

Politecnico di Milano



# RASD

## TrackMe Project

*2018*

*Version n°1*

*Software engineering 2*

### **ABSTRACT:**

The Requirements analysis and specification document (RASD) contains the descriptions of the scenarios, the use cases that describe them, and the models describing requirements and specification for the services Data4Help and AutomatedSOS offered by the company TrackMe

*Emma Bortone*

*Mohamed Gawish*

## Table of Contents

Table of Contents .....	1
I. Introduction.....	1
I.A. Purpose.....	1
I.B. Hypotheses on the assignment .....	2
I.C. Scope .....	3
I.D. Definitions, Acronyms and abbreviations .....	4
I.E. Revision history .....	5
I.F. Reference Documents .....	5
I.G. Document Structure .....	5
II. Data4Help : Overall description .....	6
II.A. Product perspective .....	6
II.B. Product functions .....	8
II.C. Users characteristics.....	9
II.D. Assumptions, dependencies and constraints.....	9
III. Data4Help : Specific requirements.....	10
III.A. External interfaces requirements.....	10
III.B. Requirements .....	16
III.C. Scenarios .....	18
III.D. Uses cases.....	19
III.E. Sequence diagrams .....	29
III.F. Performance requirements .....	32
III.G. Design constraints .....	33
III.H. Software system attributes .....	33
IV. AutomatedSOS : Overall description.....	35
IV.A. Product perspective .....	35
IV.B. Product functions .....	36
IV.C. User characteristics .....	36
IV.D. Domain constraints .....	36
V. AutomatedSOS : Specific requirements .....	37
V.A. External interfaces requirements.....	37
V.B. Functional requirements .....	39
I.B. Scenarios .....	39
V.C. Uses cases.....	40

V.D.	Performance requirements .....	42
V.E.	Design Constraints.....	42
V.F.	Software system attributes .....	43
VI.	Formal analysis using Alloy.....	43
VI. A	Data4Help .....	43
VI. B	AutomatedSOS .....	50
VII.	Effort spent.....	55
VII.A.	Common work.....	55
VII.B.	Mohamed.....	56
VII.C.	Emma .....	56
VIII.	References.....	57

# I. Introduction

## I.A. Purpose

The purpose of this document is to define the requirements analysis and specification document (RASD) of the services Data4Help and AutomatedSOS offered by the company TrackMe. This document contains the requirements of the system to be developed and its application domain. It can be used as a baseline for software evaluation and for charge control.

### I.A.1. Data4Help

Data4Help is a software-based service allowing third parties to monitor the location and health status of individuals. Data4Help supports the registration of individuals and of third parties.

- Individuals who register to Data4Help allow the company TrackMe to acquire their data.
- Third parties who register to Data4Help can access the data acquired by TrackMe by the mean of requests. They can request to :
  - Access to the data of a specific individual by providing a unique identifier. The request is then transferred to the individual who decide to accept or refuse it.
  - Access to anonymized data of groups of individuals. The request is accepted if the number of individuals satisfying it is higher than 1000.

When a request is accepted, third parties must be able to access the previously saved data and to subscribe to the request. By subscribing to a request, third parties will receive the new data corresponding to the request as soon as the data is available and without the need to renew the request process each time. Third parties can make an unlimited number of requests and subscriptions

The goals of the Data4Help service are :

- [G 1] : Third parties must be able to request to access to the data of specific individuals or to anonymized groups of individuals.
- [G 2] : At any time, third parties should never have access to data of specific individuals without their agreement.
- [G 3] : Third parties must have the possibility to subscribe to new data if their request is accepted.
- [G 4] : Individuals must be able to consult their data and accept/refuse requests

### I.A.1. AutomatedSOS

AutomatedSOS is a service build on top of Data4Help, therefore AutomatedSOS must verify all the requirements of the service Data4Help. In addition, AutomatedSOS offers the possibility to monitor the health of the subscribed individuals and to automatically send an ambulance to the location of the individuals if their health parameter are below certain thresholds.

All the goals of Data4Help are also goals of Automated SOS, but AutomatedSOS have the additional goal :

- [G 5] : An SOS is sent to an ambulance service, specifying the position of the individual, with a reaction time below 5 seconds from the time the individual's health parameters are below threshold.

## I.B. Hypotheses on the assignment

To solve the ambiguity and incompleteness of the project, we made some assumptions on the services Data4Help and AutomatedSOS. Those assumptions are listed in the two following paragraphs.

