Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

TrackMe

D.D

Design Document

Giulia Mangiaracina [905106]

Andrea Miotto [920287]

Ilaria Moschetto[915191]

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to describe the TrackMe System with respect to the design of the entire system, providing a complete view of the architectural choices and components, with a description of their run-time behavior, the interfaces that they offer and the used design patterns. All of the specified parts of the system will match the requirements and goals stated in the Requirements Analysis and Specifications Document (RASD). Moreover there will be a final section dedicated to the Implementation, Integration and Testing planning.

1.2 Scope

TrackMe is a Software that provides three different services: Data4Help, AutomatedSOS and Track4Run. Data4Help collects data of individuals through their smartwatches and similar devices. The Users can visualize their health and location data through their mobile devices, while the Third Parties can access data in real time, after authorization, to monitor the location and health status of a single individual, or a set of individuals in an anonymized way through a web application. They can receive periodically updates on target users and group request, and consult the data through useful statistics. AutomatedSOS offers a personalized and non-intrusive SOS service to people. Based on the data collected by Data4Help, it detects anomalies in the health status of the individual and alerts the NHS, allowing the dispatch of an ambulance as soon as possible. Track4Run is a service that allows the organization of local runs. Organizers can create a new run, setting a customized path and the related preferences. Users can visualize the planned runs in detail and enroll to them as participants if interested. The service uses Data4Help data to show the position of the participants on a map, allowing all the online users to follow the run live on their devices.

1.3 Definitions, Acronyms, Abbreviations

In this part of the DD Document there are some definitions, acronyms and abbreviations that will be used among the following chapters.

1.3.1 Definitions

- **User:** When referring to the *User*, we refer to a logged-in User which is not a Third Party. The software is usable only by a recognized User.
- **Visitor:** When referring to the *Visitor*, we refer to a person not yet registered to the TrackMe system.
- **Registered User:** A User registered to the TrackMe System. To use the System and performs any action, all actors must be registered.
- **Logged-in User:** Is a registered User who is logged in the TrackMe System.
- **Organizer:** Is a logged-in User who uses the Track4Run system, and decides to create a new run.
- **System:** when referring to the *System*, this document refers to the entire system infrastructure.
- **Client:** We refer to a normal logged-in User of the application, not a Third Party.
- **Normal User:** is a Client.
- **TrackMe:** is the name of our System, that includes the three main services: Data4Help, AutomatedSOS and Track4Run.
- **App:** We refer to either TrackMe Mobile application or the Web version.
- **Health data:** With health data we refer to the health data collected by User's device, among those available or requested: heart rate, blood pressure, body temperature, step counter, respiratory rate, localization.
- **Request:** A request is performed by a Third Party and concerns one or more Users. There are 4 types of requests: One-shot User request (concerning the latest available data of a specific User selected inserting his/her fiscal code), subscription User request (it requires an interval time to specify: the Third Party will receive the updates how often it has requested), one-shot group request (the Third Party selects the research preferences and the requested data that match them are anonymized) and subscription group request (the same of one-shot group request, but it requires to specify the update interval time). In all kind of requests the Third Party can specify which data it want to receive, among all the available health data (heart rate, blood pressure, body temperature, step counter, respiratory rate, localization). The User requests must be accepted by the User to be performed, while the group requests are performed only if concern a group of individuals greater than 1000 for a security factor.
- **Run:** A Run is a sports competition organized by and Organizer through the Track4Run System. The runs are divided in: scheduled runs, that have been planned and will take place in the future, and live runs, that are taking place in this moment and can be followed live. The scheduled and live runs are shown in the scheduled run list and live run list respectively.
- **Path :** Every run has a path, i.e. the route that links the points selected by the run Organizer during the event creation: initial point, the intermediate points and the final point.

1.3.2 Acronyms

- **R.A.S.D.:** Requirements Analysis and Specifications Document
- **D.D.:** Design Document
- **A.P.I.:** Application Programming Interface
- **N.H.S.:** National Health Service
- **S.M.S.:** Short Message Service
- **E.R.:** Entity Relationship
- **U.X.:** User Experience
- **REST:** Representational State Transfer
- **DB:** Database
- **HTTP:** Hyper Text Transfer Protocol
- **HTTPS:** HTTP over SSL

1.3.3 Abbreviations

These abbreviations will be used both in this document and in the following documents.

- [G k]: The k-th Goal
- [D k]: The k-th Domain Assumption
- [R k]: The k-th Functional Requirement

1.4 Document Structure

This document is divided into 6 sections:

- Introduction:
  A brief introduction of the design document.
- Architectural Design:
  Here are shown the main architectural design choices about the structure of the System. In the High Level Architecture the System is described with a layered structure of four tiers; in the Component view is specified the role of each component within the System and their interactions; the Physical Architecture is explained in the Deployment view, that illustrates the hardware and software architectural decisions; the Database view shows the database structure of the System; in the Run-time view is shown through sequence diagrams how the previously described components interact between each other and how they perform the main functionalities of the System.
- User Interface Design:
  This chapter explains the main User interfaces used in the client layer through Mock-Ups and UX diagrams.

- Requirements Traceability:
  In this section the mapping between the requirements, stated in the RASD document, and the components of the System are presented.
- Implementation, Integration and Test Plan:
  In this section in presented the plan that describes how and in which order each component will be developed and how each of them will be integrated with the others.
- Effort Spent:
  In this section is shown the total amount of hours spent by each component of this group to write the Design document.

1.5 Reference Documents

- Specification documents: *Assignments AA 2018-2019.pdf*
- RASD v.1.1 document previously delivered.

1.6 Used Tools

The tools used to develop this document are:

- GitHub
- Overleaf
- Git
- SourceTree
- draw.io
- Sketch
- Notability

Chapter 2

Architectural Design

2.1 Overview: High-level components and their interaction

TrackMe is a System that stores User's data and makes them available for Third Parties requests. It also offers the in-house services Track4Run and AutomatedSOS to the end Users. In order to provide these services, the System needs two different client applications:

- a Mobile client that sends new User data to the server, designed specifically for individual users;
- a Web client, accessible to verified Third Parties only, that gives access to the specifically requested data.

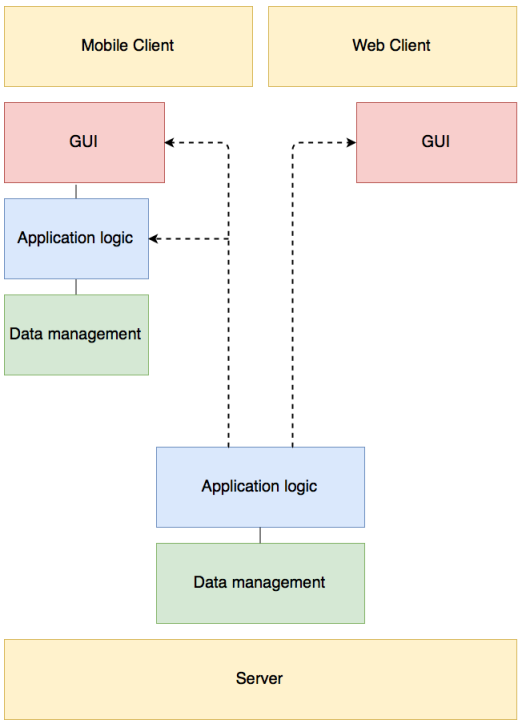


Figure 2.1: Overview of the *TrackMe* system

The picture above describes the client-server architecture of the System. The Mobile and Web clients are different in terms of thickness: in order to maintain high offline reliability, the Mobile client is provided with data management capabilities (it can, for example, manage the collected data that have not been uploaded yet), while the Web client only shows the data that is available to be accessed by the Third Party.

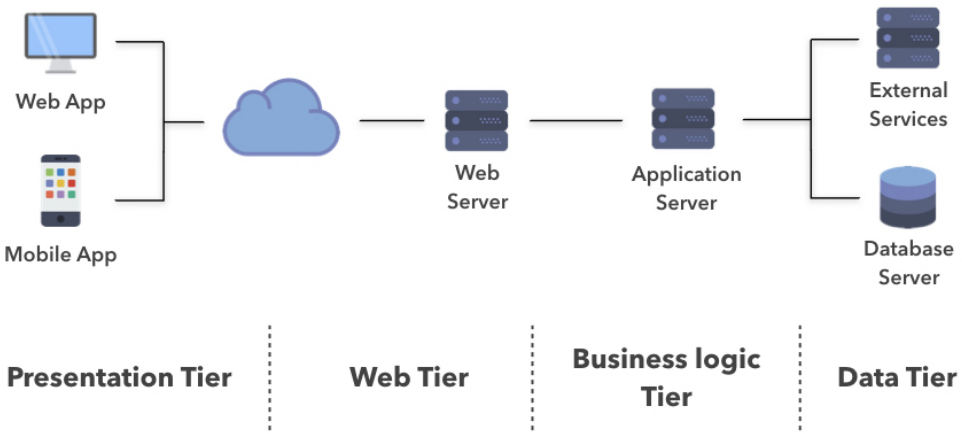


Figure 2.2: High-level architecture of the *TrackMe* system

The high-level architecture of the TrackMe system is divided into four tiers:

1. the *Presentation tier* enables the interaction with end Users, includes a Web application for Third Party access and a Mobile application, utilized by individual Users.

2. the *Web tier* consists of an Apache web server that manages RESTful HTTPS requests between clients and the application server.

3. the *Business Logic tier* includes the application server, where the main tasks, such as the request management, the run creation and the emergency call, will be effectively executed.

4. the *Data tier* manages the storage of the data needed for the functioning of the TrackMe system. It includes also the external services that will be required for the System to function properly.

2.2 Component View

In the figure below is represented the high level component architecture of the System. It is composed by two main modules that correspond to the two platforms used by two different kind of customers of our services: The "Web Application" is for the Third Parties, that use the System connecting to the site through a Web Browser, and interact with the "Third Party Mobile Services" subsystem. The "Mobile Application" is for normal Users, which use the services of the system with an IOS or Android device through the TrackMe Mobile Application, and interact with the "User Mobile Service" subsystem.

