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 on the architectural choices, components and their runtime behavior, the interfaces that they offer, the design patterns used, so that to matching the requirements and goals stated in the Requirements Analysis and Specifications Document, and a section dedicated to Integration and Testing.

1.2 Scope

TrackMe is a Software that provides three different services: Data4Help, AutomatedSOS and Track4Run. Data4Help collects data of individuals through smartwatches and similar devices. Registered users can visualize their health and location data through their mobile devices, while third parties can access data, after authorization,in real time 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 user 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:** 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
- **The 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 our 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 Beat, blood Pressure, Step counter, etc.
- **Request:** A request is performed by a Third Party and concerns normal 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 request the Third Party can specify which data it want to receive, among health data available (heart rate, blood pressure, step counter, blood pressure etc) available. 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. We divide them between scheduled run, that are runs that has been planned and will take place in the future, and are in the run list, and live run, that are those that are taking place in this moment, are in the live run list and can be followed live.
- **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 follows 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 showed the main architectural design choices taken for the system. In the high level architecture, the system is described with a layered structure of 4 tiers; in the component view is specified the role of each component within the system and their interaction; 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 through sequence diagrams, the components stated previous interact and perform the main functionalities of the system.
- User Interface Design:
  This chapter explains the main user interfaces used in the client layer through mockup and UX diagram

- Requirements Traceability:
  In this section there is presented the mapping between the requirements stated in the RASD document and the components of the system
- Implementation, Integration and Test Plan:
  this section describes the plan that will be used when each component will be develop and how it will be integrated with the others components.
- Effort Spent:

1.5 Reference Documents

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

Chapter 2

Architectural Design

2.1 Overview: High-level components and their interaction

TrackMe is a system that stores user 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

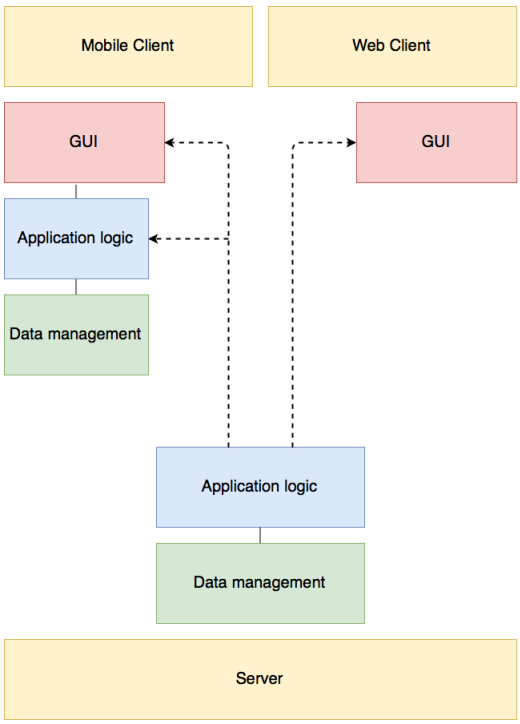


Figure 2.1: Overview of the *TrackMe* system

The picture above describes the client-server architecture of the system. The mobile and web client 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.

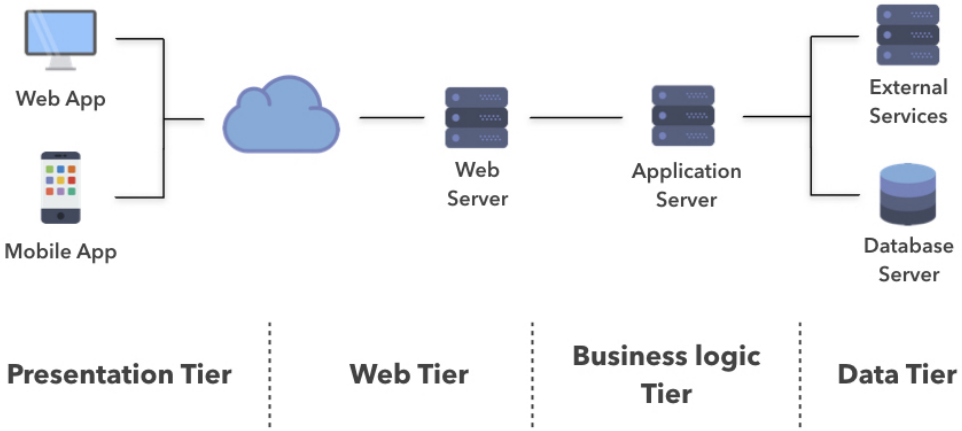


Figure 2.2: High-level architecture of the *TrackMe* system

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, corresponding to the two platforms used by the two different kind of customers of our service: The "Web Application" is for the Third parties, which 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.

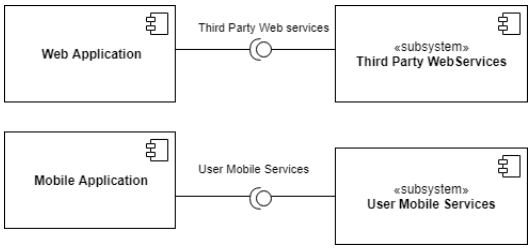


Figure 2.3: Component view of *TrackMe* System

2.2.1 Component view of the Third Party Web Services

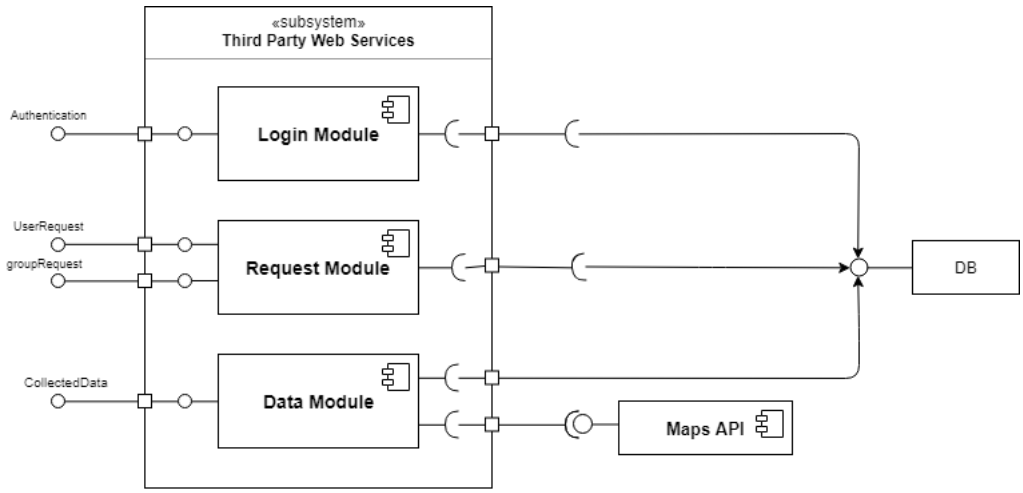


Figure 2.4: Third Party Web Service component

- The Login Module manage the operations of authentication, allowing to the Third Party to be recognized and thus use the services of the System. It interacts with the database to check the credentials and retrieve the account information.
- The Request Module is responsible of 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 an user request to notify the event, and it performs the security check in case of group request, allowing 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 frequency desired. It retrieves the selected data querying the database, that stores all the user records. In case of group request, it anonymize the data. This module is connected to Maps API to allow the consulting of user location data on an interactive map.