### I.B.1. Data4Help

- The data is collected by the mean of a smartwatch synchronized to a smartphone application. We made this choice because most of the smartwatches currently on the market are aimed to be linked to a smartphone through an application and we want our software to run on as many platforms as possible (design for portability)
- The subscription to a request on a group of individuals is automatically cancelled if the number of individuals whose data satisfy the request goes below 1000 at some point.
- The subscription to a request on a specific individual is automatically cancelled if the individual cancels his agreement.
- The individual can see the data collected by the service Data4Help. Indeed, the individual must earn something in exchange for the data he agrees to share. In this case, the individual gains the ability to control his health data.
- Data4Help must respect the General Data Protection Regulation (GDPR)
- Third parties can be companies, organizations or persons who need to acquire data (for example students or independent data scientists).
- The Data4Help service take the form of :
  - **A website and a Web API for data request.**  
We made this choice because using a website is the more convenient solution for a company or an organization who need to punctually acquire data and a web API is the most convenient solution for third parties who need to acquire data regularly (each hour for example)
  - **A smartphone application for data acquisition.**  
We made the choice to consider that the individuals have some piece of Data4Help installed on their personal device which allow Data4Help to acquire their health data. The motivation for the individual to install Data4Help on his smartphone will be to observe his own health data. Other solution would be to gathers data from some other service/system but this would mean to buy the data and the delay might be more important.

### I.B.2. AutomatedSOS

- The AutomatedSOS service is not an independent application. The individuals who want to subscribe to AutomatedSOS will first need to download the application Data4Help on their smartphone to register to the service. We made this choice because the AutomatedSOS service

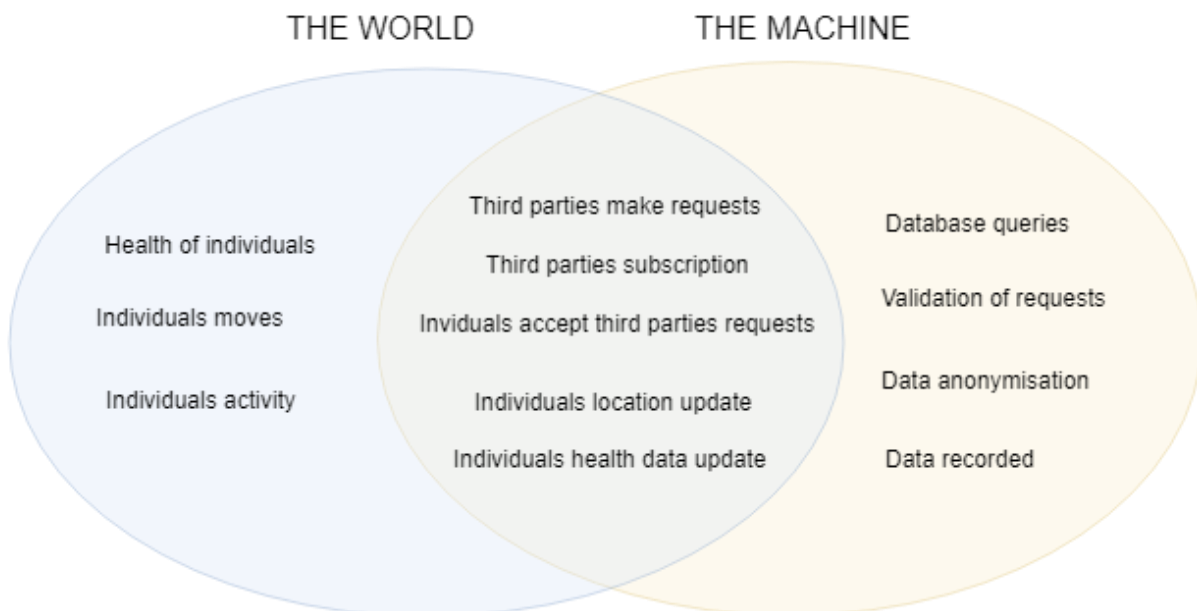
is only an extension of the Data4Help service so the requirements of the Data4Help service are also requirements of the AutomatedSOS service .

- AutomatedSOS notifies the ambulance service that an ambulance needs to be sent to a certain location by sending a SMS.
- Individuals who want to subscribe to AutomatedSOS are required to have a smartwatch and to link it to their Data4Help account. Indeed, a smartphone cannot acquire the pulse rate of the individual and this data is significant for detecting emergencies.
- The thresholds are automatically defined by AutomatedSOS depending on the individual's characteristics. We choose to prevent individuals to set themselves their thresholds to avoid unnecessary SOS or an urgent situation not being detected.
- Individuals also have the possibility to download an application on their smartwatch, in addition to the app on their smartphone, although this is not mandatory for the data acquisition. This will allow the individual to have an overview of their health data by simply looking at their watch. We made this choice because the individuals who subscribe to AutomatedSOS are elderly people who might not be comfortable with smartphones.

