Politecnico di Milano

AA 2018/2019

POLITECNICO
MILANO 1863

RASD – Requirement Analysis and Specification Document

Version 1.1 - 16/12/18

Authors:

Professor:

Elisabetta Di Nitto

Table of contents

1. Introduction

1.1 Purpose

1.1.1 General purpose   Qua spiegano molto approfonditamente l'applicazione, inserendo già i goals

TrackMe is a company that wants to offer some software-based services: Data4Help, AutomatedSOS and Track4Run. The core service is Data4Help and the others are thought as possible integrations of it. This document's purpose is to deeply describe all the proposed applications to provide a support for the stakeholders.

TrackMe wants to offer the possibility to third parties to monitor the health status and position of users through the Data4Help service. The application has to acquire the users' data in some way (ex: through a wearable device) and offers the possibility to third parties to access them. Data can be queried in a specific way or in an aggregate way: in the first case the request must be accepted by the user. Third parties can make specific requests or ask to access to data as soon as they are recorded by the application, asking for subscription. TrackMe wants to ensure also that the access to aggregate data let them anonymous: Data4Help will make data available if, and only if, anonymity can be granted. The application offers also the possibility to the user to insert some of his personal data and to consult them and his registered data and stats.

TrackMe also wills to exploit the possibility of recording users' data to offer another service: AutomatedSOS. Its aim is to support third parties in monitoring health status of the applications' users acquiring their vital signs through some device (as for Data4Help). This service is thought especially for elderly people (although it can be activated by any user) and is thought to automatically activate a request for the emergency services (ex: departure of ambulance) of third parties.

Finally, TrackMe wants to offer a service to track athletes participating to a run. This service is called Track4Run and, in this case, the application allows third parties that wants to organize a run to do it defining also its path, athletes to enroll for a run and also offers the possibility to follow the run to every user tracking runners' position during the manifestation. For this functionality it is assumed that the competitions are organized by third parties that are recognized organisms.

1.1.2 Goals

For the description of the goals the analyzed perspective is the one of the S2B:

- G1: The application must allow third parties to monitor location and health status of individuals (with their permission) and groups.

- G2: The data related to the users must be anonymized by the system in case of aggregate queries.

- G3: The system must allow the users to consult their correctly registered stats and data.

- G4: In case of real emergency, the system must guarantee a reaction time (in reporting it, providing the right information) of less than 5 seconds from the time the health parameters are out of bounds.

- G5: The system must allow recognized third parties to organize and manage a run.

- G6: The system must allow users to enroll to an organized run.

- G7: The system must allow users to see on a map the position of all runners during a run.

It is worth noting that the use of the 'must' verb points out that each goal represent a mandatory objective of the system. However, for what concerns G4, this definition is not so rigorous since that objective is clearly something to aim for, but that, in some cases, will not be possible to enforce due to technological limitations.

1.2 Scope

The Data4Help service is offered to common users and to third parties that want to acquire data (health status and location) about them, so it is thought for companies that maybe do not have the appropriate competences internally and need to be supported in the IT management: the service stands in the middle.

Figure 1 TrackMe as an intermediary

The S2B will give to the user the possibility to insert his own body measurements, possible pathologies and eating habits (it's not mandatory to insert these data) and will monitor and register his heartbeat, his position, his body temperature, his walking/running covered distance and his energy consumed. The system will also allow the user to choose which data to register and which not. Data4Help, besides helping users to monitor their health and position statistics and to consult their inserted data, supports companies in the analysis of the mentioned types of users' data and allows them, for example, to fragment their clients according to their age, their mobility, the places they visit etc. The user can, obviously, accept or refuse the data acquisition's request by the third party. It must be assumed that users' devices are capable of acquiring the mentioned data (sensors + GPS): if a sensor for some kind of data is not present, that data won't be available. The authorized personnel of the third party can access the data logging in and querying the TrackMe platform on the computer systems of the company (both users and third parties have first to register to the system). The system relies on the fact that all the users can be identified with a unique key (their fiscal code) and so the third party can ask to access their data through it. Data can be queried in two ways: the third party can make a request to the system to retrieve health status' or location's data of a single customer or it can ask for aggregate data on the base of some parameter (ex: data of all users with a certain age, with certain body measures, of all users that work in a certain area etc.). The third party can also request to the system to receive users' data in a live way, as soon as they are produced, without the necessity to make a query. Users will be notified of individual requests and will have the possibility to accept or refuse them. Individual requests may also represent requests of subscription to user's data and, also in this case, users will have the possibility to accept or refuse them. In case of acceptance, they will be then allowed to remove their subscription at any

time. The aggregate requests are handled directly by the Data4Help applicative that will provide data only if they can be showed in an anonymous way, otherwise it will notify the third party that is impossible to satisfy the request: TrackMe makes data available if, and only if, the query is satisfied by at least 1000 different users' data.

To offer the AutomatedSOS service the user directly agrees to his data processing when adding the service (he won't be queried every time, but will give his consent only once at the beginning). In this case the service monitors the users' health parameters and automatically signals the emergency to the third party that is notified when certain health's parameters go below or over certain thresholds so that an ambulance can be sent to the customer's location to help him (this responsibility is left to the third party exploiting AutomatedSOS service: AutomatedSOS has just to report the emergency). The service should guarantee a reaction time in reporting the emergency of less than 5 seconds from the moment in which the parameters go out of certain bounds: in this case it must be assumed that the users' device collect and sends data in real time to guarantee a right functioning of the service. The system provides the encoding of the call to the ambulance, the location and health status of the person as a reaction to the person's health problem that belongs completely to the environment. This service is thought to be exploited on one side by the users and on the other especially by third parties as public authorities' that, having access to such system, want to monitor the mentioned citizens' parameters and want to protect their health status (it is not very useful for companies that can't provide emergency services).

For what concerns the Track4Run application, in this case TrackMe offers a service that can be exploited by an organizer of a run to arrange the run and its path, by the users willing to participate to a run to enroll for the competition and also just to follow the evolution of the run. The system offers the possibility to organize runs to recognized third parties and to enroll for them to all users.

Both AutomatedSOS and Track4Run rely on the assumptions made for Data4Help and exploit its features.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- **User:** the 'normal' customer of the application that exploits the application only to collect his own data or to be monitored for SOS and to enroll for a run or follow its development;

- **Third party:** the customer of the application that exploits it to monitor data of 'normal' customers, to provide the SOS service or to organize a run;

- **Customer:** general TrackMe customer, can be a user or a third party;

- **Warning:** a message shown to the customer when it must be notified about something (ex: inserting a wrong input);

- **Individual Request:** request on some single user's data made by a third party;

- **Aggregate Request:** request on some users' grouped data made by a third party.

- **Health Status:** when mentioned for the application domain it refers to all the parameters that are registered for the user excluding the location: heartbeat, body temperature, walked/run distance and consumed energy.

1.3.2 Acronyms

- API = Application Programming Interface

- GPS = Global Positioning System

- UI = User Interface

- TP = Third party

- S2B = Software To Be

- PC = Personal Computer

- IT = Information Technology

- GDPR = General Data Protection Regulation

1.3.3 Abbreviations

- Gn = nth goal

- Dn = nth domain assumption

- Rn = nth requirement

1.4 Revision history

- Version 1.0:
  - First Release

- Version 1.1:
  - The 12th requirement in section 3.2.3 has been changed because it was too detailed.

- The use of Google Maps instead of other providers has been justified in section 2.1
- Some useless definitions in section 1.3.1 has been deleted and others have been changed.
- State diagram 2 has been changed: the timeout has been removed since it was useless
- A traceability matrix has been added in section 3.2.3
- The class diagram in section 2.1 has been slightly modified to make it coherent with the Alloy model.
- 'Watch a run' sequence diagram has been added and the others have been slightly changed in section 3.2 to be coherent with the DD document.
- A relevant comment for G4 has been added in section 1.1.2
- Some other very little detail has been modified/added throughout all the document.

1.5 Reference documents

- Specification document: "Mandatory Project Assignment AY 2018-2019"
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
- UML diagrams: https://www.uml-diagrams.org/
- Alloy doc: http://alloy.lcs.mit.edu/alloy/documentation/quickguide/seq.html

1.6 Document structure

The RASD document is composed by five chapters, as outlined below:

Chapter 1 is an introduction: it describes the purpose of the system informally and also by making use of the list of goals which the application has to reach. Moreover, it defines the scope, where the aim of the project is defined in greater detail and the application domain and the most important shared phenomena are shown.

Chapter 2 offers an overall description of the project. Here the actors involved in the application's usage lifecycle are identified and the boundaries of the project are defined, listing all the necessary assumptions. Furthermore, a class diagram is provided, aid to better understanding the general structure of the project, with all the related entities. Then some state diagrams are listed to make

the evolution of the crucial objects clear. Finally, the functions offered by the system are here more clearly specified, with respect to the previously listed goals.

Chapter 3 represents the body of the document. It contains the interface requirements, which are: user interfaces, hardware interfaces and software interfaces. It then lists some scenarios to show how the system acts in real world situations, followed by the description of the functional requirements, using use cases and sequence diagrams. All the requirements necessary in order to reach the goals are given, linked with the related domain assumptions. Lastly, the non-functional requirements are defined through performance requirements, design constraints and software system attributes.

Chapter 4 contains the Alloy model of some critical aspects with all the related comments and documentation in order to show how the project has been modeled and represented through the language.

Chapter 5 shows the effort which each member of the group spent working on the project.

2. Overall Description

2.1 Product perspective

The idea is to build AutomatedSOS and Track4Run upon Data4Help: they are additional services that can integrate Data4Help and they can be activated also in a second moment, providing some additional information. To monitor the position and the other data of its users, the application exploits the device's available sensors and GPS and for the managing of a run the system exploits Google Maps' APIs. Google Maps has been chosen instead of other mapping and location provider since it offers a very large library of APIs with extensive documentation and so it will be easier to exploit them and also to add possible future functionalities that has to exploit mapping services to our system.

The below high-level class diagram provides a model of the application domain: it contains only few essential attributes for the various classes and does not include every class that will be necessary to define the Model (useful data) of the system. Third parties and users communicate through the system provided by TrackMe: the third party can make a query for individual or aggregate data to the system exploiting the Data4Help service. The user, in the first case, can agree or refuse giving an answer to the system, while, in the second case, it can be the system that refuses to provide data

if it can't guarantee their anonymity; the third party can also ask to the system through a request that must be accepted to subscribe to users' data (both individual or aggregate) if it wants to receive them as soon as they are produced and, also in this case, the user will have the possibility to accept or refuse the subscription in case of individual request. The third parties can also activate the AutomatedSOS service to receive emergency calls and activate the Track4Run service to organize and manage run competitions. The same goes for the users that can have the possibility to automatically generate emergency calls and to enroll or follow run competitions activating the appropriate services.
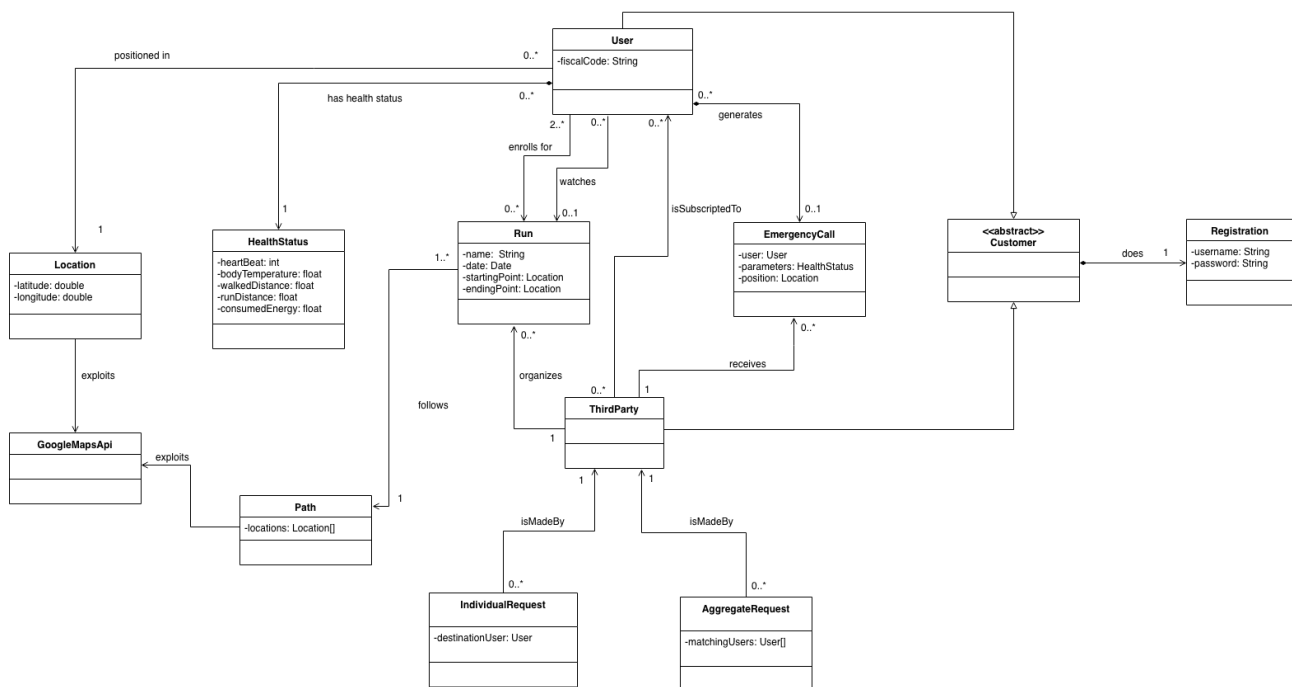
Figure 2 - Class diagram

'EmergencyCall' has been associated through the composition to the 'User' because it has no sense (on a logical level) to exist if it is not referenced by its 'User' (the same argument holds for 'HealthStatus' and 'User' and for 'Registration' and 'Customer'). 'EmergencyCall' contains a reference to the 'HealthStatus' and 'Location' of 'User' to guarantee a direct access to them even if they are retrievable from 'User' itself.

