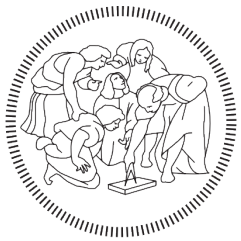


POLITECNICO
MILANO 1863

# RASD

*Software Engineering 2 Project - TrackMe*


**Authors**

- Daniele Montesi - *912980*
- Nicola Fossati - *915244*
- Francesco Sgherzi - *915377*

# Contents

# 1 Introduction

## 1.1 Purpose

TrackMe is a company that wants to offer the following services to individuals and to third party companies:

- Data4Help, that allows people to register and provide their data to TrackMe through devices such as smartwatches or smartphones, while allows companies to get data from a particular group of people;
- AutomatedSOS, that allows individuals, mainly elderly people, to register in order to have their health monitored and to receive an immediate assistance in emergency cases;
- Data4Run, a sport-oriented service that tracks athletes participating in a run. It allows organizers to define the path of the run, participants to enroll to the run, and spectators to see on a map the position of all runners during the run. Track4Run will exploit the features offered by Data4Help

## 1.2 Scope

The *The World and the Machine* approach is used in defining the scope of the project. By defining the real world entities that interact with the system and the properties of the system itself we can determine the intersection between the two sets: the *shared phenomena*.

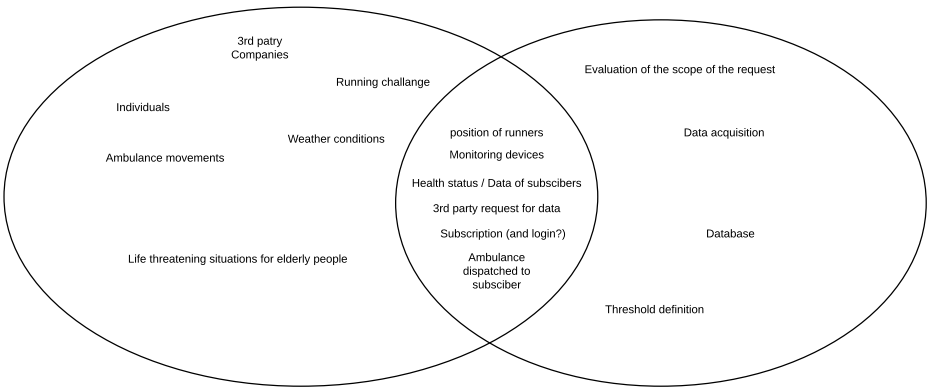



Figure 1: The World and the Machine diagram

The system-to-be uses 4 components with different roles in order to work:

- **Data4Help SmartWatch App**: Acquires the data from the smartwatch sensors (heart rate, sleep quality, position, phisical activities) and sends them via Bluetooth to the Data4Help Mobile App.
- **Data4Help Mobile App**: Gathers data from the smartwatch, shows various statistics, and sends them to the Data4Help Core Database. Each user can choose which service subscribe to.
- **Data4Help Website**: Gives third-party companies the ability to request data, either anonymized or user specific. Moreover, it allows run organizers to define the path of the run and the spectators to see the position of all runners on a map.
- **Data4Help Core**: is intended to connect all other components together providing the logic of the application. It is also responsible for the acceptance of all third-parties requests of data. It also evaluate health status of individuals deciding whether is at risk or not.

The list below shows the main goals the system should be able to accomplish:

- **G1**: The system should be able to show acquired data via the Mobile App and the Website.
- **G2**: The system should allow users to register by providing his Fiscal Code or his Social Security Number, an email and a password.
- **G3**: The system should allow companies to register.
- **G4**: The system should allow registered companies to request data from an anonymized group of individuals, only if individuals in the group are more than 1000.
- **G5**: The system should allow registered companies to request data from an individual person, only if individuals accept the request.
- **G6**: The system should let companies to be able to pay through the system in order to buy data queries/subscribe to plans.
- **G7**: The system should allow a company to subscribe to new data and receive them as soon as they are produced.
- **G8**: The system should be able to monitor user's health parameter.
- **G9**: The system should be able to react to the lowering of the health parameters below threshold in less than 5 seconds and send the position of the person to the ambulance system.
- **G10** The system should allow run organizers to register.

- **G11** If a run organizer is registered, it can define a run i.e. it can define the path that the participants should follow.
- **G12** A user should be able to enroll to a run, if he is close to the starting point and has a valid inscription code.
- **G13** Spectators of a run should be able to see each participant's position on a map.
- **G14** The system should allow users to log in
- **G15** The system should allow companies to log in
- **G16** The system should allow run organizers to log in

## 1.3 Definitions, Acronyms, Abbreviations

| | |
|---|---|
| **i.e.** | *Id est*, that is |
| **w.r.t** | with respect to |
| **w.l.o.g.** | without loss of generality |
| **Company** | Third party company |
| **BLE** | Bluetooth Low Energy |
| **RAM** | Random Access Memory |
| **UI** | User Interface |
| **GPS** | Global Positioning System |
| **API** | Application Programming Interface |

## 1.4 Revision history

## 1.5 Reference documents

—**REFD1**— The World And The Machine

—**REFD2**— WearOS

—**REFD3**— GlassFish Minimum Requirements

## 1.6 Document structure

This RASD document is made of the following parts:

- **Introduction**: this section provides a general and low-detailed description of the system-to-be.
- **Overall description**: this section provides general aspects of the system, like interfaces, constraints, domain assumptions, software dependencies and users' characteristics.
- **Specific requirements**: this section provides scenarios, use cases and a set of diagrams in order to represent the functionalities of the system-to-be.
- **Appendix**: this section contains the Alloy model used to model and verify the system, a list of the tools used to develop this document and the working hours tracking table.

# 2 Overall description

## 2.1 Product perspective

Data4Help Core is the central component of the system that connects all the other parts of the structure.

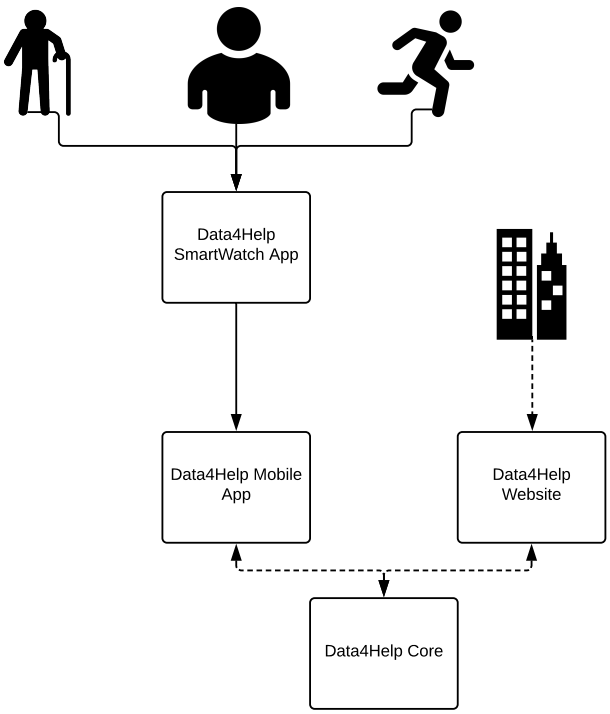



Figure 2: Use flow diagram

The Smartwatch App should be directly connected to the Mobile App to acquire data and let them be shown in the smartphone. In turn, the Mobile App has to communicate with the Data4Help Core that will register the data of the user. The Data4Help Website should communicate with the Data4Help Core as well in order to query directly on the company database. **TODO: A. Product perspective: here we include further details on the shared phenomena and a domain model (class diagrams and state charts)**

### 2.1.1 State charts

**Query request** The following state diagram shows the state of a Query made by a Company. As soon as the Query is made from a registered Company, it goes into Waiting for data states. Then the Query can be removed

by the Company or new data can satisfy the Query. In the latter case the query goes into Check Active Subscription. If a Subscription is active, the Query goes into new data available state, otherwise it goes into Payment request. If the transaction is accepted, the new state is New data available, otherwise goes into Payment failed state, until the payment method is updated.

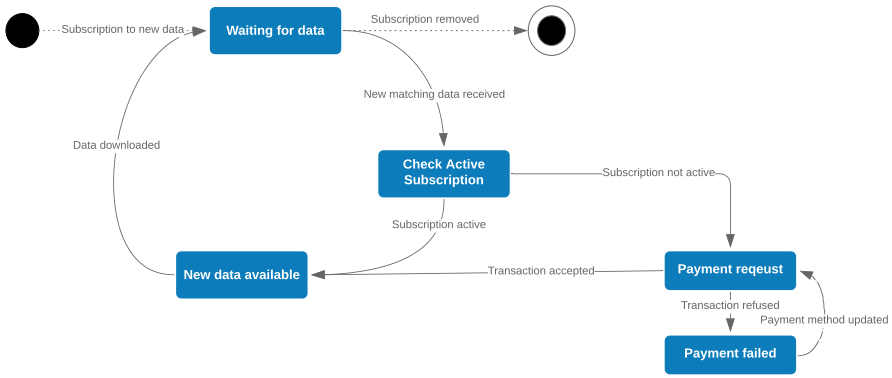



Figure 3: Data request state diagram

**Individuals data query** When a Company creates a Query, it goes into Check Active Subscription state. If the subscription to make individual queries is active, the system asks to the user for the permission. If the permission is granted, the Query goes into Company subscribed and the company can access user's data. If either the user deny the request or remove the permission to the company after having accepted the request, the Query is deleted. If the subscription is not active, the new state is payment request. If the transaction is accepted then the new state is Check Active Subscription, otherwise the transaction is refused, the new state is Payment failed and the Query waits until the user update his payment method.

