' health status.

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

More precisely, Data4Help manages the data and core logic sections while AutomatedSOS and Track4Run manage the presentation section. Actually, 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

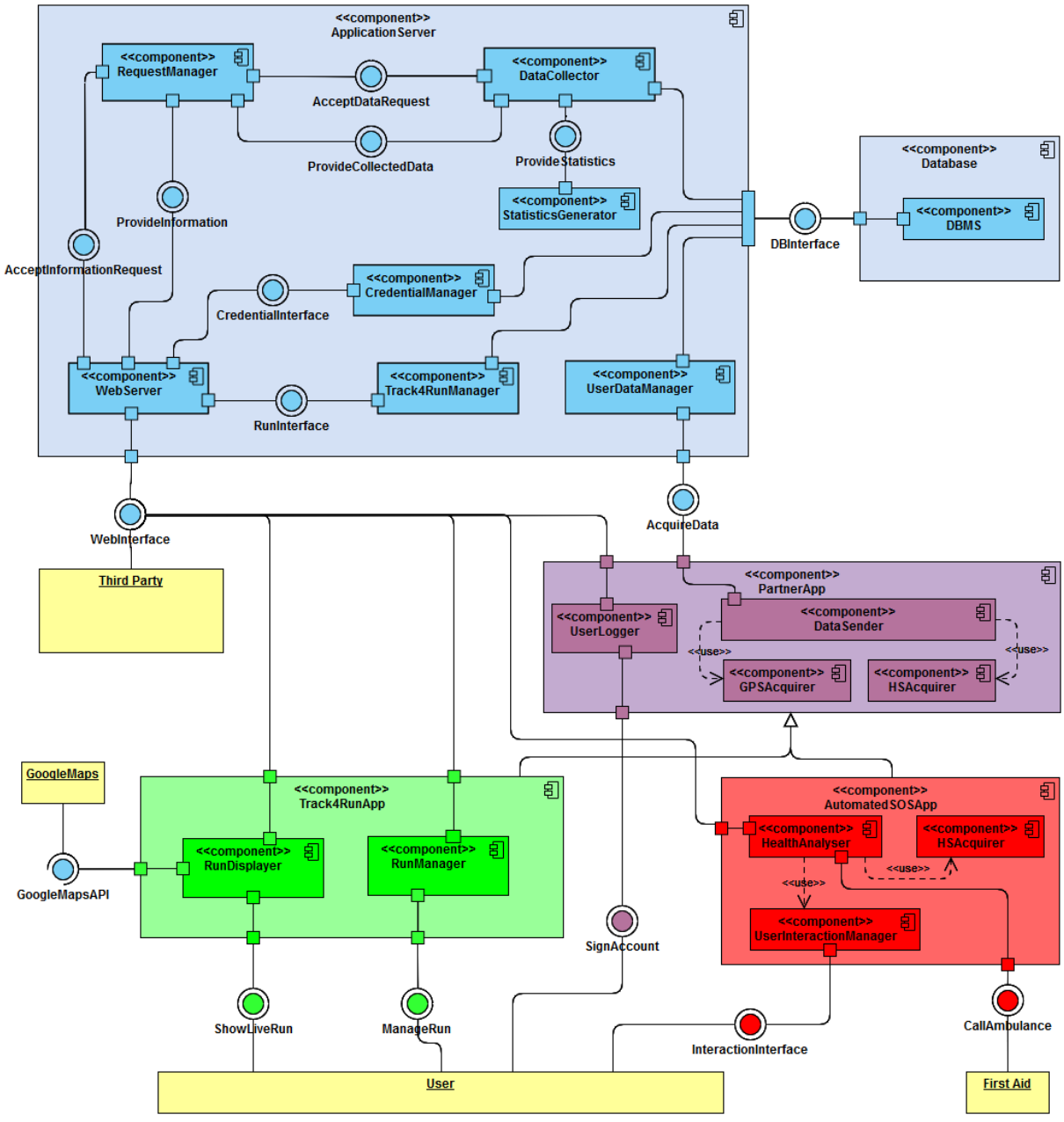

Figure 2: Component Diagram

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 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 Data Collector 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 statistic 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 users' 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 **CredentialManager:** 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 races' 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 Main Server. 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 all that is 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 call the First Aid whenever it 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 call an ambulance whenever such parameters are below a certain threshold 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 get a feedback message whenever an ambulance has been sent to his/her location.

5 **Track4RunApp:** This component extends all that is specified in the PartnerApp component to exploit all its features. Track4Run component will also provide to users the possibility to select and spectate a run, allow promoters to create and manage a run and last allow athletes to enroll to a given run.

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

5.2 **RunManager:** This internal 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 fiscal code.

2.2.2 Entity Relationship Diagram

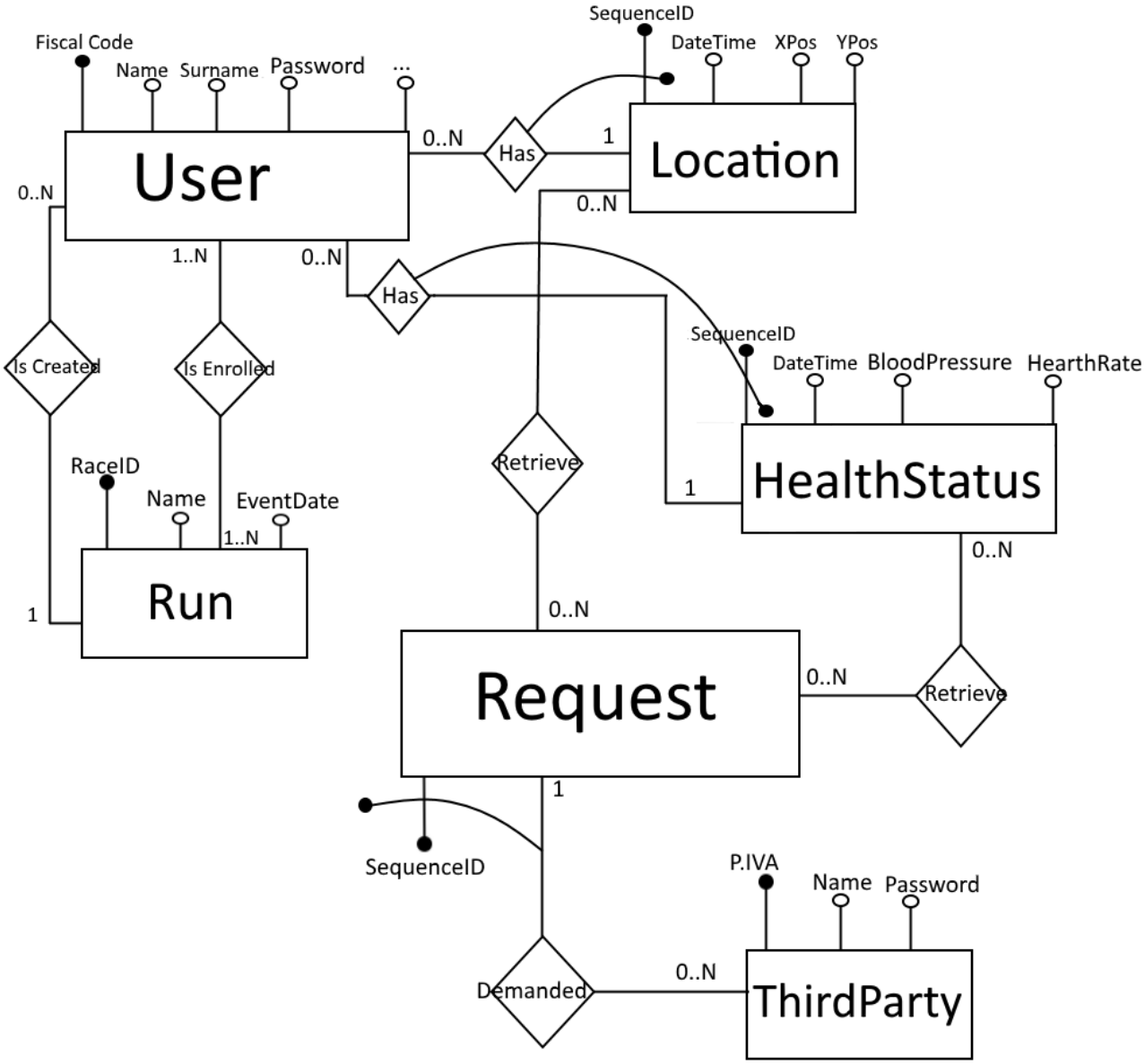

Figure 3: Entity Relationship Diagram Diagram.

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 websites in order to improve portability between 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.