Now we are going to analyze some critical aspects of the application, modeling their behaviors and showing the evolution over time of their states through appropriate state diagrams, which are reported below.
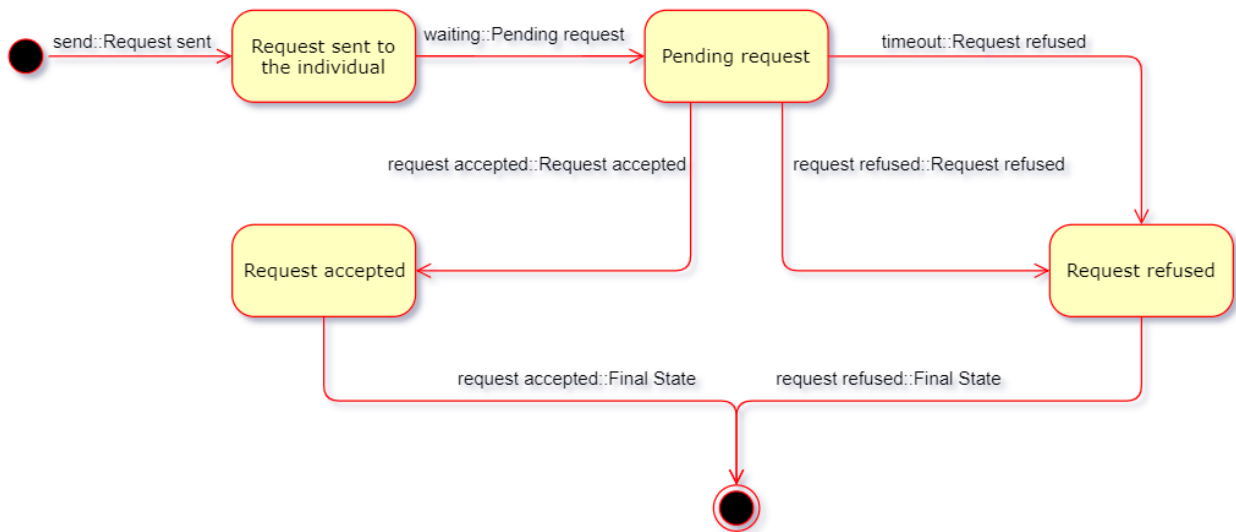
Figure 3 - State diagram 1: Request for individual data

In this first state diagram (Figure 3), the request sent to an individual by a third party to ask for his data is modeled, from the dispatch to the final state, which depends on the individual's choice.
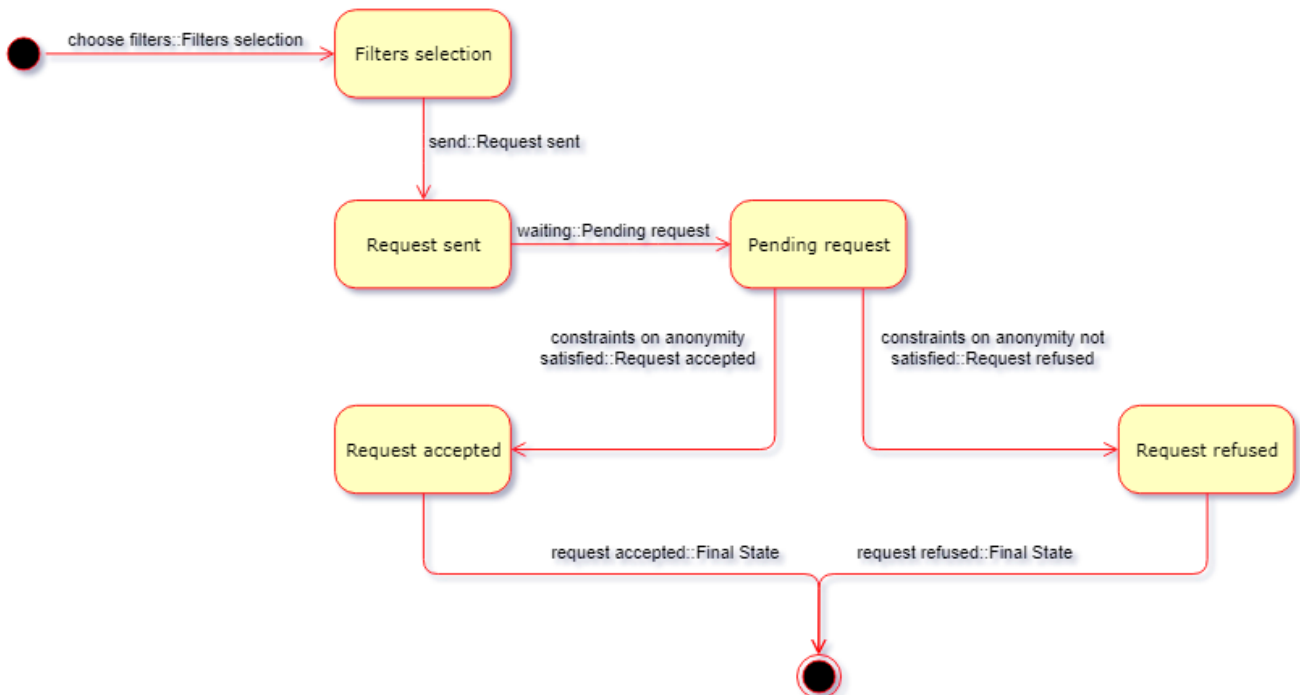
Figure 4 - State diagram 2: Request for aggregate data

Also in this case (Figure 4), to be modeled is the request by the third party, but this time it is a request for aggregate data, so the flow is different. Indeed, at first the third party is asked to select the filters it wants to apply (age, geographic area etc.), then the evolution of the states is similar to that in the previous diagram, until the check for anonymity, which results in a positive or negative response, depending on the fact that the system can guarantee the privacy of their users.
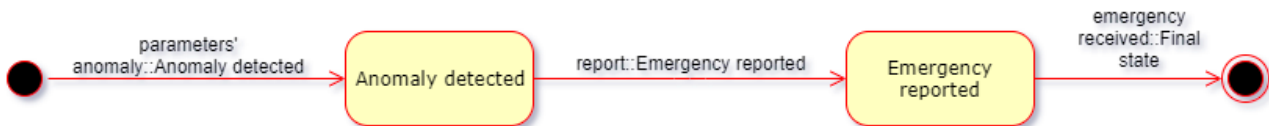
Figure 5 - State diagram 3: SOS request

This state diagram (Figure 5) is inherent to the AutomatedSOS service. Indeed, it shows the evolution of the states necessary to handle an emergency, from the detection of an anomaly in the user's parameters to the submission of the emergency with all the relevant data to the third party, so that it can send an ambulance to assist the user.
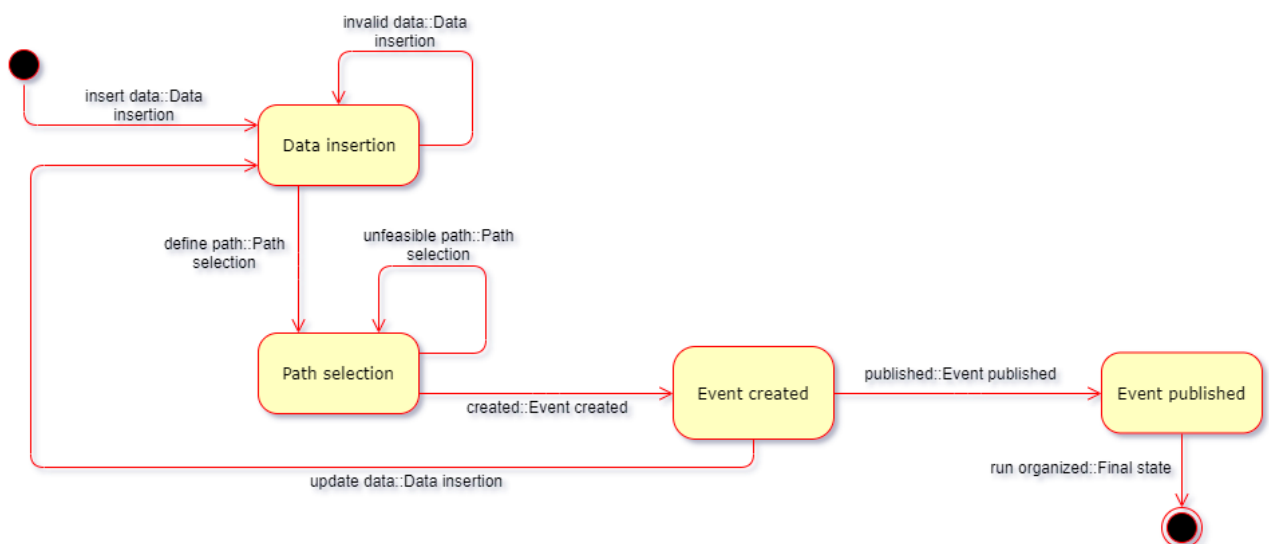
Figure 6 - State diagram 4: Run organization

Finally, this state diagram (Figure 6) represents the transformation of the event (intended as run) entity. It's important to note that 'Data insertion' and 'Path selection' are two different states. This is because the process of path definition is intended to be subsequent to the insertion of data such as the name of the event, the date on which the run will take place etc. Indeed, it could be that some of these data can be necessary for the system to decide if the path defined by the organizer is feasible or not. If an update in the organization of the run is necessary once the event has been created the system will redirect the third party to the 'Data Insertion' state regardless of the type of update.

2.2 Product functions

In the following section the most important product functions of the system are reported. It's important to underline that the AutomatedSOS and Track4Run services are built on top of Data4Help, so, if the user enables one or both of them, Data4Help and its product functions are still enabled.

2.2.1 Monitoring management

This product function is one of the most important for the system. The system will allow the user to sign up entering a username, a password and his data. Some data will be mandatory (the biographical ones), while some other data will not (the body measurements, the pathologies and diet information). Other data of the user will be collected by the S2B periodically, like the heartbeat and the location. The system will have to allow the third party that wants to retrieve some data, to choose between accessing the data of a specific individual or accessing anonymized data of groups of individuals. In the former case, the third party will provide the fiscal code of the user, who will accept or refuse to provide the data. In the latter case, the third party will write a constraint about what kind of individuals must be in the group (for example the individuals whose age is above 50 years) coupled with the required data. In order to protect the privacy of the individuals and avoid a misuse of the data, the request will be automatically refused if the group has less than 1000 people. The system will also allow the third party to subscribe to new data and receive it as soon as they are produced.

2.2.2 SOS management

This product function must be guaranteed by the system if, and only if, the customer enables the AutomatedSOS service. It is necessary that some third parties, that in this case are thought to be companies that can provide emergency services, register to the system so that it can send them the parameters of the user in case of emergency, as it will be explained in the following. The system will monitor in real time, through an appropriate wearable device, the vital parameters of the user. As soon as those parameters are found to be out of some thresholds, the system will send location and health status of the user to the nearest registered third party's application, so that it will be able to send as soon as possible an ambulance with the appropriate equipment and medical staff. The S2B will care only of reporting the emergency: the actions taken to handle it are out of the service's purpose.

2.2.3 Run management

This product function must be guaranteed by the system if, and only if, the customer enables the Track4Run service. In this case the third parties are considered recognized entities that want to organize some runs. The authorized third parties will have the possibility to create the event of a run in the system, and will put some mandatory data about it: the name, the path, the starting point, the ending point, the date and the starting time and the maximum number of participants. The system will allow the users that enabled this service to enroll for an organized run, showing them the list of organized run on a calendar. The system will also provide to the users the possibility of following the developing of a run by showing a map and the position of the runners in real time. So, the location of the runners will be captured by the system in real time, using a device provided with GPS.

2.3 User characteristics

The actors of the application are the following:

1. User: a person who registers to Data4Help and, possibly, to AutomatedSOS and/or Track4Run. The user's role will be explained better in the following distinctions (depending on the service):

    - Data4Help: the user is a person who provides his data by explicitly writing it or indirectly by a device that collects the data and send them to the system. The user can see his data on the application.

- AutomatedSOS: the user is a Data4Help's user who enables this service. He provides to the system his health status in real time and his location in case of health emergency (both collected and sent by his device).

- Track4Run: the user is a Data4Help's user who enables this service. He can enroll for run competitions. Also, during ongoing competitions, he provides his location in real time to the system (collected and sent by his device).

2. Third party: it is necessary to distinguish what the third party is depending on the service:

- Data4Help: the third party is any organization that wants to access data of the users. It has to register to the system in order to receive data, which must be explicitly asked.