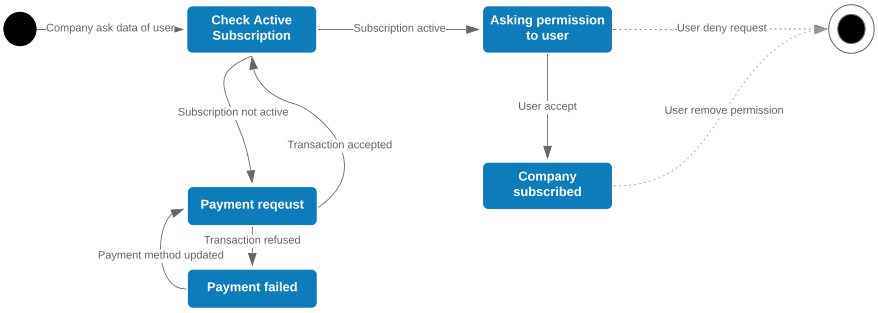



Figure 4: Individual request state diagram

## 2.2 Class diagram

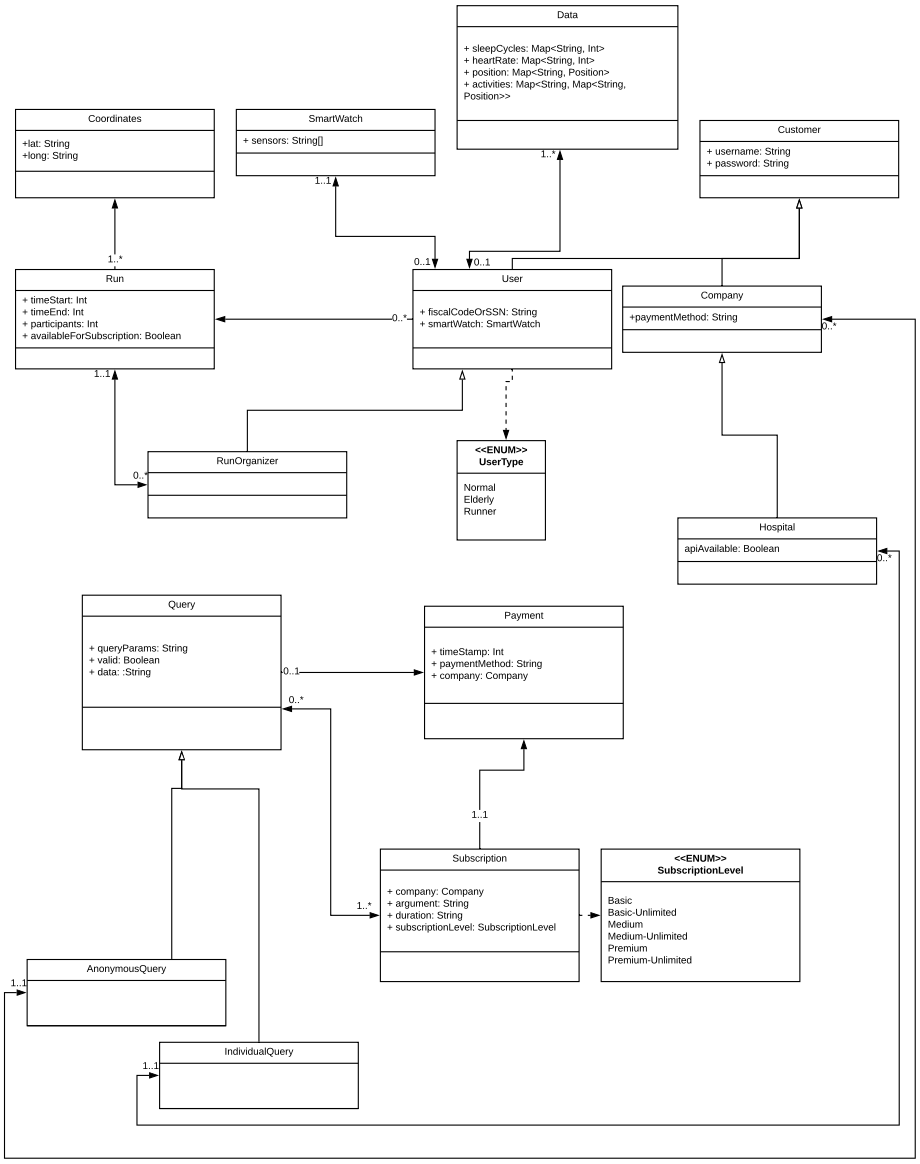



Figure 5: Class diagram

## 2.3 Product functions

This chapter listing all the functionalities that the system-to-be must offer.
EDIT: ADD SOME GENERAL FUNCTIONS

### 2.3.1 Functional requirements

Functional requirements for every system component:
Data4Help Mobile App

- [$\mathbf{RM}_M$]: Users can register, after providing a username, a password and a Fiscal Code/Social Security Number and have connected a compatible Smartwatch
- [$\mathbf{R1}_M$]: Users can log-in
- [$\mathbf{R2}_M$]: Users can only have one account
- [$\mathbf{R3}_M$]: Logged-in users can edit their account info
- [$\mathbf{R4}_M$]: Logged-in users can see their data statistics
- [$\mathbf{R5}_M$]: Logged-in can specify the nature of the daily activities (i.e. running, biking, swimming, hiking)
- [$\mathbf{R6}_M$]: Logged-in users can accept/decline a company individual monitoring request
- [$\mathbf{R7}_M$]: Logged-in users, if subscribed to *AutomatedSOS* service, can see the monitoring status of the service
- [$\mathbf{R8}_M$]: Logged-in users can register to a run before the start time
- [$\mathbf{R9}_M$]: Logged-in users can see a run information
- [$\mathbf{R10}_M$]: Logged-in users can see participant position on a map, if the run has started

Data4Help Smartwatch App

- [$\mathbf{R1}_S$]: App can read data from sensor and store it locally.
- [$\mathbf{R2}_S$]: App can send data registered locally to Data4Help Mobile App
- [$\mathbf{R3}_S$]: Users can see their health data in real time (i.e heart rate, pressure)
- [$\mathbf{R4}_S$]: Users can see their sleep monitoring of the previous night
- [$\mathbf{R5}_S$]: Users can configure the widgets showing their health activity
- [$\mathbf{R6}_S$]: Users who are participating in a run can see their timing and position through the screen of their Smartwatch

Data4Help Website

- [$\mathbf{R1}_W$]: Companies can register, after providing a username, a password and a payment method
- [$\mathbf{R2}_W$]: Companies can log-in
- [$\mathbf{R3}_W$]: Logged-in companies can see their history and account information
- [$\mathbf{R4}_W$]: Logged-in companies can subscribe to a payment service of Data4Help
- [$\mathbf{R5}_W$]: Logged-in companies can update their account information
- [$\mathbf{R6}_W$]: Logged-in companies can query on some group of individuals data
- [$\mathbf{R6bis}_W$]: Logged-in companies can subscribe to a query data
- [$\mathbf{R7}_W$]: Registered companies can request access to data of individuals
- [$\mathbf{R8}_W$]: Logged-in companies can access to an individual data, if the user has given approval
- [$\mathbf{R9}_W$]: Logged-in companies can export data previously queried using Data4Help
- [$\mathbf{R10}_W$]: Run organizers can register
- [$\mathbf{R11}_W$]: Run organizers can login
- [$\mathbf{R12}_W$]: Logged-in run organizers can organize a run
- [$\mathbf{R13}_W$]: Organizers of a run can define the path and additional information for that run
- [$\mathbf{R14}_W$]: Organizers of a run can start the run

Data4Help Core

- [$\mathbf{R1}_C$]: Can send online notifications via SMS to all users
- [$\mathbf{R2}_C$]: Can send online notifications via email to all users
- [$\mathbf{R3}_C$]: Can send online notifications via the Mobile app to its users
- [$\mathbf{R4}_C$]: Can save and store permanently user data
- [$\mathbf{R5}_C$]: Can receive and store health parameters and geographical position of registered users
- [$\mathbf{R6}_C$]: Can execute queries of companies on individuals if the user has accepted the monitoring request from the company

- [$\mathbf{R7}_C$]: Can execute queries of companies checking if the searches involve more than 1000 anonimyzed users
- [$\mathbf{R8}_C$]: Can provide a payment method for data requested by companies.
- [$\mathbf{R9}_C$]: Can communicate directly with the nearest hospital *API*.
- [$\mathbf{R10}_C$]: Can provide geographical position and critical health parameters to the emergency employee using the *API*
- [$\mathbf{R11}_C$]: Can compute for every AutomatedSOS user which are the threshold value to take care of for each health parameter
- [$\mathbf{R12}_C$]: Can compute each athlete's rank in a run and send it to each user device.
- [$\mathbf{R13}_C$]: Can send the position and rank of athletes during a run to spectator's devices.
- [$\mathbf{R14}_C$]: Can validates information provided by the user during registration
- [$\mathbf{R15}_C$]: Can charge companies on their payment method respecting Track4Me pricing policy

### 2.3.2 Non Functional requirements

Non-functional requirements:

- Users should be encouraged not to be in trouble for being monitored by companies
- Users should be encouraged to use the Mobile app with their friends
- Users should be encouraged to keep their smart watch all day and all night
- Nurses and doctors should encourage their patients to use the Mobile app
- Companies should be encouraged to use the service by other firms

### 2.3.3 Company pricing policies

The following policies are exclusively referred to the Webpage component services offered to Companies

- **Basic**: The company can query data choosing only from users of one city, one time per day **Price:** 15€/month

- **Basic - Unlimited**: : The company can query data choosing from users of only one city, with no limits **Price:** 50€/month
- **Medium**: The company can query data choosing only from users of one region of Italy, five times per day **Price:** 100€/month
- **Medium - Unlimited** The company can query data choosing only from users of one region of Italy, with no limits **Price:** 200€/month
- **Premium** The company can query data on all Data4Help users, with no limits **Price:** 1000€/month

Also, companies are also allowed to purchase single queries, with price:

- On a city: 5€/query
- On a region: 20€/query
- On all users of Data4Help: 50€/query

The following policies are exclusively referred to government companies who want to monitor individuals (i.e. hospitals, medical clinics)

- Maximum of 100 patients: 500€/month
- Maximum of 1000 patients: 1500€/month
- No limit on maximum patients: 5000€/month

It is not expected to be a pricing policy for Mobile App and Smartwatch App users, which means that the Apps will be free.
As specified in the Functional requirements, the Data4Help Core component is in charge to calculate the final price for every user using the service.

## 2.4 User characteristics

Here is shown the distinction of the users of the Data4Help services.
The common characteristic for all Customers is that all users of the service must be logged in to use it.
The users of using all possible components can be distinguished in 2 main categories:

- **Data4Help Individuals Customers**: Users who use the app to take advantage of services offered by it.
- **Data4Help Companies Customers**: Thirty-Party users who can be divided once more in 2 subcategories:
  - Companies who wants to use data for marketing purposes
  - Public companies or clinics who wants to monitor the health status of an individual patient

## 2.5 Constraints and dependencies

### 2.5.1 Constraints

The constraints for the System-to-be are listed below:

- Users are located in Italy
- Users must have a Smartwarth and a Mobile phone with a OS compatible with the applications
- Users must have Internet connection on their devices
- The system must ask the users the permission to store their data following the GDPR policy
- The system must ask users to use GPS position on their devices
- Companies must have a computer with a compatible browser to use the Data4Help services

### 2.5.2 Regulatory policies

As Data4Help will handle sensitive user data (e.g. name, birth date, location, state of health) the Application must treat them respecting the local laws, in particular the GDPR European law. Data attributable to the user must be anonymized in order to appear in a Company's query, unless the user has given explicit authorization to that company.

### 2.5.3 Security consideration

All data transferred between Data4Help Mobile App and Data4Help Core must be encrypted in order to minimize the possibility of a man-in-the-middle attack or any unauthorized access. For the same reason, all communication between Data4Help Web site and Data4Help Core must be encrypted. The Bluetooth communication between Data4Help SmartWatch App and Data4Help Mobile app is already encrypted by the Bluetooth stack itself.

### 2.5.4 Dependencies

- The system requires a DBMS to store and retrieve data
- Maps used by the system to store data relative to geographical position of ther users are provided by external APIs
- The system must rely on external Hospitals API in order to call ambulances for *[R9]* of the Core component

## 2.6 Assumptions

Each part of the system is based on the following domain assumptions.

- D1 The Storage System is reliable.
- D2 The SmartWatch on which the *Mobile App* is installed has an accelerometer, a gyroscope, a GPS antenna, and an heart rate sensor and they are always turned on.
- D3 Data taken from the previously mentioned sensors are always trusted and consistent.
- D4 The user keeps the SmartWatch on his/her wrist during day and night.
- D5 The user has a valid Fiscal Code or Social Security Number, and it is unique.
- D6 GPS signal is stable, hence the user position is always correct.
- D7 The phone on which the app will be installed has an internet access.
- D8 Every company willing to buy or subscribe to data has a credit card.
- D9 Users of *Automated SOS* have a stable internet connection.
- D10 Every Hospital in which the *AutomatedSOS* service is active has an *API* to call the ambulances.
- D11 The Hospital *API* service is active 24/7.

# 3 Specific requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

**Data4Help Mobile App:**
**TODO : ADD MOCKUPS! BOTH MOBILE PHONE AND SMARTWATCH**
The Mobile app should offer an easy interface aiming a user friendly experience of the customers. If should be possible to open a menu to navigate through the sub-sections. All the main functions (i.e. see history of activities, see account information) should be easy to access from any sub-section of the app. Descriptions of the pages has to be brief and concise. To see the details of the statistics there should be an info button that shows detailed description about the related data. If subscribed to the AutomatedSOS service, there should be a page showing the active controls on the user. In case of using the Track4Run service, there is also the possibility to see the map of a programmed run and seeing on the map all the participants, and their positions. More detailed info of the runners can be shown by tapping on their icon on the map.

The application must follow a proper design for every different mobile operating system:

- Android - Material Design
- iOS - Human Interfaces

The application should support all the screen resolutions available and optimize the item placement on the screen in the same way for every compatible device.
The user can configure the graphic of the widget visible from the Smartwatch only using the Mobile App component.

**Data4Help SmartWatch App** : The Smartwatch app should provide **widgets** that let the customer see their daily activity. There should be one widget for every type of data acquired by the device:

- Sleep monitoring
- Heart rate
- Blood pressure

The user can receive notifications about his activities in the Smartwatch and delete them through it.

**Data4Help WebSite**: The Website should offer an easy interface aiming a user friendly experience for the subscribed companies. The main menu should be visible on the top of the page, and must be used to navigate through the sub-sections. All the main functions (i.e. acquired data, account information, etc) should be accessible from any sub-section of the web page. Descriptions of the pages have to be clear and exhaustive.