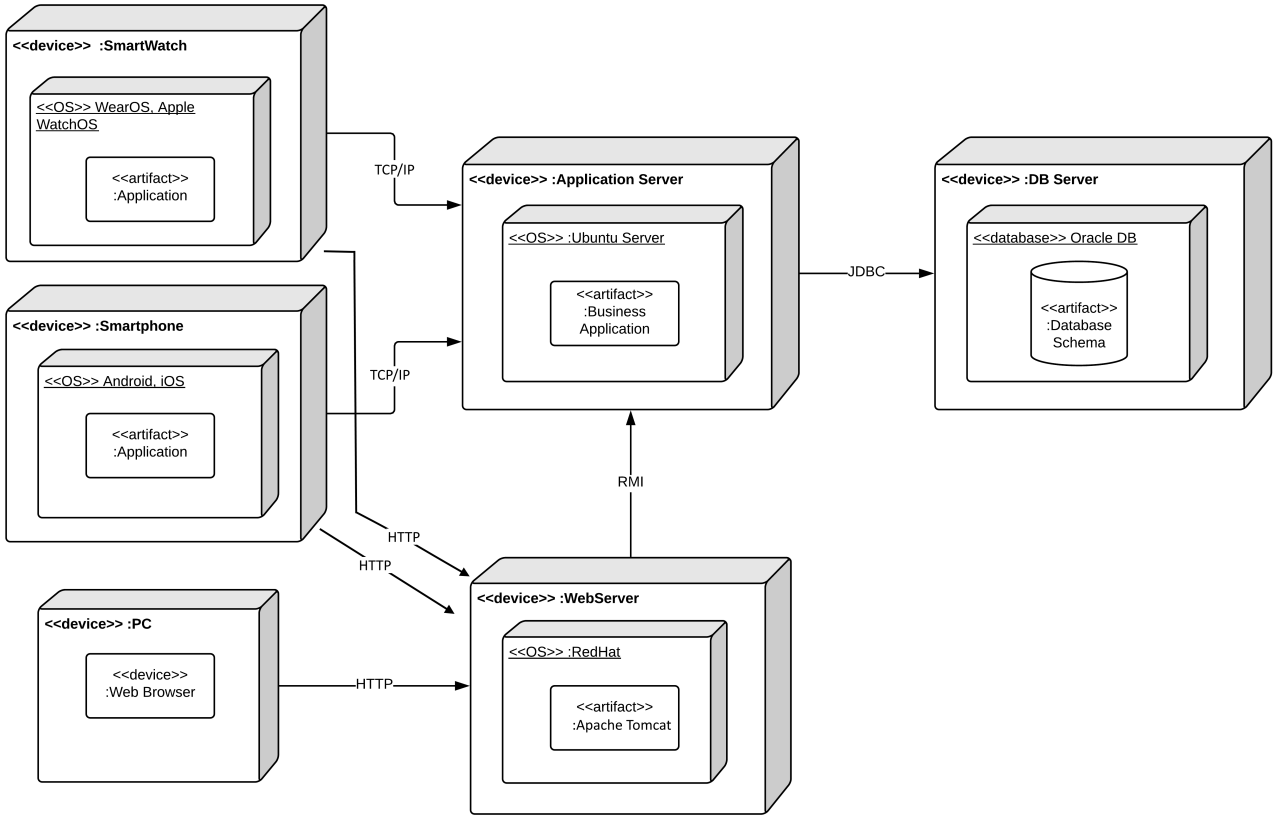

Figure 4: Deployment Diagram.

2.4 Runtime View

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

2.4.1 Individual Information Request

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

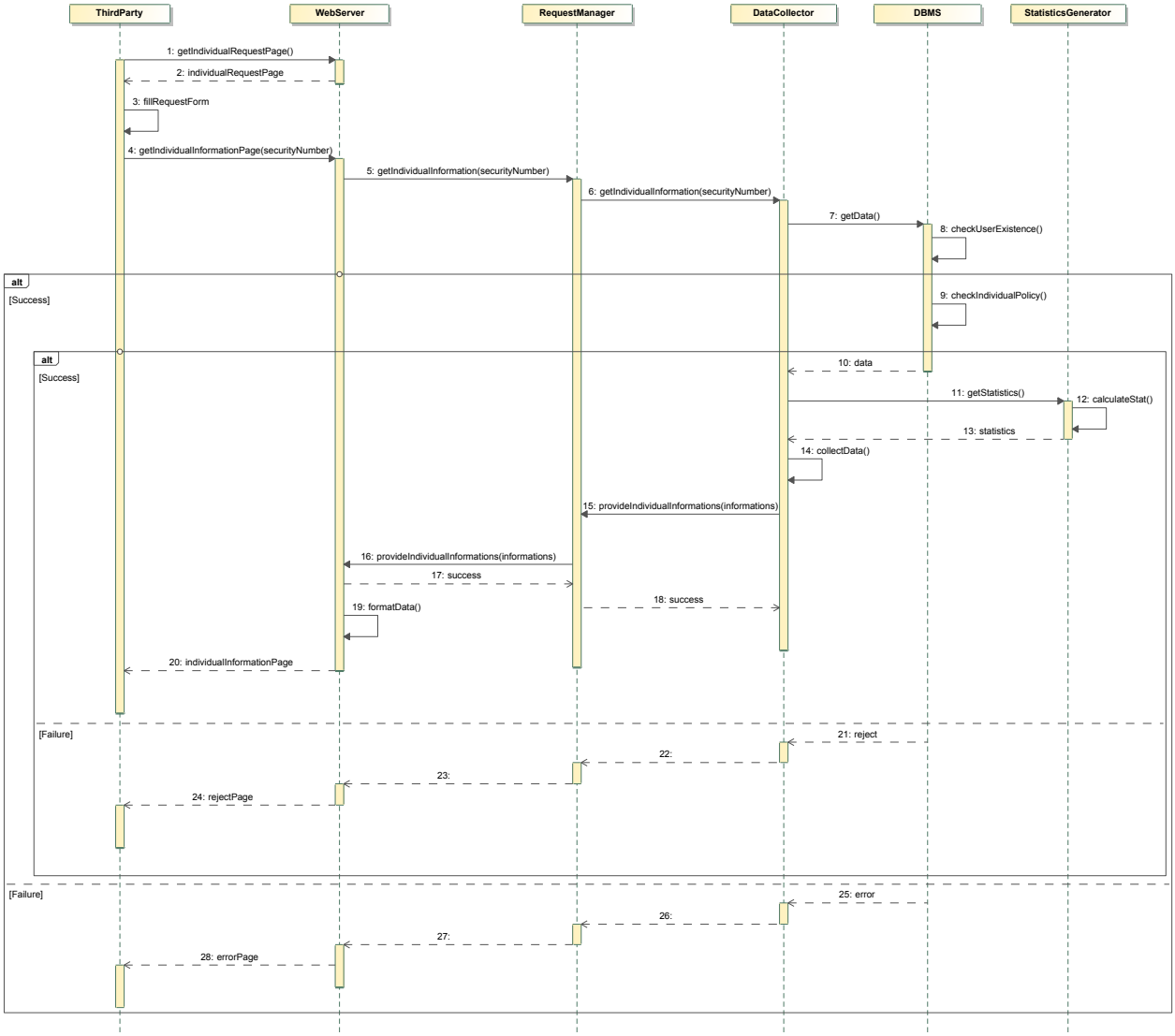

Figure 5: Sequence Diagram n.1.

2.4.2 Send Ambulance Request

In the below diagram is shown the interaction that the components have between each other to send an ambulance request to the first aid. The data retrieved by the GPS Acquirer and HealthStatus Acquirer both sent to Data4Help and sent to the HealthAnalyzer.