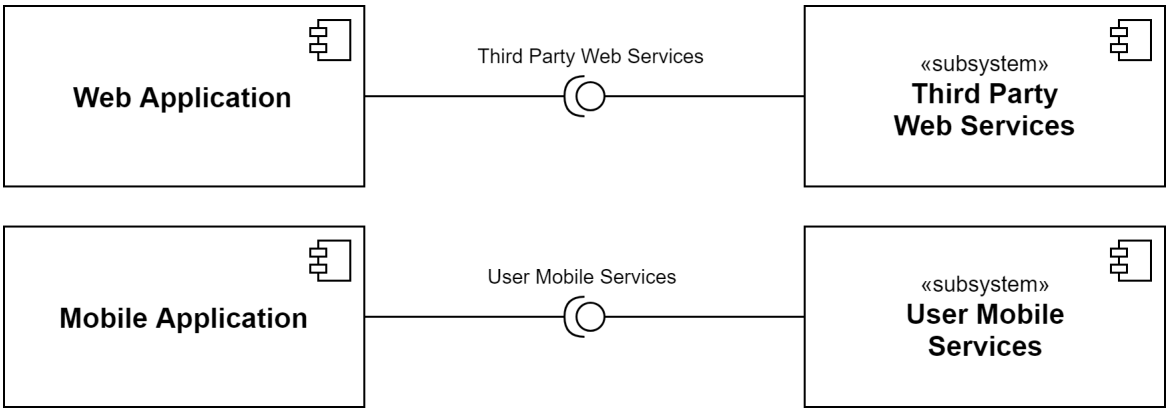


Figure 2.3: Component view of *TrackMe* System

2.2.1 Component view of the Third Party Web Services

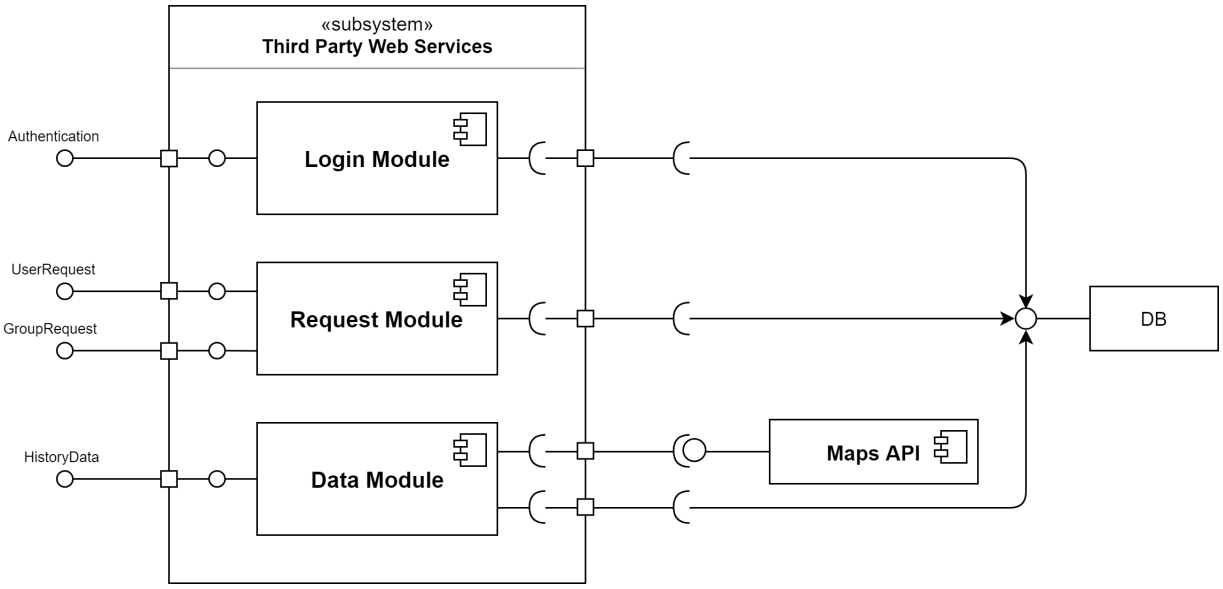


Figure 2.4: Third Party Web Service component

- The Login Module manage the operations of authentication, allowing the Third Party to be recognized and then to use the services of the System. It interacts with the database to check the credentials and retrieve the account informations.
- The Request Module is responsible for the requests dispatching: it offers to the Third Party the interfaces to fill out the forms to perform a user or a group request, and forward them to the database. Among its internal function, it sends a notification to the target User when he/she is the subject of a User request and it performs the security check in case of group request, allowing to perform the group requests only if the matched collected data belong to at least to 1000 individuals.
- The Data Module offers to the Third Parties all the functionalities to manage the performed requests, grouped by status (pending, accepted and refused), allows the consulting of the results and calculates useful statistics on collected data. In case of subscription requests, it is its responsibility to update the Third Party with the new data available with the desided frequency. It retrieves the selected data querying the database, that stores all the User's records. In case of a group request, it also anonymize the data. This module is connected to Maps API to allow the consultation of a User's location data on an interactive map.

2.2.2 Component view of the User Mobile Service

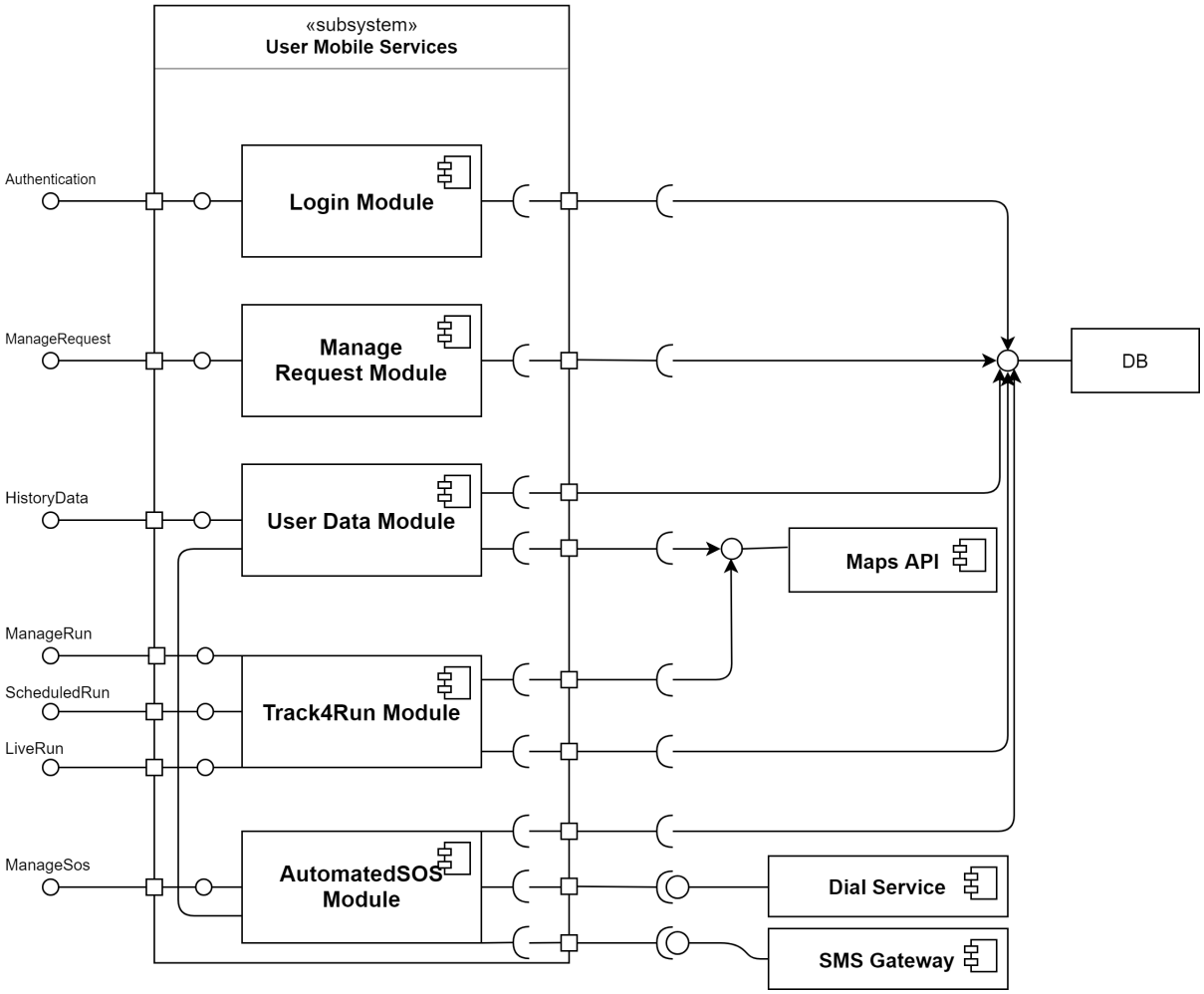


Figure 2.5: User Mobile Services component

- The Login Module manage the operations of authentication, allowing the User to be recognized and then to use the services of the System. It interacts with the database to check the credentials and retrieve the account informations.
- The Manage Request Module offers to the User the functionalities to manage the incoming requests, providing all the information about the requested data. It interacts with the database, dispatchs the User reply and notifies the sender Third Party about the occurred response.
- The User Data Module sends the new User's health and location data as soon as they have been collected by the smartwatch device and sent to the System database, where all the records are stored. It offers the interfaces to allow the User to visualize his/her History Data stored in the database, through querying it, selecting time or location setting, or performing a manual research. It is connected to the Maps API module to allow the visualization of User location data on an interactive map.

- The Track4Run Module works as Run manager for the Track4Run system: it offers to the User the interfaces that allow to create and enroll to a run, to visualize the list of the planned runs and to follow live runs. It interacts with the Maps API module to manage the location data and with the database to store new runs and retrieve all the details about the runs.

- The AutomatedSOS module provides the functionalities concerning the AutomatedSOS system: it offers to the User the interfaces that allow him/her to enable or disable the service, to manage the emergency contacts list and the emergency text message. It deals with all the tasks related to an emergency: it performs the checks on the health status and notify an emergency state if some anomalies in the health parameters are detected. It interacts with the dial service to connect to the NHS, sending all the information related to the emergency, and with the SMS gateway, to notify the user's contacts list with a text message.

2.3 Deployment View

This section is devoted to explain in details the physical architecture of the *TrackMe* system, with mentions to the communication protocols that will be used.