- AutomatedSOS: the third party, supposed to be a company that can provide emergency services, is an organization capable of providing medical aid to the users after receiving the report from the system. It has to register to the system in order to receive the data in case of emergency.

- Track4Run: the third party, supposed to be a recognized institution, will be able to organize and manage runs.

2.4 Assumptions, dependencies and constraints

- D1: The devices that acquire users' position provide location with an error of 5 meters at most.

- D2: The sensors of devices that acquire users' parameters provide information with an error of at most 1% (for each type of data) compared to the real values.

- D3: Each fiscal code number is unique.

- D4: A request for data is considered to be anonymous if, and only if, it is satisfied by at least 1000 individuals.

- D5: Each user that wills to activate AutomatedSOS service always wears the device that acquires data.

- D6: The devices on which the services are exploited can provide real time information.

- D7: Third parties that want to organize a run have demonstrated that they are acknowledged institutions.

- D8: Each user that participates to a run wears the device acquiring data during the whole competition.
- D9: The internet connection works properly without failure. This assumption is too strong

Any device that can connect to internet through a browser (for example a PC, a mobile phone, a tablet, etc.) is enough for both the user and the third party to do those activities (ex: signing up, logging in, organizing a run, requiring data from a user, accepting or refusing data requests from third party) that don't require data acquisition through devices, but to exploit these various functionalities needing an internet connection, devices with a 2G/3G/4G and, possibly, WiFi connection, must be present on the customer's device.

However, some services offered by the system require some other specific hardware: in order to locate the user it's necessary for him to have a GPS device and to correctly report emergencies all the appropriate sensors must be present on the wearable device (ex: smartwatch).

For what concern the exploited software interfaces, the system uses an external suitable service so that its architecture is simpler. The Google Maps APIs are used when providing the Track4Run service. The organizer will choose the path of its run by managing a map provided by the APIs. The user who wants to assist at a run, as well, will be provided of a map by the APIs.
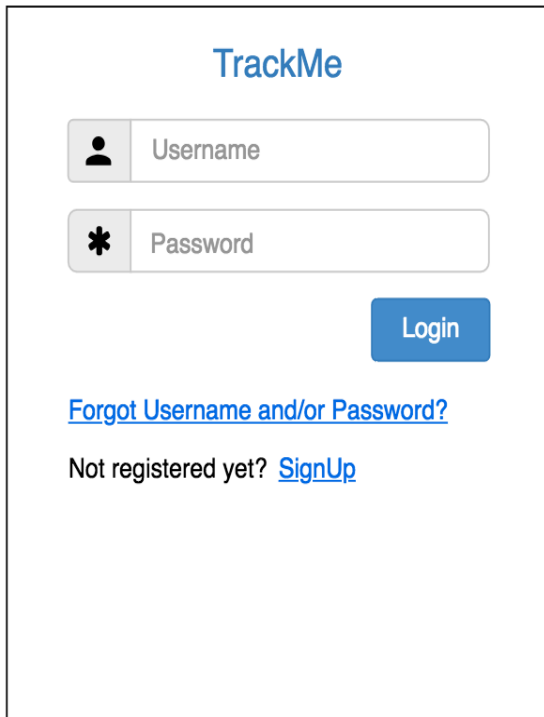
The S2B is thought to be distributed in Italy, that's why it exploits the fiscal code number in order to distinguish the users. Finally, the S2B must be compliant with the GDPR normative for the privacy.

3. Specific Requirements

3.1 External interface requirements

3.1.1 User Interfaces

The following mockups give an approximate idea of how the application's interfaces should appear: some of the most important screenshots of the interactions between the system and each of its customers (User and Third Party) are represented.
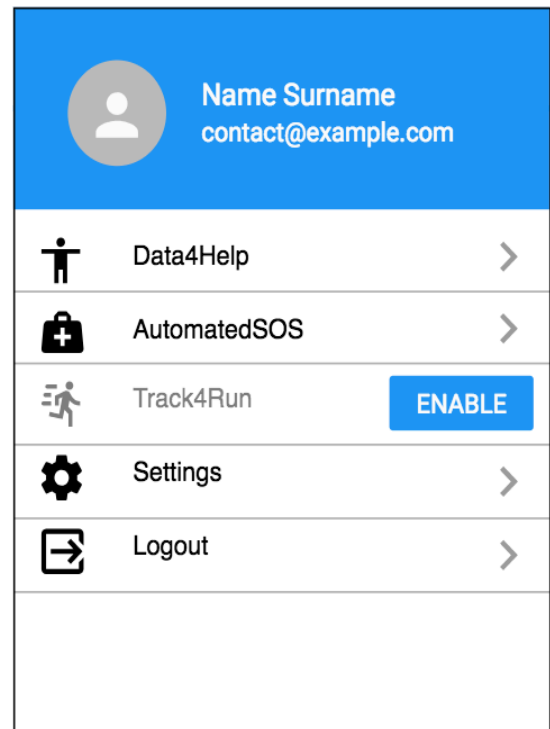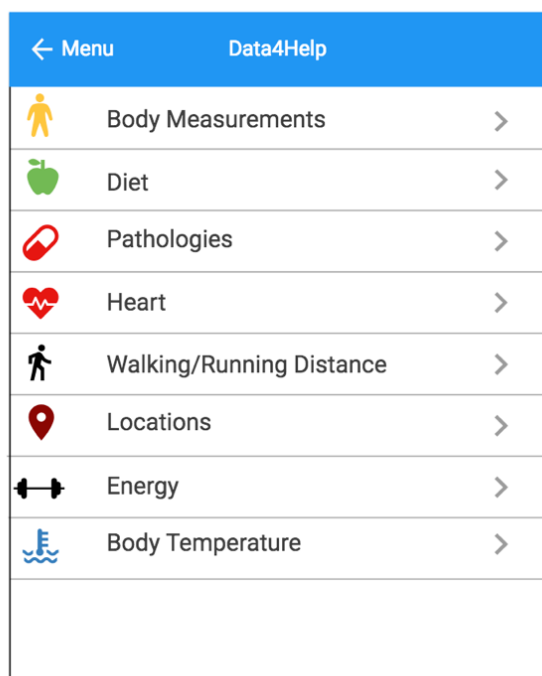
Figure 7 – Mockup: Login

Figure 8 - Mockup: Main menu

Figure 9 – Mockup: User UI: Data4Help menu

Figure 10 - Mockup: User UI: access to 'Heart' functionality

Figure 11 – Mockup: User UI Track4Run: view of days with at least one competition
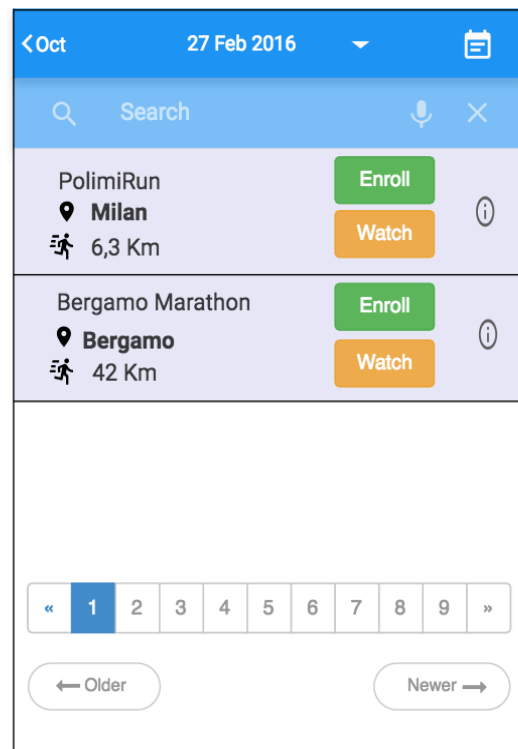
Figure 12 – Mockup: User UI Track4Run: list of scheduled competitions
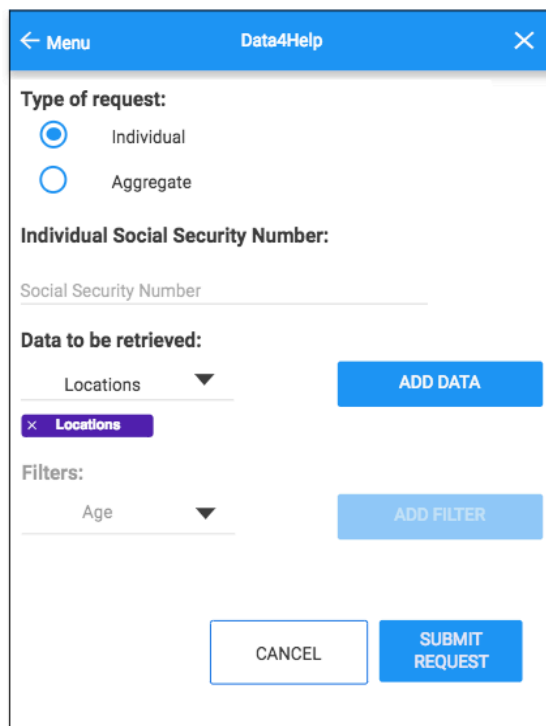
Figure 13 – Mockup: TP UI Data4Help: individual request
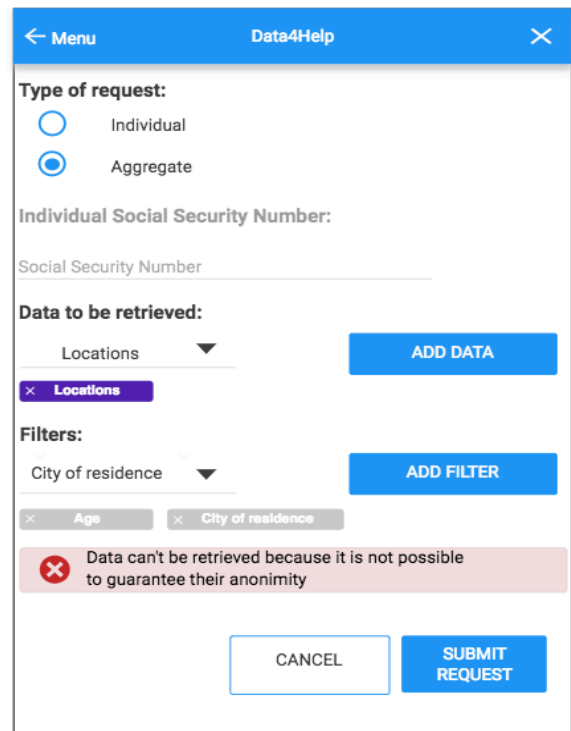
Figure 14 – Mockup: TP UI Data4Help: aggregate request
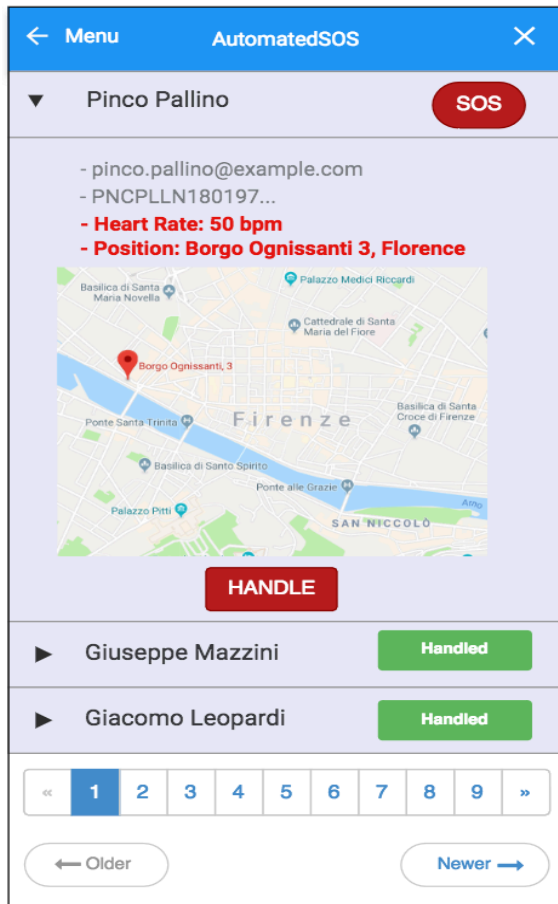
Figure 15 – Mockup: TP UI AutomatedSOS: list of reportings
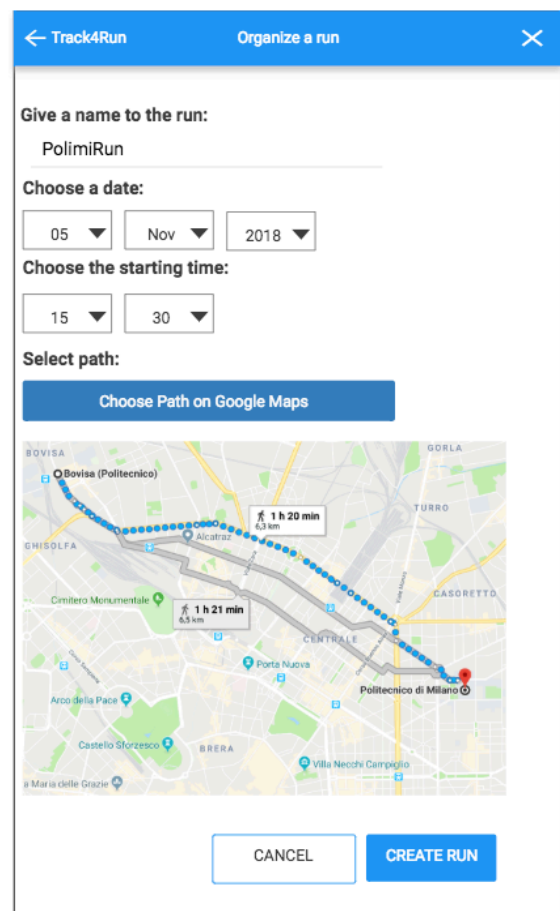
Figure 16 – Mockup: TP UI Track4Run: organize a run

Figure 7 and Figure 8 represent a possible login and menu interface for any kind of customer (both User and Third Party) while the other mockups represent just one side of the interaction with the system offered by TrackMe (specified in the captions).