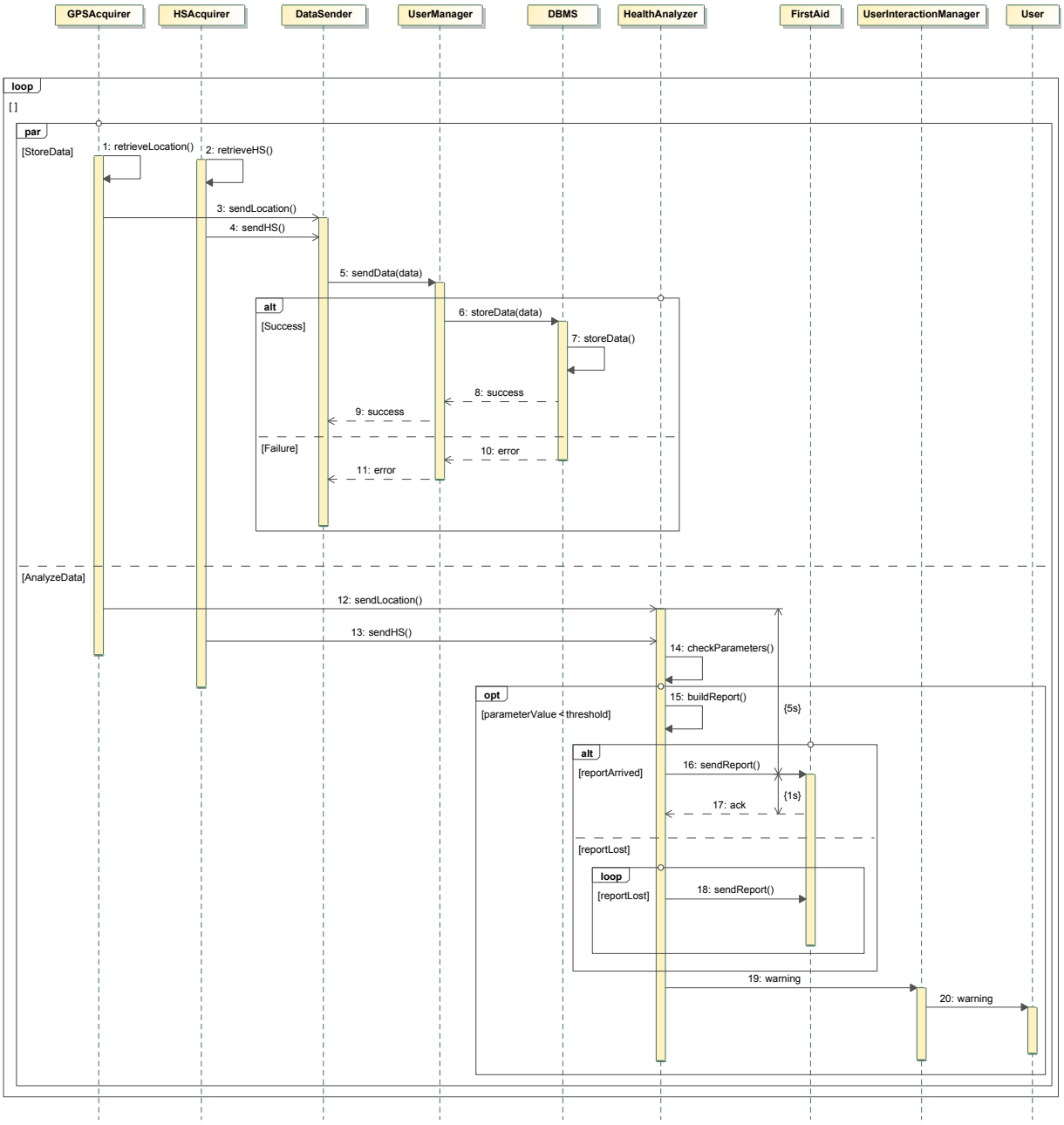

Figure 6: Sequence Diagram n.2.

2.4.3 Spectate to a Run

The next Sequence Diagram represents the interaction that the components have between each other 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.

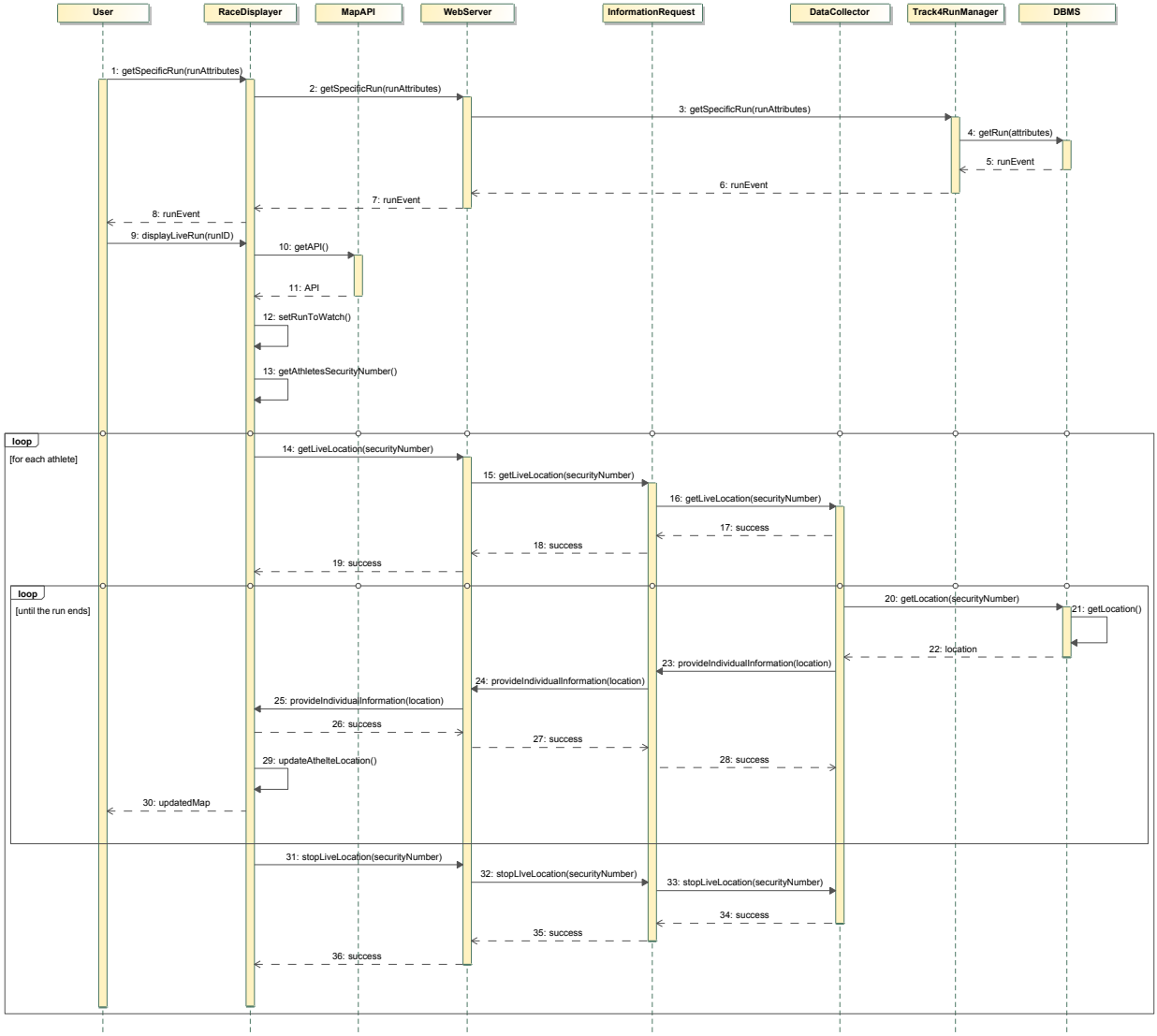

Figure 7: Sequence Diagram n.3.

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.

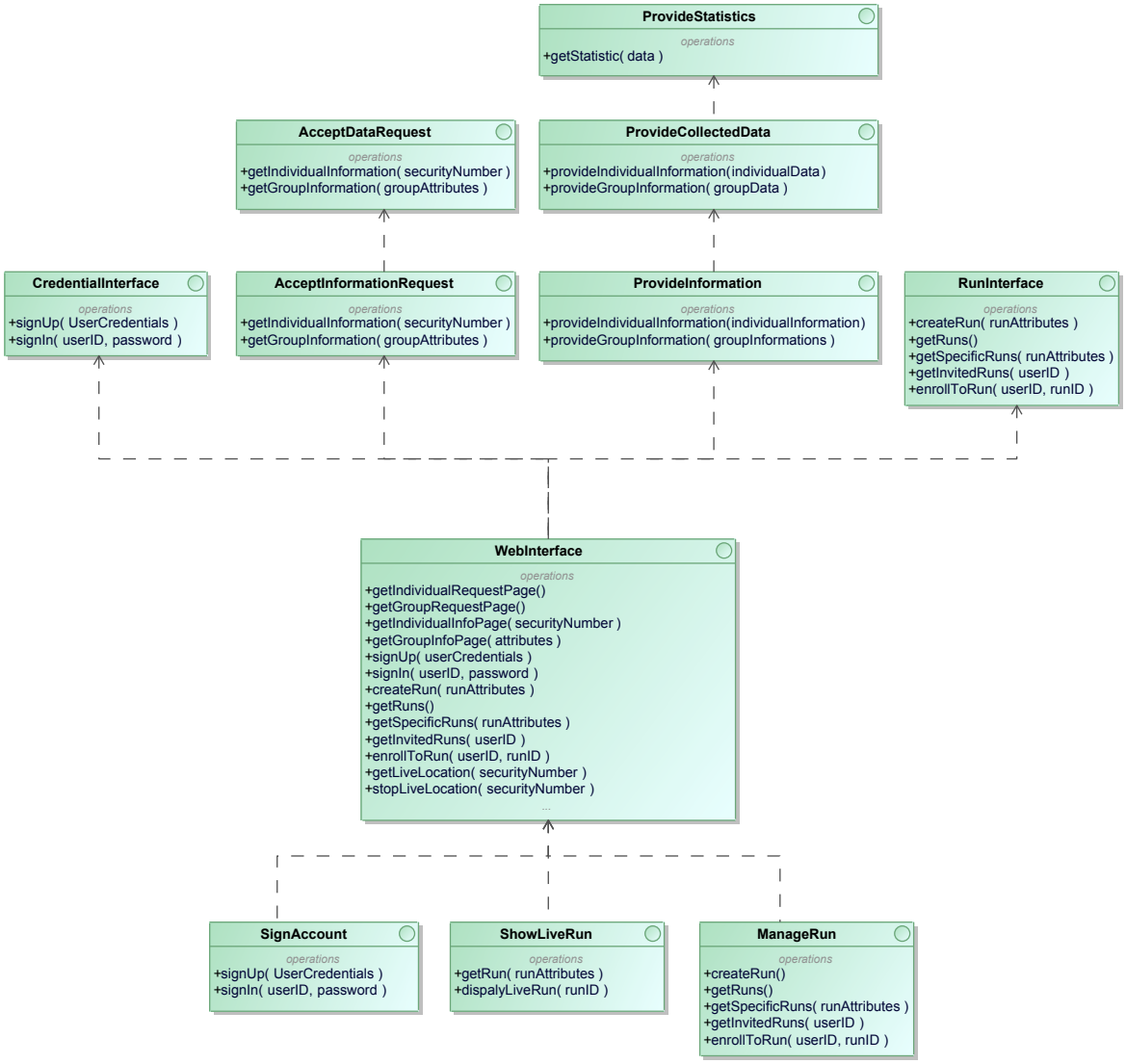

Figure 8: Component Interfaces Diagram

1.1 WebServer

1.1.1 **WebInterface:** Interface for third parties that want interact with the system in order to obtain informations from Data4Help service. Interface for Track4Run and for AutomatedSOS in order to perform the right tasks.