2.2.2 Component view of the User Mobile Service

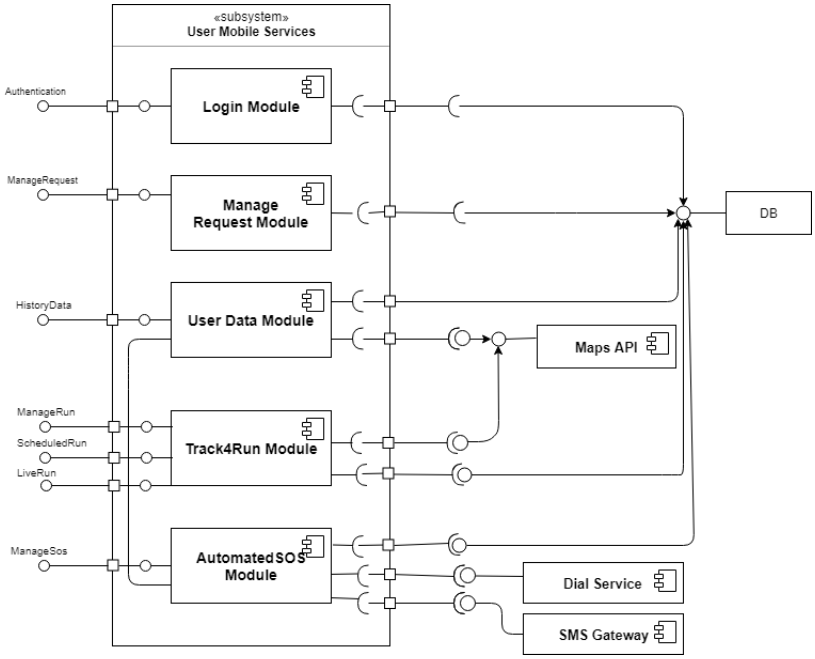


Figure 2.5: User Mobile Services component

- The Login Module manage the operations of authentication, allowing to the User to be recognized and thus use the services of the System. It interacts with the database to check the credentials and retrieve the account information.

- 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, dispatching 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 to the system database, that stores the records.

  It offers the interface to allow the User to visualize his/her History Data stored in the database, querying it with the time and location setting selected by the user, 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, visualize the list of the planned runs and follow live runs. It interacts with the Maps API module to manage the location data, and with the database to store new created runs and retrieved all the details about the runs.

- The AutomatedSOS module provides the functionalities concerning the AutomatedSOS system: it offers to the user the interface to enable or disable the service, to manage the emergency contact 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 launches an emergency state if it detects some anomalies in the health parameters. 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 contact list with a text message.

2.3 Deployment View

This section is devoted to explain in detail 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 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. Finally, the *Application Server* needs to interact with the external map service, which is of paramount importance for the *TrackMe* system because it provides important utility for the final users. The choice is *OpenStreetMap* for reliability and privacy.