3.1.2 Hardware interfaces

The system has no hardware interface.

3.1.3 Software interfaces

The system doesn't provide any API to external applications: APIs are offered only to and from the various TrackMe subsystems as it will be shown in the DD document.

3.2 Functional requirements

3.2.1 User

Scenarios

Scenario 1

Tyrion, an elderly man who lives alone, taking advantage of the sunny day, decides to fix up the yard, despite the fact that his doctor ordered him to not push himself too hard to avoid unpleasant inconveniences. Indeed Tyrion should have listened to the doctor because after a little more than one hour he starts to feel fatigued. Fortunately, he is wearing a wearable device with the AutomatedSOS service offered by the company TrackMe installed, which detects that the man's parameters are below the threshold and immediately reports the emergency, allowing Tyrion to be still alive.

Scenario 2

Marco is a competitive and sportsman person, so he wants to enroll for a run. He decides to use the Track4Run service, in order to find a run. He finds it and enrolls for it. The run will start in a month, so he decides to train for it with his friend Federico, who will also participate in the competition. While they are running together, Marco stumbles into a stone and breaks his knee. Unfortunately he won't be able to take part to the competition, but he really would like to support his friend Federico at least. He finds out that it is possible to monitor the position of a runner in real time during the competition thanks to the Track4Run service. So, while he is lying down on the bed of his hospital, he is able to keep track of how well is Federico's run going.

Use Case Diagram
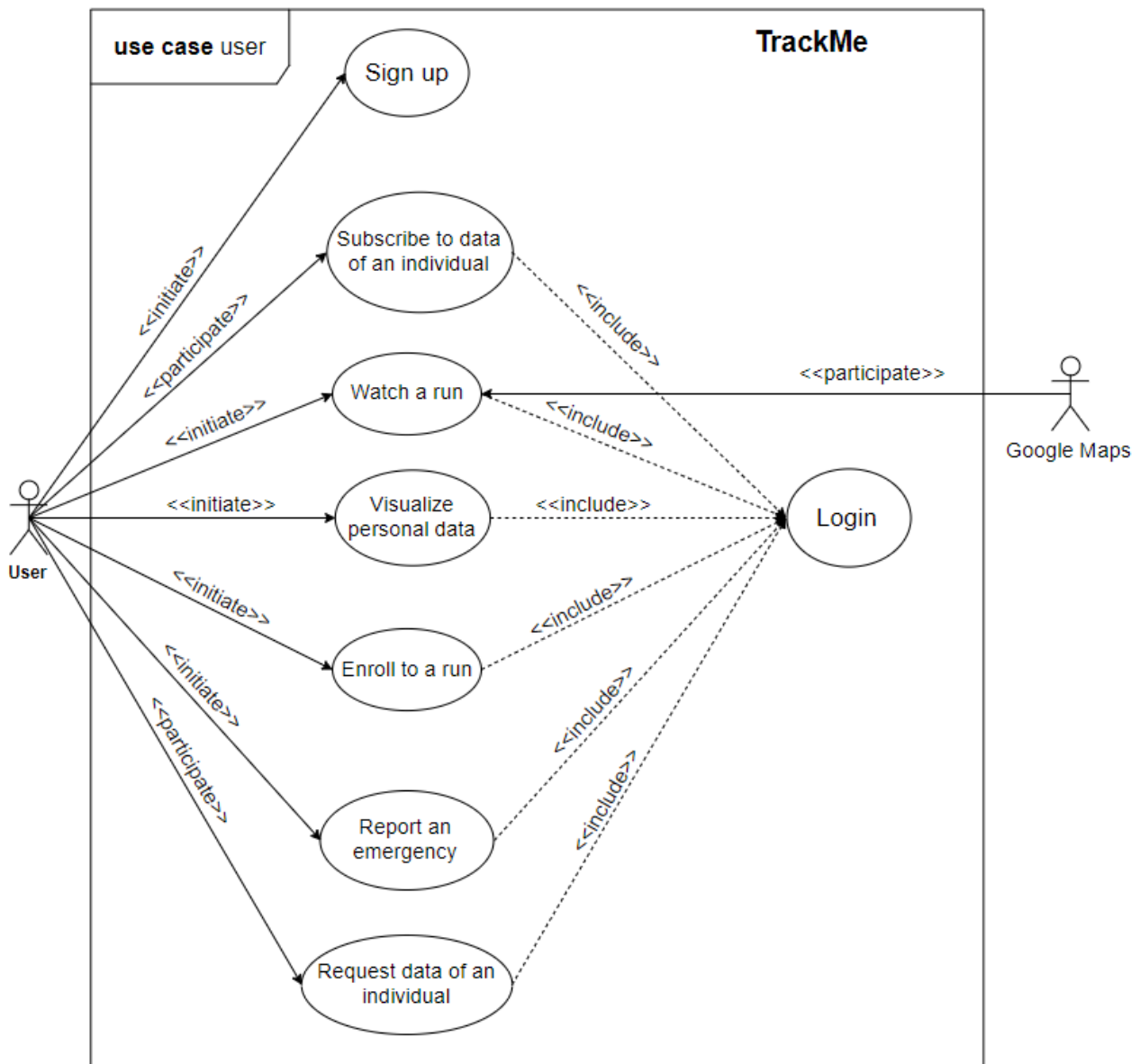
Figure 17 - Use case diagram: User

Use Cases

Name	Sign up
Actor	User
Entry conditions	The user has opened the application on his device.
Events flow	1. The user chooses the "Sign up" option. 2. The user fills the mandatory fields.

	3. The user fills the optional fields with not mandatory data. 4. The user chooses the confirmation option. 5. The systems saves the data.
Exit conditions	The user is registered and the system has his data stored.
Exceptions	1. The user was already registered. In this case the system warns the user and suggests him/her to do the login. 2. The username is already taken. In this case the system warns the user and suggests him to change the username. 3. The username doesn't fill all the mandatory fields. In this case the system warns the user and notifies him which fields were left unfilled.

Name	Login
Actor	User
Entry conditions	1. The user has opened the application on his device. 2. The user has already done the "Sign up" activity.
Events flow	1. The user chooses the "Login" option. 2. The user enters username and password in the respective fields. 3. The user chooses the confirmation option.
Exit conditions	The user is logged in and the system allows the user to visualize and manage his account and his data.
Exceptions	1. The user enters the wrong username. 2. The username enters the wrong password.

	In both cases, the system warns the user and notifies him which field is wrong, suggesting to correct it.

Name	Visualize personal data
Actor	User
Entry conditions	The User has already done the "Login" activity.
Events flow	1. The user selects the type of data that he wants to consult. 2. The system provide the single data if they are of 'static' type (data inserted manually by the user such as his weight) or with average and last measurement if they are of 'dynamic' type (data measured by the system such as the heartbeat).
Exit conditions	The user is provided with the requested data.
Exceptions	\

Name	Report an emergency
Actor	User, Third party
Entry conditions	1. The user is a member of the service AutomatedSOS. 2. The user has successfully logged in.
Events flow	1. The system detects that the parameters of the monitored user are below or above the defined threshold. 2. The system sends an alert to the third party, notifying the emergency and sending the location and the status of the user. 3. The third party receives the emergency call.

Exit conditions	The emergency is correctly reported to the third party.
Exceptions	\
Special Requirements	The alert is received by the third party within 5 seconds from the detection of parameters below or above the threshold.

Name	Watch a run
Actor	User
Entry conditions	1. The user has activated the service Track4Run. 2. The user has successfully logged in.
Events flow	1. The user selects the run he wants to see. 2. The user is allowed to select a runner to follow him or to navigate on the map and focus on a certain area.
Exit conditions	The user can see the progress of the run and the participants' position on the map.
Exceptions	\

Name	Enroll to a run
Actor	User
Entry conditions	1. The user has activated the service Track4Run. 2. The user has successfully logged in.
Events flow	1. The user opens the service Track4Run. 2. The user selects the run he wants to enroll for from a list provided by the system containing only the competitions to come. 3. The user sees a recap of the information about the run. 4. The user confirms the participation.

	5. The system sends to the user a receipt of his enrollment.
Exit conditions	The user is regularly registered to the run.
Exceptions	The user tries to enroll for a run that reached the maximum number of participants. The system notifies the user with a warning.

Sequence Diagrams
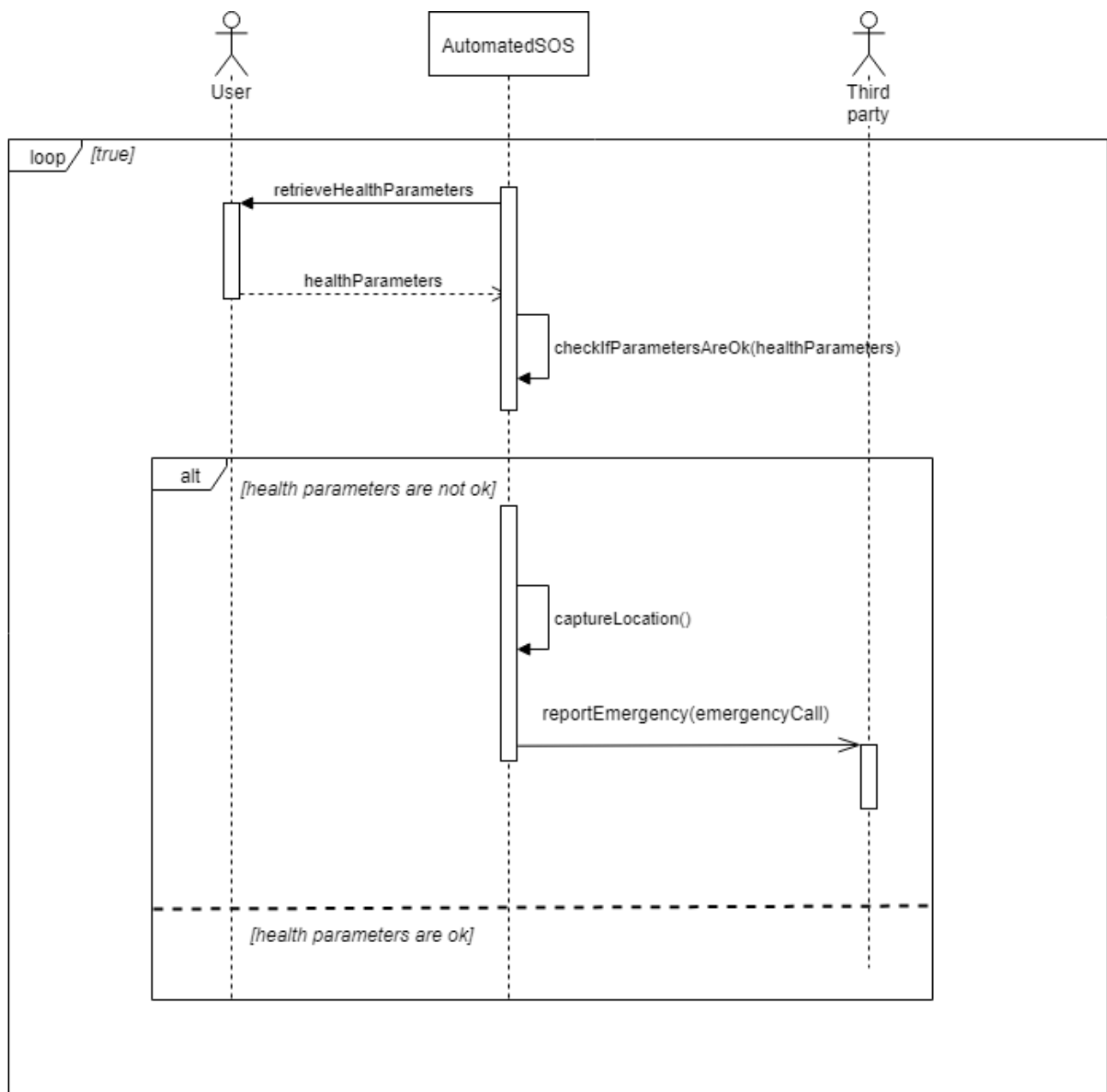
Figure 18 - Sequence diagram: Report an emergency
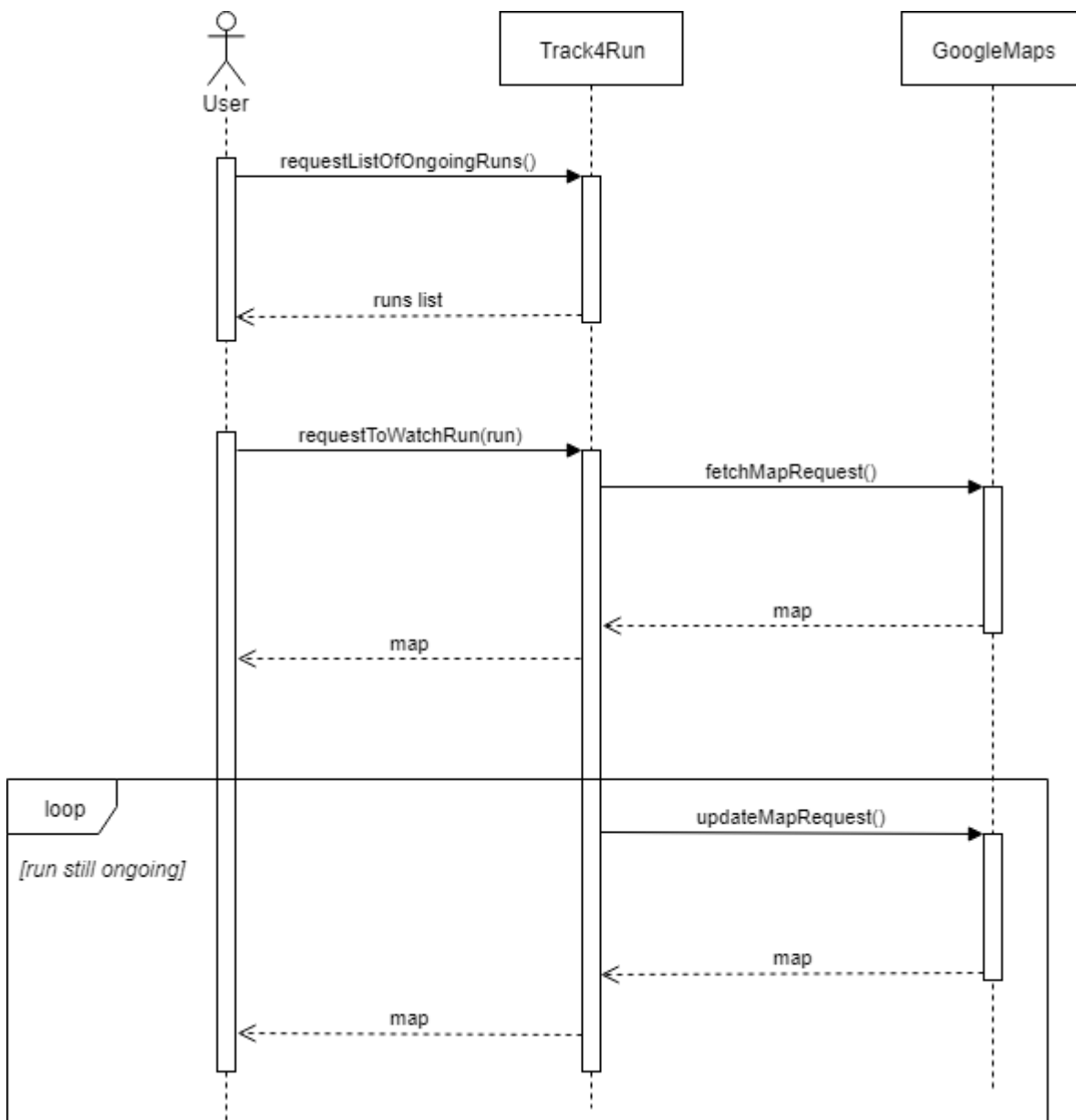
Figure 19 - Sequence diagram: Watch a run