**Data4Help Core:** This component does not have a user interface since it is intended to be accessible only by the qualified staff that manages it.

### 3.1.2 Hardware Interfaces

**Data4Help Mobile App:** The application should require the location services in order to work. If GPS is unavailable on the mobile phone, it can be requested to the Smartwatch app and viceversa. If both unavailable, the application wouldn't work and show an error window to notify the user. The app should also require a connection from either mobile network or Wi-Fi, but is sufficient to turn it at least once per day. If not, the app will notify the user asking to turn on connectivity.

**Data4Help SmartWatch App:** As written for Data4Help Mobile app, but also requires Bluetooth connectivity. If not available, an error message will appear on the Mobile app.

**Data4Help WebSite:** There is not any special hardware interface needed for the WebSite component.

**Data4Help Core:** There is not any special hardware interface needed for the Core component.

### 3.1.3 Software Interfaces

- **Data4Help SmartWatch App**: Development will focus on the production of a WatchOS and a WearOS app in order to properly communicate with their respective smartphone app.
  As an app built for the latest version of those OSes is also backwords compatible with their previous versions, there are no particular *minimum version* requirements. By developing an app for WatchOS and WearOS, the app will reach the 84% of all available Smartwatches.

- **Data4Help Mobile App**: Due to the fact that iOS and Android are the only OSes that provide a seamless integration with their smartwatch counterparts, the app will be developed for those platforms only. In order to support smartwatch communication there is a minimum version required for those OSes, namely:

  - Android $>$ 4.4 (API level 19) (about 94,7% of devices)

– iOS > 9.3 (about 96,3% of devices)

- **Data4Help Website**: It will require the use of a modern Web Browser to be accessed. It will work either on desktop and mobile Web Browsers.
- **Data4Help Core**: As the *Core* component will need only to provide **REST** endpoints for the communication (with ambulance, Website, App) there are no specific requirements on this component.

### 3.1.4 Communication Interfaces

In the system there are 2 types of communication.

1. The Mobile App and the Web Site need a bidirectional channel with the Core component of the system to operate properly.
   This can be achived by providing a REST API on the *Data4Help Core* component.
2. The Smartwatch App needs a direct connection with the Mobile app to give it user's data.
   The communication between the smart device and the smartphone is achieved via *BLE* and, once the channel is established, the Smartwatch App sends JSON messages to the Mobile App concerning all activities performend from the last syncronization.

```
{
    userId: "ka8c57pno3",
    last_synchronization: "1512518400",
    heart_rate: {
        "1512518400": 60,
        "1512519000": 61,
        ...
    },
    activities: {
       "1512518400": {
            "_lat": "",
            "_long": ""
        },
        "1512519000": {
            "_lat": "",
            "_long": ""
        },
        ...
    }
}
```

### 3.1.5 Memory Constraints

- **Data4Help SmartWatch App** As the devices on which the app will run have generally less than 1GB of RAM and less than 16GB of non volatile storage the smartwatch app should offload all the unnecessary computations to the mobile app, therefore reducing it size (which could be kept under 10MB) and its memory footprint.

- **Data4Help Mobile App** Based on the functionalities it will provide, the overall size of the app should not exceed 50MB, without taking into account the saved user data.

- **Data4Help Core** The server on which the *Core* component will run will need at least 2GB of RAM and 250MB of non volatile storage in order to host the application. An additional 2TB of storage should be added in order to retain the user-generated data.
  An estimate on the number of users using the service will suggest at least 64GB of RAM in order to ensure responsive operations for all the services provided.

## 3.2 Functional requirements

### 3.2.1 Scenarios

Here there are shown some scenarios to better understand the system usage from multiple viewpoints.

**Scenario 1 User registration and log-in**
Pierluigi is a Runner, he runs at least once per day and is very focused on the performance he gets during the training session. In order to do so, he downloaded the Data4Help App on both its Smartwatch and on its Mobile phone, so he registered using its email, confirmed the email, and then inserted personal data such as Age, Nationality, City where he lives and the sport he practices. Once having completed the subscription, he logs in through the Mobile app, turns on Bluetooth on smartphone and smartwatch and automatically the Data4Help apps synchronizes in both his devices. So he set run activity function on Mobile app and starts the run.

**Scenario 2 User parameter consulting**
Matteo has just waken up in the morning after a long night. However he feels very tired and acciaccato EDIT like if he hasnt slept at all. He suspects that something happened, so, in order to verify its hypotesis, he sees the the Data4Help widget on its smartwatch for sleep monitoring and finally, finds out that the quality of the sleeping process was very low. In order to see the details of the sleep monitoring, he opens the mobile app and check

the reasons of the bad night, and he finds out that its heart rate was really speedy on the first hours and he moved a lot during the night.

**Scenario 3 Data synchronization between devices**
Letizia is a student that has just finished her Parkour lesson at Milano Gravity sport center. She would like to see the calories she consumed, the maximum heart-bpm and other health parameters. Luckily, she has a Smartwatch with Data4Help application installed. Firstly she turns on the Bluetooth of her smartphone, that automatically synchronizes data with the smartwatch app, then she opens the mobile app of Data4Help and gives a look of the recent statistics.

**Scenario 4 Hospital registration and purchase**
Villa Serena is a clinic set in Jesi that wants to experiment an innovative way of monitoring their patients. To do so, the director of the clinic goes to the website of the Data4Help services, subscribes using the institute email, click on Subscribe to a Premium service choosing the 1000-patient limited option. Then he adds a payment method and complete the operation.

**Scenario 5 Patient registration**
Dr. Verdi is a private doctor of Dermatology in Milan that wants to exploit the features of Data4Help to monitor his patients. To do so, he has already subscribed to a 100-Patient limit option. To use the service, he asked Maria, one of his patients, to download the Mobile app and subscribe to the service. Maria subscribes as a normal user using her mail and Fiscal Code and, when finished, communicates her email to the Doctor that, immediately, add Maria as her patient. Maria receives a notification asking if she agrees to be monitored by dr. Verdi (identified by his email) and she clicks on accept.

**Scenario 6 Company subscription and purchase**
Clear-Water Spa is a company that produces energy drinks for runners that wants to start a marketing campaign in Milan. In order to know where people usually do activities in Milan, they decides to subscribe to a payment option of Data4Help services. The Marketing director goes on their Web page, subscribes to the service using the e-mail and Fiscal Code, inserts a payment method and purchases the 1-City unlimited query option, specifying Milan as the preferred city. Then he starts to query on the city, inserting:

- Age of the people to search (i.e. from 20 to 25 year old)
- The hour when to search people (i.e. from 9.00 to 10.00)

- The health parameter filters (i.e. heart rate $< 100$ bpm, pressure $>$ 130 mmHg)

The Data4Help system firstly checks if the query is satisfiable (i.e. is too specific), if yes, it shows a map with most frequent places of Milan where people go, with average of every health parameter registered from their users.

**Scenario 7** Iscrizione a Automated SOS
Ottavio is a patient of dr Verdi using AutomatedSOS services offered in the Data4Help mobile app.The smartwatch of Ottavio has detected that his health parameters has gone below the threshold computed by Data4Help, so the service is immediately notified by and calls an ambulance of the closest Hospital to the patient. The hospital confirms the emergency notification arrival and prepare an ambulance that is immediately sent to the position of Ottavio.

**Scenario 8** Organizzatore che organizza
Luigi is a run organizer for a famous sportive brand in Milan. He wants to use the services offered by Track4Run in order to attract more runners, so first he goes to the Data4Help web page and registers inserting his name, surname, email, Fiscal Code and specifying he is a Run organizer. After consolidating his email, he log in through the website and clicks Organize a new run. He fills in a module specifying the city of the race, the date, the starting time, the length in km and a description. Once finished, he clicks on post the run and obtain a link where he can see all the details of the joining partecipants.

**Scenario 9** partecipante a gara che si iscrive e inizia
Salvatore is a professional runner of Milano. He has recently searching for some running races where to partecipate to test his performances, so he opens Data4Help app on his smartphone and clicks on See nearby run When clicks, he is showed a map with all the races of Milan starting that day. Salvatore clicks on one starting near his house and a window appears on the screen of the app showing all the relevant information of the app. Then he decided to join the run, so clicks on join the run, finally receives the run number and a confirmation mail.

**Scenario 10** AAAAAAAAAAAAAAAA
Amanda and Dario have a son, Roberto, who has just joined a race organized through the Data4Help app. Since Robertos family wants to know their son geographical position during the race, they also download the app on their smartphones and register, specifying they are spectators of a race. Then they log in, click on See nearby races, click on the interested race and click on See spectators. Then in the screen appears a map showing the position of

all runners in blue points. The parents type on the searching bar of the app the name of their son and, finally, his position becomes red and is possible to distinguish his position.

### 3.2.2 Requirements mapping

In the following section are explained the functional requirements of the application, grouped under each goal.

- **G1**: The system should be able to show acquired data via the Mobile App and the Website.

  **Requirements**

  - [$\mathbf{R1}_S$]: App can read data from sensor and store it locally.
  - [$\mathbf{R2}_S$]: App can send data registered locally to Data4Help Mobile App

  **Domain assumptions**

  - **D1** The Storage System is reliable.
  - **D2** The SmartWatch on which the *Mobile App* is installed has an accelerometer, a gyroscope, a GPS antenna, and an heart rate sensor.
  - **D3** Data taken from the previously mentioned sensors are always trusted and consistent.
  - **D4** The user keeps the SmartWatch on his/her wrist during day and night.

- **G2**: The system should allow users to register by providing his Fiscal Code or his Social Security Number, an username and a password.

  **Requirements**

  - [$\mathbf{RM}_M$]: Users can register, after providing a username, a password and a Fiscal Code/Social Security Number and have connected a compatible Smartwatch
  - [$\mathbf{R14}_C$]: Can validates information provided by the user during registration

**Domain assumptions**

- D5 The user has a valid Fiscal Code or Social Security Number, and it is unique.

- **G3**: The system should allow companies to register.

  **Requirements**

  - [$\mathbf{R1}_W$: Companies can register, after providing a username, a password and a payment method
  - [$\mathbf{R14}_C$]: Can validates information provided by the user during registration

- **G4**: The system should allow registered companies to request data from an anonymized group of individuals, only if individuals in the group are more than 1000.

  **Requirements**

  - [$\mathbf{R2}_S$]: App can send data registered locally to Data4Help Mobile App
  - [$\mathbf{R5}_C$]: Can receive and store health parameters and geographical position of registered users
  - [$\mathbf{R7}_C$]: Can execute queries of companies checking if the searches involve more than 1000 anonimyzed users

  **Domain assumptions**

  - D1 The Storage System is reliable.

- **G5**: The system should allow registered companies to request data from an individual person, only if individuals accept the request.

  **Requirements**

  - [$\mathbf{R6}_C$]: Can execute queries of companies on individuals if the user has accepted the monitoring request from the company
  - [$\mathbf{R1}_C$]: Can send online notifications via SMS to all users
  - [$\mathbf{R2}_C$]: Can send online notifications via email to all users
  - [$\mathbf{R3}_C$]: Can send online notifications via the Mobile app to its users

- **G6**: The company should be able to pay through the system in order to buy data queries/subscribe to plans.

**Requirements**

- [$\mathbf{R4}_W$]: Logged-in companies can subscribe to a payment service of Data4Help
- [$\mathbf{R15}_C$]: Can charge companies on their payment method respecting Track4Me pricing policy

**Domain assumptions**

- D8 Every company willing to buy or subscribe to data has a credit card.

- **G7**: The system should allow a company to subscribe to new data and receive them as soon as they are produced.

**Requirements**

- [$\mathbf{R6bis}_W$]: Logged-in companies can subscribe to a query data
- [$\mathbf{R2}_C$]: Can send online notifications via email to all users

**Domain assumptions**

- D1 The Storage System is reliable.
- D2 The SmartWatch on which the *Mobile App* is installed has an accelerometer, a gyroscope, a GPS antenna, and an heart rate sensor and they are always turned on.
- D3 Data taken from the previously mentioned sensors are always trusted and consistent.

- **G8**: The system should be able to monitor user's health parameter.