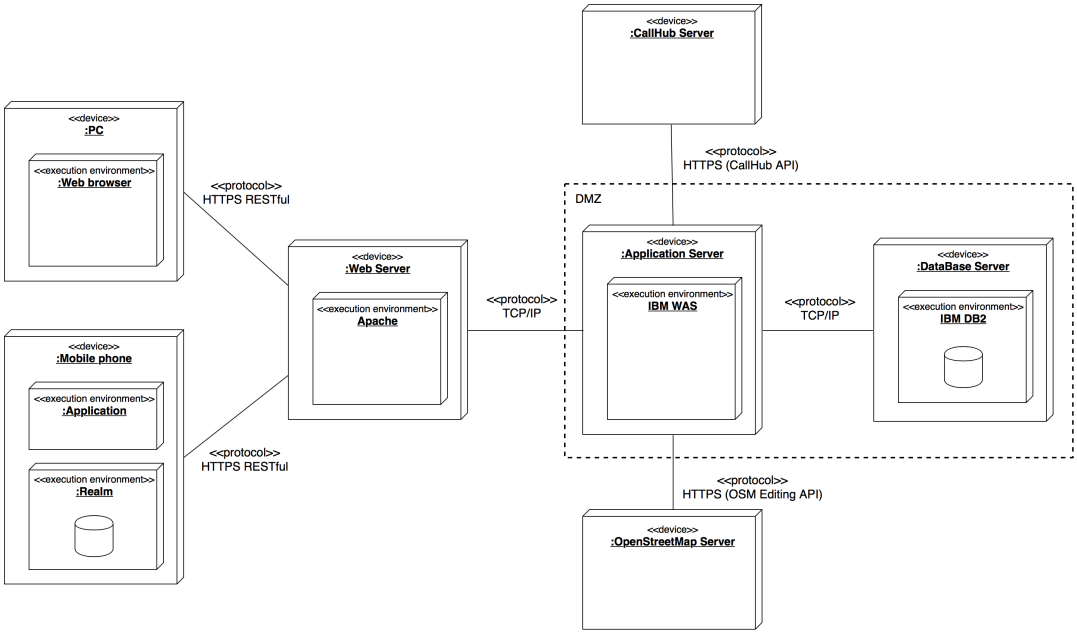


Figure 2.7: Deployment of the *TrackMe* system

2.4 Database View

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

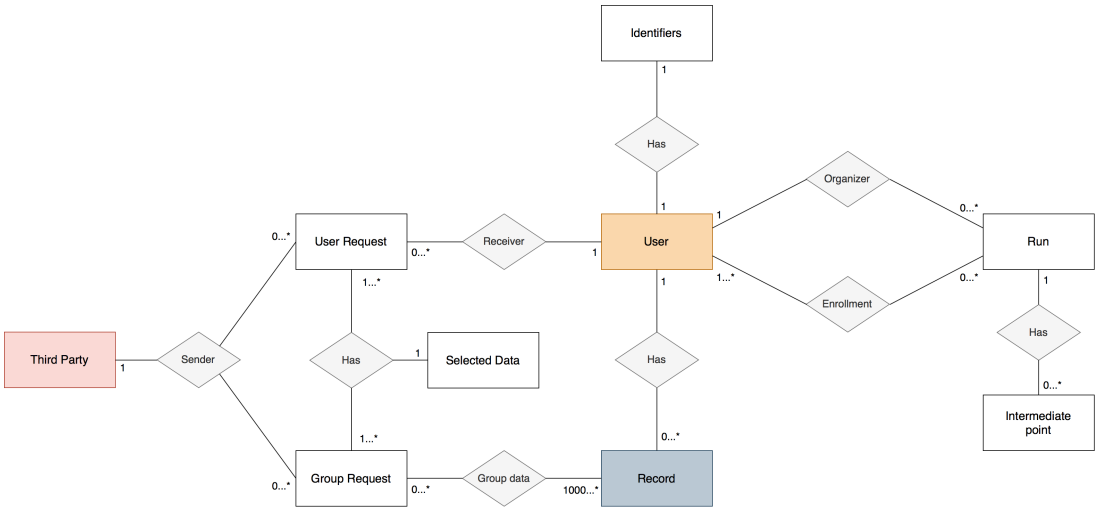


Figure 2.8: Simplified ER diagram for the *TrackMe* system

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 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.

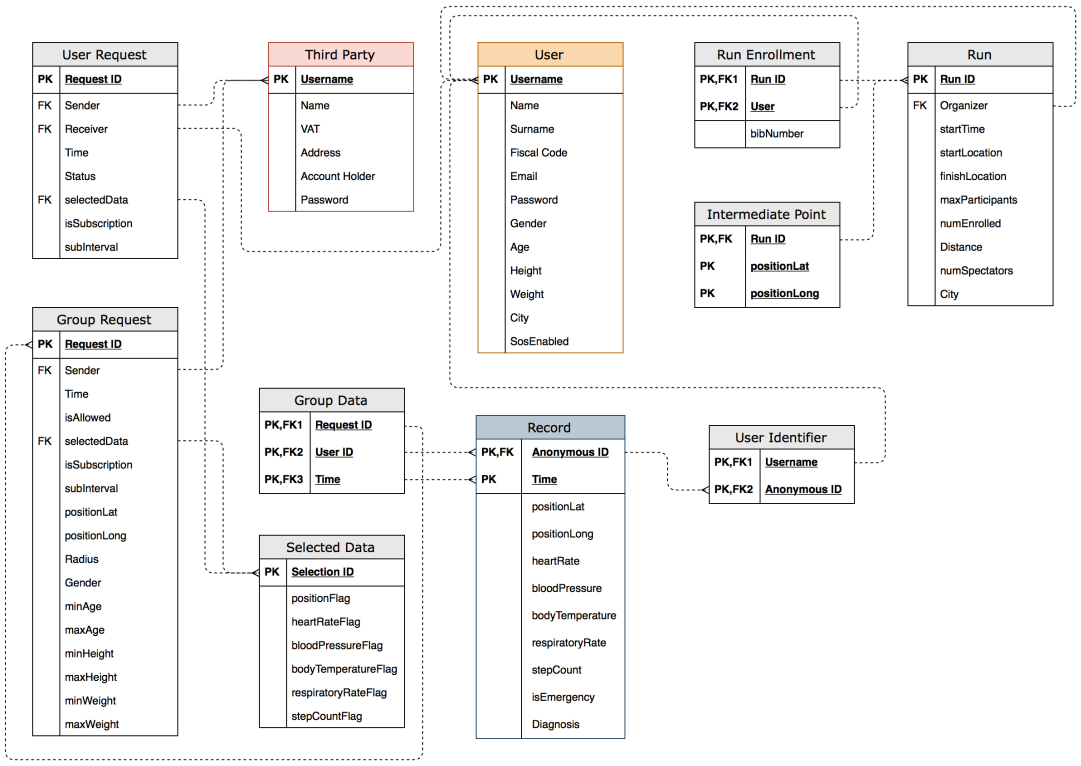


Figure 2.9: Tables and relative attributes of the *TrackMe* database

Figure 2.7 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 the 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 cain either become Accepted or Refused, depending on the user choice. The *Status*

attribute is not present in the *Group Request* because this type of request does not require user 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 request 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 sensor 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 on 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.

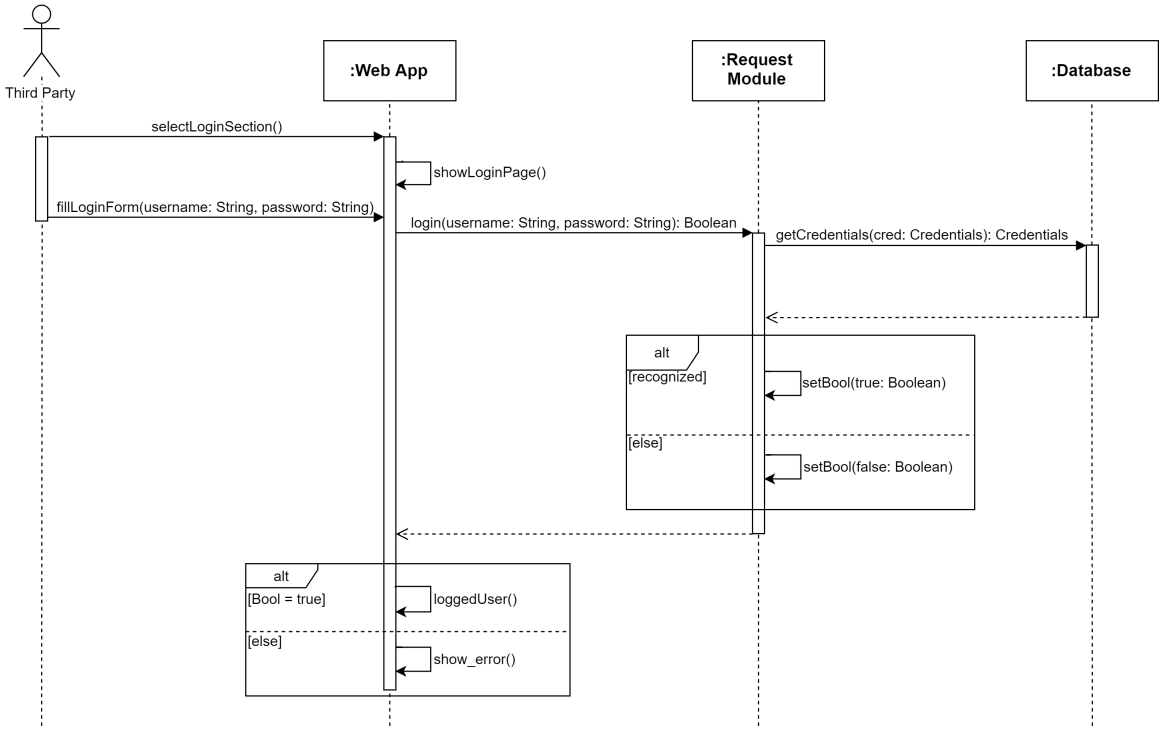


Figure 2.10: Login sequence diagram

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, with subscription mode, in the individuals section; 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.