1.2 RequestManager

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

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

1.3 **DataCollector**

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

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

1.4 **StatisticGenerator**

1.4.1 **ProvideStatistics:** Interface that provides statistics to DatCollector.

1.5 **UserDataManager**

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

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 CredentialsManager 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 informations 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.

4.1 **HealthAnalyser ———— TO SEE**

4.1.1 **CallAmbulance:** Software interface that allows AutomatedSOS to call, through the internet or in the worst case through telephone call, First Aid whenever is necessary in order to provide help to user. Moreover it provides to FirstAid a special report on which parameters are critical.

4.2 **UserInteractionManager**

2.2.1 **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 display the warning message into the smartwatch display.

5.1 **RunDisplayer**

5.1.1 **ShowLiveRun:** Interface that allows to shows 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 to a user to create and manage a run providing all the useful information. This interface allows also to 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 tier that well separate the different components as shown in Overview description in order to implement MVC software design pattern.

- **Architectural Design:**

  - Client-Server architectural pattern is chosen because it is the most used and easily implemented communication pattern in fact, in our application, is the best one to supply on third parties information requests and to acquire data from users via internet. Regarding the acquisition of users' data, is the smartphone that decides, when data are successfully collected from the device, to call Data4Help server in order to provide and store data, furthermore even third parties call Data4Help server first and then the exchange of the information can be proceed; these two operations are the main core of the service then is logic that client-server architecture will be the right choice.
  - Three tier architecture should be implement to assure reuse and maintainability of the system, in fact this division can permit to change some parts of the system without reconfigure the entire solution.
    * Presentation Tier, as described above, is in charge of display on final users screen a human interface that allows the interaction with our system. This tier is present only on users' device: regarding third parties the software that is in charge to display human interface will be his own browser so the exchanged information will be an HTML page on 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 to the Application Tier is the WebServer that allocate 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 to acquire all the data incoming from users, to store them on Data Tier and to handle the request on viewing data. This Tier is present on the largest part on Application server but even AutomatedSOS has inside his software an 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 Presentation Tier, acquire data that users submit and communicate through RMI with the business application that runs on the Application Server. The business application is the main core of Data4Help service, and is composed by all the other components inside ApplicationServer (as Request Manager,Data Collector,Statistic Generator..); is required that it can interface with Web server, in order to acquire the external requests, and then using the components described above supply the information required; furthermore is in charge to manage all the information of users, as login or registration,or manages 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 partner application in order to acquire users' data. As mentioned before even AutomatedSOS implements a small part of application tier inside his software that is in charge to continuously monitor health status acquired, compare it with old data and call an ambulance via web if there is a necessity filling out a form which are indicated all the critical parameters detected.

    * Data Tier, as described above, is in charge to store and provide informations. This tier is present only on 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 that stores races' information. This Tier offers an SQL interface that can be exploit 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 three tier architecture because can split the entire software solution in the three main region: the Model implements all that is described in data tier, the controller all that is described on Presentation tier and View all that is described in Presentation tier. All these 3 groups communicates each others through the so called Adapter Pattern.
  - The Adapter design pattern is developed to create an interface that permits to the MVC groups to easily communicate each others using different languages or different operating system that is mandatory in our system. This pattern is the representation of what is described on Component Interface section.
  - Strategy design pattern can be so much useful in the implementation of RequestManager component because it can dynamically change how to handle request queue in order to supply efficiently on heavy request load.
  - Proxy design pattern is an obvious consequence of Web Server, because this component stay in the middle between the two tier and creates a sort mask that obscure what is behind it and simplify a lot the Presentation Tier software that has only to display what Web server indicates

2.7 Other Design Decisions

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 systems, and how the mockups are correlated each other.

- Data4Help

  Third parties can interact with Data4Help system through a Website which what they can made the several kind of requests.

Figure 9: Data4Help's homepage

This is the homepage of Data4Help's website, reachable from the user's browser by the link *www.data4help.com*. In the homepage it's possible to read a brief overviwe of the service offerd by Track4Me. Several actions are possible from here click on the different buttons present in the head **HOME**, **ABOUT**, **SERVICES**, **MADE NEW REQUEST**, and since the user is already logged into thesystem also the options **MY REQUESTS** and **LOG OUT** are available.Besides the options selectable into the header, scrolling down the page it is possible to select a button to directly made a monitoring request.

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

Figure 10: Individual request form

Clicking on **MADE 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 made a 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

Otherwise, clicking on **GROUP MONITORING REQUEST** button the system loads a new page containing the form to made 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's then possible to select a minimum and a maximum age. Click one the **More options** button it's possible to limit furthermore the request adding other filters to the group of individuals. Once the user has fill al the fields, clicking on the **SUBMIT** button the request is forward 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.

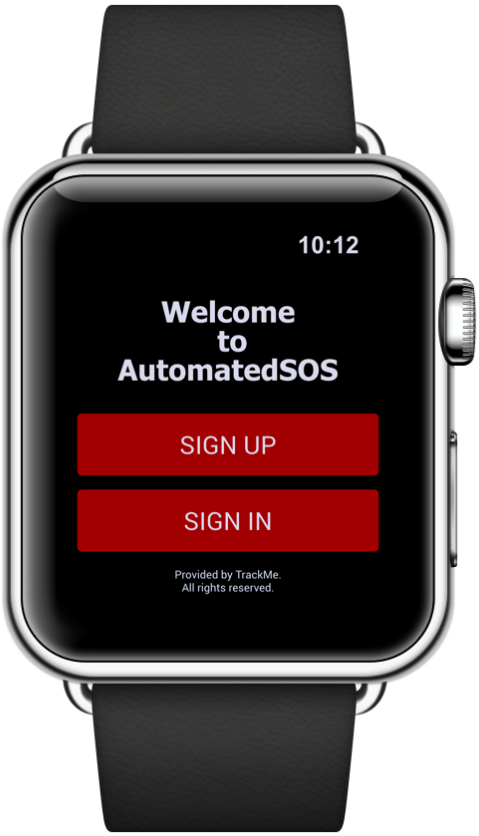

Figure 12: Welcome page.

Figure 13: Registration form.

In the first access to AutomatedSOS App the user is asked to sign up or to sign in (Figure 2). Based on the fact that the user already has a TrackMe account or not will choose the right option. 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 3), 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 would be possible to select the **NEXT PAGE** button.

Figure 14: Privacy Policy Conditions.

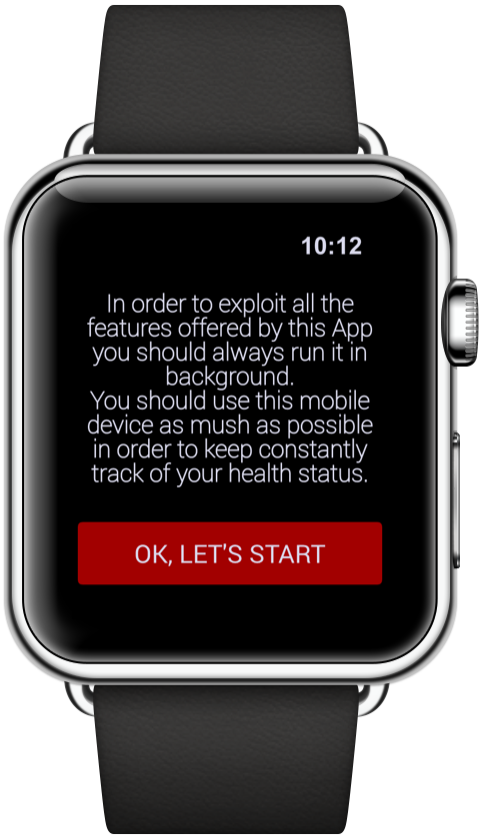

Figure 15: Usage Conditions.

During the first access to the App, the next step after sign up/sign in is to agree to the Privacy Policy Conditions (Figure 4). 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. The last step before starting use the App is taking note of the importance of wearing the Smartwatch as much as possible and to let the App runs in background (Figure 5).

Figure 16: Main menu.

Figure 17: Warning message.

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 6). Selecting **Monitor Health** the user can see live health informations, 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 allow to the user to set own threshold parameters according to the particular illnesses he/she is affected, own age and so on. As soon as an ambulance request is done due to the user's critical health status a warning message (Figure 7) 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.