3.2.2 Third party

Scenarios

Scenario 3

Giovanni suffers from heart's problem, and a private hospital is taking care of him. After some analysis, the doctors of the hospital find out that his health has improved, and it is only necessary to monitor his heartbeat. The hospital would like to allow Giovanni to go home so that his bed is free for another patient, but monitoring his heartbeat constantly is needed. The hospital finds out that monitoring his heartbeat is possible even though he is at home or in any other place, thanks

to Data4Help. It just needs to sign up and log in to the service and ask Giovanni's heartbeat by providing his fiscal code. Giovanni, who also needs to sign up and log in to the service, will just accept the request and the hospital will see his heartbeat. The hospital can also ask for a subscription in order to get new data as soon as they are produced.

Scenario 4

ISTAT is doing some research activity in order to analyze the health status of smoking people. In order to do that, it requests to the system to access the data of those who live in Milan and smoke. Specifically, ISTAT wants the heartbeat of those people. Data4Help provides this data to ISTAT.

Scenario 5

An emergent start-up, located in a small city, provides a food delivery service, and has the peculiarity of selling vegan food only. The company wants to know the location of vegan people in the small city, in order to do some targeted advertising. So, they ask that piece of information to Data4Help which stores the eating habits of its users. Because the city is small, there are only 853 vegan people in there, so Data4Help refuses to provide the requested data in order to protect their anonymity.

Scenario 6

Politecnico di Milano decides to organize the annual run called PolimiRun. Once registered to the Track4Run service offered by the TrackMe Company, the organizers can define all the useful information to make the run enjoyable by the participants, including the path which the latter will have to travel.

Furthermore, through the services offered by TrackMe, Politecnico can ask for the data of the users involved into the run to eventually analyze them.

Use Case Diagram
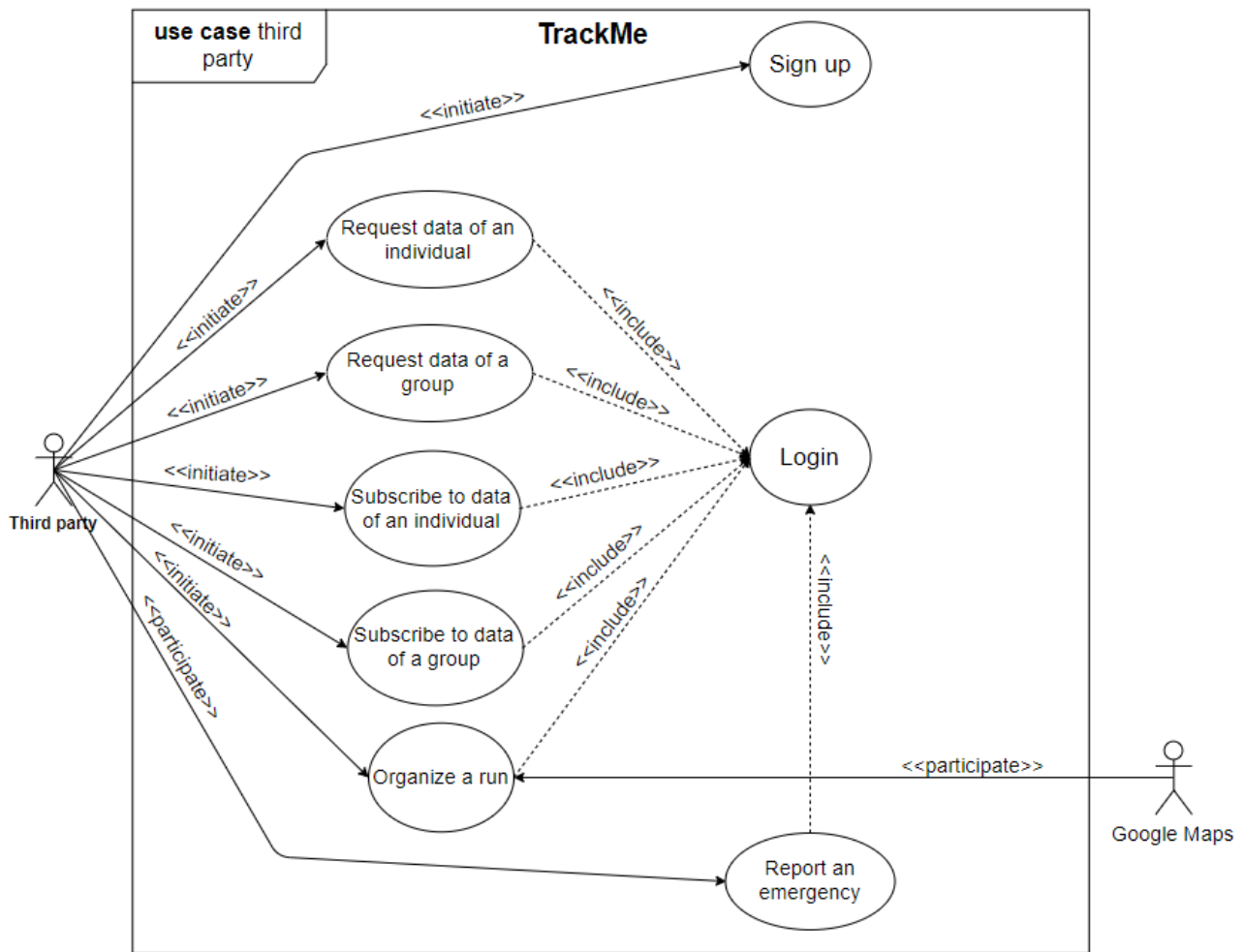
Figure 20 - Use case diagram: Third party

Use Cases

The Login and Sign up use cases are identical to the ones described for the user so they are not repeated here.

Name	Make an individual request
Actor	Third party, user
Entry conditions	The third party has already done the "Login" activity.
Events flow	1. The third party chooses the "Make an individual request" option.

	2. The third party inserts the type of data it wants to retrieve. 3. The third party writes the identifier of the individual (his fiscal code). 4. The third party chooses the confirmation option. 5. The system queries its database with the third party's request and finds who the user is. 6. The system forwards to the user the third party's request. 7. The user sees the request forwarded by the system. 8. The user chooses the "Accept" option. 9. The system provides to the third party the requested data about the user.
Exit conditions	The third party obtains the data about the requested individual.
Exceptions	1. The user chooses the "Refuse" option instead of the "Accept" option. In this case, the system notifies the third party that the request was refused by the user. 2. The timeout for the waiting of the answer expires. In this case, the system notifies the third party that the request was refused by the user. 3. There is no user registered in the system with the identifier specified by the third party. In that case, the system warns the third party and suggest him to try with another identifier.

Name	Make an aggregate request
Actor	Third party

Entry conditions	The third party has already done the "Login" activity.
Events flow	1. The third party chooses the "Make an aggregate request" option. 2. The third party inserts the type of data it wants to retrieve. 3. The third party fills the fields with the constraints of which groups of individuals the third party is looking for. 4. The third party chooses the confirmation option. 5. The system queries its database with the third party's request. 6. The system gives the requested data to the third party.
Exit conditions	The third party obtains the data about the group of individuals that respect the imposed constraints.
Exceptions	The individuals that respect the constraints found by the system are less than 1000. In this case, the system notifies the third party that the requested data can't be given in order to protect the anonymity of the users.

Name	Subscribe to data of an individual
Actor	Third party, User
Entry conditions	"Make an individual request" activity has been successfully done.
Events flow	1. The third party chooses the "Subscribe to data" option that appears as soon as the data requested have been shown by the system. 2. The system forwards the request of subscription to the user.

	3. The user sees the request forwarded by the system.
	4. The user chooses the "Accept" option.
	5. The systems registers that the third party is subscribed to that data.
	6. The user produces some new data.
	7. The system provides the data to the subscribed third party.
Exit conditions	The third party is registered to the data and the system provides them to it as soon as they are produced.
Exceptions	The user refuses the request for subscription. The third party is notified by the system with a warning.

Name	Subscribe to data of a group
Actor	Third party
Entry conditions	"Make an aggregate request" activity has been successfully done.
Events flow	1. The third party chooses the "Subscribe to data" option that appears as soon as the data requested have been shown by the system.
	2. The systems registers that the third party is subscribed to that data.
	3. The group of users produce some new data.
	4. The system provides the data to the subscribed third party.
Exit conditions	The third party is registered to the data and the system provides them to it as soon as they are produced.
Exceptions	The number of users satisfying the request has become less than 1000 so the subscription is canceled until

	the matching users become again more than 1000 and the third party is notified with a warning.

Name	Organize a run
Actor	Third party
Entry conditions	1. The third party has activated the service Track4Run. 2. The third party has successfully logged in.
Events flow	1. The third party opens the service Track4Run. 2. The third party gives a name to the event. 3. The third party defines the date and time on which the run will take place. 4. The third party defines the path which the participants will travel. 5. The third party defines the maximum number of participants for the competition. 6. The third party publishes the event.
Exit conditions	The event for the run is online and joinable.
Exceptions	The third party enters invalid data (an invalid name or date and time for the run) or it defines an unfeasible path. A warning is showed, indicating the wrong parameter(s) and the motivation.