**Requirements**

- [$\mathbf{R1}_S$]: App can read data from sensor and store it locally.
- [$\mathbf{R2}_S$]: App can send data registered locally to Data4Help Mobile App
- [$\mathbf{R4}_C$]: Can save and store permanently user data
- [$\mathbf{R5}_C$]: Can receive and store health parameters and geographical position of registered users

**Domain assumptions**

- D1 The Storage System is reliable.
- D2 The SmartWatch on which the *Mobile App* is installed has an accelerometer, a gyroscope, a GPS antenna, and an heart rate sensor and they are always turned on.
- D3 Data taken from the previously mentioned sensors are always trusted and consistent.
- D4 The user keeps the SmartWatch on his/her wrist during day and night.
- D7 The phone on which the app will be installed has an internet access.

- **G9**: The system should be able to react to the lowering of the health parameters below threshold in less than 5 seconds and send the position of the person to the ambulance system.

**Requirements**

- [$\mathbf{R1}_S$]: App can read data from sensor and store it locally.
- [$\mathbf{R2}_S$]: App can send data registered locally to Data4Help Mobile App
- [$\mathbf{R5}_C$]: Can receive and store health parameters and geographical position of registered users
- [$\mathbf{R9}_C$]: Can communicate directly with the nearest hospital *API*.
- [$\mathbf{R10}_C$]: Can provide geographical position and critical health parameters to the emergency employee using the *API*
- [$\mathbf{R11}_C$]: Can compute for every AutomatedSOS user which are the threshold value to take care of for each health parameter

**Domain assumptions**

- D2 The SmartWatch on which the *Mobile App* is installed has an accelerometer, a gyroscope, a GPS antenna, and an heart rate sensor and they are always turned on.
- D3 Data taken from the previously mentioned sensors are always trusted and consistent.
- D4 The user keeps the SmartWatch on his/her wrist during day and night.
- D7 The phone on which the app will be installed has an internet access.

- D9 Users of *Automated SOS* have a stable internet connection.
- D10 Every Hospital in which the *AutomatedSOS* service is active has an *API* to call the ambulances.
- D11 The Hospital *API* service is active 24/7.

- **G10** The system should allow run organizers to register.

  **Requirements**

  - [$\mathbf{R10}_W$]: Run organizers can register
  - [$\mathbf{R11}_W$]: Run organizers can login
  - [$\mathbf{R14}_C$]: Can validates information provided by the user during registration

- **G11** If a run organizer is registered, it can define a run i.e. it can define the path that the participants should follow.

  **Requirements**

  - [$\mathbf{R12}_W$]: Logged-in run organizers can organize a run
  - [$\mathbf{R13}_W$]: Organizers of a run can define the path and additional information for that run
  - [$\mathbf{R14}_W$]: Organizers of a run can start the run

- **G12** A user should be able to enroll to a run, if he is close to the starting point and has a valid inscription code.

  **Requirements**

  - [$\mathbf{R8}_M$]: Logged-in users can register to a run before the start time
  - [$\mathbf{R9}_M$]: Logged-in users can see a run information

- **G13** Spectators of a run should be able to see each participant's position on a map.

  **Requirements**

  - [$\mathbf{R12}_C$]: Can compute each athlete's rank in a run and send it to each user device.
  - [$\mathbf{R13}_C$]: Can send the position and rank of athletes during a run to spectator's devices.

- **G14** Users should be able to log in

**Requirements**

- [$\mathbf{R1}_M$]: Users can log-in rank of athletes during a run to spectator's devices.

- **G15** Companies should be able to log in

  **Requirements**

  - [$\mathbf{R12}_C$]: Can compute each athlete's rank in a run and send it to each user device.
  - [$\mathbf{R13}_C$]: Can send the position and rank of athletes during a run to spectator's devices.

- **G16** Run organizers must be able to

  **Requirements**

  - [$\mathbf{R12}_C$]: Can compute each athlete's rank in a run and send it to each user device.
  - [$\mathbf{R13}_C$]: Can send the position and rank of athletes during a run to spectator's devices.

### 3.2.3 Use cases

**User registration**

| | |
|---|---|
| **Actors** | Not registered user, Mobile app, Track4Me core App |
| **Start conditions** | None |
| **Event flow** | • not registered user clicks on Subscribe<br>• not registered user inserts an username, an email, a Fiscal Code or SSN and a password<br>• The system checks if username is not used by any other user<br>• The system checks if email is not used by any other user<br>• The system checks if Fiscal Code / SSN is valid<br>• The mobile app checks if a compatible smartwatch is connected and has the Smartwatch Data4Help app installed<br>• The system generates a confirmation mail<br>• The user confirms the registration inserting the code received via email<br>• The system shows a registration complete screen |
| **Exit condition** | User information used for registering are stored in the company database |
| **Exceptions** | • The username is used by another user<br>• The email is used by another user<br>• The Fiscal Code/SSN is not valid<br>• Smartwatch is not compatible<br>• Smartwatch has not Data4Help app installed |
| **Goals** | G2 |

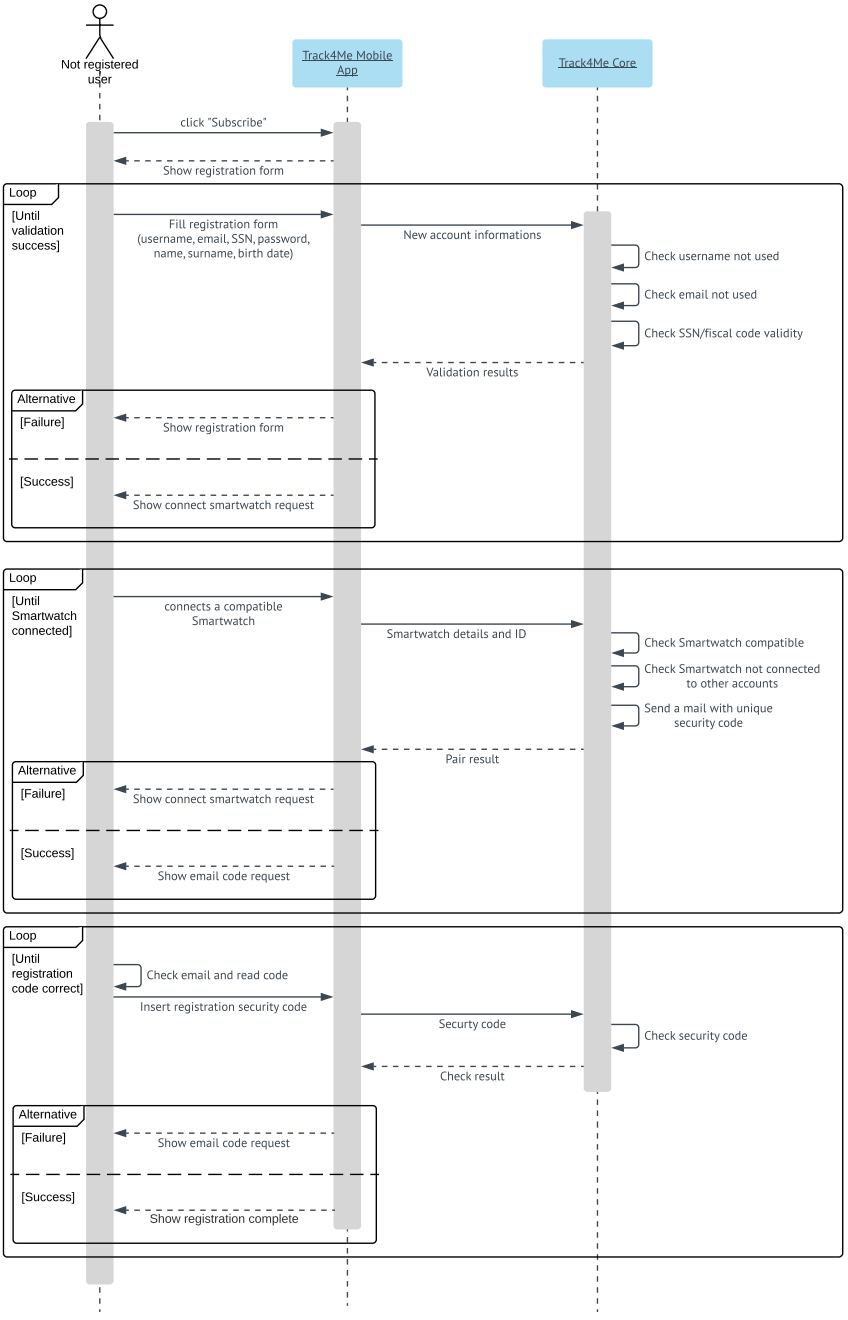



Figure 6: User registration sequence diagram

**Consulting of activity history**

| | |
|---|---|
| **Actors** | Registered User |
| **Start conditions** | The user is logged in |
| **Event flow** | • Registered user clicks on Show past activities on the Mobile phone<br>• the app shows a screen for searching data information<br>• registered user inserts year, month and the day to search and clicks on the type of data searched<br>• the system searches for a correspondence in the company database<br>• the system shows a screen showing the required data in a graph |
| **Exit condition** | None |
| **Exceptions** | • The specified day of the calendar is not valid<br>• The specified day of the calendar shows no activities of the user |
| **Goals** | **G1** |

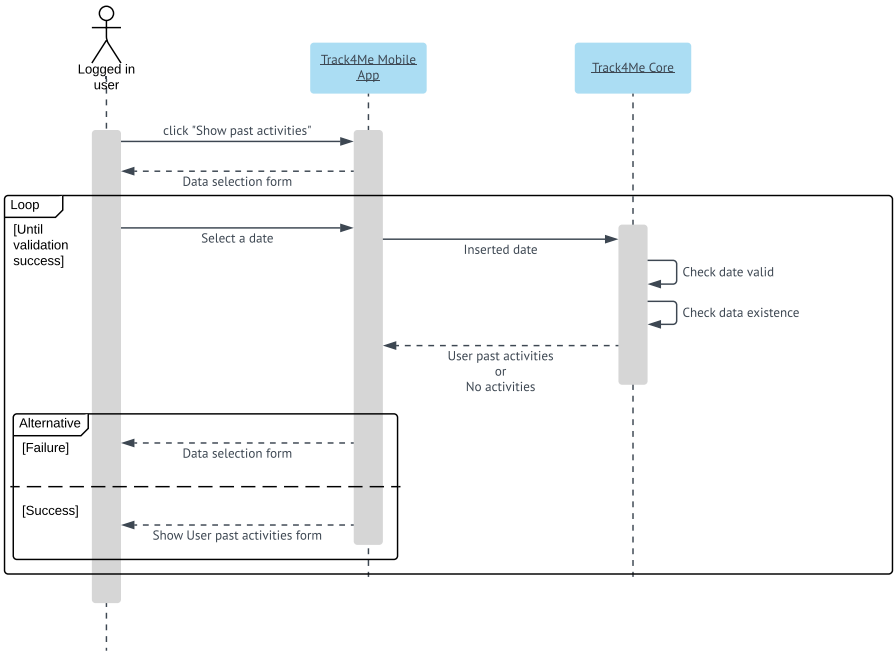



Figure 7: Consulting of activity history sequence diagram

### Data Synchronization between Smartwatch and Smartphone

| | |
|---|---|
| **Actors** | Mobile app, smartwatch app |
| **Start conditions** | Bluetooth connection enstablished between smartwatch and smartphone |
| **Event flow** | • User turns on Bluetooth of the smartwatch and smartphone<br>• The smartphone requests connection to smartwatch<br>• Smartwatch confirms connection<br>• Smartwatch send data registered in the day<br>• Smartphone accept data and store locally |
| **Exit condition** | Data in the user smartphone is synchronized with data of the smartwatch and stored locally |
| **Exceptions** | Connection Refused |
| **Goals** | G1 |