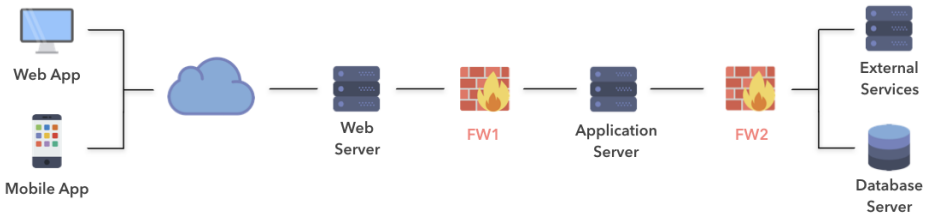


Figure 2.6: High-level architecture

First of all, two different clients are put in place: a lightweight Web application that will be accessible from any modern Web Browser and a full Mobile application with support for both iOS and Android, the two most popular mobile operating systems. The Mobile application will manage User's data through *Realm*, an ODBMS which provides support for both iOS and Android. In order to have a highly maintainable system, the communication between the clients and the server will adhere to the *REST* protocol, which is built on top of *HTTPS*. The requests will be received from an *Apache Web Server*, where they will be forwarded to the application server after being filtered by a firewall. The *Application Server* is the central element of the system architecture and it needs high performance, security assurance and high availability. Our engineering team found in *IBM WebSphere Application Server* the product that best meets these needs. The server will run the application on the *EJB* environment and it will be the only element to interact with the database. The *Database Server* will be an *IBM DB2*, so that when technical support is needed only a single vendor has to be contacted for both the application and database server. To enhance the security of the system, the Application Server and the Database Server are set in a demilitarized zone delimited by firewalls, in order to prevent unwanted connections. Finally, the *Application Server* needs to interact with the external service, such as . Both the map service and the call/sms service are of paramount importance for the *TrackMe* system, because they provide important utility

for the final users. The choice of the map service is *OpenStreetMap* for its reliability and privacy. The call and sms service is entrusted to CallHub.

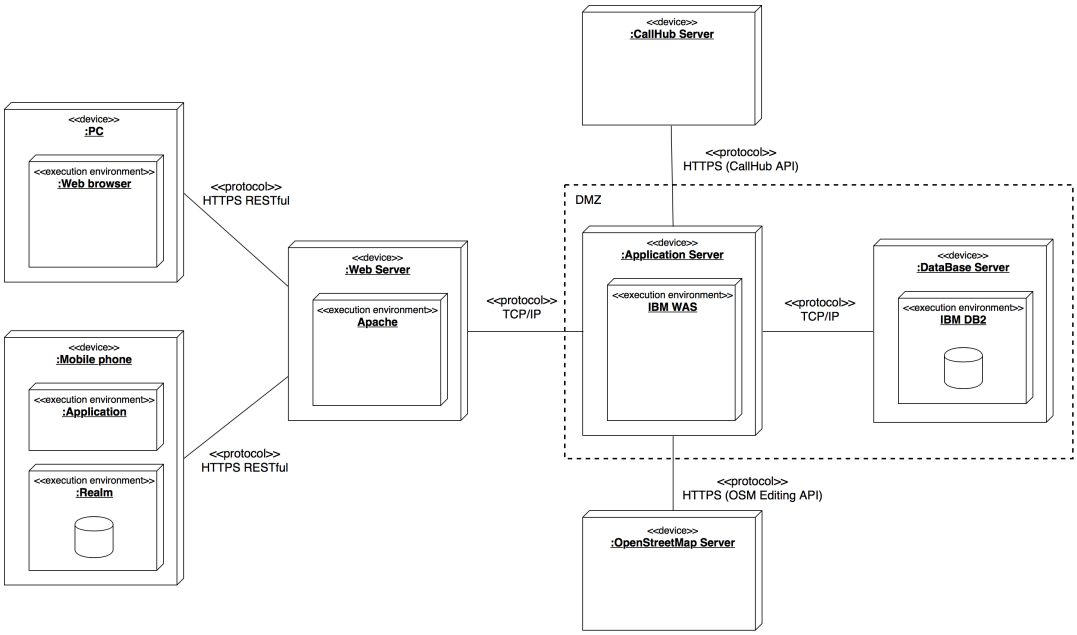


Figure 2.7: Deployment of the *TrackMe* system

2.4 Database View

In this section is described the design of the database that will be used for the *TrackMe* system.

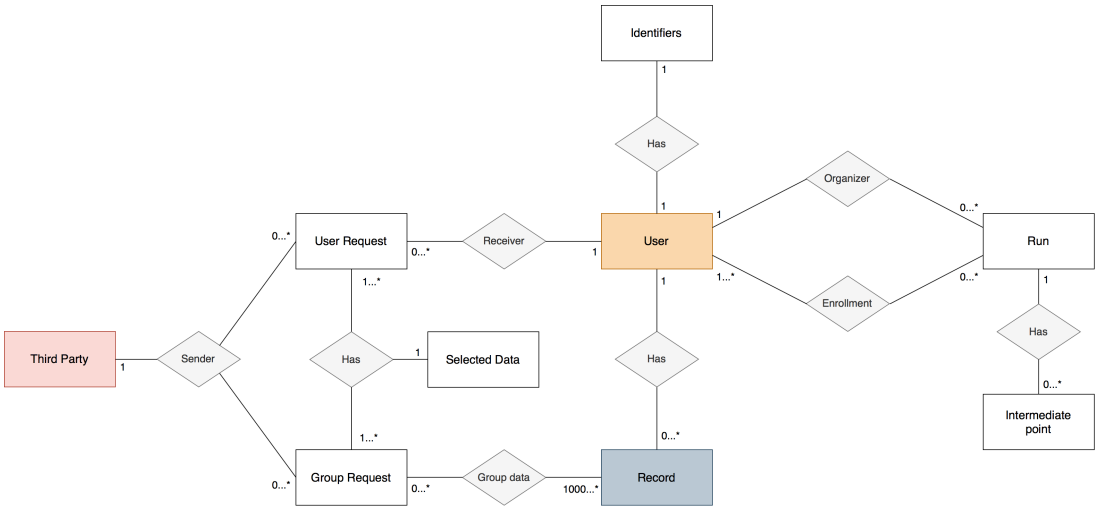


Figure 2.8: Simplified ER diagram for the *TrackMe* system

The picture above shows the main entities of the database. There is a *User* entity, used to store the personal information of individual Users that is requested at sign up. Every User has many records associated to itself, each of them containing location and

health data. This information is put in the *Record* entity. The *Third Party* entity contains meaningful data about the company that signed up to the *TrackMe* system. Third Parties can make requests to the System in order to get access to User's data, so the *Third Party* entity is related with the *User Request* and the *Group Request* entities. Both these entities contain information about the request, such as its status and the specific type of data to be accessed, but there is an important difference between the two regarding the anonymity of data. While a User request is sent to an individual User directly, a group request has a higher scope, and the data that is accessed after this type of request must be completely anonymous. The sender of the request must not get access to the personal data of the Users that have generated the records. To enforce this, the *Group Request* entity is not related with *User*, but instead directly with *Record*. This solution is permitted by the use of anonymous identifiers, present in the *Record* entity, that have a 1 to 1 relation with the Users of the system.

The *Track4Run* service is represented by the Run entity, which has the information about the location, time and participants.

The *AutomatedSOS* service does not rely on a complex data structure so it's not displayed in the simplified ER diagram. Nevertheless, two attributes are present in the *User* and *Record* entities: the *SosEnabled* attribute is a flag that indicates if the service is enabled or not; the *isEmergency* attribute indicates if the record contains critical health parameters.

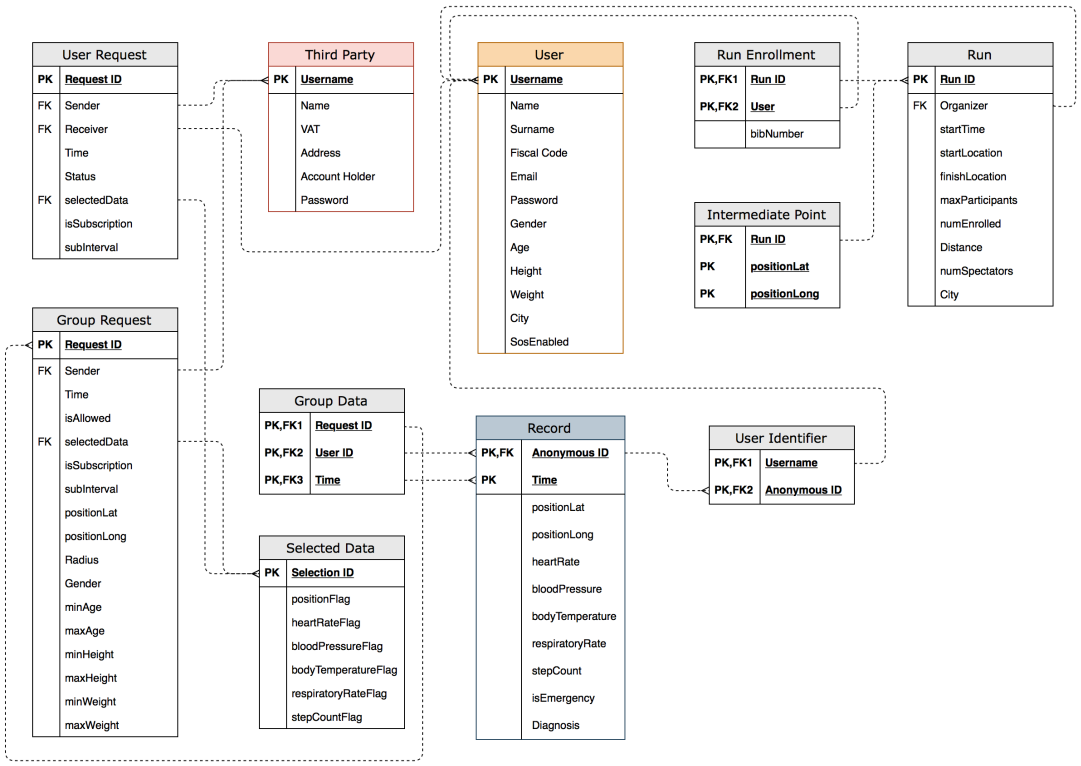


Figure 2.9: Tables and relative attributes of the *TrackMe* database

Figure 2.9 represents a more complete representation of the *TrackMe* database. Some of the attributes included in the diagram require some further explanation:

– the *Status* attribute of the *User Request* table can take one of this three values:

Accepted, Refused or Pending. As soon as the Third Party makes a request in the System, its *Status* is set to Pending. When the User replies to the request it can either become Accepted or Refused, depending on the User's choice. The *Status* attribute is not present in the *Group Request* because this type of request does not require User's interaction to be effective. In place of the *Status* attribute, the *isAllowed* attribute is used: it is a boolean that is defined automatically by the system after checking if the minimum number of 1000 interested Users is reached or not.