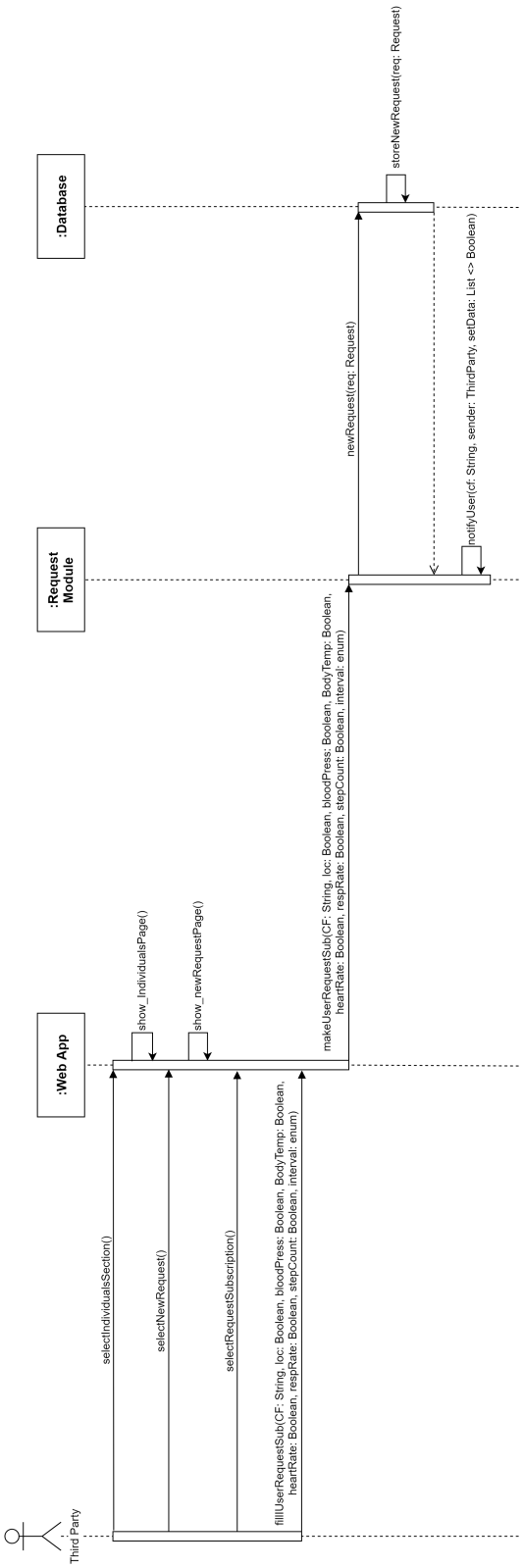


Figure 2.11: User request by a Third Party

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 not more pending.

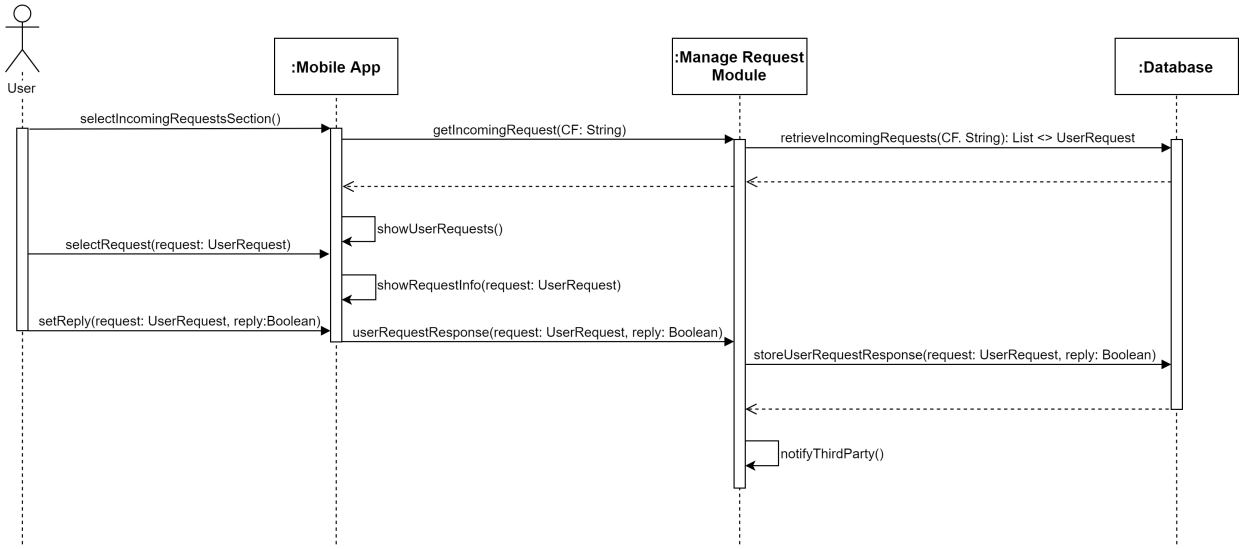


Figure 2.12: User acceptance of a request made by a Third Party

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 information from the database, creating a group with data that matche 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.