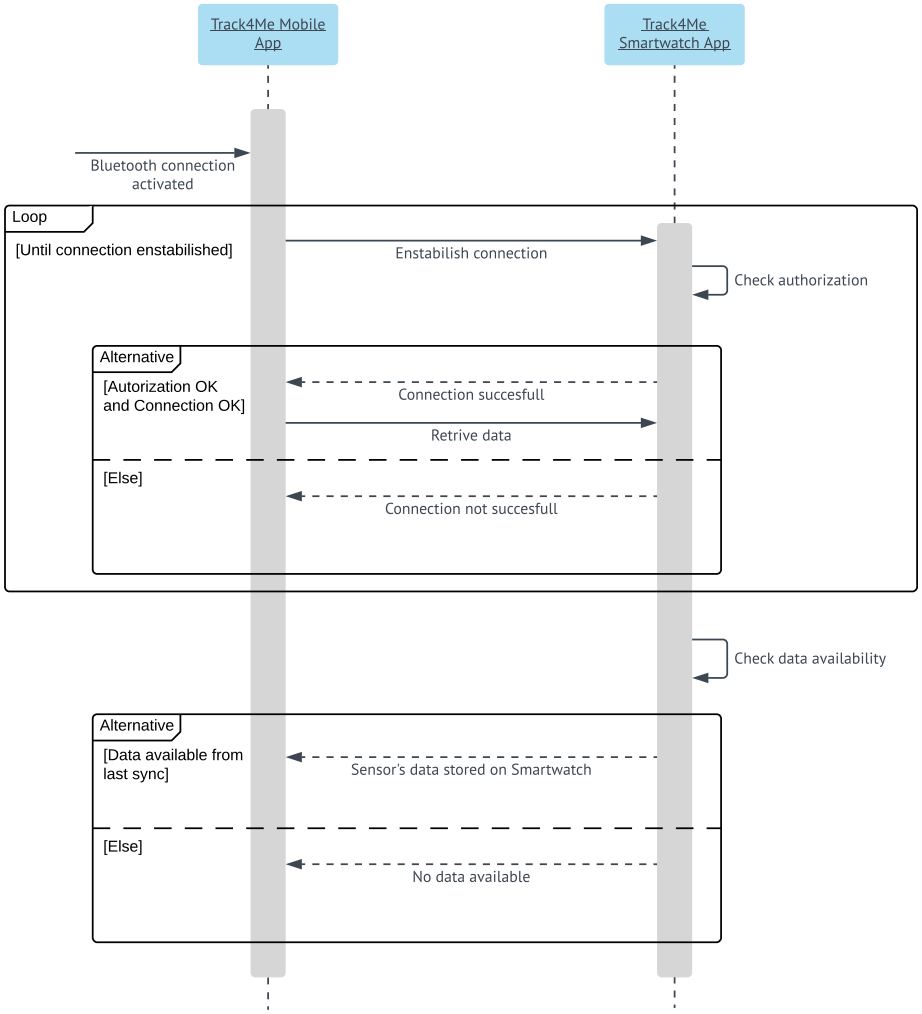



Figure 8: Data Synchronization between Smartwatch and Smartphone sequence diagram

**Company registration**

| | |
|---|---|
| **Actors** | Not registered company |
| **Start conditions** | None |
| **Event flow** | • not registered company clicks on Subscribe<br>• not registered company inserts a username, an email, a payment method and a password<br>• The system checks if email is not used by any other user<br>• The system checks if Payment method is valid<br>• The system generates a confirmation mail<br>• The company confirms the registration click on Confirm email<br>• The system shows a registration complete screen |
| **Exit condition** | Company information used for registering is stored in the company database |
| **Exceptions** | • The username is used by another company<br>• The email is used by another company<br>• The Payment method is not valid |
| **Goals** | G3 |

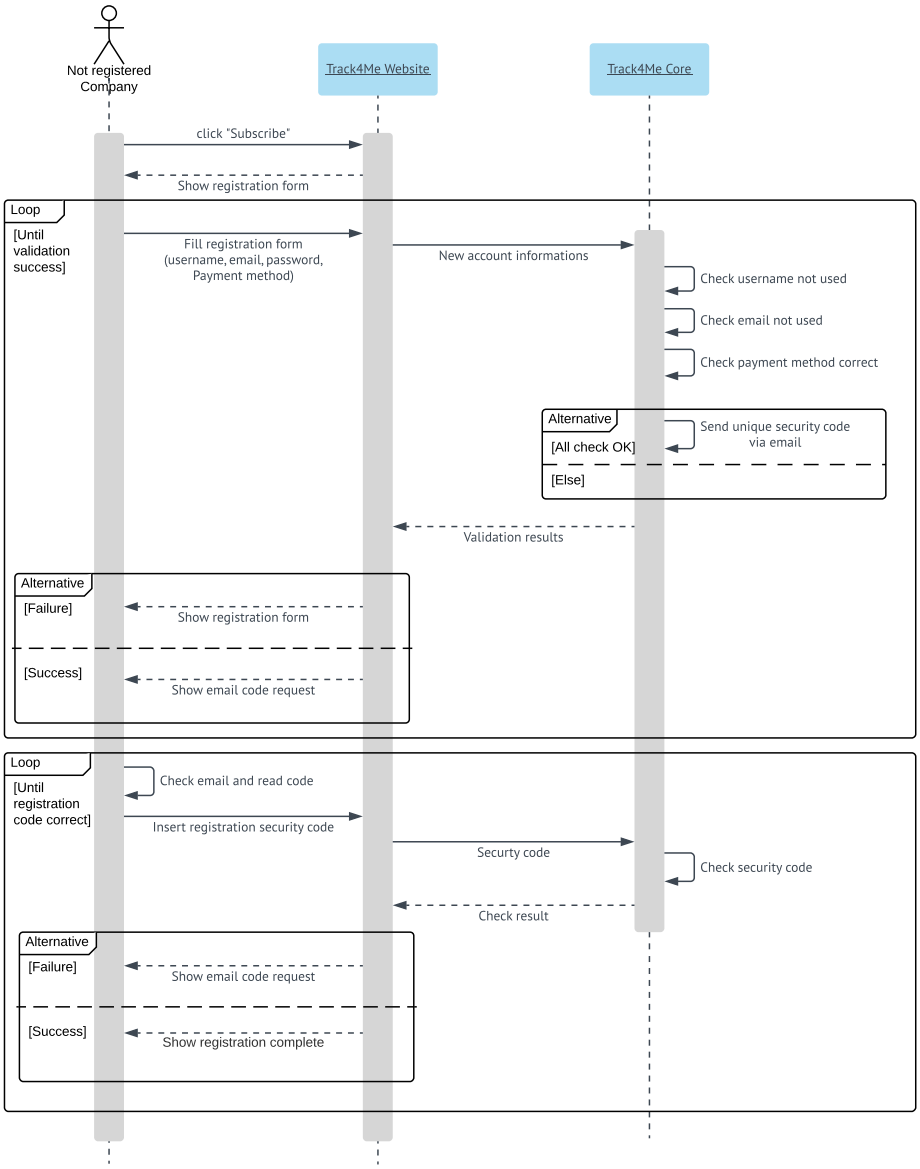



Figure 9: Company registration sequence diagram

**Company query search on multiple individuals**

| | |
|---|---|
| **Actors** | Company, WebSite, core component |
| **Start conditions** | None |
| **Event flow** | • Company clicks on Start a query on the website<br>• System checks if the company has query left<br>• System shows the page on the website<br>• Company specifies age of people to search, hours of the day, health parameter filters<br>• System validates the request<br>• System send the required data formatted on a pdf page on the website<br>• Company download the data |
| **Exit condition** | Query generated as pdf file and stored in the Data4Help database |
| **Exceptions** | • No query left for the company type of subscription<br>• Not valid query |
| **Goals** | G4 |

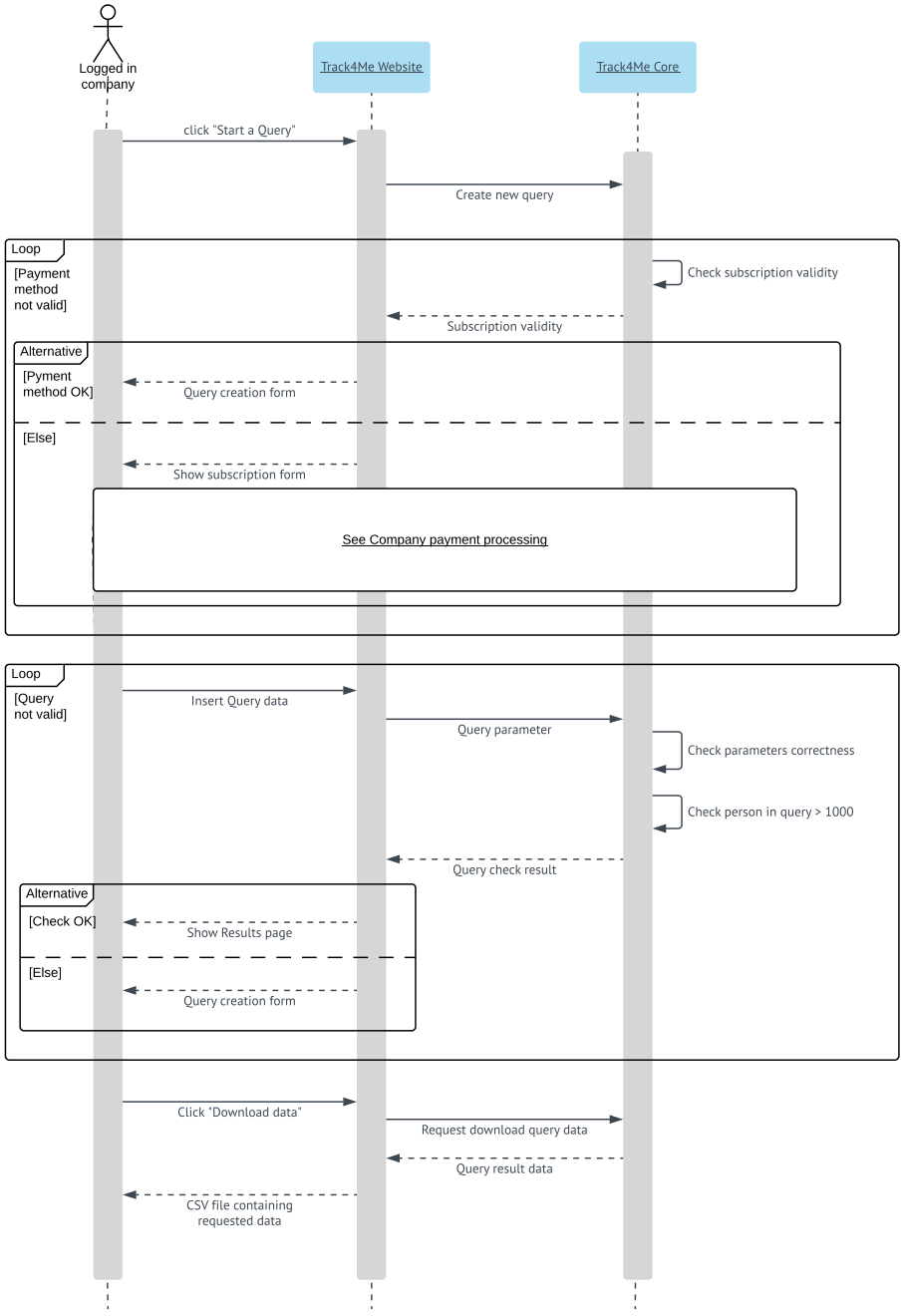



Figure 10: Company query search on multiple individuals sequence diagram

**Company request for individual monitoring**

| | |
|---|---|
| **Actors** | Company, User, Core component |
| **Start conditions** | None |
| **Event flow** | • Company clicks on add new patient<br>• System checks if the company can add new patients<br>• Company insert the username of the patient<br>• System checks if username exists in the database<br>• System send a monitoring request notification to the user<br>• Username clicks on Accept requests through the Mobile app notification or through the email notification |
| **Exit condition** | • System adds information of the monitoring company to the user in the database<br>• The system update number of patient left to monitor of the company in the database |
| **Exceptions** | • No new patients left for the company type of subscription<br>• Username not found<br>• User refuses the request |
| **Goals** | G5 |