### I.C. Scope

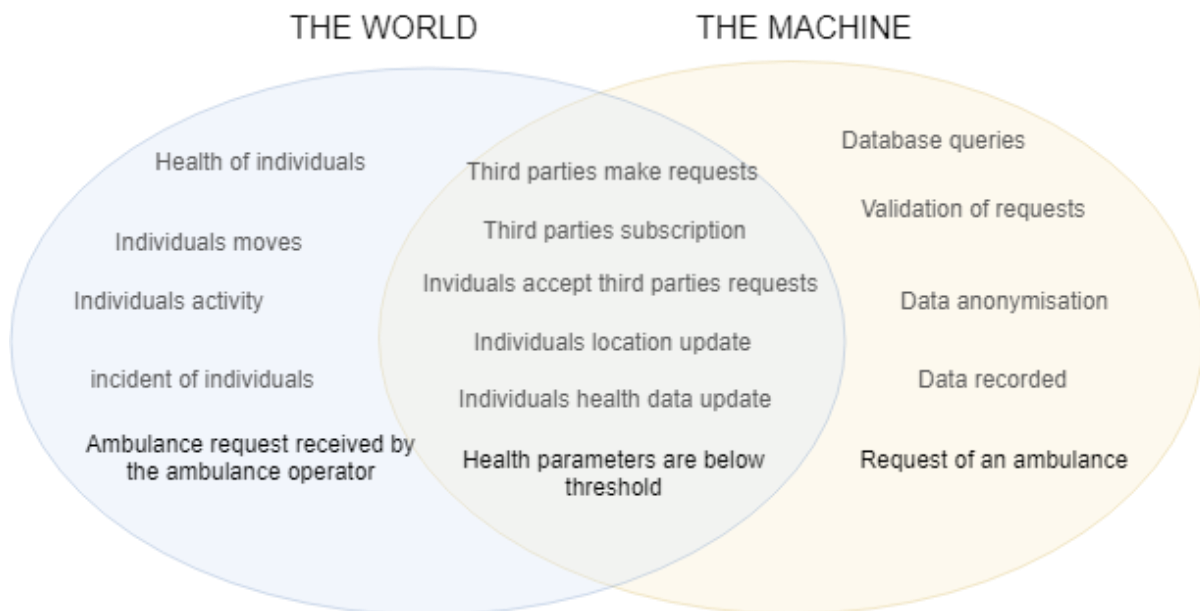
In this section, we will give a brief description of the world and of the shared phenomena. By “world” we intend the portion of the real world that is affected by the machine and by “machine” we mean the portion of system to be developed. A phenomenon shared by the world and the machine can either be controlled by the world and observed by the machine or controlled by the machine and observed by the world.

#### I.C.1. Data4Help



### I.C.2. AutomatedSOS

The world and the machine are mainly the same for the AutomatedSOS service as it is built on top of Data4Help. Indeed, AutomatedSOS must have all the functionalities of Data4Help in addition to the SOS service.



### I.D. Definitions, Acronyms and abbreviations

Third party: company or organization or person who need to access data of individuals

Individual: person willing to share his data with a third party and who want to monitor his health.

Smartwatch: device aimed to be worn on the wrist. A smartwatch can collect data, can be linked to a smartphone application and can support wireless technologies like Bluetooth, Wi-Fi, and GPS.

Subscription: an arrangement for automatically receiving data corresponding to a specific accepted request as soon as the data is available and without the need to renew the request process each time.

Request: the act, for a third party, of asking for data. A request can be individual or anonymized.

Individual request: A request that concerns a unique person known by his fiscal code.

Anonymized request: A request that concerns a group of individuals who fit some criteria. The result of this type of request is anonymized.

Anonymize: to remove any information that shows which particular person a data relates to.

SOS: SMS sent by Data4Help to an ambulance service asking to send an ambulance to a specific location. An SOS is sent when the health parameters of a monitored individual go below threshold.

### I.E. Revision history

<i>Version</i>	<i>Date</i>	<i>Description</i>
V1.1	14/10/2018	Creation of the document
V1.2	20/10/2018	Creation of parts I (Introduction) and II (Specific requirements)
V1.3	25/10/2018	Update of parts I and II + creation of parts III (Specific requirements)
V1.3	30/10/2018	Update of parts I and II + creation of parts III (Specific requirements)
V.1.final	8/11/2018	Update parts I, II and III. Verification coherence of the document.
V.2	02/12/2018	Adding non functional requirements that add been forgotten

### I.F. Reference Documents

[1] SOMMERVILLE, Iam. *Software engineering 9*. International edition.

[2] Assignment *Mandatory Project: goal schedule, and rules*.