Sequence Diagrams
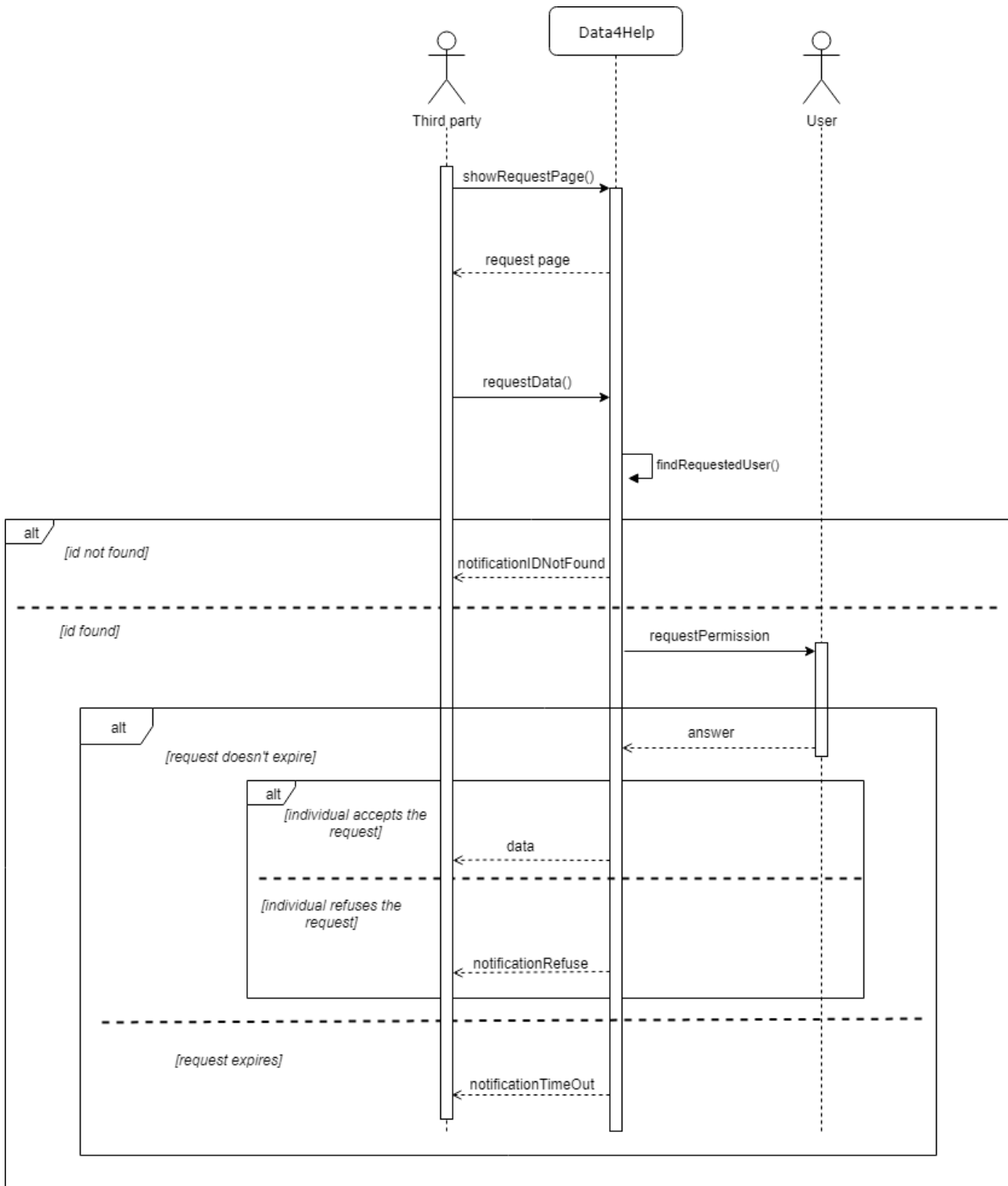
Figure 21 - Sequence diagram: Make an individual request
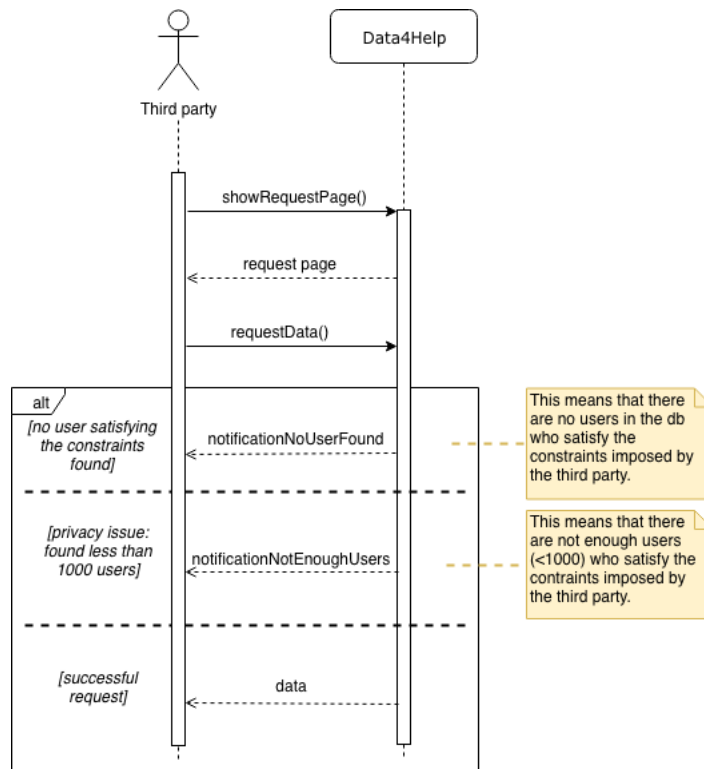
Figure 22 - Sequence diagram: Make an aggregate request
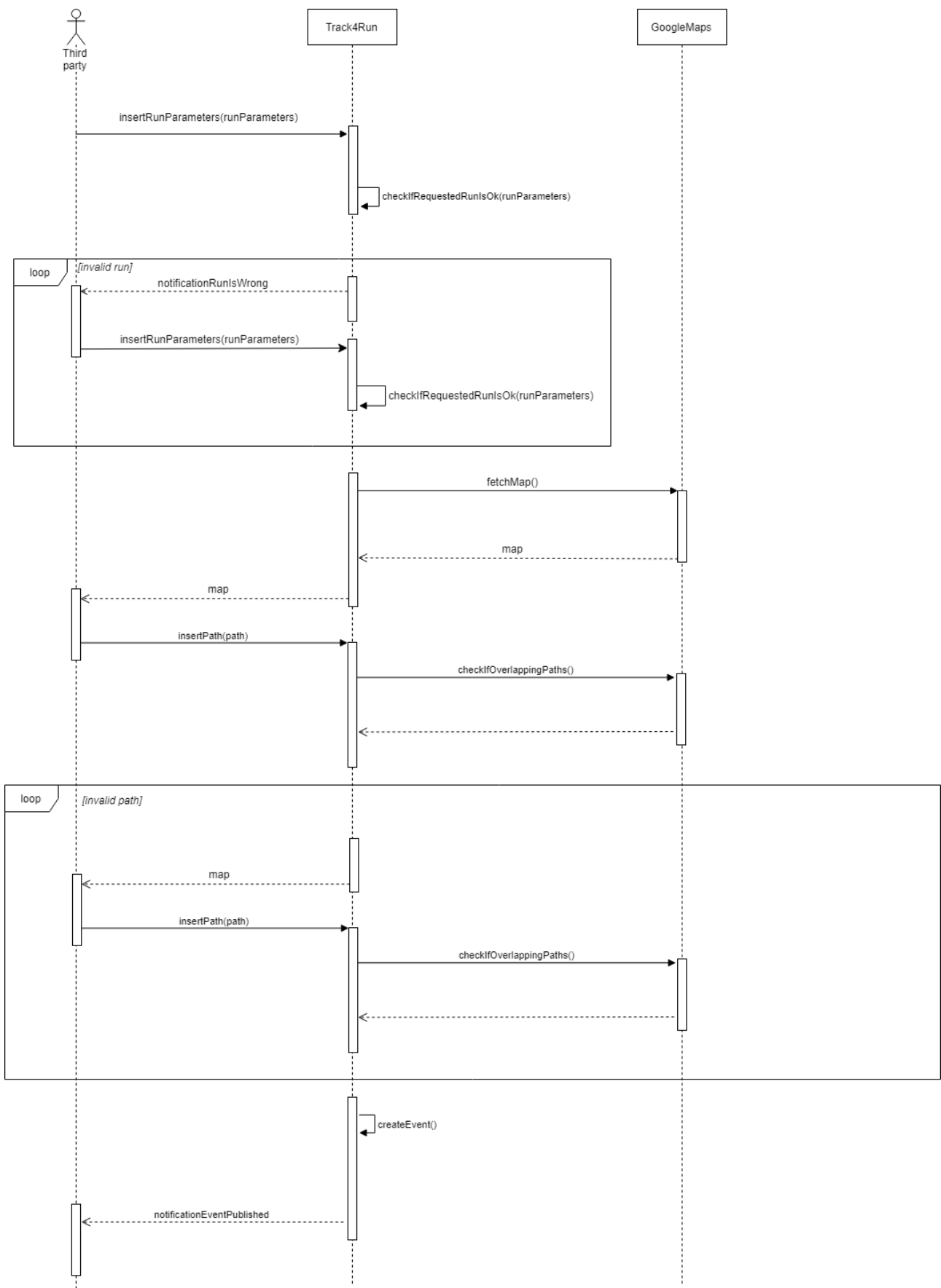
Figure 23 - Sequence diagram: Organize a run

In this section we show that the requirements ensure the satisfaction of the goals in the context of the domain assumptions: the list of requirements and domain assumptions under each goal have this purpose.

G1: The application must allow third parties to monitor location and health status of individuals (with their permission) and groups.

- **R1**: The system must save the collected data of users registered to Data4Help in real time.

- **R2**: The system has to allow the third party that wants to retrieve some data to choose between an individual request or an aggregate one.

- **R3**: In case of a query for data of an individual the system has to ask to the third party the individual's fiscal code.

- **R4**: In case of a query for aggregate data the system has to ask to the third party which parameters to use to filter data.

- **R5**: When a request for a specific individual's data is made the system must allow the individual to accept the request or not.

- **R6**: When a request for data is approved the system has to make the previously saved data available to the third party.

- **R7**: The system must provide to the third party the possibility to subscribe to new data and receive them as soon as they are produced both for individuals and for group of individuals.

- **R8**: The system must allow the user to accept or refuse a possible request for subscription by a third party.

- **D1**: The devices that acquire users' position provide location with an error of 5 meters at most.

- **D2**: The sensors of devices that acquire users' parameters provide information with an error of at most 1% (for each type of data) compared to the real values.

- **D3**: Each fiscal code number is unique.

- **D6**: The devices on which the services are exploited can provide real time information.

- **D9**: The internet connection works properly without failure.

G2: The data related to the users must be anonymized by the system in case of aggregate queries.

- **R9**: The requests by third parties on aggregate data must provide them if, and only if, the number of individuals whose data satisfy the request is higher than 1000.

- **R10**: When the number of users satisfying an aggregate request (for which a subscription was made) becomes less than 1000 the subscription is automatically canceled until the matching users become again more than or equal to 1000.

- **D4**: A request for data is considered to be anonymous if, and only if, it is satisfied by at least 1000 individuals.

G3: The system must allow the users to consult their correctly registered stats and data.

- **R1**: The system must save the collected data of users registered to Data4Help in real time.

- **R11**: The system must allow the User to analyze its own data and stats providing him a way to access to all registered data and stats and giving him the possibility to consult both their aggregate (ex: daily average) values and precise measurements.

- **D1**: The devices that acquire users' position provide location with an error of 5 meters at most.

- **D2**: The sensors of devices that acquire users' parameters provide information with an error of at most 1% (for each type of data) compared to the real values.

G4: In case of real emergency, the system must guarantee a reaction time (in reporting it providing the right information) of less than 5 seconds from the time the health parameters are out of bounds.

- **R12**: The application, if AutomatedSOS is activated, must send the user's health parameters and location to the third party that is the nearest to him as soon as it is detected that his parameters are out of the defined bounds.

- **D1**: The devices that acquire users' position provide location with an error of 5 meters at most.

- **D2**: The sensors of devices that acquire users' parameters provide information with an error of at most 1% (for each type of data) compared to the real values.

- **D5**: Each user that wills to activate AutomatedSOS service always wears the device that acquires data.

- **D6**: The devices on which the services are exploited can provide real time information.

- **D9**: The internet connection works properly without failure.

G5: The system must allow recognized third parties to organize and manage a run.

- **R13**: The system has to allow third parties that have activated the Track4Run service to schedule a run providing name, starting and ending point coordinates, the path, the date and time of the competition and the maximum number of participants.

- **D7**: Third parties that want to organize a run have demonstrated that they are acknowledged institutions.

- **D9**: The internet connection works properly without failure.

G6: The system must allow users to enroll to an organized run.

- **R14**: The system has to allow users who have activated the Track4Run service to enroll for an organized run showing them the organized competitions on a calendar.

- **D9**: The internet connection works properly without failure.

G7: The system must allow users to see on a map the position of all runners during a run.

- **R1**: The system must save the collected data of users registered to Data4Help in real time.

- **R15**: The system has to allow users who have activated the Track4Run service to follow the development of a run selecting an ongoing competition from a list that identifies runs by their name.

- **D6:** The devices on which the services are exploited can provide real time information.

- **D8**: Each user that participates to a run wears the device acquiring data during the whole competition.

- **D9**: The internet connection works properly without failure.