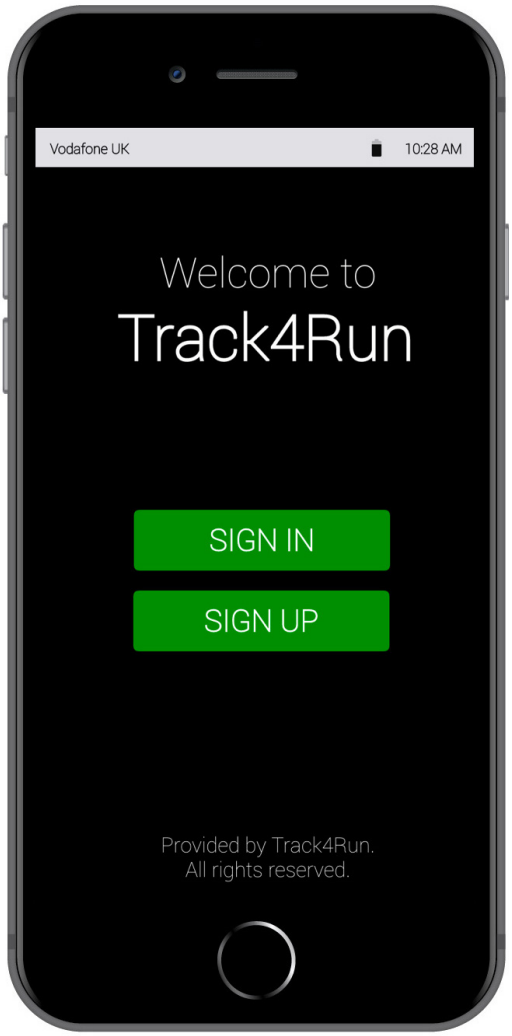

Figure 18: Welcome page.

Figure 19: Registration form.

In the first access to Track4Run App the user is asked to sign up or to sign in (Figure 2). Based on the fact that the user already has a TrackMe account or not will choose the right option. 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 3), the user has to fill all the mandatory fields in order to complete the registration form. Once every field is correctly filled, it would be possible to complete the registration selecting the **Register Now** button.

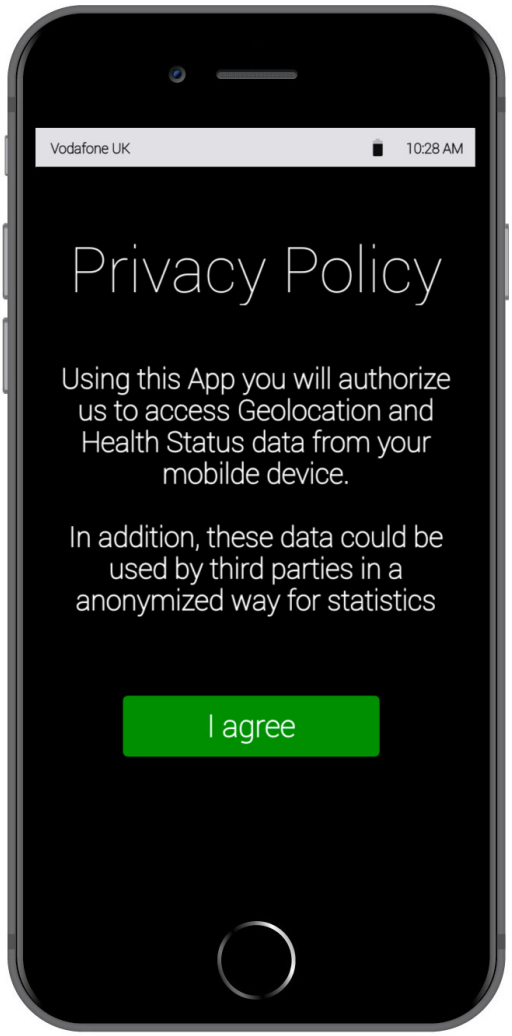

Figure 20: Privacy Policy Conditions pt.1

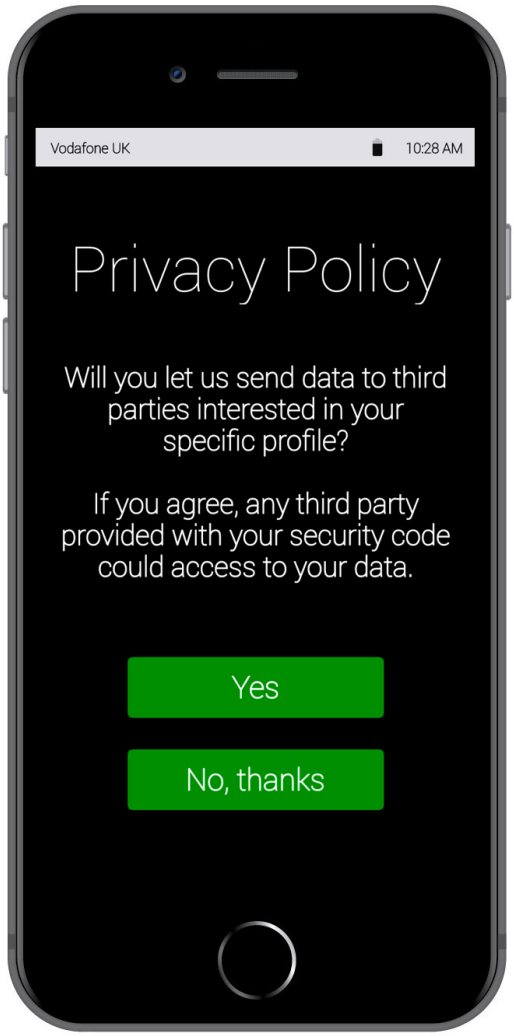

Figure 21: Privacy Policy Conditions pt.2

During the first access to the App, the next step after sign up/sign in is to agree to the Privacy Policy Conditions (Figure 4). 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 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 9). In this case it is not strictly necessary to accept it, the user simply can select the **No, thanks** option.

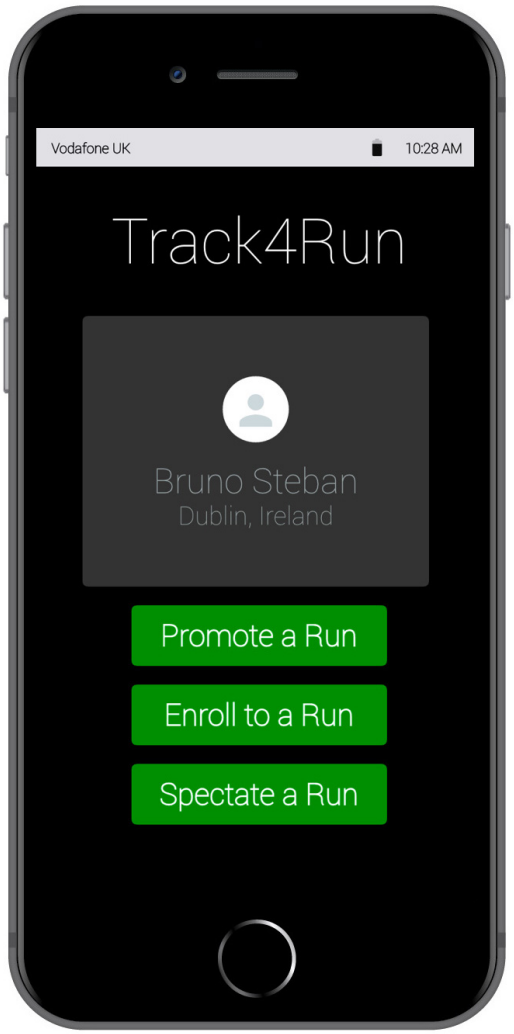

Figure 22: Main menu.

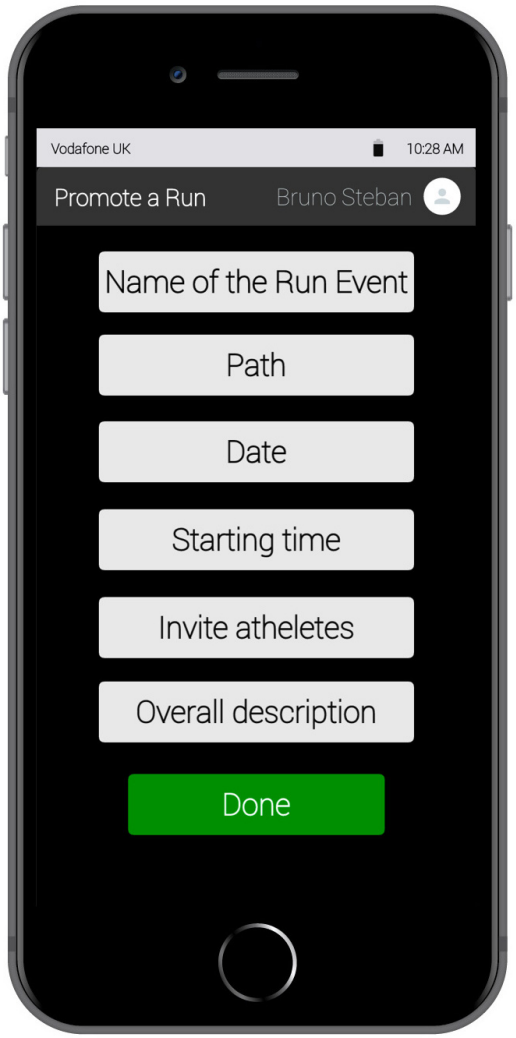

Figure 23: Pomote a run view.