[3] Les numériques, *COMPARATIF / Quelle montre connectée choisir ?*

<https://www.lesnumeriques.com/montre-connectee/comparatif-montres-connectees-a1781.html>

### I.G.Document Structure

This document contains the requirements analysis of two services: the Data4Help service and the AutomatedSOS service . The description of the two services is kept separated as the Data4Help service has to exists independently of the existence of the AutomatedSOS service . Indeed, the AutomatedSOS service is built after the Data4Help service and on top of it.

- The first section concerns the Data4Help service and contains three subsections:
  - **Overall description** which gives a general description of the software to be. This includes a domain model and the domain assumptions, the description of the most important requirements and the description of the user characteristics
  - **Specific requirements** which gives a more detailed description of the software. This chapter includes the description of the external interface requirements, the list of the functional and nonfunctional requirements and of the performance requirements. Furthermore, this chapter describes the design constraints and the software system attributes.
    - The **functional requirements** specify what the product or service must do. They are actions that the product or service must take, such as check, calculate, record, and retrieve.
    - The **non-functional requirements** demonstrate the properties that the product or service should have to do what it must do. These requirements are the characteristics or qualities that make the product or service attractive, or usable, or fast, or reliable. Most non-functional requirements are associated



with performance criteria and are usually those requirements that establish the product or service boundary.

- The **constraints** can be of two types: design constraints and project constraints. Design constraints are those pre-existing design decisions that mandate how the final product must look or how it must comply technologically. Project constraints cover the areas of budget and schedule along with deadlines.

- The second section is about the AutomatedSOS service and has the same structure as the previous one.
- This document contains also a section called **Formal analysis using Alloy** which includes the alloy model of the software and the discussion of its purpose
- Finally, this document contains two additional sections which describe the effort spent on the project and the list of reference documents.

## Data4Help

### II. Data4Help : Overall description

#### II.A. Product perspective

The Data4Help service must include a user-friendly application available on mainstream mobile devices (android, smartwatches) and a back-end server to stock anonymized data and manage requests. Furthermore, third party must be able to make requests and acquire data through a website and a web API.

Individuals use the Data4Help service through their smartphones, if they own a smartwatch, they must synchronize their smartwatch to their smartphone. The data transfer between the smartwatch and the smartphone is made by the Bluetooth technology.

A global overview of all the major components of the Data4Help service is given by the class diagram:

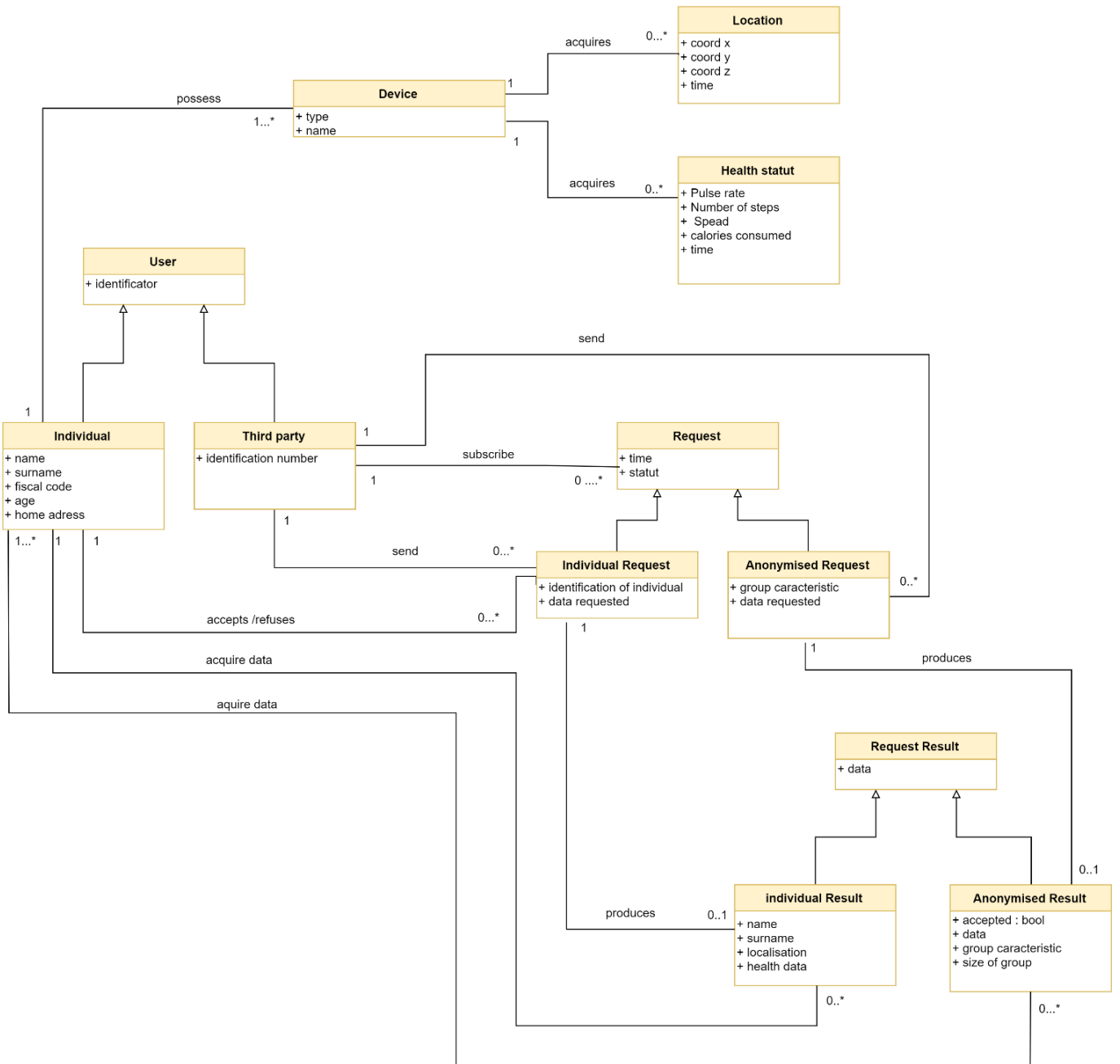


Figure 1: Class Diagram Data4Help

As we can see, the project is constituted by four main elements : The users, the requests, the results of those requests and the data acquired by the individual's personal devices.

The central element of the Data4Help service is the request management process. For this reason, we choose to provide a state diagram which focuses on changes occurring in the class Request :

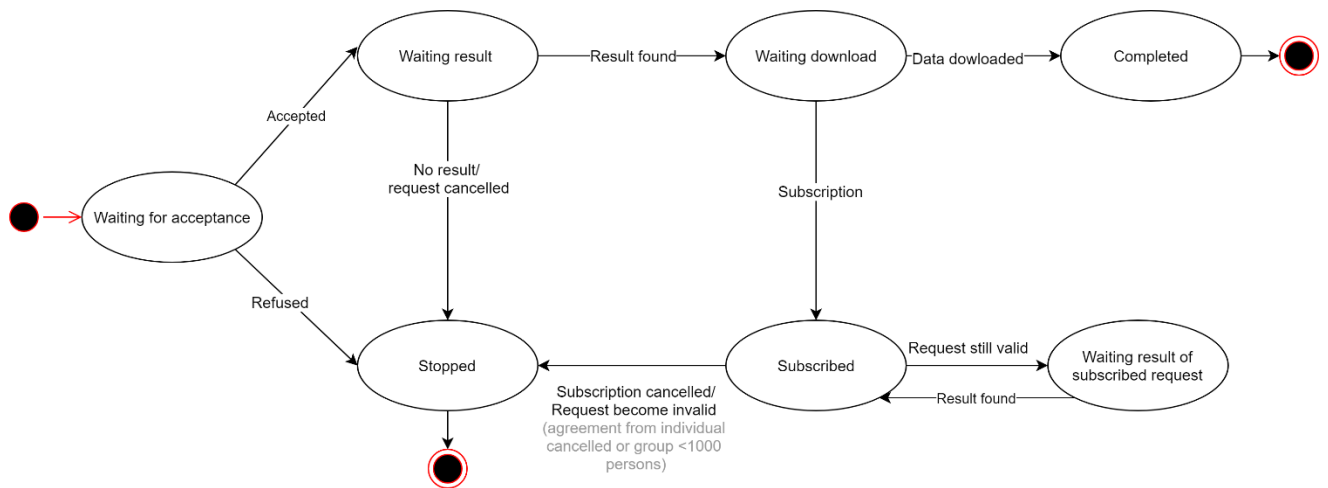


Figure 2: State Diagram : REQUEST

Once created, a request can either be stopped or accepted. In the case of an individual request, the request is stopped if the individual refuses it ; and in the case of a anonymized request, the request is stopped if the number of individuals is lower than 1000. Otherwise the request is accepted.

An accepted request can be stopped if it is cancelled or if no result is produced (for example, if the request concerns an individual who do not use the application anymore). If the request produces a result, the request enters the state “Waiting download”. A request waiting for download enters the final state “Completed” once the data has been downloaded. If the request is subscribed, then it enters in a cycle where the result of the request is continuously produced ; as long as the request stays valid.