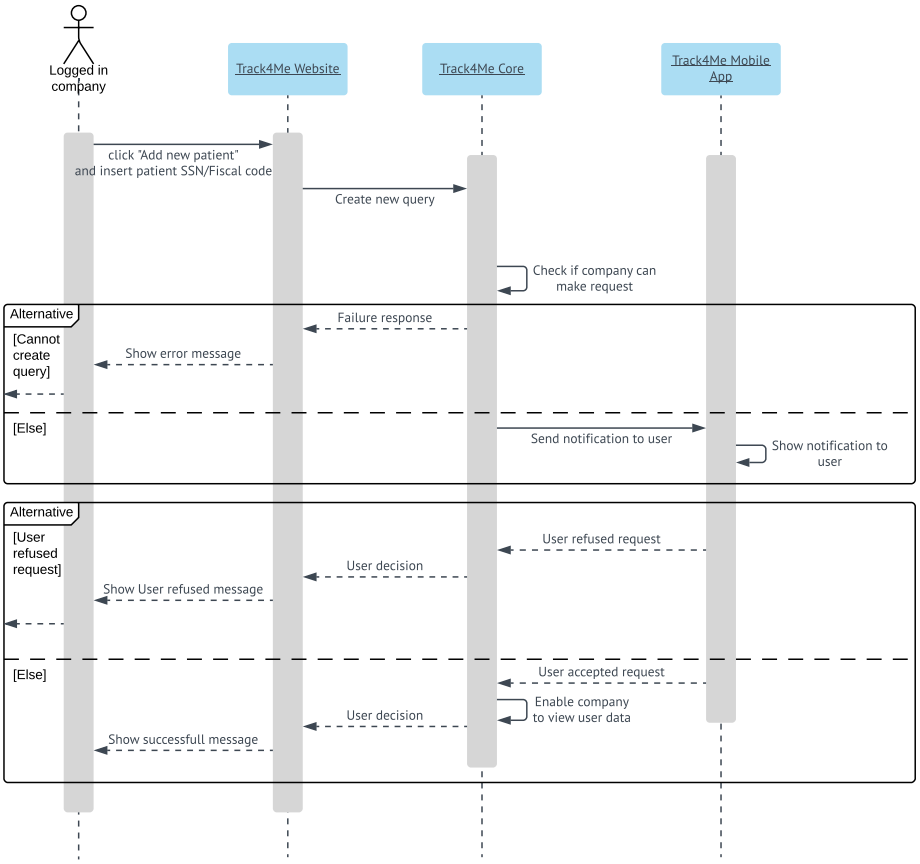



Figure 11: Company request for individual monitoring sequence diagram

**Company consulting of individual data**

| | |
|---|---|
| **Actors** | Company, core component |
| **Start conditions** | None |
| **Event flow** | • Company clicks on See patients information on the website<br>• System shows a page listing all active patients for the company<br>• Company clicks on the desired patient name<br>• System shows all information of the patient and a menu where to choose a day<br>• Company chooses a day of the calendar from the menu<br>• System shows the data of the patient on the specified day |
| **Exit condition** | None |
| **Exceptions** | • No query left for the company type of subscription<br>• Not valid query |
| **Goals** | G5 |

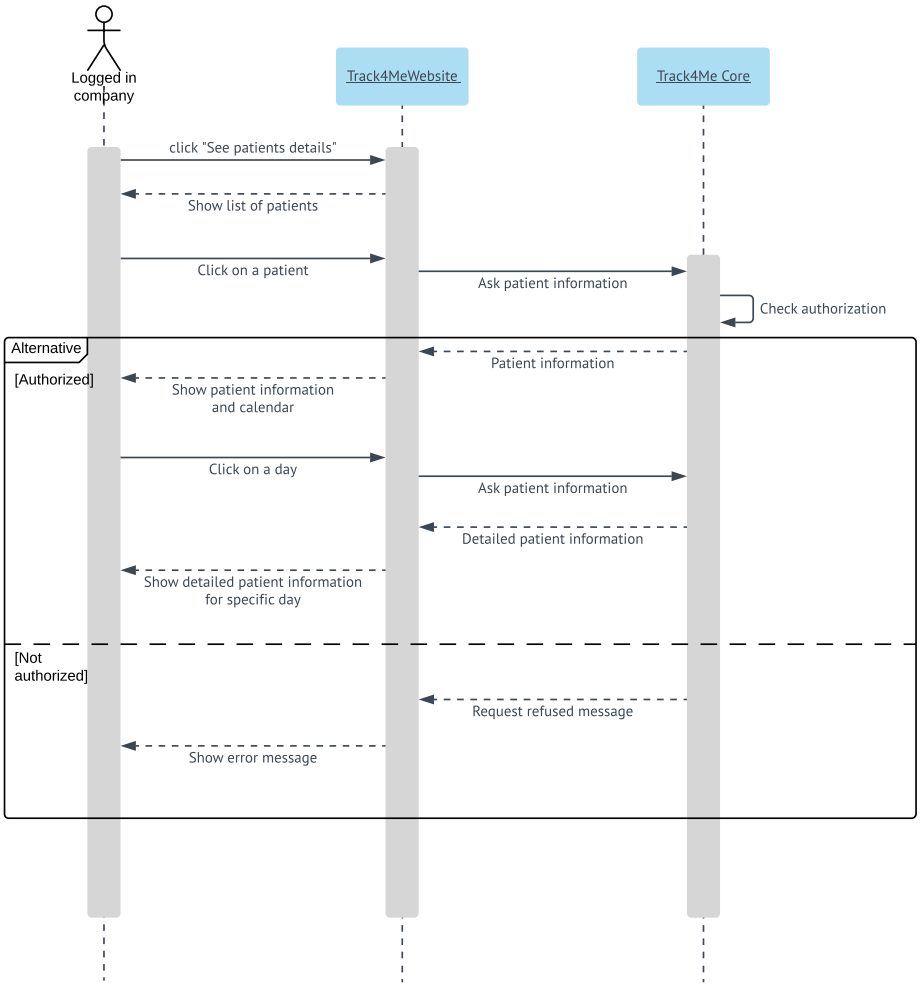



Figure 12: Company consulting of individual data sequence diagram

**Company payment processing**

| | |
|---|---|
| **Actors** | Company, Core component |
| **Start conditions** | Company purchases a subscription plan |
| **Event flow** | • Company insert the number of the credit card and the CVV<br>• System verify the card has enough money<br>• The system get the money from the credit card<br>• The system show a Successful operation |
| **Exit condition** | The system stored the information of the subscription of the company in the database |
| **Exceptions** | • Not enough money<br>• Not valid credit card information |
| **Goals** | G6 |

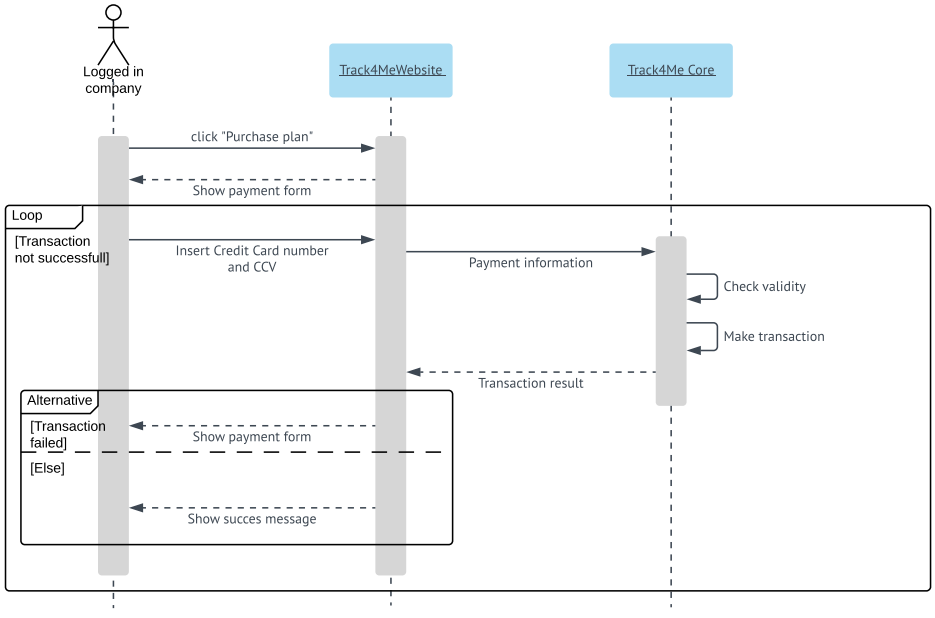



Figure 13: Company payment processing sequence diagram

**Company receives data of subscribed query**

| | |
|---|---|
| **Actors** | Company, core component |
| **Start conditions** | Is 0:01 of one month passed after the last data sent of the query |
| **Event flow** | • System retrieve the query information on the database<br>• System compute query on the database<br>• System export a CSV of the data stored<br>• System send the document on the email of the company |
| **Exit condition** | None |
| **Exceptions** | None |
| **Goals** | **G7** |

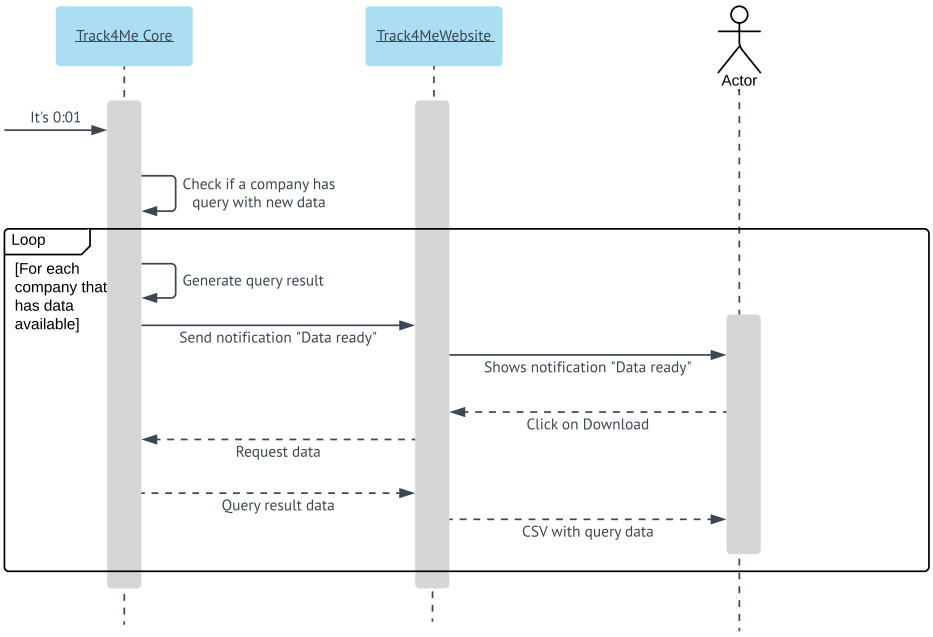



Figure 14: Company receives data of subscribed query sequence diagram

**Emergency situation and ambulance call**

| | |
|---|---|
| **Actors** | User, core component |
| **Start conditions** | User health parameter under the threshold value |
| **Event flow** | • System change the user status in AMBULANCE NEED<br>• System connects hospital API<br>• Hospital API answer to the connection<br>• System send emergency request to the hospital API sending location of the user and the health parameters under the threshold<br>• Hospital API confirms that an ambulance has been sent to the position |
| **Exit condition** | The user is labelled as AMBULANCE SENT |
| **Exceptions** | Hospital API refuses the connection |
| **Goals** | G9 |

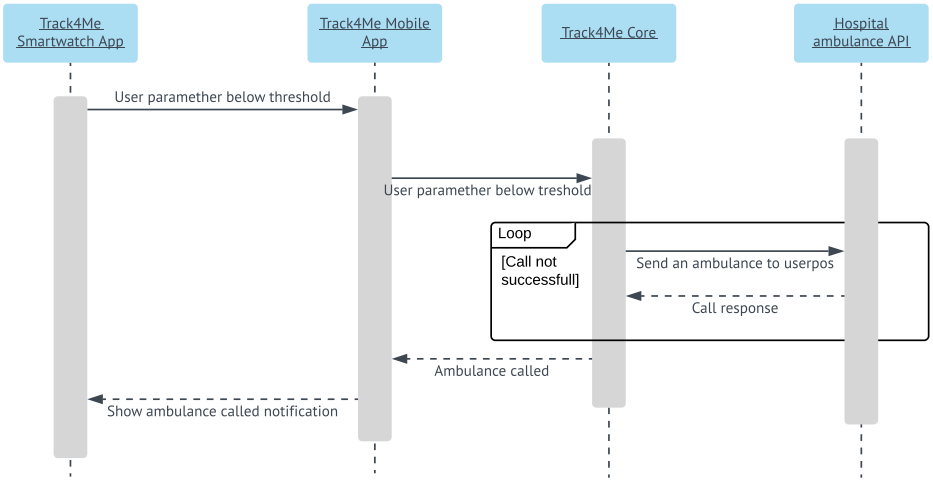



Figure 15: Emergency situation and ambulance call sequence diagram

**Run organizer registration**

| | |
|---|---|
| **Actors** | Not registered run organizer |
| **Start conditions** | None |
| **Event flow** | • not registered company run organizer clicks on Subscribe<br>• not registered run organizer inserts a username, an email and a password<br>• The system checks if email is not used by any other user<br>• The system generates a confirmation mail<br>• The run organizer confirms the registration click on Confirm email<br>• The system shows a registration complete screen |
| **Exit condition** | Run organizer information used for registering is stored in the Data4Help database |
| **Exceptions** | • Not valid username<br>• Not valid email |
| **Goals** | G10 |