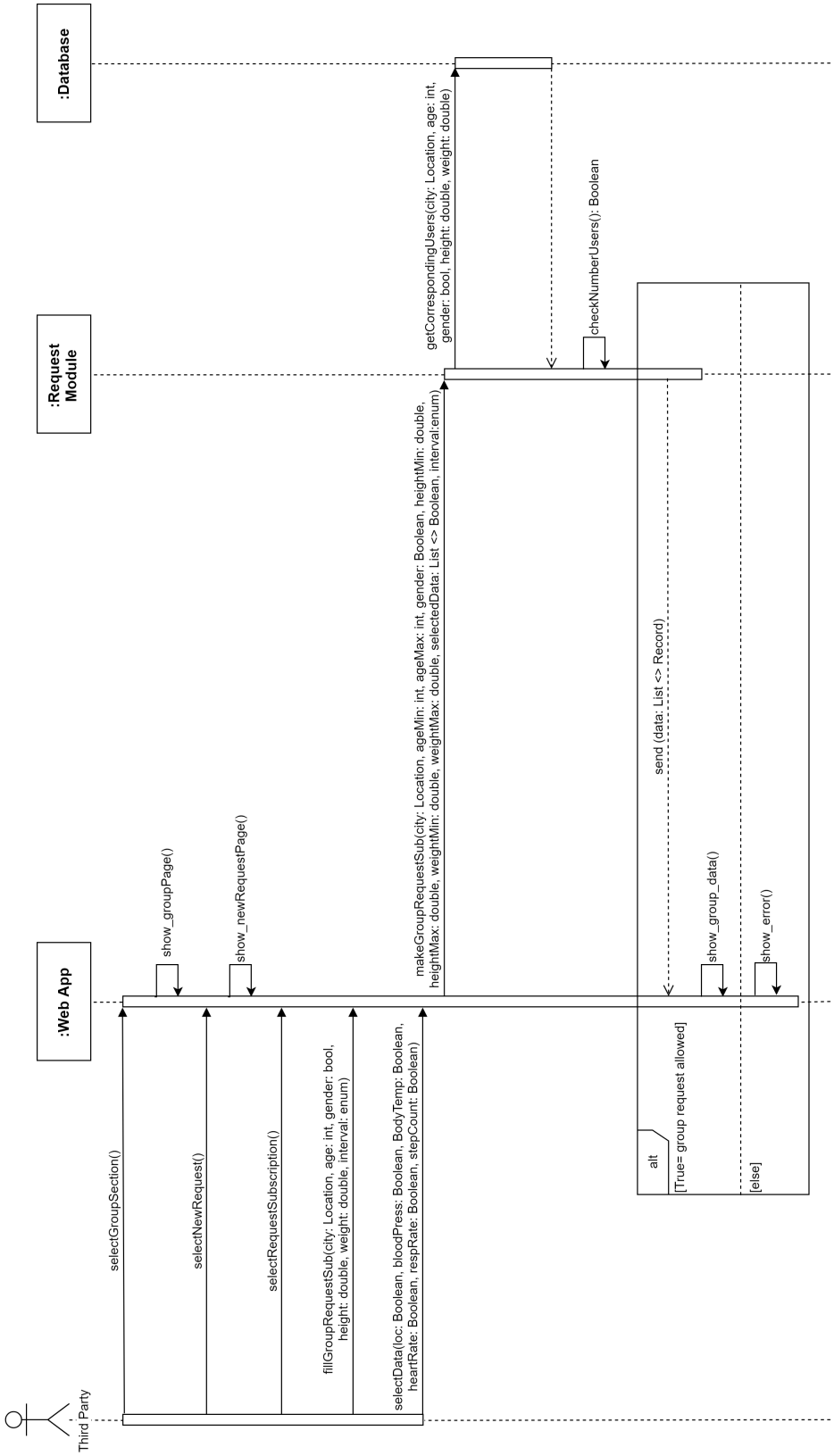


Figure 2.13: Group request by a Third Party

2.5.5 Visualization of User data

The Third Party can see the list of user performed requests, divided for 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.

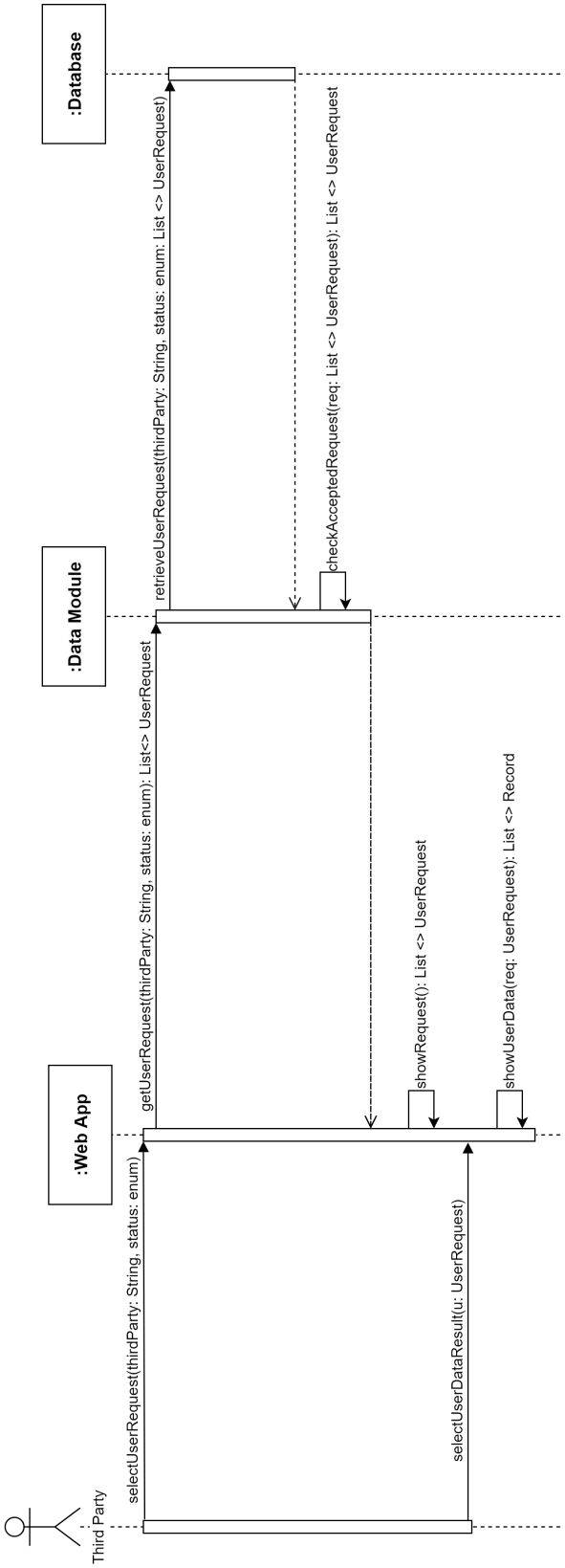


Figure 2.14: Visualization of data of a User that has already accepted the request

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. (continues...)