## II.B. Product functions

In order to fulfill the goals, the main functions that the Data4Help service should offer to third parties are the possibility for third parties to make anonymized and individual requests and to subscribe to them. Moreover, the main functions that the Data4Help service should offer to individuals are the possibility to see their data and to accept or refuse requests. In this paragraph, those main product functions are more precisely specified from the perspective of the third party and of the individual

### Third party

The request management is the center of the service Data4Help. Indeed, the service has to allow third parties to make two types of requests: anonymized requests and individual requests. Moreover, the system must allow third parties to subscribe to the requests that have been accepted

### Individual

The system must provide an interface for individual that allows them to consult their health data. Furthermore, at any time, third parties should never have access to data of specific individuals without their agreement. This implies that for an individual request to be accepted, the individual concerned must give his agreement. The consequence is that the system must allow individuals to accept or refuse requests. Another consequence is that an anonymized request must be accepted by the system only if the number of individuals whose data satisfy the request is higher than 1000.

## II.C. Users characteristics

The users of the Data4Help service are Individuals, third parties and the manager of the service.

- **Manager:** The manager can access all functionalities of the system such as login, logout, accept or refuse the registration of a third party, add user, delete user.
- **Individual:** An individual is a person who download the app Data4Help on his smartphone and agrees that TrackMe acquires his data. An individual can access the functionalities: register, login, logout, consult his health data, accept and refuse requests from third parties
- **Third party:** A third party is an entity that desire acquiring data. It can be a company, an organization or a person (a student, a researcher...). A Third-party can access the functionalities: request for individual's data (on specific individuals or on anonymized groups of more than one thousand persons), register, login, logout and subscribe to an already accepted request.

## II.D. Assumptions, dependencies and constraints

### II.D.1. Domain assumptions

[D.1] Fiscal code is unique

[D.2] The data recorded by external devices (smartphone and smartwatches) are assumed to be correct.

[D.3] A smartwatch can detect it has been removed from the wrist which disable the data recording.

[D.4] For an account, there is only one individual.

### II.D.2. Constraints

[CON.1] The Price of the entire project must not exceed 85.000,00 €.

[CON.2] The system must be delivered before 12 October 2019.

[CON.3] With the delivery of the system, it must be also submitted a user manual that:

- Should be written that is understandable for all users.
- Should provide screenshots of the user interface and explain its various components.
- Should contain information about the system and how to operate the system.

## III.Data4Help : Specific requirements

### III.A. External interfaces requirements

#### III.A.1. User interfaces

The two end users are third parties and individuals. Both types of users have to register to Data4Help in order to be able to use the services.

In this section, the user interfaces are presented for both types of users. The phases presented are :

- For third parties :
  - Registration
  - Making request on specific individual or on anonymized groups
  - Subscribing
- For individuals :
  - Registration
  - Observation of own data
  - Accepting/refusing requests

Individuals will use Data4Help on their smartphones. Indeed, even if the smartwatch is the device used to acquire data but the small size of the screen makes it unsuitable to accept/refuse requests.

Third party will use Data4Help through a website and a web API.

##### *1.A.1.a. Registration phase*

The third party will register to Data4Help using a computer while individuals will register to Data4Help using their smartphone. The interface is the same whether the user uses a computer or a smartphone. So, only the smartphone version is presented here. For the computer version, the content is the same, only the dimension changes.

**Data4Help**

E-mail address

Password

Remember me ☐ **log in**

[Forgot my password](#)

You don't have an account yet ?

**Sign Up**

Progress bar and icons: ECG, Heart, Running person

Figure 3 : Sign up for both types of users

**Data4Help**

Your first steps with Data4Help

**Create your account**

email address

Name Surname

Choose a password Confirm password

\*\*\*\*\*

☐ Sign up for personal usage

Allows you to use Data4Help service to monitor your own health data

☐ Sign up for data request services

Allows you to use Data4Help service to acquire data from a specific individual or from an anonymised group of individuals

Progress bar and icons: ECG, Heart, Running person

Figure 4: Log in and Log up for both types of users

**Create your account**

email address

Name Surname

Choose a password Confirm password

\*\*\*\*\*

☒ Sign up for personal usage

Allows you to use Data4Help service to monitor your own health data

Place of birth Date of birth

Gender Weight

Fiscal code

City Postal code

House number Street name

+39 ...

[Terms and conditions](#)

☒ I agree

☐ Sign up for data request services

Allows you to use Data4Help service to acquire data from a specific individual or from an anonymised group of individuals

Progress bar and icons: ECG, Heart, Running person

Figure 5 : Sign up for individuals

☒ Sign up for data request services

Allows you to use Data4Help service to acquire data from a specific individual or from an anonymised group of individuals

**Select user type**

☐ Compagny

☐ Other

Progress bar and icons: ECG, Heart, Running person

Figure 6: Sign up for third parties

☒ Sign up for data request services

Allows you to use Data4Help service to acquire data from a specific individual or from an anonymised group of individuals

Select user type

☒ Compagny

Name

VAT Registration Number

City  Postal code

Building number  Street name

Expected usage of data

[Terms and conditions](#)

☒ I agree

Figure 7 : Sign up for third parties (companies)

☒ Sign up for data request services

Allows you to use Data4Help service to acquire data from a specific individual or from an anonymised group of individuals

Select user type

☐ Compagny

☒ Other

Fiscal code

City  Postal code

House number  Street name

You are

- Student
- researcher
- Journalist
- Other

[Terms and conditions](#)

☒ I agree

Figure 8 : Sign up for third parties (Others)

#### 1.A.1.b. Smartphone interface for individuals

Individuals will use their smartphone to keep track of their health data and to accept or refuse requests from third parties.