- the selection of the type of data to be accessed with a request is represented in the *Selected Data* table. It contains 6 boolean values that indicate if the sender actually requests access to the specific type of data. Given that a selection can be done in $2^6 = 64$ different ways, a unique integer identifier *Selection ID* that ranges from 1 to 64 is put in place.

- the attributes in the *Record* table represent all the types of data supported by the system. Users' smart devices may not be equipped with the sensors needed to acquire certain type of data (for example, smart sensors for respiratory rate monitoring are not easily available in the market), so instances of this entity do not have to contain all of them. To signal the absence of data, a special value is stored.

- the *subInterval* attribute in the *User Request* and *Group Request* tables indicates the time between two data updates in a subscription request. It is therefore used only if the *isSubscription* attribute contains the value True and it can take 4 different values: hour, day, week or month.

2.5 Run-time View

These sequence diagrams show how the different components interact through each other when the main features are used by various actors.

2.5.1 Login

A Third Party goes to the login Web page of the TrackMe Web application for authentication in order to be able to use the provided services. It inserts its credential (username and password). The Request Module checks if the inserted data match with an existent account, and if the Third Party is recognized, it is logged in the System.

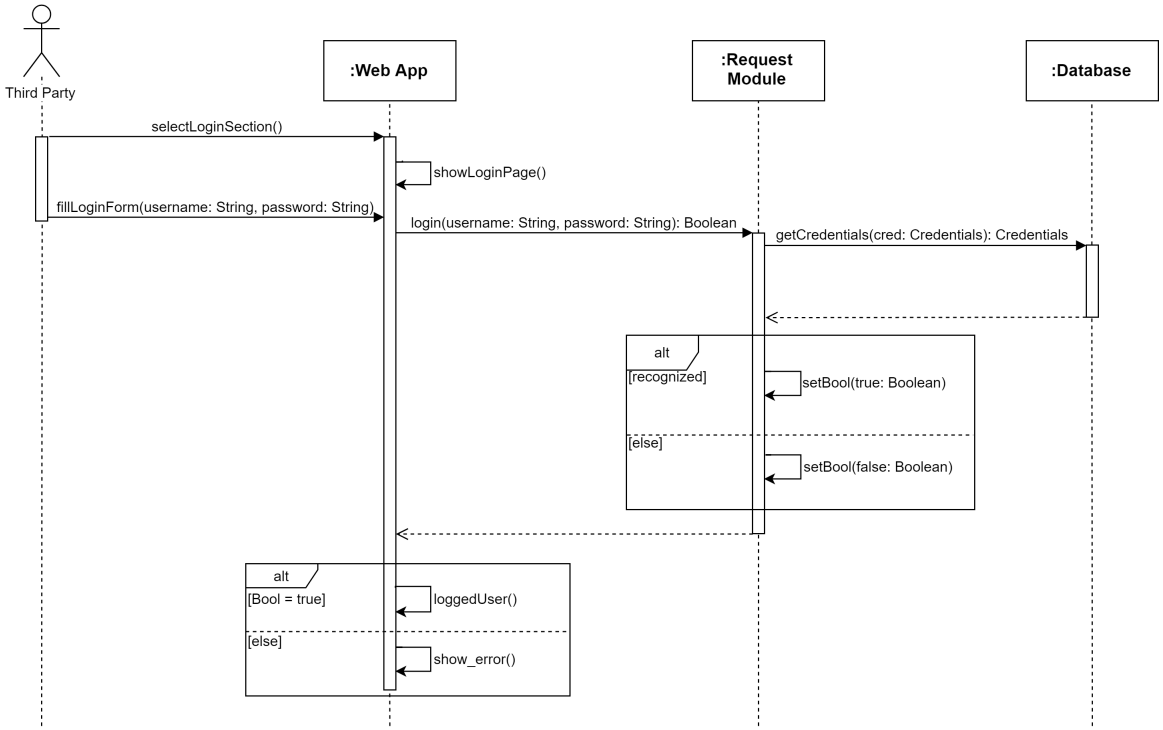


Figure 2.10: Login sequence diagram

The Login process for a User of the Mobile application is the same.

From now on we consider that the Users are logged in the System.

2.5.2 Dispatch of a User subscription request

A Third Party selects the new request button in the individuals section with subscription mode; the Third Party fills out the form with the fiscal code of the target individual, the data that it want to receive (encoded as booleans) and an interval time corresponding to the desired update frequency. The Request Module creates a new request with the selected data and stores it in the database. After that, the module notifies the target User that there is a new request to manage in his/her incoming request section.

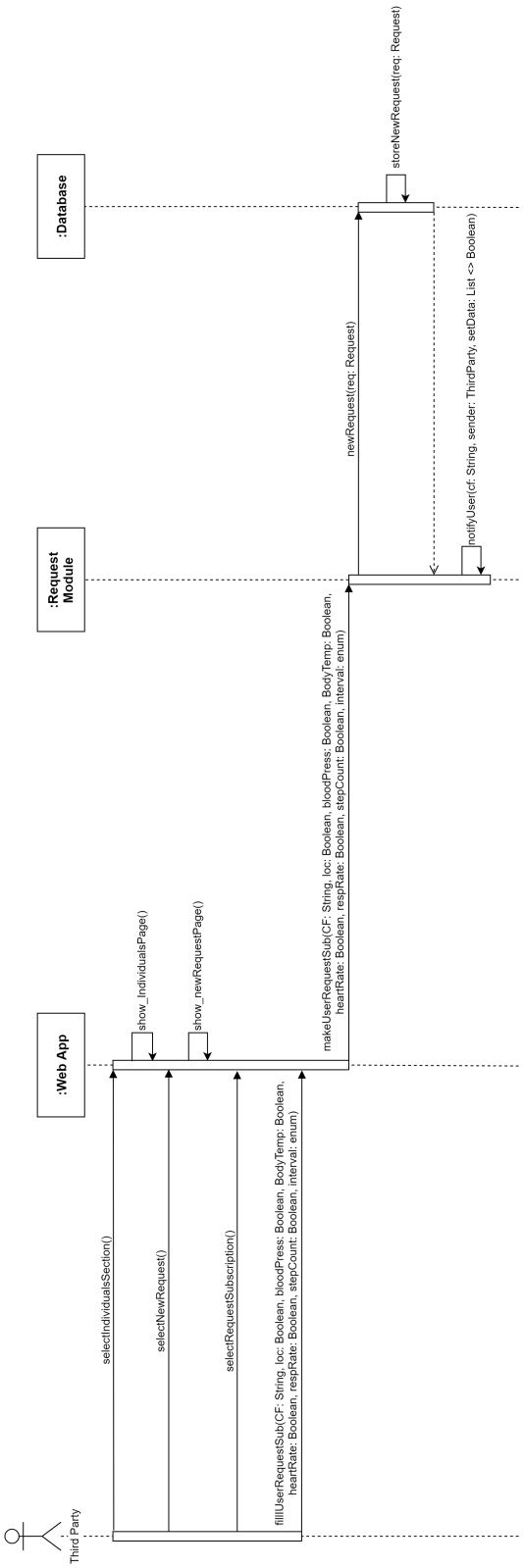


Figure 2.11: User request by a Third Party

2.5.3 Manage of incoming User Requests

A User can manage incoming requests clicking on the appropriate section. The Manage Request Module asks to the database the list of requests that the User has received and shows it. The User can selects a specific request and the application shows all the details: the name of the Third Party which performed the request, the list of data in which it is interested in, the update interval time it selected. At this point the User can decide to accept or deny the request; the Manage Request Module dispatch the response to the database, updating the status of the request. Finally, it notifies the request sender, to inform that the request is no more pending.

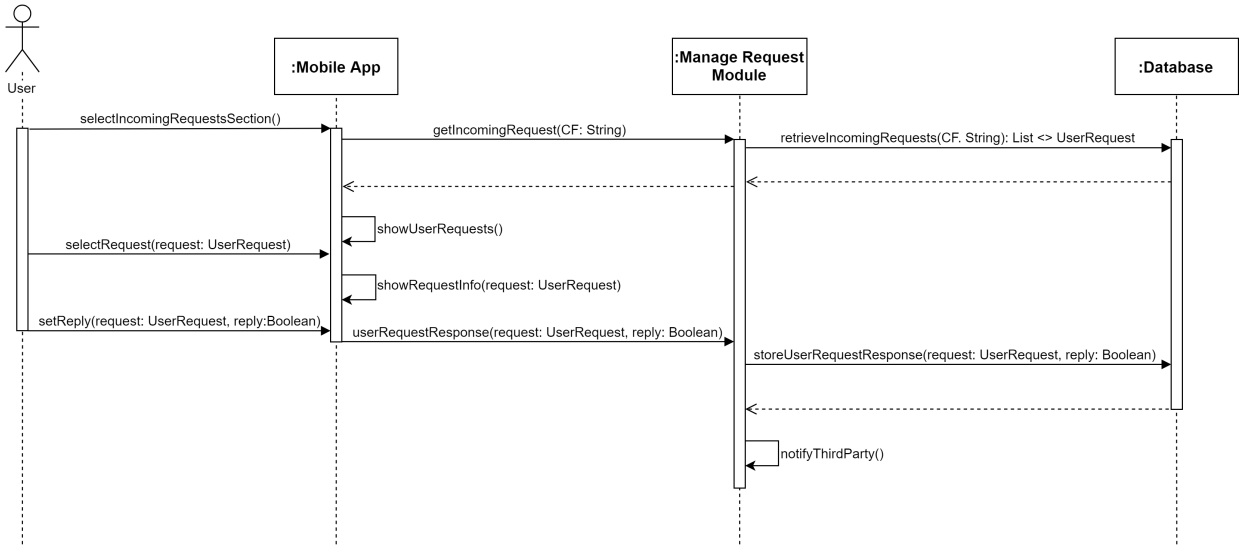


Figure 2.12: User acceptance of a request made by a Third Party

2.5.4 Dispatch of a group subscription request

A Third Party selects the new request button with subscription mode, in the group section; it fills out the form with the kind of data it desire to receive, the characteristics on which will be perform the group research (age range, gender, location, weight and height range) and the update interval time. The Request Module retrieves the informations from the database, creating a group with data that match the parameters selected, and checks if the number of individuals in this group is greater than 1000. If it is, the module anonymizes the data covering the identities with the anonymous id associated to each individual and sends the data to the Third Party.