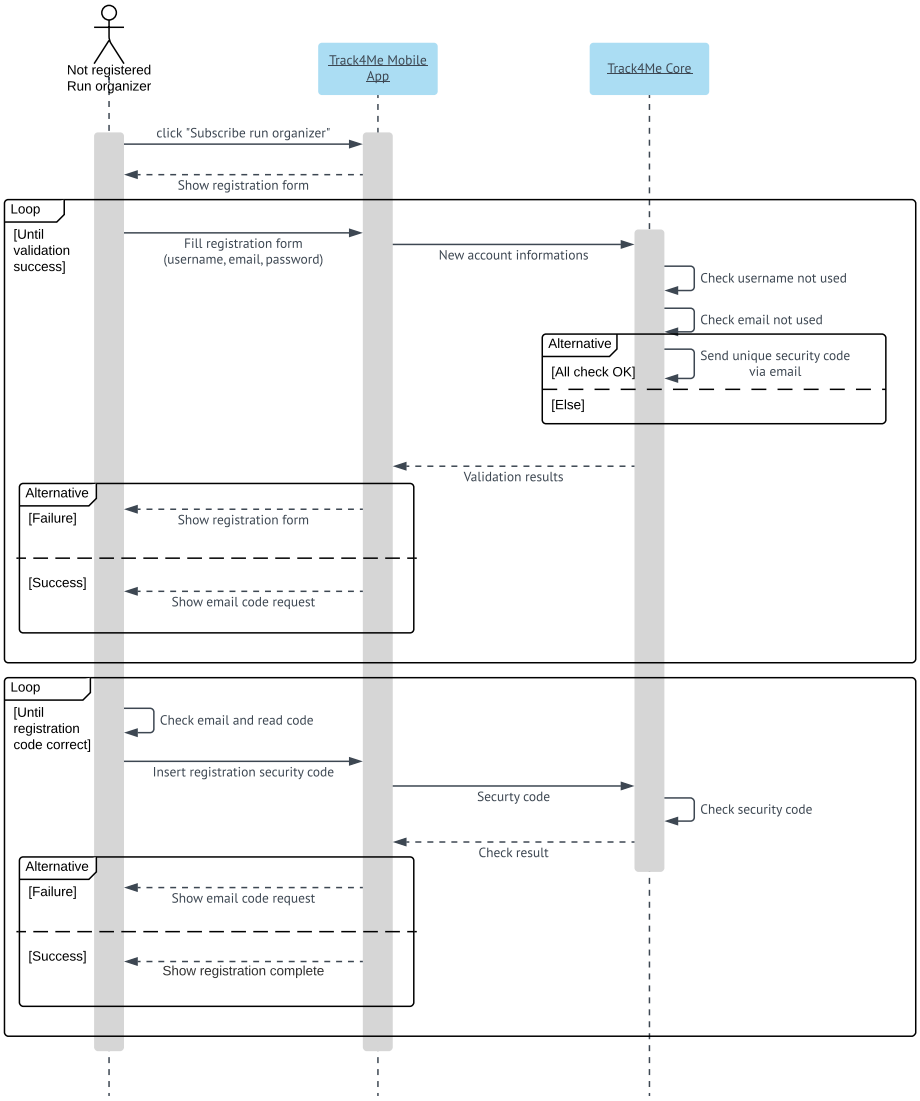



Figure 16: Run organizer registration sequence diagram

**Run organizer adds a new race**

| | |
|---|---|
| **Actors** | Run organizer |
| **Start conditions** | None |
| **Event flow** | • Run organizer clicks on "Organize new run" on the Data4Help website<br>• The system shows a screen with empty spaces about run information<br>• Run organizer insert the name of the run, the city where the run will take place, the date and the starting time<br>• The system checks if date is correct<br>• The system shows a map where user can define the path of the run<br>• Run organizer clicks on the path he wants the run to be taken<br>• The system calculates the km of the run<br>• The system show a "Correctly created run" screen |
| **Exit condition** | The system stores the information of the run in the Data4Help database and makes it public to users |
| **Exceptions** | • Not valid date |
| **Goals** | G11 |

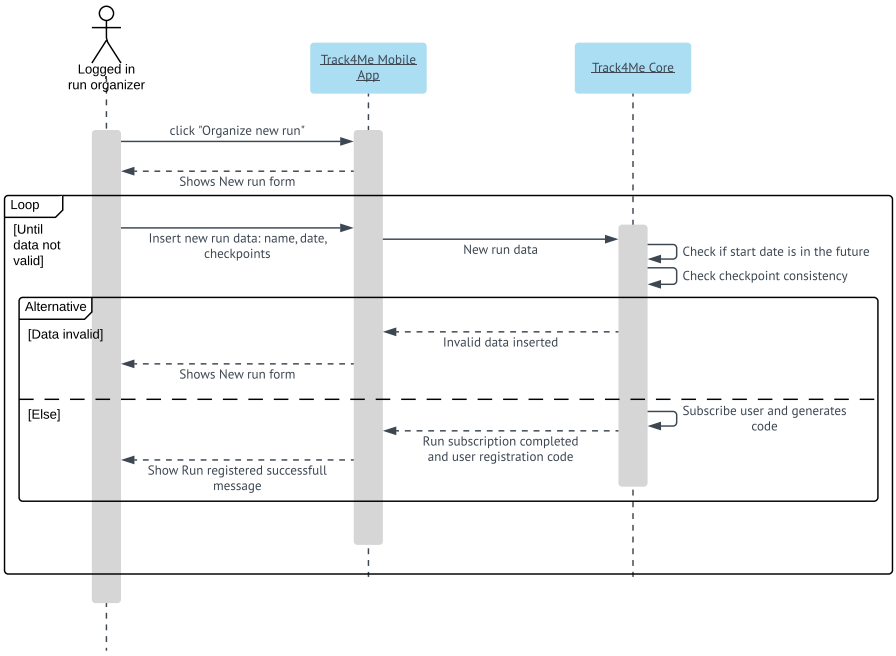



Figure 17: Run organizer adds a new race sequence diagram

**Runner subscription to a race**

| | |
|---|---|
| **Actors** | Run organizer |
| **Start conditions** | None |
| **Event flow** | • Partecipant clicks on 'See nearby races'<br>• System shows a map with the races closer than 2 km from the user<br>• Partecipant click on a run and clicks subscribe<br>• The system verifies if it's still possible to subscribe<br>• The system generates an runner ID number for the user<br>• The system send the ID number to the user email<br>• The user shows a correct subscription screenshot on the Mobile app |
| **Goals** | G12 |

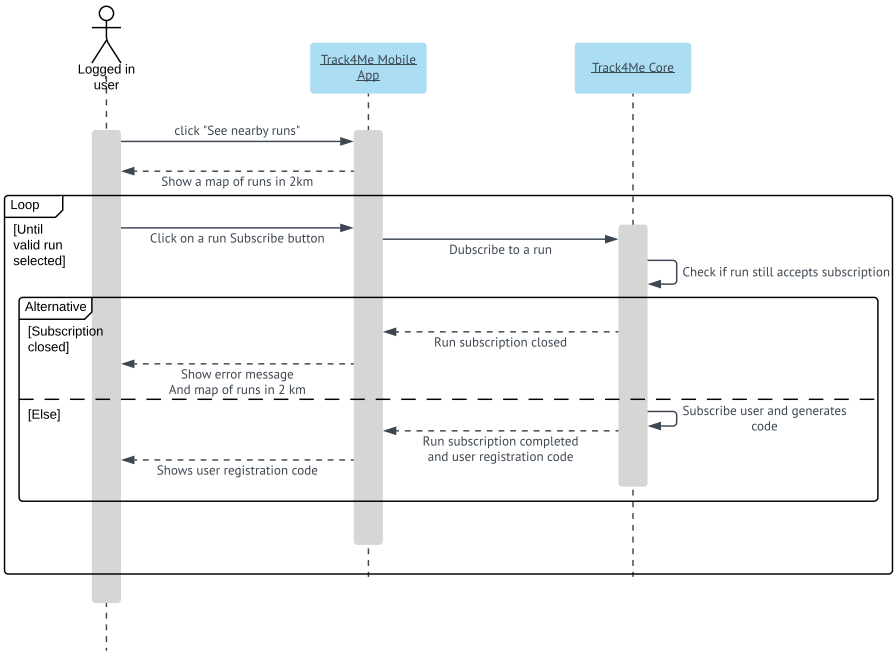



Figure 18: Runner subscription to a race sequence diagram

**Spectator or a run requests for runner position**

| | |
|---|---|
| **Actors** | User |
| **Start conditions** | None |
| **Event flow** | • User clicks on "See nearby races"<br>• System shows the map in radius of 2km from the user current position<br>• User clicks on the desired run he wants to see and clicks on see current runner positions<br>• The system shows a map with runners geographical position with their relative username and rank |
| **Exit condition** | None |
| **Exceptions** | • Run has already finished<br>• Run hasn't already started |
| **Goals** | G13 |