Through the smartphone application, individuals also have the possibility to subscribe to AutomatedSOS.

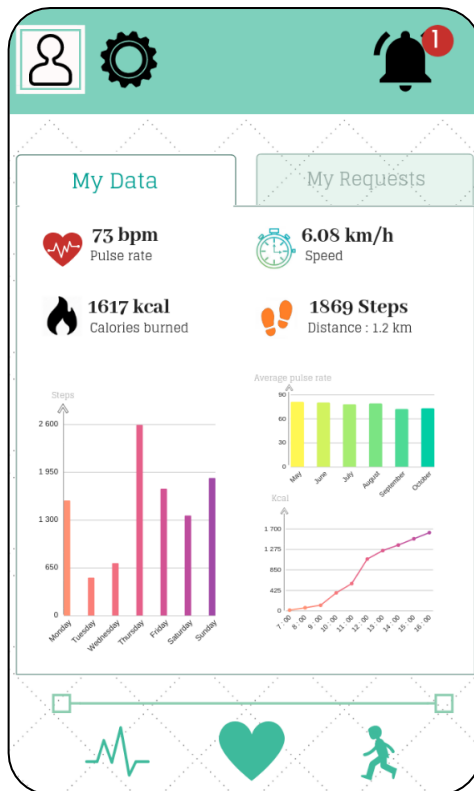


Figure 9 : Health data visualization for individuals

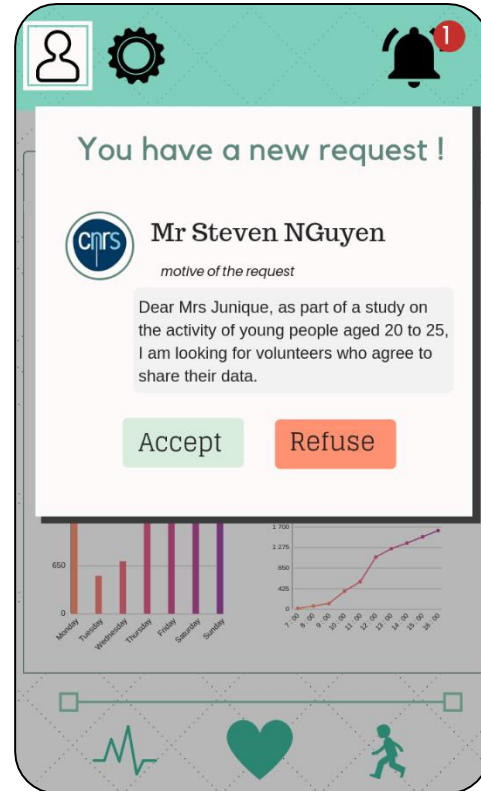


Figure 10 : Notification for new request

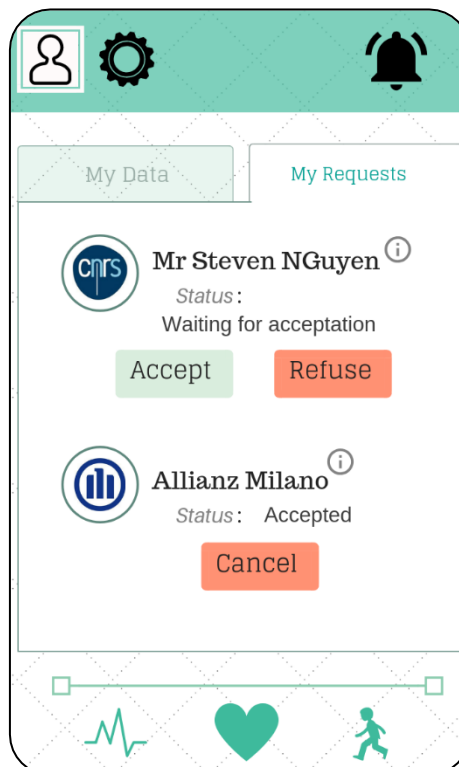
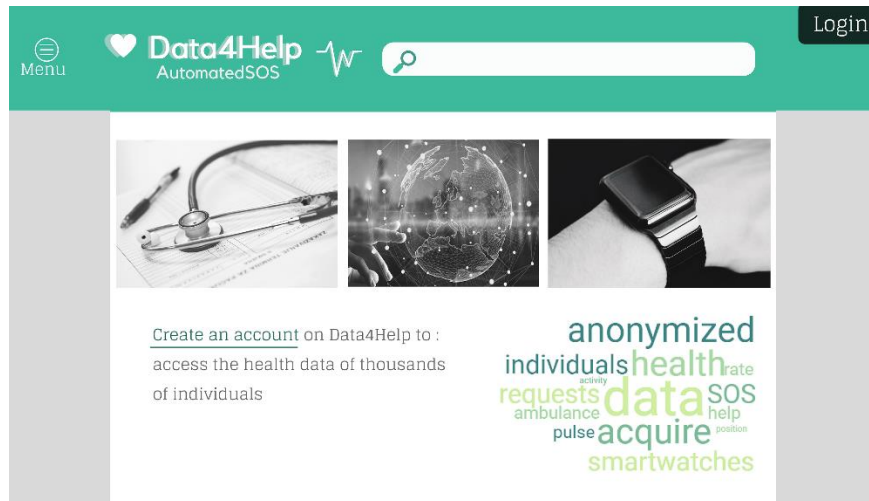