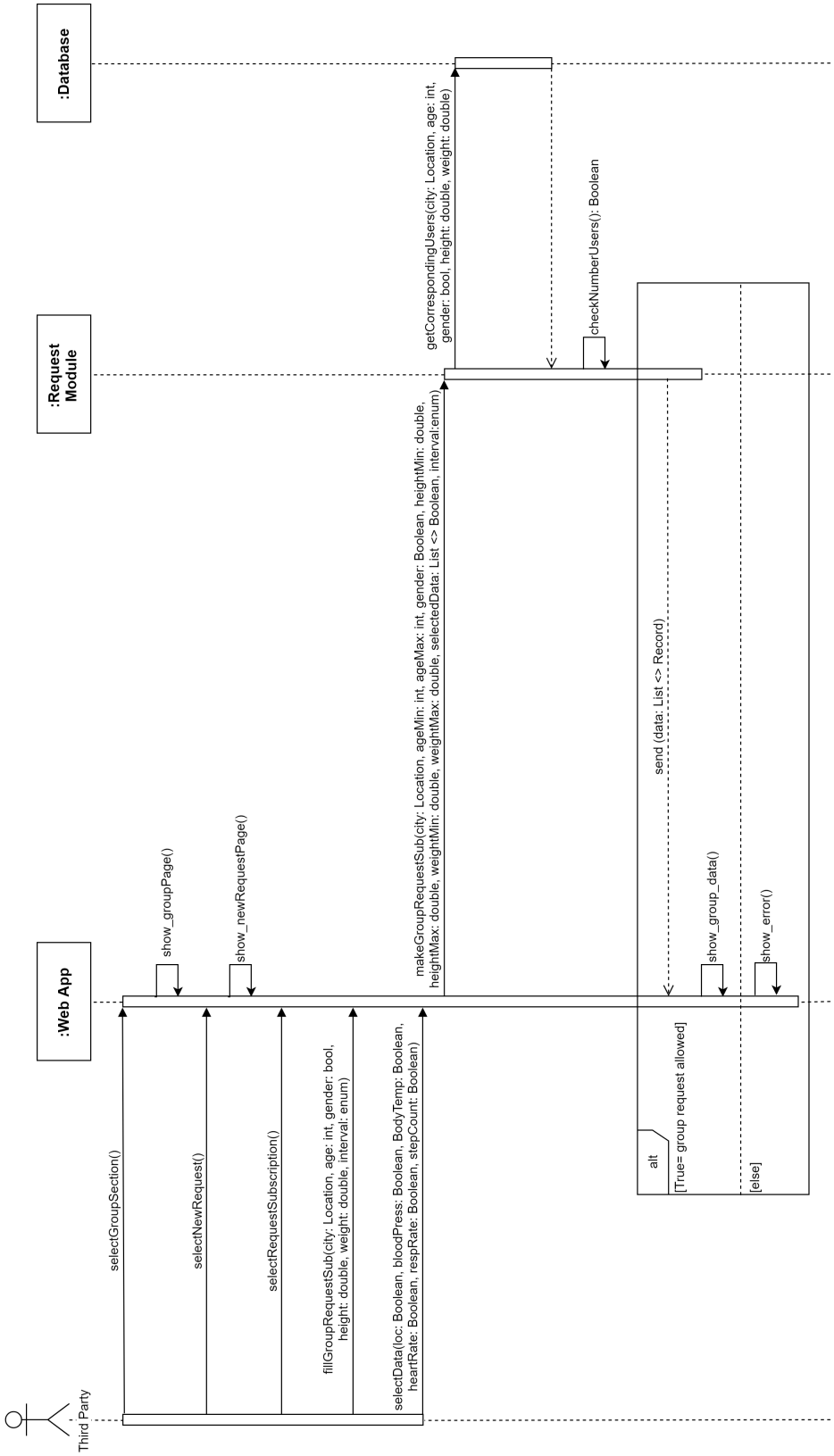


Figure 2.13: Group request by a Third Party

2.5.5 Visualization of User data

The Third Party can see the list of performed User requests, divided by status: Accepted, Refused, Pending. The Data Module retrieves them from the database and shows the list of requests, taking only those of the selected status. Assuming that the Third Party is in the Accepted Requests section, now it can select the target request and the application shows the related records of the request.

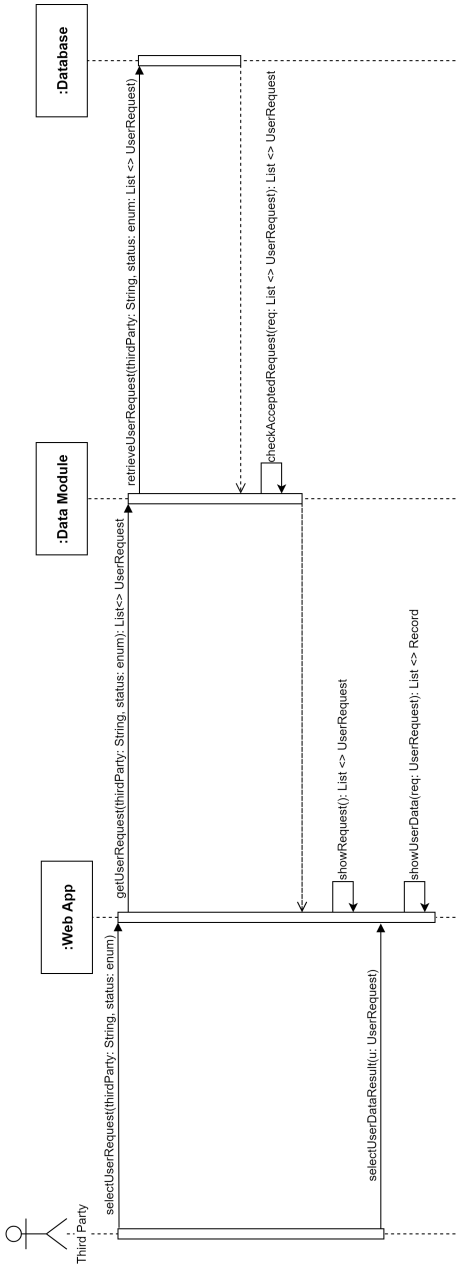


Figure 2.14: Visualization of data of a User that has already accepted the request

2.5.6 Emergency

The smart device collects parameters that exceed predeterminate thresholds: this is a signal of an emergency. The User History Module receives the critical data, constructs a Record and, in case the User has the AutomatedSOS service enabled, forwards it to the AutomatedSOS Module to be analyzed in more detail. The AutomatedSOS Module performs checks in order to identify if the data correspond to an emergency. If no critical parameters have been found, the module simply send the record back to the User History Module, after updating it with the correct emergency status (in this case false). If an emergency is actually detected, the AutomatedSOS Module proceeds to create a brief textual diagnosis of the health status. From the textual diagnosis, an audio diagnosis is generated with the use of the Festival Speech Synthesis System, an open source tool that offers a full text to speech system. At this point everything is ready for the call to the NHS through the CallHub external service. The same service is used to send a text message to the contacts specified by the user.

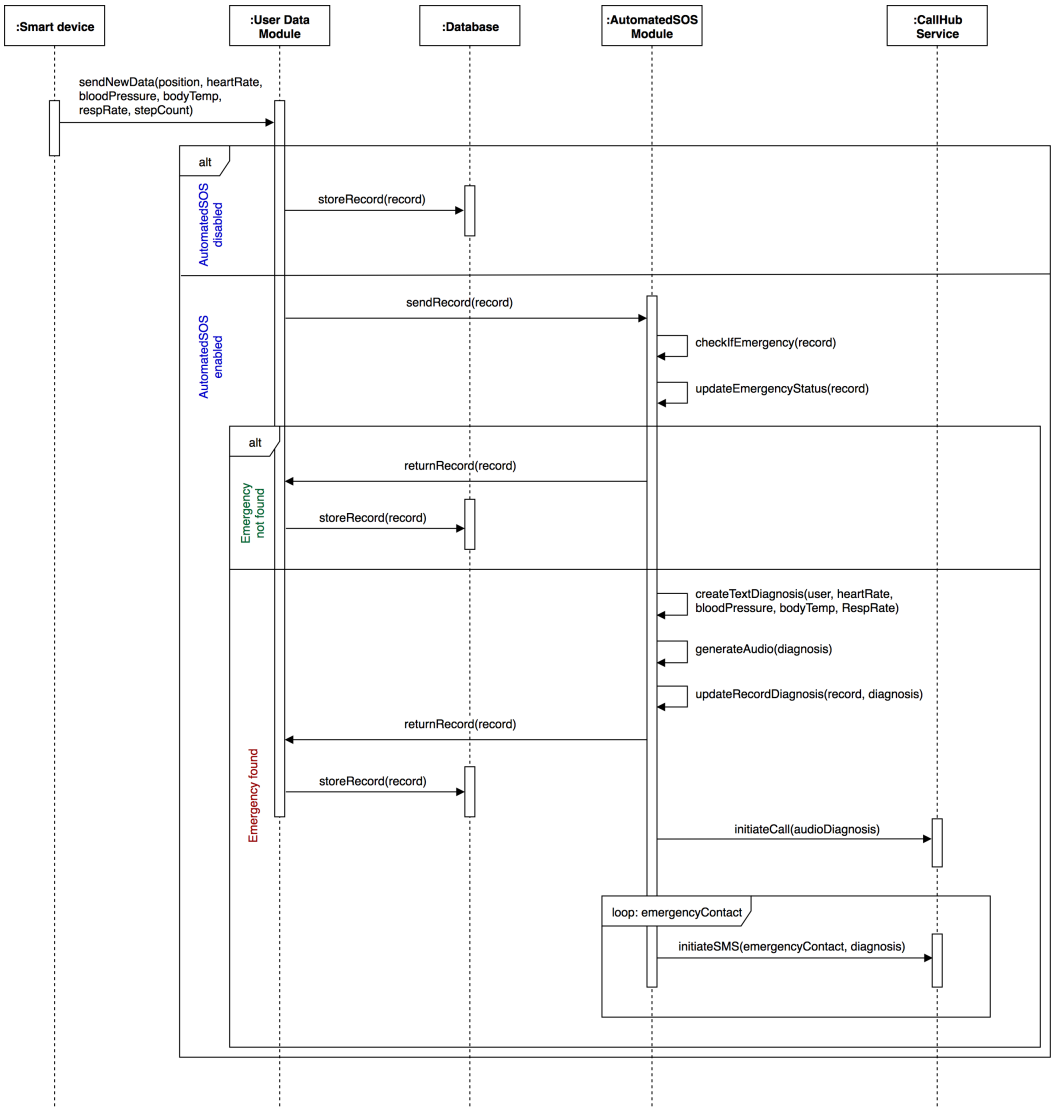


Figure 2.15: Emergency management in AutomatedSOS

2.5.7 Creation of a new run

An Organizer creates a new run filling out the form with all the details: day of the run, start time, starting point, ending point, list of intermediate points and maximum number of participants. The Track4Run Module defines the path that contains all the points selected by the Organizer through the map service and performs the checks to validate the run. If it is allowed, the module adds it to the list of scheduled run and the application shows a confirmation message to the Organizer. Otherwise, the application shows an error message.