Traceability Matrix

In the following table we are going to map, for each requirement, all the use cases which are related to it. It's important to emphasize that the reported use cases are those strictly connected to the requirement of the case, while the ones that are touched indirectly are not listed for the sake of clarity. For example, the requirement R1, as it is written, it is the only one which claims that the system has the feature of collecting and saving data, i.e. the feature exploited by most of the use cases, including the requests and the subscriptions by third parties, otherwise there would be no data to retrieve. However, since the requirement specifies how these data are collected, i.e. in real time, we decided to focus only on those use cases that exploit this specific characteristic.

Requirement	Use Case
R1	Report an emergency Watch a run
R2	Make an aggregate request Make an individual request
R3	Make an individual request
R4	Make an aggregate request
R5	Make an individual request
R6	Make an aggregate request Make an individual request
R7	Subscribe to data of individual Subscribe to data of a group
R8	Subscribe to data of individual
R9	Make an aggregate request
R10	Subscribe to data of a group
R11	Visualize personal data
R12	Report an emergency
R13	Organize a run
R14	Enroll to a run
R15	Watch a run

3.3 Performance requirements

The system has to be able to serve a great number of users and third parties simultaneously. It has to guarantee quick, reactive and correct responses. Also, it's important to underline that AutomatedSOS provides a critical and vital service, so it must be ensured, when possible, a reaction time of less than 5 seconds from the time the parameters are out of bounds.

3.4 Design constraints

3.4.1 Standards compliance

The system adopts precise units of measure:

- Heartbeat: [bpm]
- Distance: [Km]
- Temperature: [Degrees]
- Consumed Energy: [Kcal]

With regard to the privacy of data, since the application processes sensitive ones, the entire project is subject to the General Data Protection Regulation (GDPR), a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA).

3.4.2 Hardware limitations

As specified in the "Dependencies and constraints" section, hardware requirements are present only in relation to specific functionalities. For example, every device with an internet connection is fine in order to sign up or log in, while in order to use the AutomatedSOS service, instead, the GPS and other sensors are needed. So in the following, to be concise, are reported all the hardware requirements needed to use every functionality:

- Connection to internet (Wi-Fi/4G/3G/2G)
- GPS
- Sensor for heartbeat monitoring, temperature measuring and to measure walking/running covered distance

3.4.3 Any other constraint

The system must respect privacy policies, in particular the privacy of the users. In that sense, this is a critical aspect because third parties can request sensible data of the users. In case of a data request of a specific user, no data will be given to the third party unless the user authorizes it. Moreover, the user will see precisely what data he is asked for. In case of a data request of a group of users, data won't be provided if the matching group has less than 1000 users in order to avoid a misuse of the data (for instance, if the third party is asking for data about 10-year-old children living in a certain street in Milano and the number of these children is two, then the third party could be able to derive

their identity simply having people monitoring the residents of the street between 8.00 and 9.00 when kids go to school).

3.5 Software system attributes

3.5.1 Reliability

The system must be able to run continuously without any interruptions. In order to do that, it must be ensured that the system is fault tolerant. For example, the central server, which contains the data, should be duplicated, just like the running processes, which provide the services. Some techniques, like the FloodSet algorithm, can be adopted to ensure the required reliability.

3.5.2 Availability

As mentioned before, a fault tolerant architecture is needed. It's important to make different assumptions for each one of the services. Indeed, if Data4Help and Track4Run can be expected to be available 99.9% of the time, AutomatedSOS contains some critical aspects tied to the nature of the service itself. That's why it is expected to have an availability of 99.999%.

3.5.3 Security

The data provided by the user contains sensitive information, so the security aspect is of primary importance. The central database on which the data reside must be protect by all the necessary measures to avoid any external and internal attack and also to handle malfunctions of the hardware. For the purpose of sending the data, encryption technique must be used in order to guarantee privacy and consistency.

3.5.4 Maintainability

The development of the application must be done so that in the future it will be easy to fix and modify it, according to the circumstances, and also in order to let cost of these operations be cheap. Appropriate design patterns will be used, as it will be better explained in a further document.

3.5.5 Compatibility

The application offers multiple services and can be used by a variety of people (it's a quite heterogeneous application). That is why it must be compatible to as many devices and technologies as possible, in order to meet the constraints contained in "Hardware and Software interfaces" sections.

4. Formal Analysis Using Alloy

In this section an analysis of some critical aspects of the system is provided exploiting Alloy. The focus is on some static constraints, in particular:

- It cannot happen that two different customers have the same username or that two different users have the same fiscal code.

- In case of an individual request refused by the user, the third party that made the request must be properly notified of the problem and it must be notified only in that case. Moreover, an individual request can be either accepted or refused by the user, not both.

- In case of an aggregate request that violate privacy constraints the third party author of the request must be properly notified of the problem and it must be notified only in that case.

- An emergency call is present if the user is having some health problems (parameters out of bounds) and it must be created only in this case.

Then, some other necessary, but less important, structural constraints are expressed and properly commented in the Alloy formal notation. These bonds are verified analyzing the worlds generated running the model and checking some relevant assertion.

It's important to note that some numerical values that were too big for having a correct analysis in Alloy have been reduced.

sig FiscalCode{}

sig Username{}

sig Password{}

sig Registration{

username: one Username,

password: one Password

}

abstract sig Customer{

```
        registration: one Registration

}


sig User extends Customer{

        fiscalCode: one FiscalCode,

        healthStatus: one HealthStatus,

        location: one Location,

        emergencyCall: lone EmergencyCall,

        runsWatching: lone Run,

        runsEnrolledFor: set Run

        --It may happen that a user enrolls for a run and then doesn't participate to it and

        --wants to follow it, for this reason it is NOT stated that 'runsWatching' and

        --'runsEnrolledFor' are disjoint

}

sig ThirdParty extends Customer{

        organizedRuns: set Run,

        warningsReceived: set Warning,

        receivedEmergencyCalls: set EmergencyCall,

        subscriptedUsers: set User

}

sig Location{

        latitude: one Int,

        longitude: one Int

}
```

```
--{latitude >= -90 and latitude <= 90 and longitude >= -180 and longitude <= 180}

--NB: scaled values for simplicity

{latitude >= -3 and latitude <= 3 and longitude >= -6 and longitude <= 6 }



sig HealthStatus{

        heartbeat: one Int,

        bodyTemperature: one Int

}



--{heartbeat >= 30 and heartbeat <= 210 and bodyTemperature >= 30 and bodyTemperature <=
45}

--NB: scaled values for simplicity

{heartbeat >= 3 and heartbeat <= 7 and bodyTemperature >= 3 and bodyTemperature <= 5}



--Contains HealthStatus and Location even if they could be retrieved from User

sig EmergencyCall{

        userToAssist: one User,

        userHealthStatus: one HealthStatus,

        userLocation: one Location

}



--The health status and location contained in the emergency call must be the health status and
location of the user for

--which is done

{userHealthStatus = userToAssist.healthStatus and userLocation = userToAssist.location}
```

```
abstract sig Warning{}


one sig RefuseWarning extends Warning{}


sig SingleRequest{

        authorOfRequest: one ThirdParty,

        subjectOfRequest: one User

}
```

--Represent the result of a single request and is associated to all the single requests that gave this result
```
abstract sig SingleRequestResult{

        request: set SingleRequest

}


one sig SingleRequestAccepted extends SingleRequestResult{}

one sig SingleRequestRefused extends SingleRequestResult{}

one sig PrivacyWarning extends Warning{}


sig AggregateRequest{

        matchingUsers: set User,

        authorOfRequest: one ThirdParty,

        privacyWarning: lone PrivacyWarning

}
```

--The request generates a warning if, and only if, the privacy cannot be guaranteed

```
--{#privacyWarning = 1 iff #matchingUsers < 1000}

--NB: scaled values for simplicity

{ #privacyWarning = 1 iff #matchingUsers < 5}



sig Path{}

sig Date{}

sig RunName{}


sig Run{

        name: one RunName,

        date: one Date,

        path: one Path,

        startingPoint: one Location,

        endingPoint: one Location

}



--All FiscalCodes have to be associated to a User

fact FiscalCodeUserConnection{

        all fc: FiscalCode | some u: User | fc in u.fiscalCode

}



--All Usernames have to be associated to a Registration

fact UsernameRegistrationConnection{

        all u: Username | some r: Registration | u in r.username

}
```

--All Passwords have to be associated to a Registration

fact PasswordRegistrationConnection{

      all p: Password | some r: Registration | p in r.password

}

fact RegistrationCustomerConnection {

      all r: Registration | some c: Customer | r in c.registration

}

--Every Customer has a unique username

fact NoSameUsername {

      no disj c1,c2: Customer | c1.registration.username = c2.registration.username

}

--Every User has a unique SSN

fact NoSameFiscalCode {

      no disj u1,u2 : User | u1.fiscalCode = u2.fiscalCode

}

--The third party is notified of the refusing if the request has not been accepted

fact RefuseWarningIfRefuse{

      all sr: SingleRequest | one r: SingleRequestRefused | sr in r.request implies (RefuseWarning

      in sr.authorOfRequest.warningsReceived)

}

--The third party has made a request that has not been accepted if it has been notified with

--the warning

```
fact RefuseWarningOnlyIfNeeded{

        all tp: ThirdParty | RefuseWarning in tp.warningsReceived implies (some sr: SingleRequest
        | tp in sr.authorOfRequest and (one r: SingleRequestRefused | sr in r.request))

}
```

--Every single request can be either accepted or refused, not both.

```
fact SingleRequestAcceptedOrRefused{

        all sr: SingleRequest | (some srr: SingleRequestResult | sr in srr.request) and (no disj
        srr1,srr2: SingleRequestResult | sr in srr1.request and sr in srr2.request)

}
```

--The third party is notified of the problem if the privacy cannot be guaranteed

--NB: scaled values for simplicity (the 5 should be 1000)

```
fact NotMoreThan1000MatchingUsers{

        all ar: AggregateRequest | #ar.matchingUsers < 5 implies (PrivacyWarning in
        ar.authorOfRequest.warningsReceived)

}
```

--The privacy cannot be guaranteed if the third party is notified

--NB: scaled values for simplicity (the 5 should be 1000)

```
fact PrivacyWarningOnlyIfNeeded{

        all tp: ThirdParty | PrivacyWarning in tp.warningsReceived implies (some ar:
        AggregateRequest | tp in ar.authorOfRequest and #ar.matchingUsers < 5)

}
```

```
--The emergency call is created if, and only if, the user needs it (his parameters are out of bounds)

fact EmergencyCallIfAndOnlyIfNeeded{

        all u: User | (

        (#u.emergencyCall = 1) iff (u.healthStatus.heartbeat <= 4 or u.healthStatus.heartbeat >= 6

        or u.healthStatus.bodyTemperature <= 3 or u.healthStatus.bodyTemperature >= 5)

        )

}



--All Health Status have to be associated to a User

--NB: two users CAN have the same health status

fact HealthStatusUserConnection {

        all hs: HealthStatus | some u: User | (u.healthStatus = hs)

}



--All present Emergency Calls have to be associated to a User (the reference is mutual) and to a
thirdParty

--NB: the emergency has to be received only from one third party

fact EmergencyCallConnection {

        all u: User, ec: EmergencyCall | (ec.userToAssist = u iff u.emergencyCall = ec)

        and (one tp: ThirdParty | ec in tp.receivedEmergencyCalls)

}



--All Paths have to be associated to a Run

fact PathRunConnection{

        all p: Path | some r: Run | p in r.path
```

```
}
```

--All Dates have to be associated to a Run

```
fact DateRunConnection{

        all d: Date | some r: Run | d in r.date

}
```

--There can't be two runs with the same name and with the same date

```
fact NoRunsWithSameNameAndDate{

        no disj r1,r2: Run | r1.name = r2.name and r1.date = r2.date

}
```

--All Locations have to be associated to a User or to a Run

```
fact LocationConnection{

        all loc: Location | (some u:User | loc in u.location) or (some r: Run | loc in r.startingPoint or
loc in r.endingPoint)

}
```