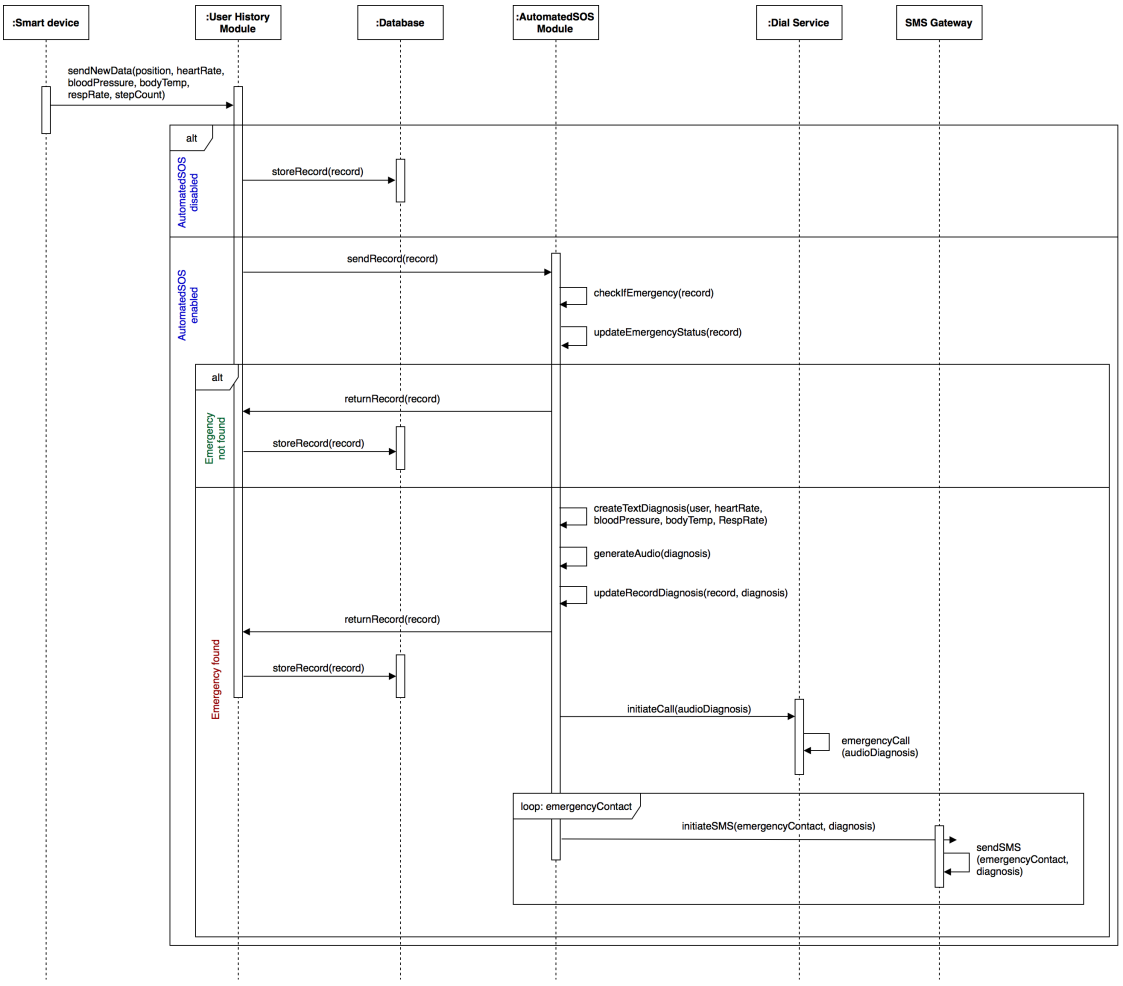


Figure 2.15: Emergency management in AutomatedSOS

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 max 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 to the list of planned run, and the application show a confirmation message to the organizer. Otherwise, the application show an error message.

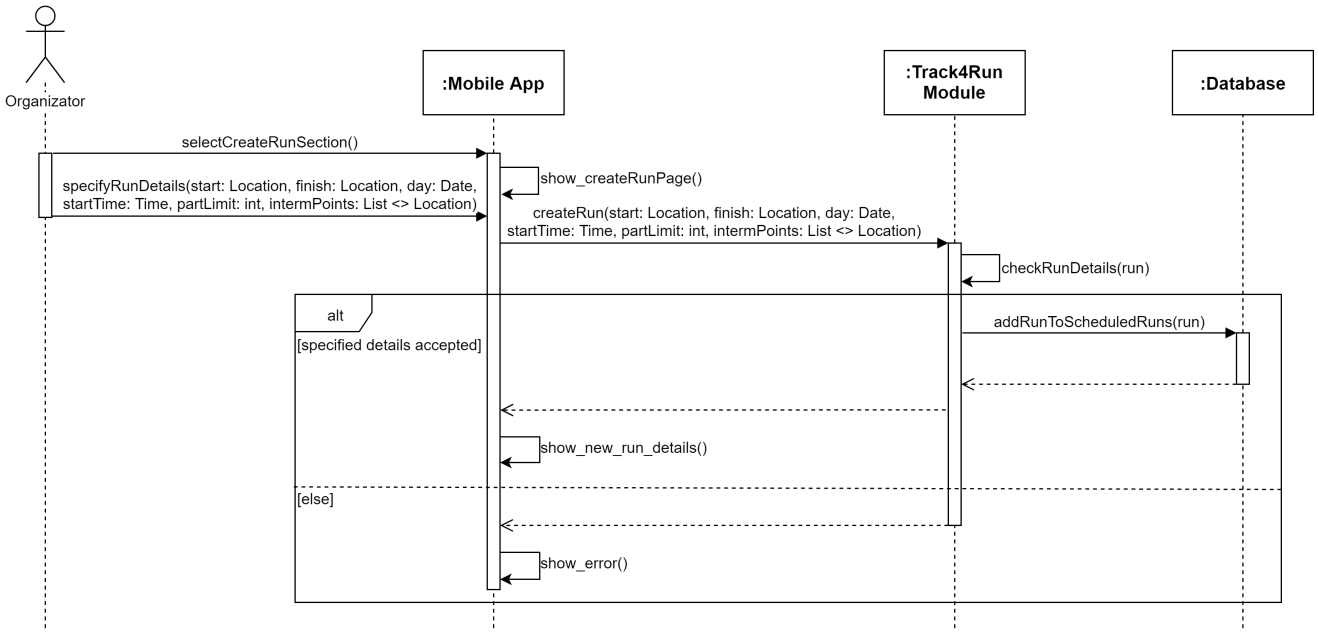


Figure 2.16: Create a new run by an Organizer

2.5.8 Enroll to a run

An user can views the list of planned run, performing a custom research on date, location and max number of participants; the Track4Run Module retrieves the runs from the database. The user can select a specific run and consults the detail of the event. At this point he/she can decide to enroll to the run: the module checks if there are yet available entries, and the enrollment is allowed adds the user to the list of participant to that run.