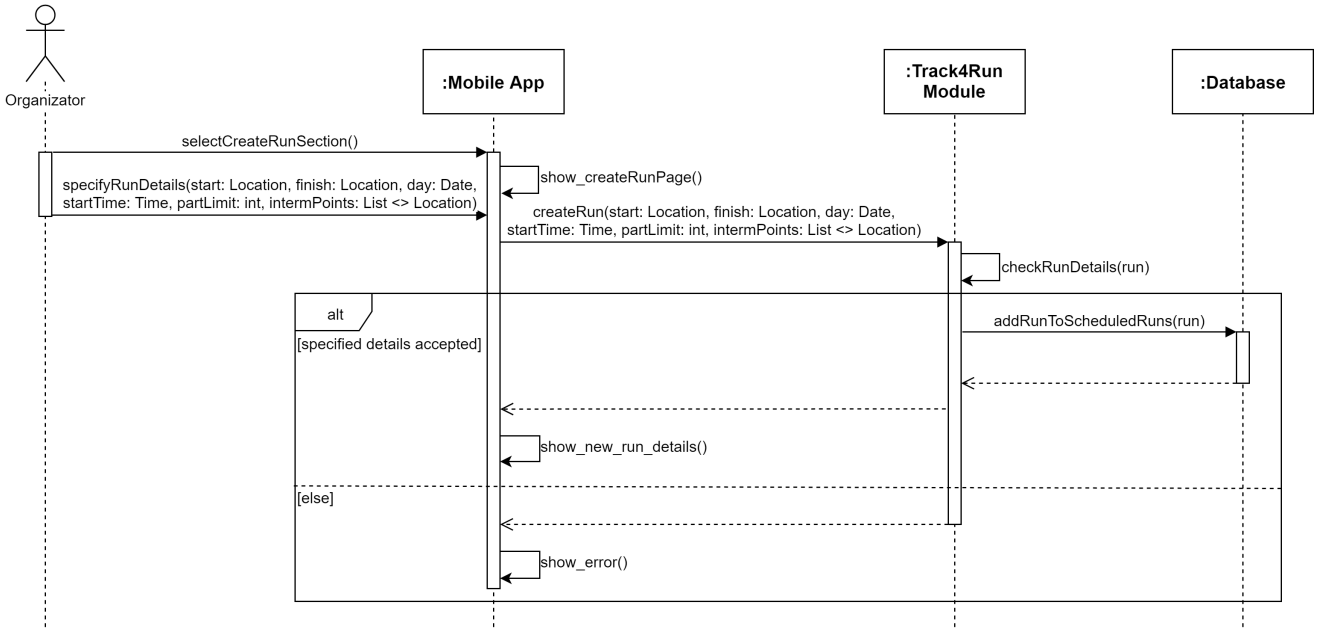


Figure 2.16: Create a new run by an Organizer

2.5.8 Enroll to a run

Users can view the list of scheduled runs performing a custom research on date, location and maximum number of participants and the Track4Run Module retrieves the runs from the database. Users can select a specific run and consult the detail of the event. At this point he/she can decide to enroll to a run: the module checks if there are already available entries and, in case of a positive response, the enrollment is allowed and the User is added to the list of participant of that run.

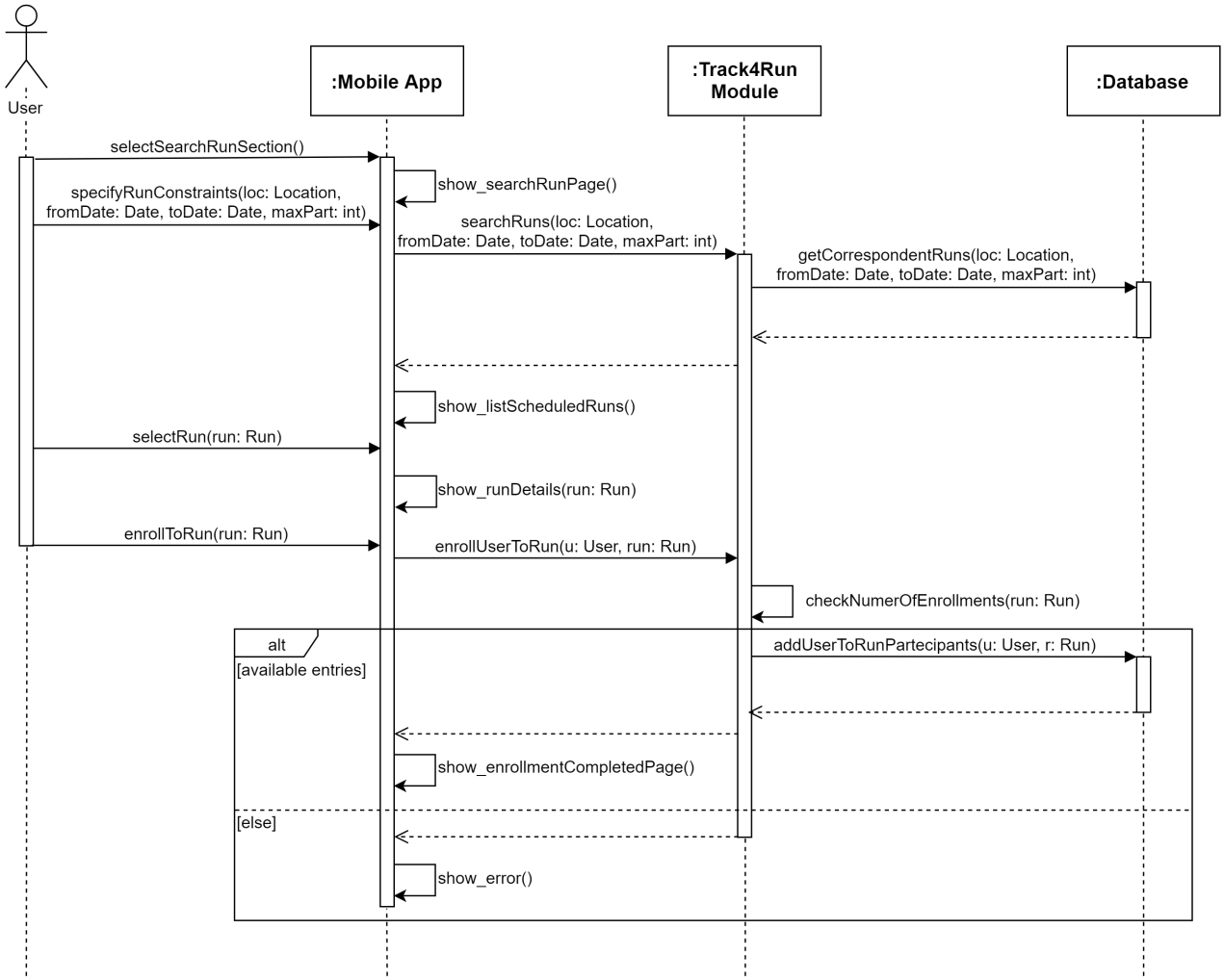


Figure 2.17: Enrollment to a scheduled run by a User

2.6 Component Interfaces

In these diagrams are shown the main methods offered by the different interfaces of the components to the Users of our application.

- The interface UserMobileServices provides all the methods that Users will call to manage their requests, data visualization, emergency services and services related to run enrollment and run organization.
- The interface ThirdPartyWebServices extends the interfaces of the Third Party Web Services subsystem. These interfaces provide all the methods relative to the request of data of individual User or group of Users and also relative to the visualization of the already accessible data.

2.7 Selected architectural styles and patterns

- Layered Architecture:

  As described in the High-level overview section, the architecture of the System is composed of four tiers: the Presentation level, the Web level, the Business Logic level and the Data level. Client executables for iOS and Android constitute the Presentation level, which will interface with the logic of the System through an Apache Web server. The database server and the external map service form the Data level, which will communicate with the server application through a firewall. A layered structure improves reuse and maintainability of the System.

- Publish-Subscribe:

  The subscription requests follow the event based paradigm (often called publish-subscribe): with this modality, a Third Party subscribes itself to the data of a User or a set of Users (group) and is notified with the new data each interval time specified, if they are available.

- Observer:

  AutomatedSOS service can be seen as an Observer to the User health status (Observables): it continuosly monitorizes them, and activate emergency functionalities when an anomaly is detected.

- Client-Server:

  The main functionalities of the System are based on the well-known client-server structure: there are two kind of clients (Web Users and Mobile application Users) that send requests to a unique server, that elaborate them passing through the Application logic tier and reply to the clients.

- Facade:

  All the complexity of the System is hidden by providing to the User simple interfaces. This is evident in the AutomatedSOS Service, where all the tasks related to the management of an emergency are responsibility of the appropriate module, and the User does not need to interact directly with them.