In Track4Run Home three options are possible. 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 enrol to a run. Finally, choosing **Spectate a Run** a spectator can see on a map the position of all runners 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 X). Here, it is possible to set all the necessary features of the run, like to define the path on a map, set the date, the starting time and so on. It is also possible to invite athletes to the run in order to promote the event.

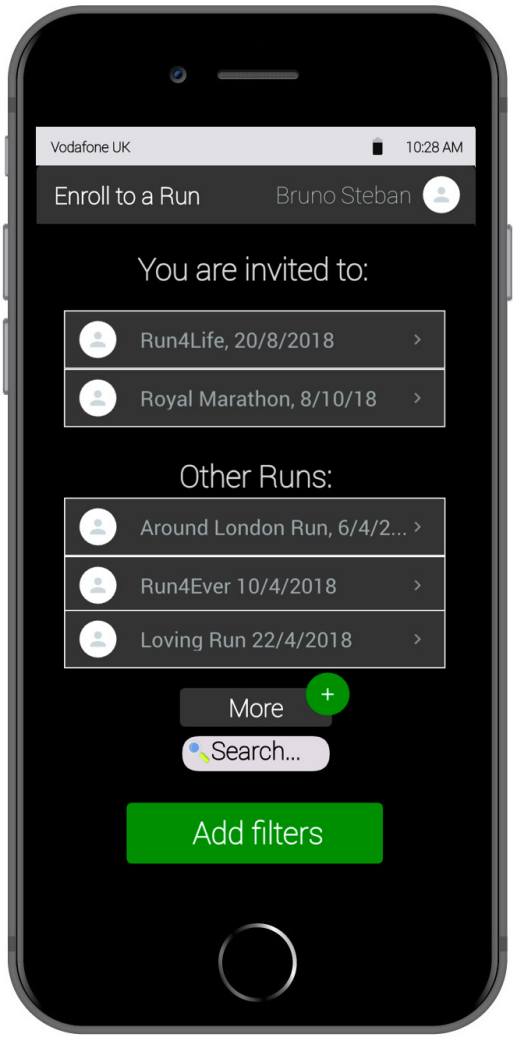

Figure 24: Enroll to a run.

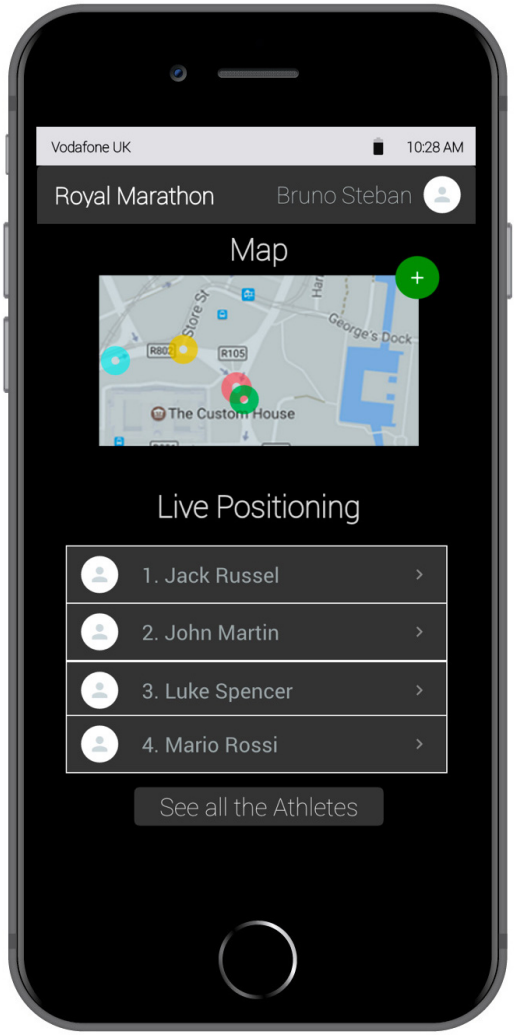

Figure 25: Spectate a run view

.

In the section Enroll to a Run (Figure X) a user can see a list of all the runs in which he/she has been invited. The user can select each 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 spefic 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 X), it is possible to spectate to a specific run in a interactive way. A map shows all the runners enrolled in it, each one represented by a different coloured circle. Selecting the **+** button is possible to zoom in the map that will appear on full-screen mode. Below the map the live positioning show 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: Receive from SignAccount interface user's credentials then it store them locally. After this operation waits for privacy policy appceptance.

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

- UserLogger: Receive from SignAccount interface that user has accepted the first part of pivacy policy then, if decide finally to create an account pass all these information to WebInterface.
- WebServer: Receive from WebInterface the credentials and provide them to credential manager through CredentialInterface.
- CredentialManager: Check the credentials inserted and if they are valid store them into DB using DBInterface.
- DBMS: Store 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: Receive from SignAccount interface that user has accepted the als the second part of pivacy policy then, if decide finally to create an account pass all these information to WebInterface.
- WebServer: Receive from WebInterface the credentials and provide them to credential manager through CredentialInterface.
- CredentialManager: Check the credentials inserted and if they are valid store them into DB using DBInterface.
- DBMS: Store data incoming from DBInterface.

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

- UserLogger: Whenever a user start the application automaticcaly this component inform the WebServer.
- WebServer: Receive from WebInterface users' log request and send those to Credential-Manager.
- CredentialManager: Check the users' availability and communicate it in reverse order allowing UserLogger to decide if user is admissible or not.

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, exploting the data acquired by GPSAcquirer and HSAc-quirer, send every ten minutes all the information acquired since the last message to UserDataManager using AcquireData interface.
- UserDataManager: When a data update is arrived through AcquireData interface, this component generates and sends an SQL query the allows the insertion of the new data in the DB.
- DBMS: Store data incoming from DBInterface.

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

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

- WebServer: Provide to third parties using WebInterface a web page that allows them to insert all the necessary credentials; then retrieve these informations. Moreover send these data to CredentialManager using CredentialInterface.
- CredentialManager: Check the credentials inserted and if they are valid store them into DB using DBInterface.
- DBMS: Store data incoming from DBInterface.

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

- WebServer: Provide to third parties using WebInterface a web page that allows them to make a request inserting all the necessary information. Then pass the request using AcceptInformationRequest interface.
- RequestManager: Receive requests from AcceptInformationRequest interface and stack them in a queue sorted by urgence 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 check the queue in order to select the more urgent request. Once an on-demand request is picked up it wraps the request and send it to DataCollector.
- DataCollector: This component once a data request is received builds a DB query that match what third partie wants and send it to DB using DBInterface.
- DBMS: This component when the query is arrived, compute it and provide data to the DataCollector using DBInterface.

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

- DataCollector: This component once data are arrived from DB, asks through ProvideStatistic interface to compute some kind of statistic on the data.
- StatisticGenerator: This component receive data from ProvideStatistic interface, compute statistics and send back them to 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: This component once data are arrived from DB and statistics are generated, wrap them and send this package to RequestCollector using ProvideCollectedData interface.
- RequestCollector: This component receive information from DataCollector, unwrap and send them using ProvideInfromation interface to WebServer.
- WebServer: Provide to third parties using WebInterface a web page that allows them to see statisitics 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 selcted in order to make the subscription. Using a parallel thread automatically add into the priority queue data request to supply live acquisition.
- DataCollector: This component once a data request is received builds a DB query in order to retreive only new data that match what third partie wants and send it to DB using DBInterface.

- DBMS: This component when the query is arrived, compute it and provide data to the DataCollector using DBInterface.

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

- DataCollector: This component once data are arrived from DB, wrap them and send this package to RequestCollector using ProvideCollectedData interface.
- RequestCollector: This component receive information from DataCollector, unwrap and send them using ProvideInfromation interface to WebServer.
- WebServer: Provide to third parties using WebInterface a web page that allows them to download retrieved information.

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

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

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

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

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 check the queue in order to select the more urgent request. Once a single mode request is picked up it wraps the request and send it to DataCollector.
- DataCollector: This component once a data request is received builds a DB query that match what third parties wants and send it to DB using DBInterface, requiring also all the information of the user involved.
- DBMS: This component when the query is arrived, compute it and provide data to the DataCollector using DBInterface.
- DataCollector: Once data are finally retrieved this component check, before return an answer to RequestManager, if the user involved has accepted the second part of the privacy policy; if yes wrap the information package and proceed as always, otherwise communicate to RequestManager that the request cannot be accepted.

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

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

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

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

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

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 check the queue in order to select the more urgent request. Once a group mode request is picked up it wraps the request and send it to DataCollector.

- DataCollector: This component once a data request is received builds a DB query that match what third parties wants and send it to DB using DBInterface, requiring also all the number of users involved in the specific query.
- DBMS: This component when the query is arrived, compute it and provide data to the DataCollector using DBInterface.
- DataCollector: Once data are finally retrieved this component check, before return an answer to RequestManager, if the number of users involved are less then 1000; if yes wrap the information package and proceed as always, otherwise communicate to RequestManager that the request cannot be accepted.