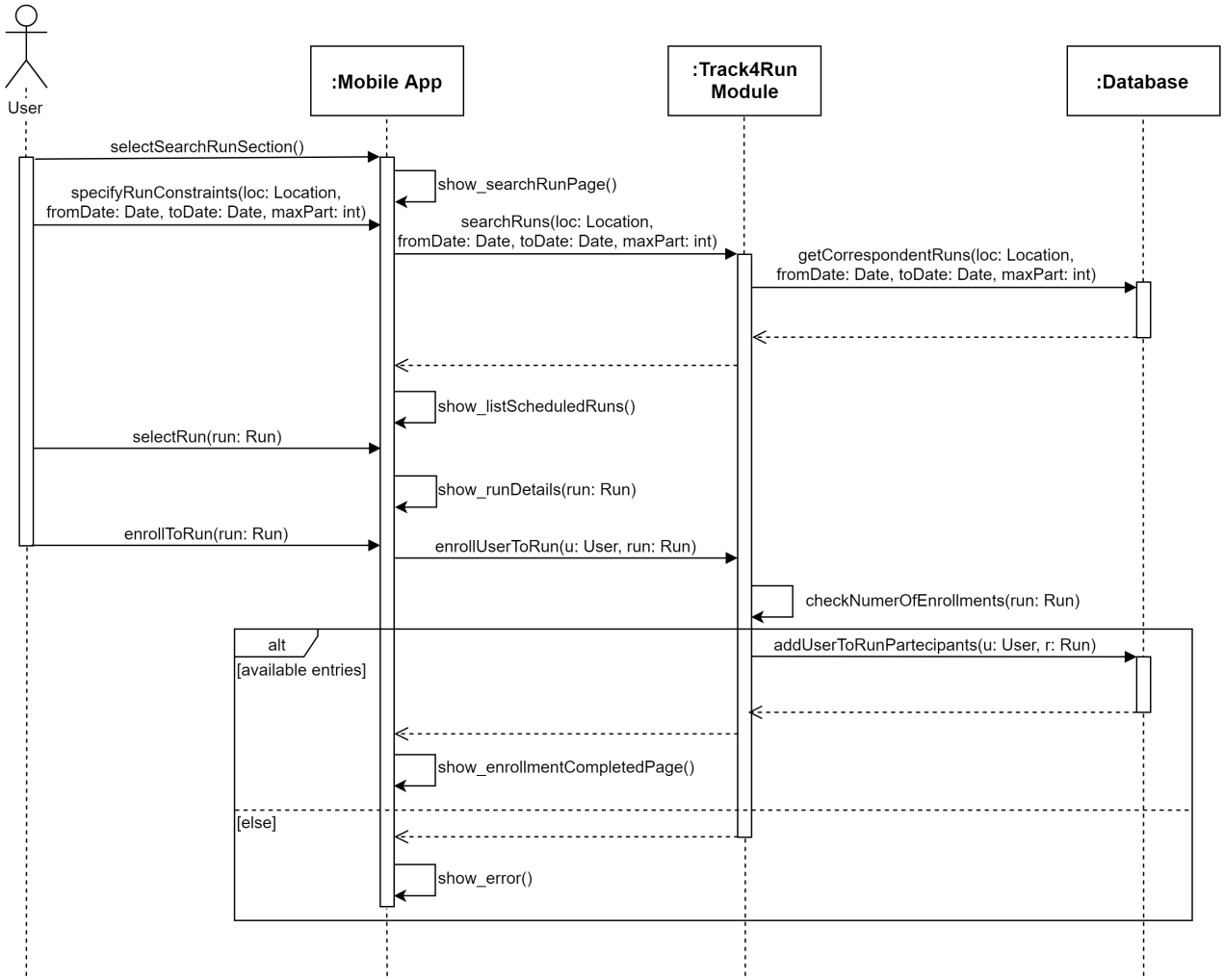


Figure 2.17: Enrollment to a scheduled run by a User

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 to users and group of users and also 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 4 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 subscribe itself to the data of a user or a set of user (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 client.

- 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.

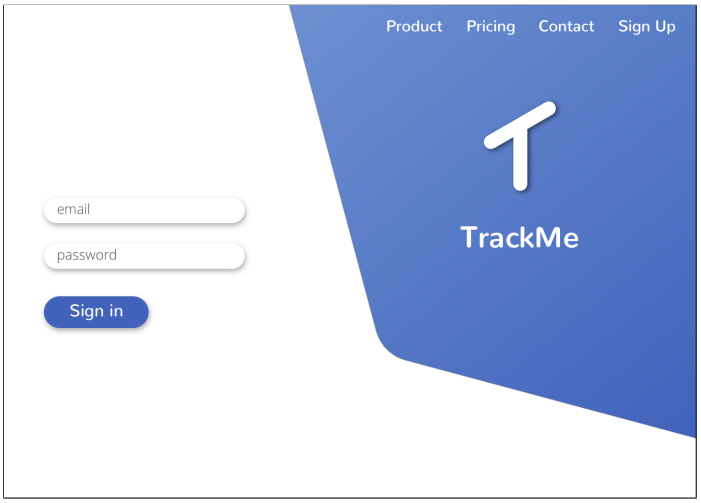
2.8 Other design decisions

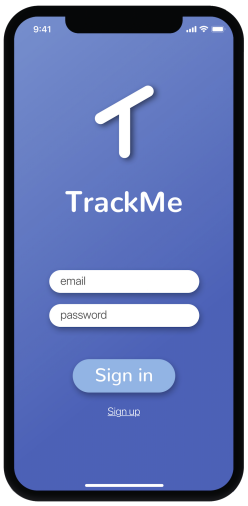
Our System uses an external map service, Open Street Map, in order to allow the visualization and browsing on an interactive map of localization data, both on web app and mobile application.