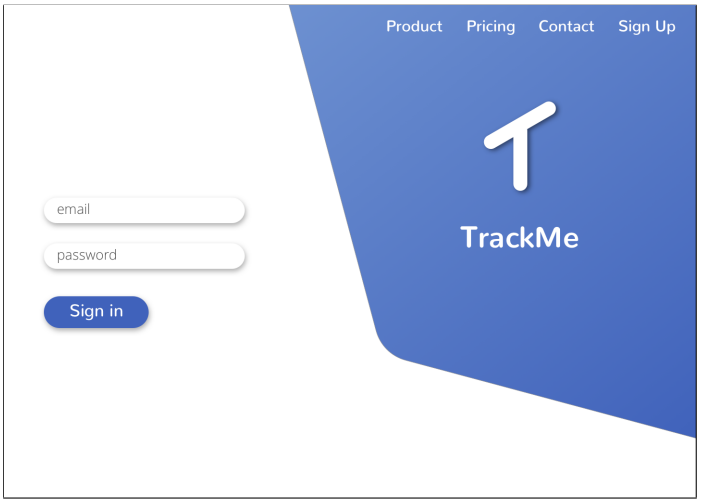
2.8 Other design decisions

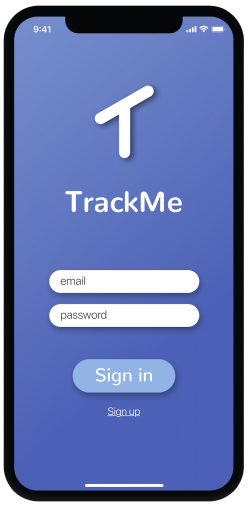
Our System uses an external map service, Open Street Map, in order to allow the visualization and browsing on an interactive map of localization data, both on Web and Mobile application.

Chapter 3

User Interface Design

The complete User interface view of the applications can be found in the RASD Document and it does not report significant changes.





3.1 UX Diagrams

The figures below are the UX Diagrams of the Mobile application and the Web application, that explain how the User interfaces for the end-Users should be.

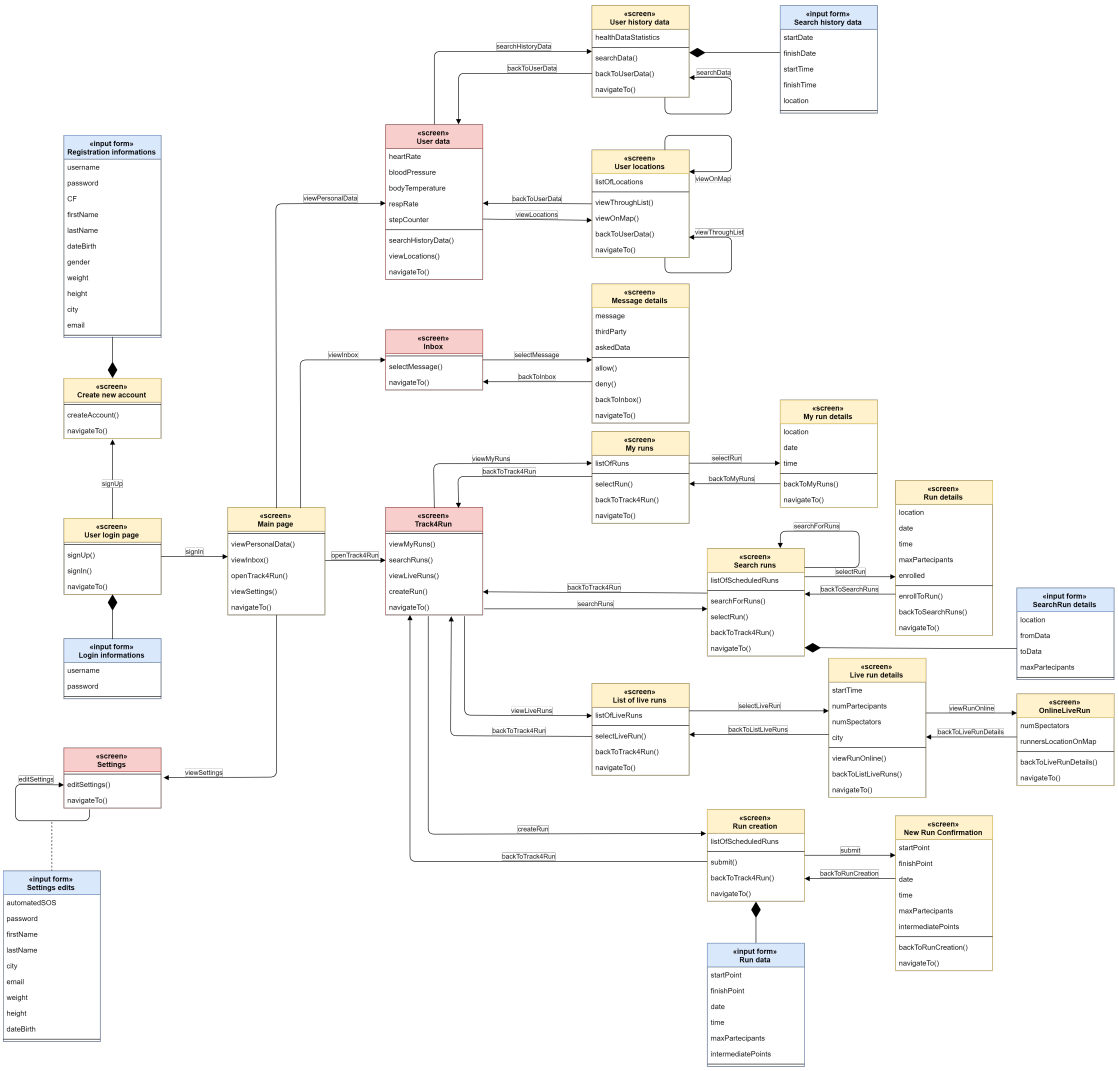


Figure 3.1: UX diagram of the User Mobile application

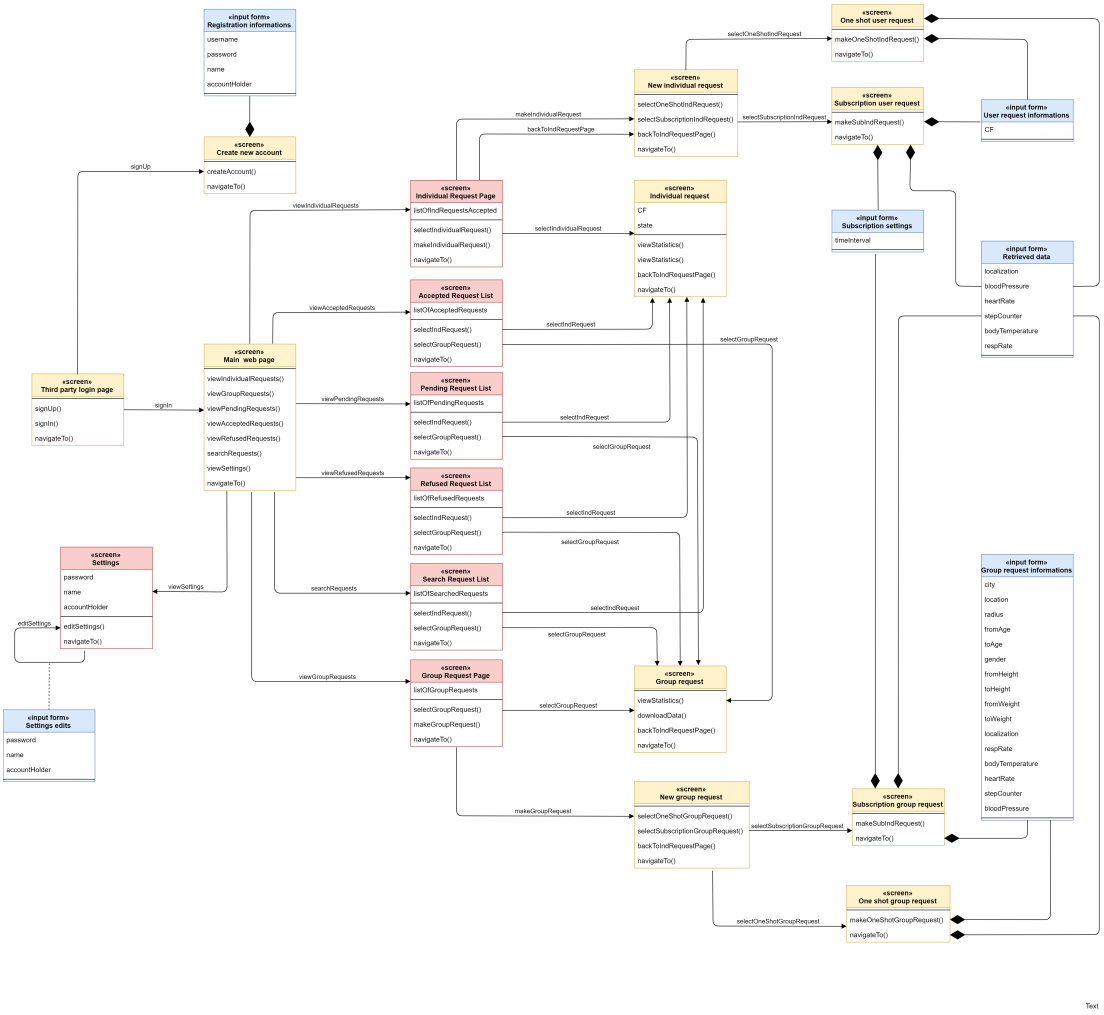


Figure 3.2: UX diagram of the Third Party Web application

The screens in red of the UX diagrams are the parts of the application that are always reachable from the bottom menu in the Mobile application and from the leftmost menu in the Web application. These screens are reachable from every page of the Mobile and Web application, except from the *User login page*, *Third Party login page* and the *Create new account* screens.