--All runs are organized by one and only one third party

```
fact RunThirdPartyConnection {

        all r: Run | one tp: ThirdParty | r in tp.organizedRuns

}
```

```
assert NoDifferentThirdPartiesReceivingTheSameEmergencyCall{

        no disj tp1, tp2: ThirdParty | some ec: EmergencyCall | ec in tp1.receivedEmergencyCalls
        and ec in tp2.receivedEmergencyCalls
```

```
}



assert AggregateRequestHasWarningCorrectly {

        all ar: AggregateRequest | #ar.matchingUsers >= 5 iff #ar.privacyWarning = 0

}



assert NoSingleRequestBothAcceptedAndRefused{

        no sr: SingleRequest | (one a: SingleRequestAccepted | sr in a.request) and (one r:

        SingleRequestRefused | sr in r.request)

}



check NoDifferentThirdPartiesReceivingTheSameEmergencyCall for 3



check AggregateRequestHasWarningCorrectly for 5



check NoSingleRequestBothAcceptedAndRefused for 5

----------------------------------------------------------------------------------------------------------------

pred world1 {

        #SingleRequest = 2

        #SingleRequestAccepted.request = 1

        #SingleRequestRefused.request = 1

        #ThirdParty = 2

        (some disj t1, t2: ThirdParty | some disj s1, s2: SingleRequest  | t1 in s1.authorOfRequest

        and t2 in s2.authorOfRequest and s1 in SingleRequestAccepted.request and s2 in

        SingleRequestRefused.request)

}
```

```
run world1 for 4 but 0 Run, 0 EmergencyCall, 0 AggregateRequest


pred world2{

        #AggregateRequest = 2

        #User = 5

        #ThirdParty = 2

        (some disj a1,a2: AggregateRequest | a1.authorOfRequest != a2.authorOfRequest and

        #a1.matchingUsers = 5 and  #a2.matchingUsers = 2)

}


run world2 for 7 but 0 Run, 0 EmergencyCall, 0 SingleRequest


pred world3 {

        #ThirdParty = 1

        #User = 1

        #EmergencyCall = 1

        (some u:User | some tp:ThirdParty | some ec:EmergencyCall | ec.userToAssist = u and

        u.emergencyCall = ec and ec in tp.receivedEmergencyCalls)

}


run world3 for 2 but 0 Run, 0 SingleRequest

pred world4{

         #Run = 2

}
```
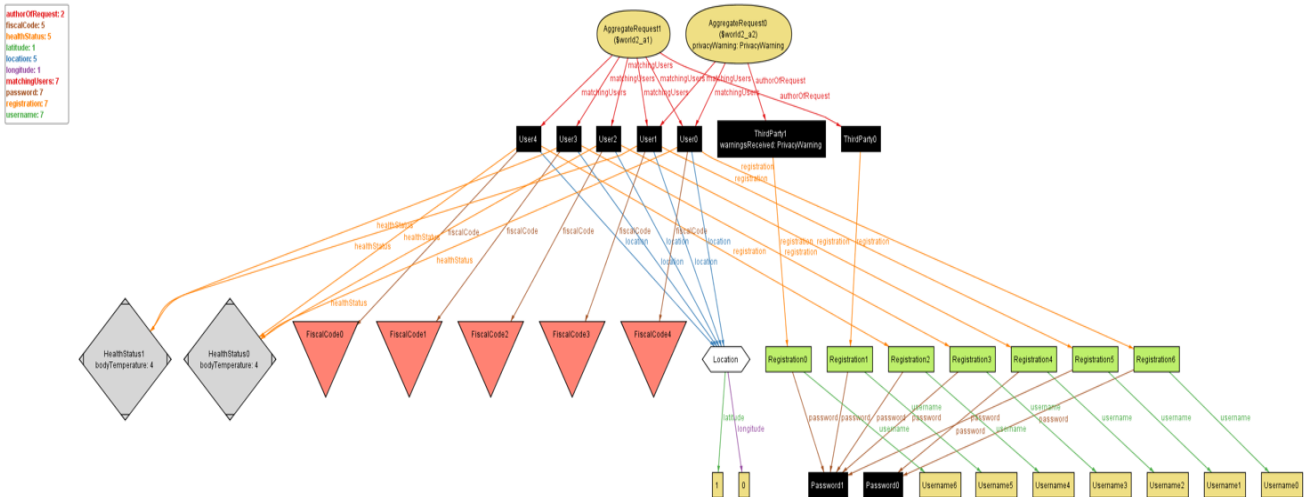
run world4 for 2 but 0 EmergencyCall, 0 SingleRequest, 0 AggregateRequest

Here there are the worlds generated through the execution of the three predicates:

From the first world in Figure 23 (and from the other worlds generated by the first 'run') it can be noticed that all usernames are actually different one from the other, that the third parties are notified with a refuse warning if, and only if, the individual requests have been refused (only 'Third Party 1' receives the warning) and that every individual request can't be both refused and accepted.

Figure 24 World 2
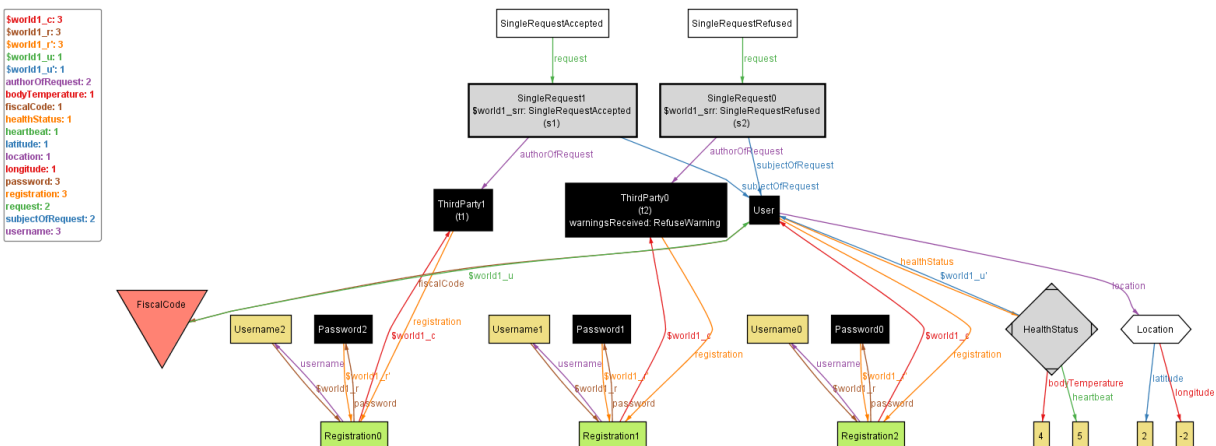
Looking at the second world in Figure 24 (and at the other worlds generated by the second 'run') we can see that also all fiscal codes are actually different one from the other and that the only third party notified with a privacy warning is 'ThirdParty1' that has made an aggregate request ('AggregateRequest0') with a number of matching users (only two) not sufficient to guarantee their privacy while 'ThirdParty0' is not notified since its aggregate request is satisfied by a sufficient number of users (five). Obviously the numerical values used in Alloy are different from the real world to allow a correct execution in the tool without any loss of general meaning.
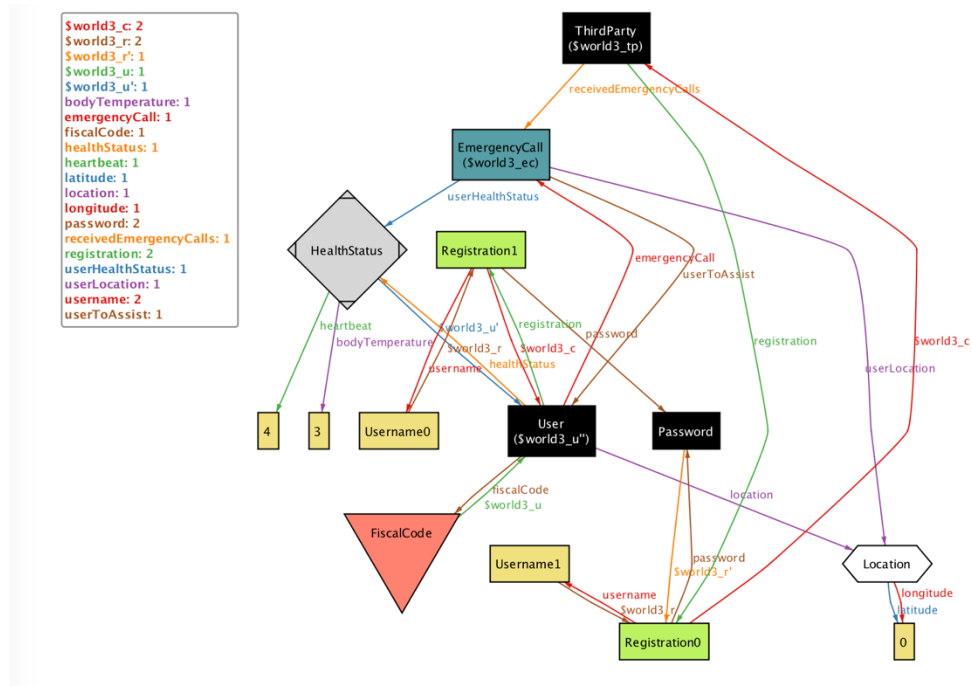

Figure 25 World 3

This world in Figure 25 (and the other worlds generated by the third 'run') shows that the presence of an emergency call for some user is always coupled with an effective health problem of that user (parameters out of bounds). In particular here 'User' has both its body temperature and its heartbeat (3 and 4) out of the defined thresholds. Once again the numerical values used are different from those of the real world to allow a correct execution in the tool without any loss of general meaning.
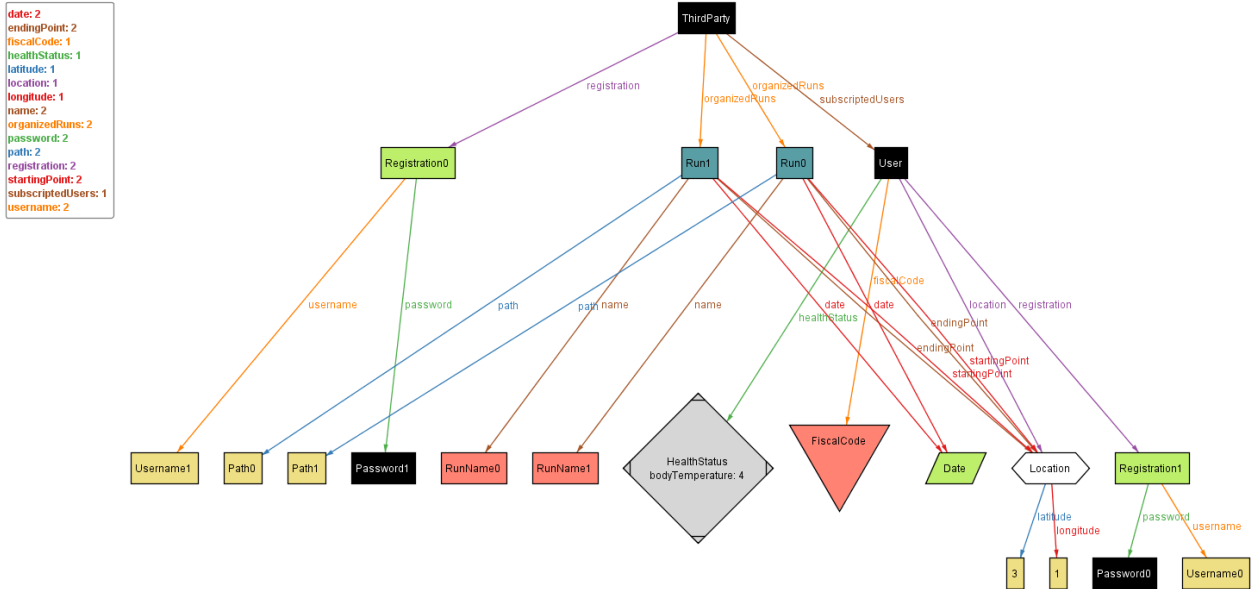
Figure 26 World 4

Finally this world (and the other worlds generated by the fourth 'run') only shows one trivial constraint about the runs: no competitions can be scheduled in the same date and with the same name. No relevant constraints have been modelled for the Track4Run service.

In the end, the results provided by the Alloy analyzer are showed below.

Executing "Check NoDifferentThirdPartiesReceivingTheSameEmergencyCall for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  7587 vars. 546 primary vars. 18687 clauses. 24ms.
  No counterexample found. Assertion may be valid. 5ms.

Executing "Check AggregateRequestHasWarningCorrectly for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  16840 vars. 1210 primary vars. 39408 clauses. 43ms.
  No counterexample found. Assertion may be valid. 13ms.

Executing "Check NoSingleRequestBothAcceptedAndRefused for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  16792 vars. 1210 primary vars. 39207 clauses. 41ms.
  No counterexample found. Assertion may be valid. 11ms.

Executing "Run world1 for 4 but 0 Run, 0 EmergencyCall, 0 AggregateRequest"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8837 vars. 568 primary vars. 22506 clauses. 25ms.
  Instance found. Predicate is consistent. 35ms.

Executing "Run world2 for 7 but 0 Run, 0 EmergencyCall, 0 SingleRequest"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  20567 vars. 1253 primary vars. 49953 clauses. 50ms.
  Instance found. Predicate is consistent. 85ms.

Executing "Run world3 for 2 but 0 Run, 0 SingleRequest"
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
  3663 vars. 242 primary vars. 10031 clauses. 10ms.
  Instance found. Predicate is consistent. 10ms.

Executing "Run world4 for 2 but 0 EmergencyCall, 0 SingleRequest, 0 AggregateRequest"
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
  3692 vars. 244 primary vars. 9788 clauses. 8ms.
  Instance found. Predicate is consistent. 13ms.

Figure 27 Alloy Analyzer results

5. Effort Spent

ST1

Description of the task	Hours
Introduction	4
Product perspective	0.5
Product functions	2.5
Domain assumptions	1
External interface requirements	1
Functional requirements	11
Non-functional requirements	1
Formal analysis using Alloy	23

ST2

Description of the task	Hours
Introduction	10
Product perspective	3.5
Product functions	1
Domain assumptions	4
External interface requirements	8
Functional requirements	10
Non-functional requirements	0.5
Formal analysis using Alloy	14

ST3

Description of the task	Hours
Introduction	6
Product perspective	6
Product functions	1
Domain assumptions	2
External interface requirements	2
Functional requirements	15
Non-functional requirements	1
Formal analysis using Alloy	8