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

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: Receive from SignAccount interface that user has fill all the fields on registration page and has accepted the first part of pivacy policy (the only one presented) then, if decide finally to create an account pass all these information to WebInterface.
- WebServer: Receive from WebInterface the credentials and provide them to credential manager through CredentialInterface.
- CredentialManager: Check the credentials inserted and if they are valid store them into DB using DBInterface.
- DBMS: Store data incoming from 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: This component, exploting the data acquired by GPSAcquirer and HSAcquirer, acquire health status parameters every 5 seconds and GPS location every ten minutes as 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: This component, exploting the data acquired by GPSAcquirer and HSAcquirer, acquire health status parameters every 5 seconds and GPS location every ten minutes as before.

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

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

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

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

- HealthAnalyser: This component, when user requires it or is necessary on check the health parameters, sends to WebServer a request in order to retreive all the historical data about the user using WebInterface. From this point in ahead the sequence of action takes the standard procedure of managing request (R.8 , R.10) until WebServer.
- WebServer: Once the retrieved information is arrived to WebServer this component sends the package to 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 use it when is 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 check the last parameters acquired and compare them to the historical ones and then decides with a special algorthm if the user urgent 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: This component when the analysis founds something bad, sends an ambulance request to the nearest hospital specyfing the location of the user.

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

G.8 Allow user to manage a run.

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 displyaing a map in which he 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 he want to invite in 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 athletes allowed.

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

- RunManager: This component once all the information above are submitted and is also specified the name and description of the run, provides all these information to the WebServer.
- WebServer: Receive from WebInterface the inforamtion about the new race and forwards it to Track4RunManager component.
- Track4RunManager: Check all the information about the new run that the user want to create and if they are good creates and send an SQL query in order to store these information to DBMS.
- DBMS: Store data incoming from DBInterface.

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).

R.34 Allow user to enroll in a run.

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

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's 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 is useful for describing the order of implementation from top to bottom following this rule: a node is implemented before its children.

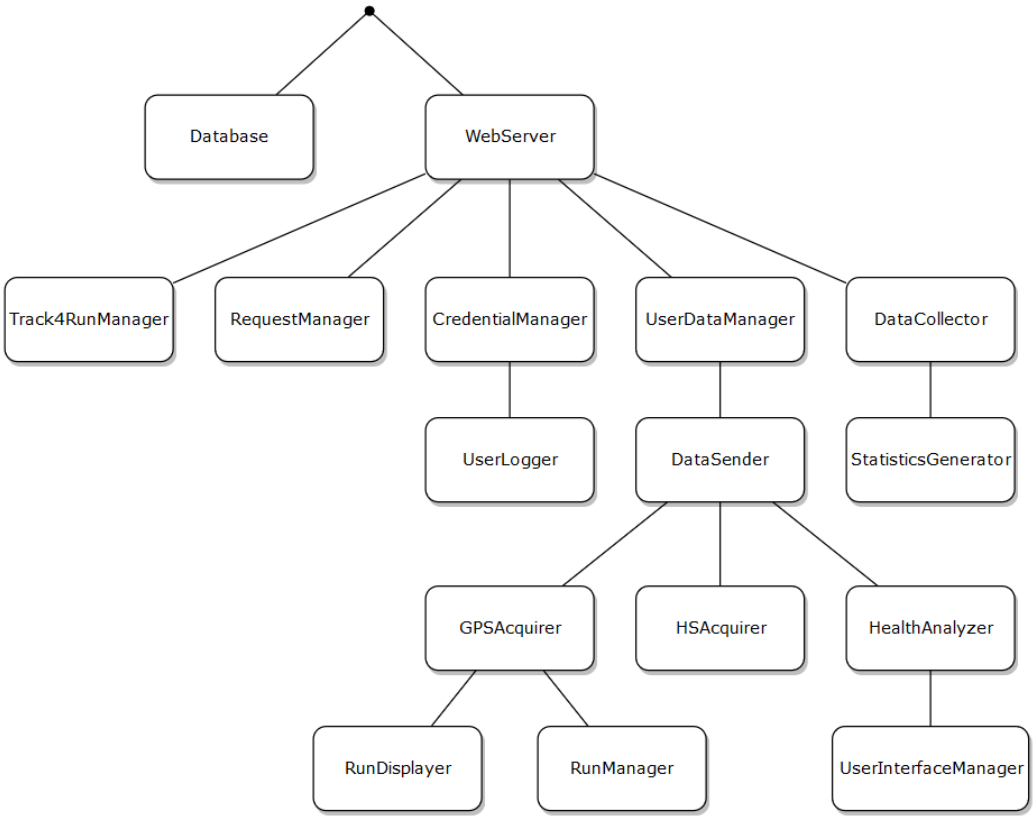

Figure 26: Implementation tree.

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's one of the most critical parts of the system, if it fails then all our system will as well, and lastly it forms the model of our system.

The other component that will be implemented in parallel to the Database is the WebServer because 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 then the AutomatedSOS component will be developed, especially the HealthAnalyzer since, in order to have faster reaction time, it contains part of the logic of our

system and also this component performs a critical functionality: calling an ambulance in case of need.

Lastly the 2 components of the Track4Run application will be developed.

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 which couples really well with the dependency priority implementation order, this way as soon as components that depends 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 unit tested is integrated with. The element just created is represented by a red square, while grey ones represent already existing and unit tested components. Note that in these diagrams if there are more that one grey square it does not mean that all the elements are integrated all together but just one by one, coupled with the one just developed.

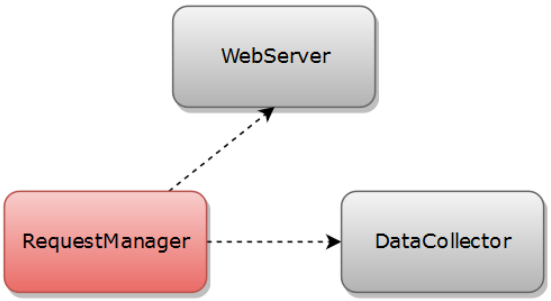

Figure 27: RequestManager Integration.

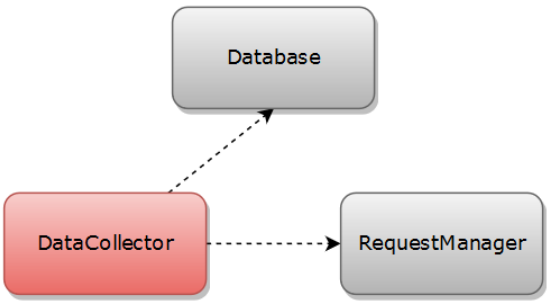

Figure 28: DataCollector Integration.

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

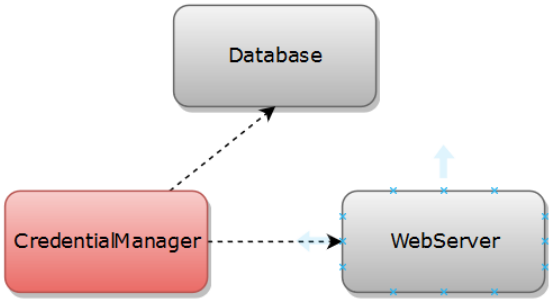

Figure 29: CredentialManager Integration.

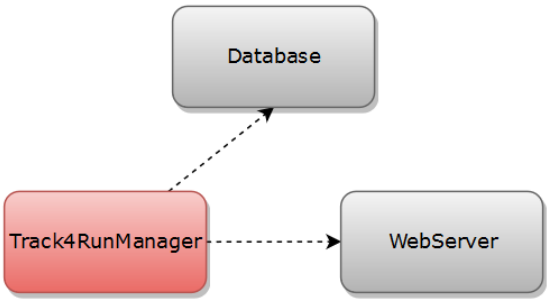

Figure 30: Track4RunManager Integration.

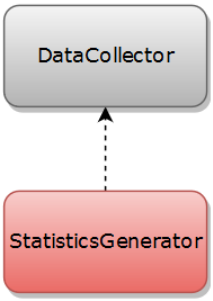

Figure 31: UserDataManager Integration.

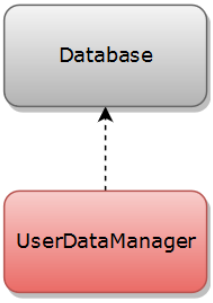

Figure 32: StatisticsGenerator Integration.

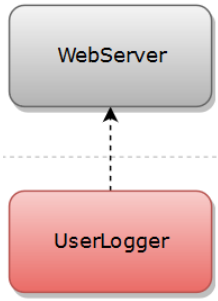

Figure 33: UserLogger Integration.

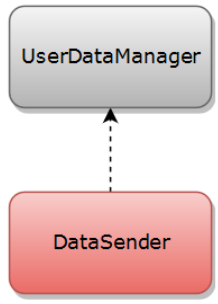

Figure 34: DataSender Integration.

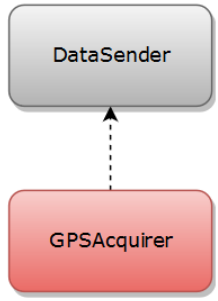

Figure 35: GPSAcquirer Integration.