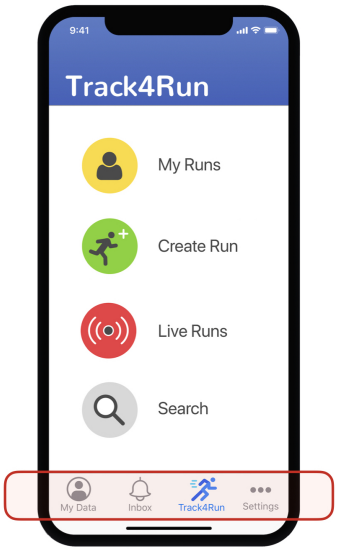


Figure 3.3: Bottom menu from which is possible to open My Data, Inbox, Track4Run and Settings in the Mobile application

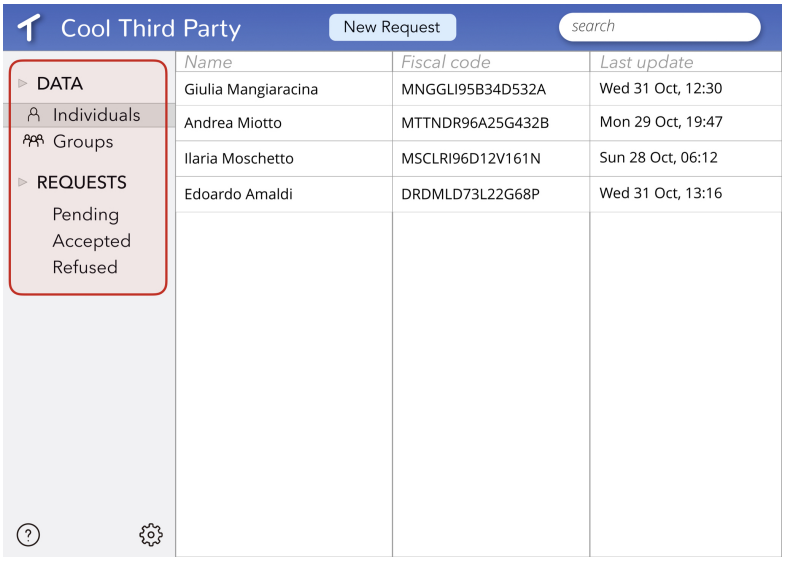


Figure 3.4: Leftmost menu from which is possible to open Individuals, Groups, Pending, Accepted and Refused requests views in the Web application

Chapter 4

Requirements Traceability

In this section is shown the mapping between the requirements stated in the RASD document and the component modules defined in this paper.

[G1]The Registered User can access to the services offered by TrackMe System with a single account.

- Mobile Application: [R1 - R2]
- Web Application: [R3 - R4]
- AutomatedSOS Module: [R5]

[G2]The Registered User can be recognized by providing his/her username and password.

- Login Module (of Mobile Application): [R6]
- Login Module (of Web Application): [R7]

[G3.1]Allow a User to manage the accesses to his/her personal data.

- Manage Request Module: [R8 - R9 - R10]

[G3.2]Allow a User to visualize his/her actual health parameters and position.

- User Data Module: [R11 - R12 - R13]

[G3.3]Allow a User to visualize his/her past data history.

- User Data Module: [R14 - R15 - R15.1 - R16 - R16.1]

[G3.4]Allow a Third Party to send an authorization request to a User for the access to the his/her data.

- Request Module:[R17 - R18 - R19]
- Manage Request Module: [R20]

[G3.4.1]Allow a Third Party to request the latest available data of a User.

- Data Module:[R21]

[G3.4.2]Allow a Third Party to request a subscription to the data of a User.

- Request Module:[R22]
- Data Module : [R23]

[G3.5]Allow a Third Party to request anonymized data of a set of Users.

- Request Module: [R24 - R25]
- Data Module : [R26]

[G3.5.1]Allow the Third party to request the latest data of a set of Users.

- Data Module : [R27]

[G3.5.2] Allow a Third Party to request a subscription to the data of a set of Users.

- Request Module:[R28]
- Data Module : [R29]

[G3.6] Allow a Third Party to visualize the available data through useful statistics.

- Data Module : [R30]

[G4.1] NHS is alerted when the User gets in a critical state.

- AutomatedSOS Module : [R31 - R32 - R33]

[G4.2]Allow the User to create a list of contacts to be alerted in case of emergency.

- AutomatedSOS Module : [R34 - R35 - R36]

[G5.1] Organizer can create a new run.

- Track4Run Module : [R37 - R38 - R39 - R40 - R41 - R42]

[G5.2] Allow a User to enroll to a run as participant.

- Track4Run Module : [R43 - R44 - R45]

[G5.3]Allow a User to visualize the list of scheduled runs.

- Track4Run Module : [R46 - R47 - R48]

[G5.4] Allow a User to visualize the list of live runs.

- Track4Run Module : [R49 - R50]

[G5.5] Allow a User to visualize on a map the positions of the participants in a live run.

- Track4Run Module : [R51 - R52]

Chapter 5

Implementation, Integration and Test Plan

This section revolves around the way in which the components and modules of the system should be developed, implemented and integrated with each other, with the objective to perform an efficient testing phase.

5.1 Implementation

The development of the TrackMe System requires a first initialization phase, in which the physical servers will be deployed and configured. The Database Server, which plays a key role in our system for it communicates with all the modules, must be configured with the definition of the database, according to the ER structure presented in chapter 2. At this point the implementation phase can begin.

The implementation will require three different teams: the first team will focus on the server side, which includes the components defined in the component view; the other two teams will develop one client each, the web Application (used by Third Parties through the web browser) and the mobile application (used by individual users through smartphones) respectively. With the client-server interfaces already defined in the Overview section, the work on the two client applications can be pursued in parallel since there is no dependency among them. The system is composed of three main services, it is useful to set an order in which they will be implemented. Data4Help is the main service offered by the system, so it will be the first to be developed. Since Data4Help data are generated by users, the development of its modules should begin as soon as possible, given that the functioning of the entire system is based on these data.

The first module to be developed should be the User data module in the user mobile service sub-system, because it offers the main functions of the entire system, allowing the acquisition of user's health and location data and their storage in the database. After that, the Third Party request Module will be implemented, followed by the User manage request module, since these two modules are strictly dependent: The interaction with the Maps API is implemented in this phase, with respect to each component. Data4Help implementation will be completed after the development of the Login Module and Data Module in the Third Party Web service Sub-system. After the completion of

the Data4Help related components, developers can shift the focus on the AutomatedSOS Module and the Track4Run Module. They are independent with each other, so they can be developed in parallel by two different teams. The AutomatedSOS Module must be integrated with the User Data Module since it depends on it. In this phase, the communication with the external CallHub service will be put in place.

5.1.1 Integration and Test Plan

The integration strategy consists fundamentally in a bottom-up approach: following the order just exposed, the modules are developed as separate entities, starting from bottom to top, and their interaction is tested gradually according to the following building schema:

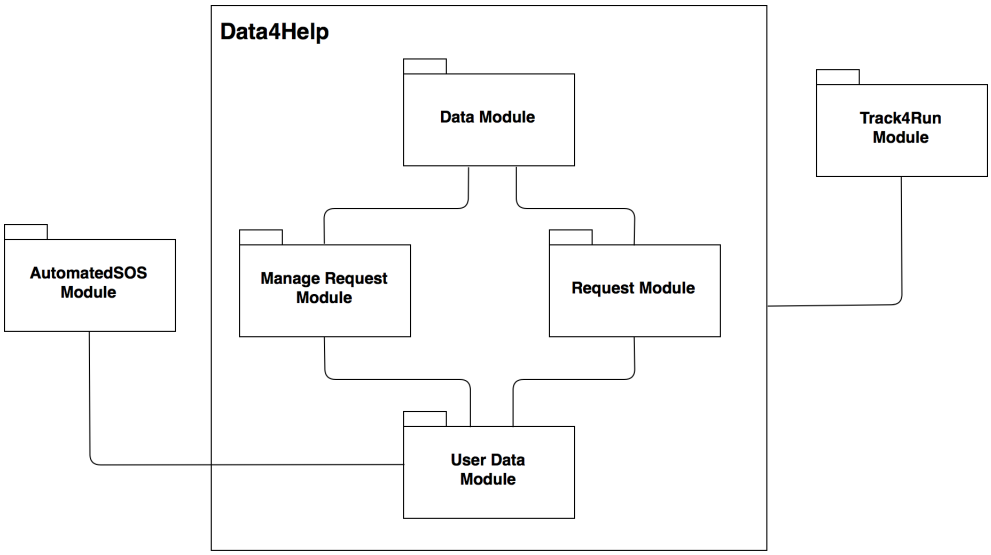


Figure 5.1: Module interdepencies

For example, after completing the manage request module and the request module, the integration between these modules must be done by testing their interaction, i.e. simulating the dispatch and the reply to the requests by the users. At the end, load testing and stress testing will be performed on the entire System.

Chapter 6

Effort Spent

- Giulia Mangiaracina:
  - [19/11/2018]Team Reunion: 3h
  - [20/11/2018]Team Reunion: 3h
  - [21/11/2018]: 2h
  - [22/11/2018]Team Reunion: 4h
  - [23/11/2018]: 1h
  - [25/11/2018]Team Reunion: 4h
  - [26/11/2018]Team Reunion: 4h
  - [28/11/2018]: 2h
  - [29/11/2018]Team Reunion: 3h
  - [02/12/2018]Team Reunion: 4h
  - [03/12/2018]: 1h
  - [04/12/2018]Team Reunion: 2h
  - [05/12/2018]: 2h
  - [06/12/2018]Team Reunion: 3h
  - [/12/2018]:

  Total: 38 h
- Andrea Miotto:
  - [19/11/2018]Team Reunion: 3h
  - [20/11/2018]Team Reunion: 3h
  - [22/11/2018]Team Reunion: 4h
  - [23/11/2018]: 1h
  - [25/11/2018]Team Reunion: 4h
  - [26/11/2018]Team Reunion: 4h
  - [27/11/2018]: 1h
  - [28/11/2018]: 2h
  - [29/11/2018]Team Reunion: 3h

- [02/12/2018]Team Reunion: 4h
- [03/12/2018]: 2h
- [04/12/2018]Team Reunion: 2h
- [05/12/2018]: 2h
- [06/12/2018]Team Reunion: 3h
- [/12/2018]:

Total: 38 h

- Ilaria Moschetto:
  - [19/11/2018]Team Reunion: 3h
  - [20/11/2018]Team Reunion: 3h
  - [21/11/2018]: 2h
  - [22/11/2018]Team Reunion: 4h
  - [21/11/2018]: 1h
  - [25/11/2018]Team Reunion: 4h
  - [26/11/2018]Team Reunion: 4h
  - [27/11/2018]: 2h
  - [29/11/2018]Team Reunion: 3h
  - [02/12/2018]Team Reunion: 4h
  - [03/12/2018]: 2h
  - [04/12/2018]Team Reunion: 2h
  - [05/12/2018]: 2h
  - [06/12/2018]Team Reunion: 2h
  - [/12/2018]:

  Total: 38 h