Chapter 3

User Interface Design

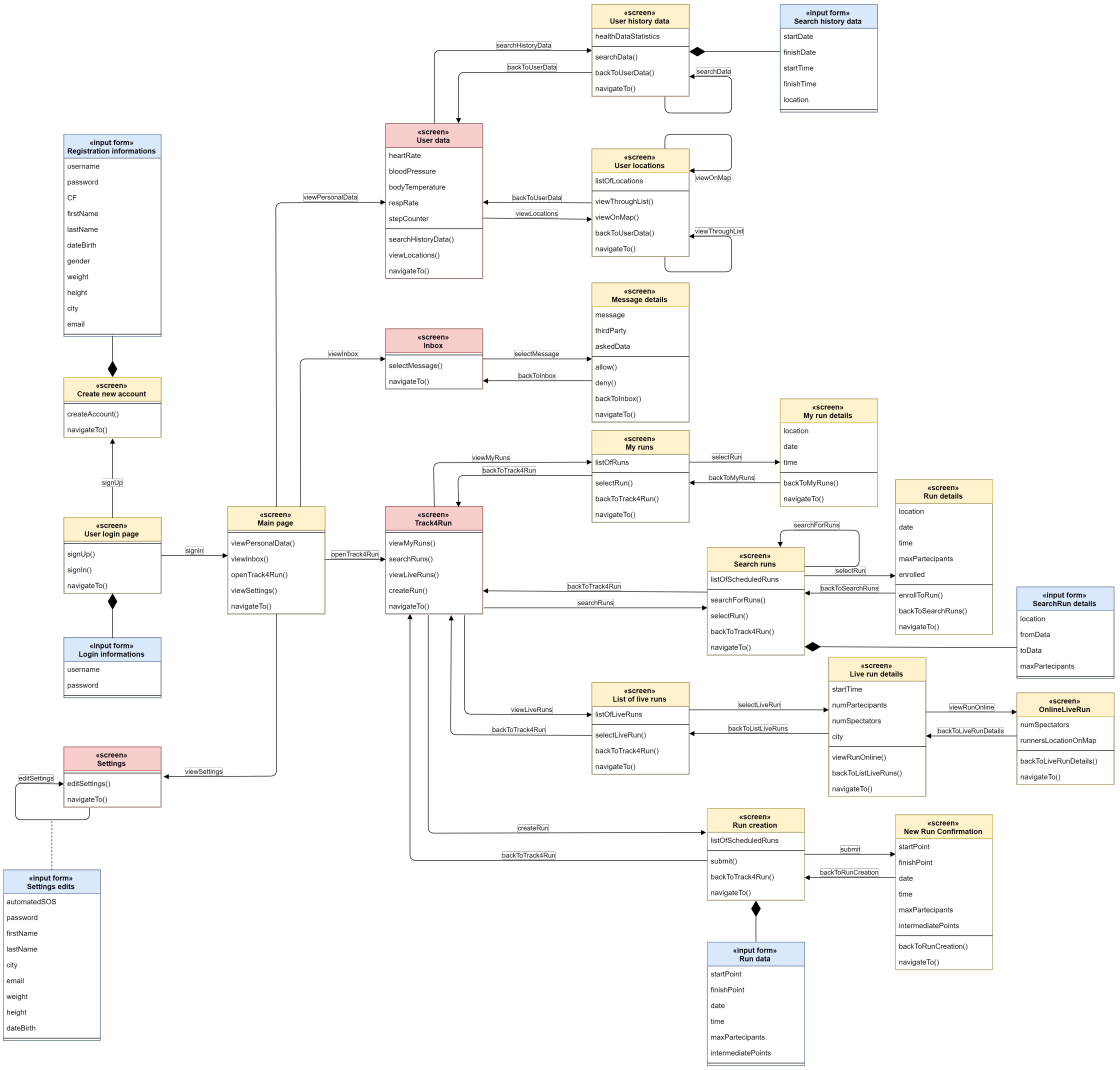
The complete User interface view of the applications can be found in the RASD Document, and it does not report significant changes.





3.0.1 UX Diagrams

The figures below are the UX Diagrams of the web application and the mobile application, that explain how the user interfaces for the end-users should be.



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-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 Users.

- Request Module:[R22]
- Manage Request Module: [R20]
- 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 the set of Users.

- Data Module : [R27]

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

In this section will be expose the way in which the components and modules of the system should be developed, implemented and integrated each other, to perform an efficient testing phase.

5.0.1 Implementation

The development of the TrackMe System requires a first initialization phase, in which will be deployed and configured the physical servers, the security measures and the protocols which allows the component to communicate within the internal net and to the external one. Then, will be deployed the database Server (which plays a key role within our system, since it communicates with all the modules), accompanied by the DBMS, ready and configured, and it will proceed with the definition of the database , according to the ER structure stated in chapter 2 (so the Data layer should be done).

At this point can begin the real implementation phase, concerning the development of the business logic tier, inside the application server.

The implementation phase should develop two software-based application: the first is a web application, used by Third Parties, and the second one is a mobile application, used by users; thus, the two sub-projects can be entrusted to two different development teams that interacts each other and work in parallel, each with specific programming skills. Since Data4Help data are generated by users, the development of the mobile application should begin as soon as possible, given that the Web Application bases its work on those data. In this way possible problems will be find out first. The first module that will be develop in the user mobile service sub-system should be the User data module, because it offers the main function of the entire system, allowing the acquisition of user's data (health and location data) through user smartwatch's sensors and their storage in the database. After that, The Web application's team can start to develop the request Module of the Third Party Web service Sub-system, while the other team will focus on the manage request module, since these two modules are dependent. The interaction with the Maps API is implemented in this phase, within each component development. After the completion of these parts, the two teams can work in parallel, proceeding with the development of the remaining modules; in particular, the AutomatedSOS Module and The track4Run Module are independent, so can be develop by two different groups within the mobile application team. The AutomatedSOS Module must be connected to

the User Data Module since it is dependent to it; here are performed the connection of the module with the dial service and to the SMS gateway.

5.0.2 Integration and Test Plan

The strategy chosen for the integration plan is the bottom-up one.

Chapter 6

Effort Spent

- Giulia Mangiaracina:
  - [17/10/2018]Team Reunion: 1h
  - [19/10/2018]: 1h
  - [22/10/2018]Team Reunion: 2h
  - [23/10/2018]Team Reunion: 3h
  - [24/10/2018]: 1h
  - [25/10/2018]Team Reunion: 2h
  - [28/10/2018]Team Reunion: 4h
  - [30/10/2018]: 2h
  - [31/10/2018]Team Reunion: 2h
  - [2/11/2018]Team Reunion: 5h
  - [4/11/2018]: 3h
  - [5/11/2018]Team Reunion: 4h
  - [6/11/2018]: 2h
  - [7/11/2018]Team Reunion: 5h
  - [8/11/2018]Team Reunion: 2h
  - [9/11/2018]: 1h

  Total: 40 h
- Andrea Miotto:
  - [17/10/2018]Team Reunion: 1h
  - [20/10/2018]: 1h
  - [22/10/2018]Team Reunion: 2h
  - [23/10/2018]Team Reunion: 3h
  - [24/10/2018]: 1h
  - [25/10/2018]Team Reunion: 2h
  - [26/10/2018]: 2h
  - [28/10/2018]Team Reunion: 4h

- [31/10/2018]Team Reunion: 2h
- [2/11/2018]Team Reunion: 5h
- [5/11/2018]Team Reunion: 4h
- [6/11/2018]: 2h
- [7/11/2018]Team Reunion: 5h
- [8/11/2018]Team Reunion: 2h
- [9/11/2018]: 3h
- [10/10/2018]: 1h

Total: 40 h

- Ilaria Moschetto:
  - [17/10/2018]Team Reunion: 1h
  - [19/10/2018]: 1h
  - [22/10/2018]Team Reunion: 2h
  - [23/10/2018]Team Reunion: 3h
  - [25/10/2018]Team Reunion: 2h
  - [26/10/2018]Team Reunion: 2h
  - [27/10/2018]: 1h
  - [28/10/2018]Team Reunion: 4h
  - [30/10/2018]: 2h
  - [31/10/2018]Team Reunion: 2h
  - [1/11/2018]: 1h
  - [2/11/2018]Team Reunion: 5h
  - [4/11/2018]: 2h
  - [5/11/2018]Team Reunion: 4h
  - [7/11/2018]Team Reunion: 5h
  - [8/11/2018]Team Reunion: 2h
  - [9/11/2018]: 1h

  Total: 40 h