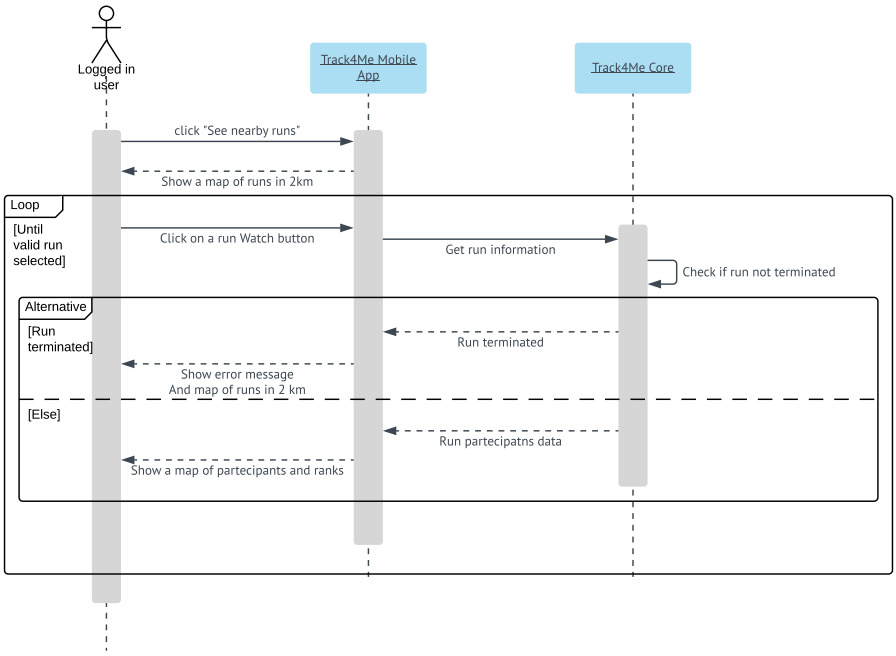



Figure 19: Spectator or a run requests for runner position sequence diagram

# 4 Appendix

## 4.1 Alloy

```
    open util/boolean

// Signatures

sig CharSet {}

sig Position {
    _lat: one CharSet,
    _long: one CharSet
}

sig Smartwatch {
    user: one User,
    compatible: one Bool
}

abstract sig Customer {
```

```
    username: lone CharSet,
    password: lone CharSet,
    canRegister: one Bool,
    isRegistered: one Bool
} {
    isRegistered = True => canRegister = True
    canRegister = False => isRegistered = False
    (#username = 0 || #password = 0) => isRegistered = False
}

sig User extends Customer {
    fiscalCodeOrSSN: lone CharSet,
    smartwatch: one Smartwatch,
    notifications: set Notification,
    acceptsDataRequestFrom: set Company,
    hasInscriptionCode: lone Bool,
} {
    (
        #fiscalCodeOrSSN = 0 ||
        smartwatch.compatible = False
    ) <=> (canRegister = False)
}

sig Elderly extends User {
    isInDanger: one Bool,
    inDangerFrom: lone Int
} {
    inDangerFrom >= 0 => isInDanger = True
    #inDangerFrom = 0 => isInDanger = False
}

sig AmbulanceRequest {
    hospital: one Company,
    person: one Elderly,
    timeElapsed: one Int,
    accepted: one Bool
} {
    timeElapsed >= 5 => accepted = True
}
sig RunOrganizer extends User {
    runPath: set Position,
    runOrganized: lone Run
} {
    isRegistered = False => (#runPath = 0 && #runOrganized = 0)
```

```
}

sig Company extends Customer {
    paymentMethod: lone CharSet,
    queries: set Query
} {
    #paymentMethod = 0 <=> canRegister = False
    isRegistered = False => #queries = 0
}

abstract sig Query {
    company: one Company
}

sig AnonQuery extends Query {
    people: set User,
    isValid: one Bool
} {
    isValid = True <=> #people >= 3
    no p: User | p in people && p.isRegistered = False
}

sig IndividualQuery extends Query {
    person: one User,
    userAccepts: lone Bool,
}

sig Notification {
    user: one User,
    company: one Company
}

sig Run {
    hasStarted: one Bool,
    participants: set User,
    organizer: one RunOrganizer
}

// Facts: Consistency

fact UsernameConsistency {
    // There are no 2 Customer(s) with the same username
    all disj c, c': Customer | c.username != c'.username
}
```

```
fact FiscalCodeOrSSNConsistency {
    // There are no 2 User(s) with the same fiscalCodeOrSSN
    all disj u, u': User | u.fiscalCodeOrSSN != u'.fiscalCodeOrSSN
}

fact SmartWatchConsistency {
    // Let's suppose, wlog, that the cardinality of the relation
    // between smartwatch and user is 1 to 1
    all s: Smartwatch, u: User | s.user = u <=> u.smartwatch = s
}

fact QueryConsistency {
    // If a query has been made by a company
    // it must be in the set of all the queries
    // made by the company
    all q: Query, c: Company | q.company = c <=> q in c.queries
}

fact NotificationConsistency {
    // If a notification has been sent to a user
    // the user must have it in the set of
    // all notifications
    all n: Notification, u: User | n.user = u <=> n in u.notifications
}

fact AmbulanceRequestConsistency {
    all e: Elderly, a: AmbulanceRequest | a.person = e <=> a.hospital in e.acceptsDa
}

fact RunAndRunOrganizerConsistency {
    all r: Run, ro: RunOrganizer {
        r.organizer = ro <=> ro.runOrganized = r
    }
}

// Assertions

// Goal G2: The system should allow users to register by providing his
// Fiscal Code or his Social Security Number, an username and a password.

assert UserCanRegister {
    all u: User {
        (
```

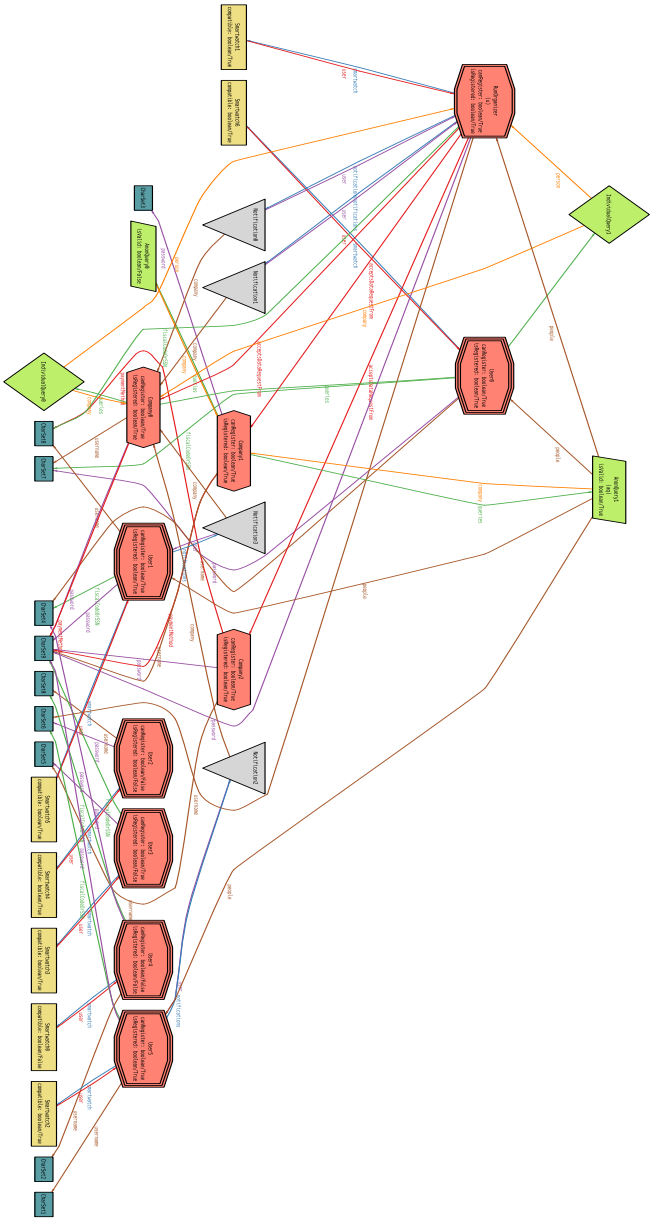
```
            #u.username = 1 &&
            #u.password = 1 &&
            #u.fiscalCodeOrSSN = 1 &&
            u.(smartwatch.compatible) = True
        ) => u.canRegister = True
    }
}

check UserCanRegister for 5

// Goal G3: The system should allow companies to register

assert CompaniesCanRegister {
    all c: Company {
        (
            #c.username = 1 &&
            #c.password = 1 &&
            #c.paymentMethod = 1
        ) => c.canRegister = True
    }
}
check CompaniesCanRegister for 5

// Goal G4: The system should allow registered companies to request data
// from an anonymized group of individuals, only if individuals in the
// group are more than 1000.

assert CompaniesCanMakeAnonimizedQueries {
    all c: Company, q: AnonQuery{
        (
            c.isRegistered = True &&
            #queries > 0 &&
            #q.people >= 5
        ) => q.isValid = True && q in c.queries

    }
}

check CompaniesCanMakeAnonimizedQueries for 5

// Goal G5: The system should allow registered companies to request data
// from an individual person, only if individuals accept the request.

assert CompaniesCanMakeIndividualQueries {
```

```
    // If a company requests data from a single person
    // either
    // the person accepts <=> company is in the person acceptance list
    // or
    // the person still hasn't accepted => there's a notification
    // concerning the company and the user in the person notification list

    all q: IndividualQuery, c: Company, n: Notification {
        (q.company = c) &&
        (
            (q.userAccepts = True <=> c in q.(person.acceptsDataRequestFrom)) ||
            (#q.userAccepts = 0 => (
                n.user = q.person &&
                n.company = c &&
                n in q.(person.notifications)
            ))
        )
    }
}

check CompaniesCanMakeAnonimizedQueries for 5

// Goal G11: If a run organizer is registered, it can define a run. For instance, it
// define the path that the participants should follow.

assert OrganizerCanDefineARun {
    all ro: RunOrganizer {
        ro.isRegistered = True => (#ro.runOrganized >= 0 && #ro.runPath >= 0)
    }
}

check OrganizerCanDefineARun for 5

// Goal G9: The system should be able to react to the lowering of the health
// parameters below threshold in less than 5 seconds
// and send the position of the person to the ambulance system

assert SystemProcessesRequestInLessThan5Seconds {
    no ar: AmbulanceRequest {
        ar.timeElapsed > 5 && ar.accepted = False
    }
}
```

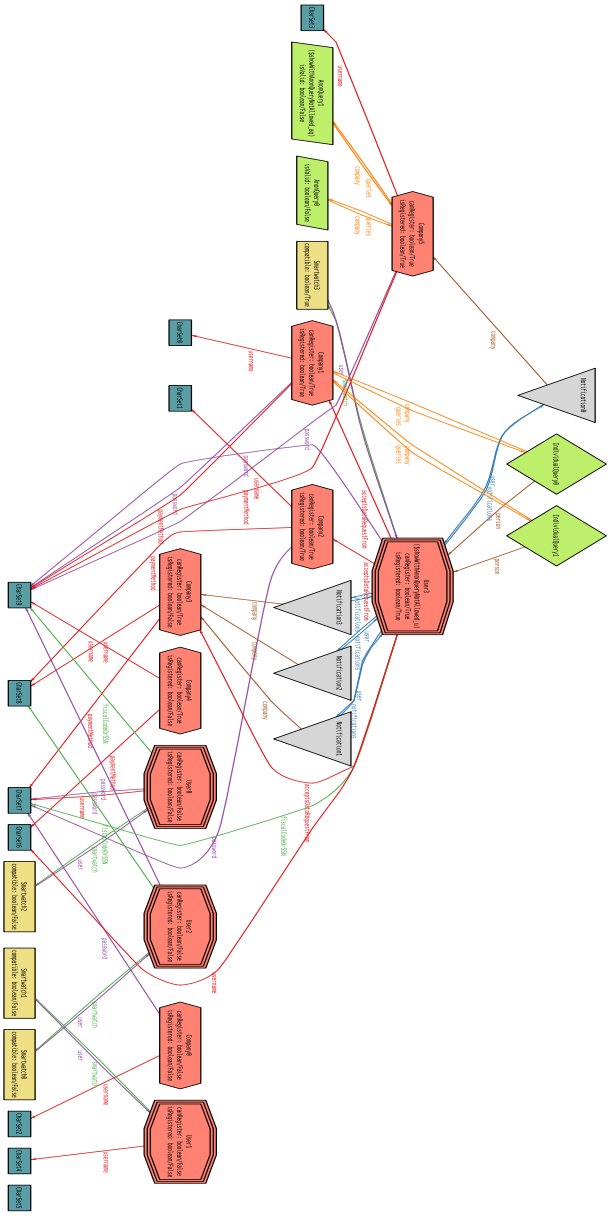
```
check SystemProcessesRequestInLessThan5Seconds for 5

pred showWithAnonQueryAllowed{
    #User > 6
    some u: User {
        u.isRegistered = True
        #u.acceptsDataRequestFrom > 2
    }
    some aq: AnonQuery {
        #aq.people > 3
    }
    #Company > 2
    #AnonQuery > 1
    #IndividualQuery > 1
    #Notification > 3
}

pred showWithAnonQueryNotAllowed {
    #User > 3
    some u: User {
        u.isRegistered = True
        #u.acceptsDataRequestFrom > 2
    }
    some aq: AnonQuery {
        #aq.people < 3
    }
    #Company > 2
    #AnonQuery > 1
    #IndividualQuery > 1
    #Notification > 3
}

run showWithAnonQueryAllowed for 10
run showWithAnonQueryNotAllowed for 10
```

## 4.2 Worlds generated

### 4.2.1 With Anonymous queries allowed



### 4.2.2 With Anonymous queries not allowed



# 5 Hours tracking

| Date | Nicola Fossati | Daniele Montesi | Francesco Sgherzi |
|---|---|---|---|
| 15/10/2018 | 2,5 | 2,5 | 2,5 |
| 20/10/2018 | 0 | 4,0 | 4,0 |
| 21/10/2018 | 1,5 | 1,5 | 2,5 |
| 24/10/2018 | 0 | 2,5 | 0 |
| 27/10/2018 | 3 | 2 | 0 |
| 29/10/2018 | 5,5 | 5,5 | 5 |
| 31/10/2018 | 0 | 4,5 | 0 |
| 05/11/2018 | 4 | 0 | 4 |
| 06/11/2018 | 5 | 1 | 6 |
| 07/11/2018 | 10 | 2 | 9 |
| 08/11/2018 | 4 | 0 | 2 |
| 09/11/2018 | 3 | 0 | 4 |
| 10/11/2018 | X | X | X |