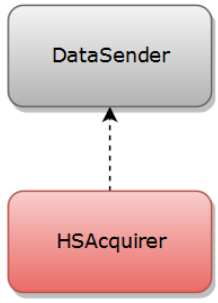

Figure 36: HSAcquirer Integration.

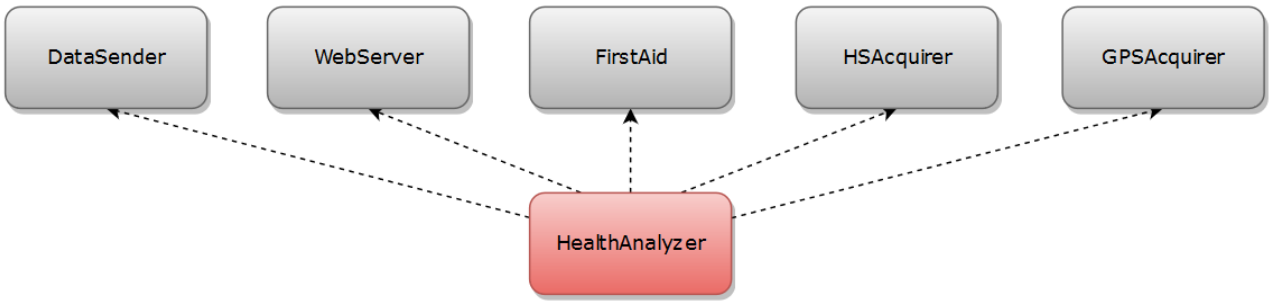

Figure 37: HealthAnalyzer Integration.

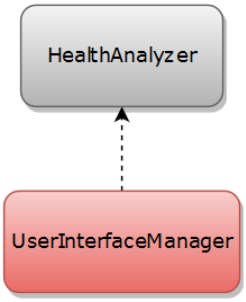

Figure 38: UserInterfaceManager Integration.

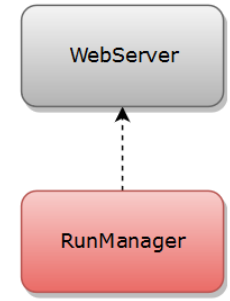

Figure 39: RunManager Integration.

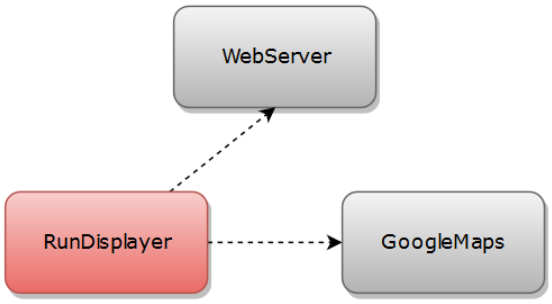

Figure 40: RunDisplayer Integration.

5.2.2 Third Party Request Integration

In the following two figures, integration including more that two components at the same time are shown. More precisely they explain the order in which components are integrated together in order test the com-

plete 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.

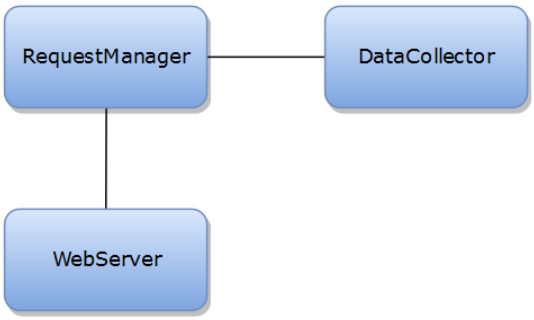

Figure 41: ThirdPartyRequest semi-chain integration.

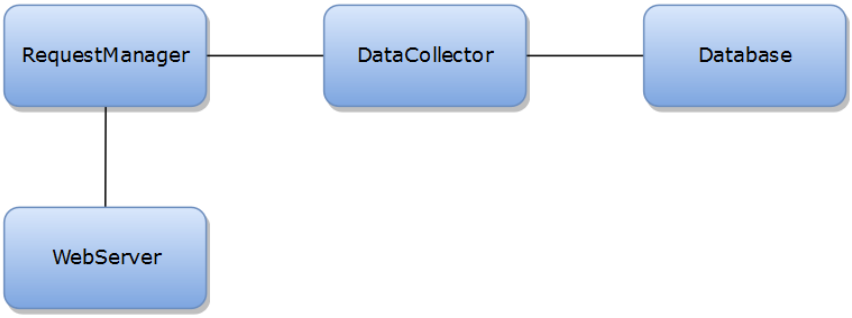

Figure 42: ThirdPartyRequest full-chain integration.

5.2.3 Run Event Storing Integration

In the following two figures, integration including more that two components at the same time are shown. More precisely they explain the order in which components are integrated together in order test the complete functionality of storing into the Database all the information regarding run events. 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.

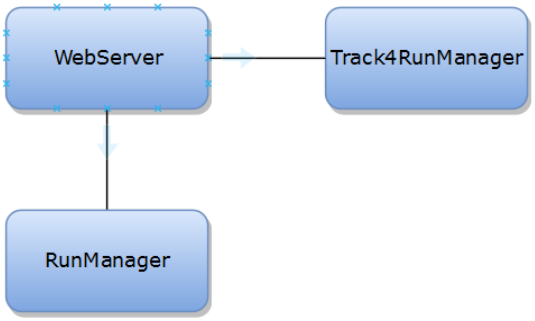

Figure 43: Run event storing semi-chain integration.

Figure 44: Run event storing full-chain integration.

5.2.4 Non so cosa mettere

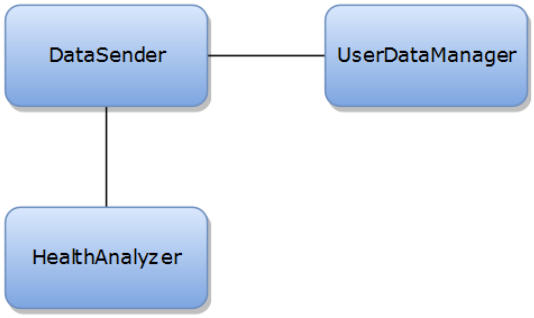

Figure 45: SpectateRun semi-chain integration.

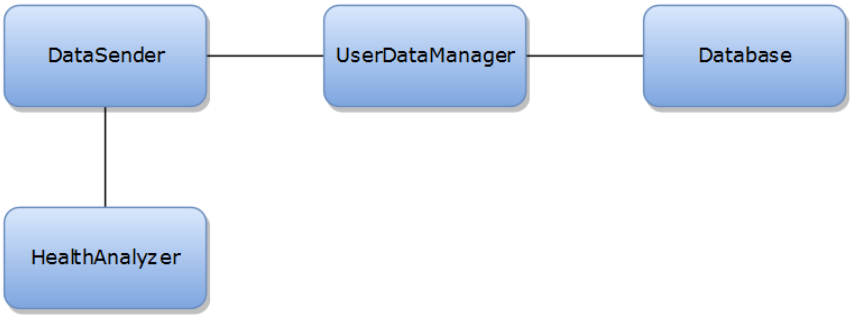

Figure 46: SpectateRun full-chain integration.

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.
- Query optimization problems: taking into account that the system's functionality heavily relies on queries this will for sure be a very useful test.
- Hardware issues.

- Network issues.

More precisely two types of System Testing will be done:

Load Testing -> 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 -> 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

In this section contains information about how much hours each group member 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
Overview		1
Deployment View		4.5
Runtime View		10
Component Interface		3
Component Diagram		3
Implementation,Integration and Test Plan		5
Implementation,Integration and Test Plan		7.5
Total		34

6.0.2 Andrea Caraffa

Date	Task	Hours
18/10/18	Goals	2
19/10/18	Domain Assumptions	3
20/10/18	Text Assumptions	3
21/10/18	Introduction	2
27/10/18	Goals	2
30/10/18	Product Functions	3
1/11/17	Mockups	3
3/11/18	Mockups	3
4/11/18	Goals	2
4/11/18	Mockups	2
5/11/18	External requirements	2
5/11/18	Alloy	2
6/11/18	External requirements	3
9/11/18	Revisioning	2
9/11/18	Alloy	3
10/11/18	Revisioning	3
11/11/18	Revisioning	3
11/11/18	Alloy	3
Text Assumptions		3
Goals		6
Domain Assumptions		3
Introduction		2
Product functions		3
External requirements		5
Mockups		8
Alloy		8
Document Revisiong		8
Total		46

6.0.3 Andrea Bionda

/	Task	Hours
	Text Assumptions	3
	Goals and Introduction	6
	Domain Assumptions	3
	Functional requirements	13
	Class Diagram	5
	Sequence Diagram	5
	Performance requirements and Constraints	3
	Alloy	7
	Total	45

7 Reference Documents

- Specification Document "Mandatory Project Assignment AY 2018-2019".
- Slides "Structure of RASD".
- Slides "Use of Alloy in RE".
- Use Case Diagrams created with https://www.lucidchart.com
- Mockups created with https://www.fluidui.com