Figure 11 : Request management for individuals

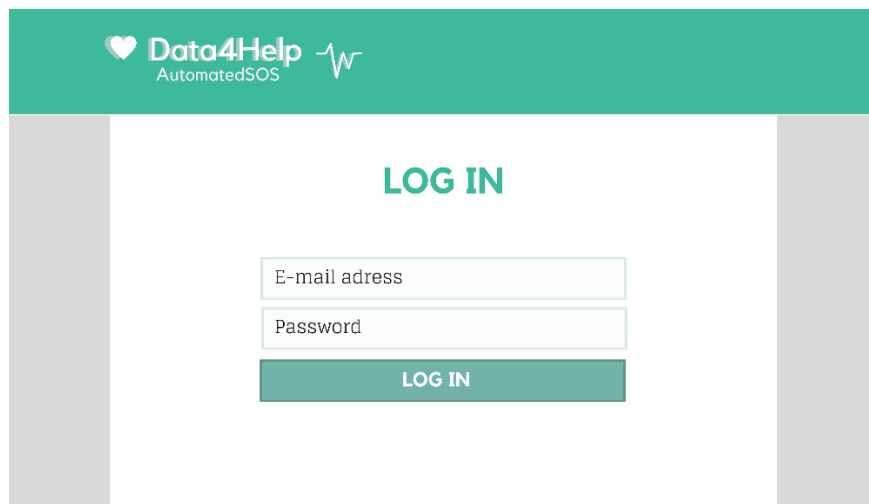


*I.A.1.c. Website interface for third parties*

The interface offered by the Data4Help service to third parties is a web site. Through this website, third parties will be able to register, log in, make new requests and subscribe to them. The registration phase is not presented again as the interface will be mainly the same as the registration phase on the smartphone application.



*Figure 12: Home page of Data4Help Website*



*Figure 13 : Data4Help website : Log in for third parties*




				
 Make a new request				
Current requests				
Identifier	Date	Type	Status	Result
001F5DF64	31/10/2018	Individual • MRLPLP80A01F205M	Accepted	<ul style="list-style-type: none"> <li>Download data</li> <li>Subscribe</li> <li>Cancel request</li> </ul>
95D8G6DSF	15/09/2018	Anonymised • 20 to 25 • women • Disco	Refused • Less than 1000 individuals	✗
13F8G6LOS	04/09/2018	Anonymised • 20 to 25 • Men • Milano	Accepted	<ul style="list-style-type: none"> <li>Download data</li> <li>Subscribe</li> <li>Cancel request</li> </ul>
156AZR5FG	04/09/2018	Individual • IMEBTN80A01F205G	Subscribed	<ul style="list-style-type: none"> <li>Download last data</li> <li>unsubscribe</li> </ul>

Figure 14 : Data4Help website : Request management for third parties



### New request

Select type :

☒ Specific individual
 


fiscal code of the individual

motive for the request

☐ Anonymised group
 

Submit request

Figure 15 : Data4Help : make a new request for a specific individual



### New request

Select type :

☐ Specific individual
 

☒ Anonymised group  
 Select the cells of the array corresponding to the desired request

AND

Age	0 - 25	25 - 50	50 - 75	>75
Genre	M	F		
City	Milano	Roma		
Weight (kg)	< 50	50 - 80	80-100	> 100

OR

Submit request

Figure 16 : Data4Halp : make a new request for a anonymised group

### III.A.2. Hardware interfaces

Data4Help is a software-based service application that do not use any hardware interfaces.

### III.A.3. Software interfaces

The Data4Help service will provide a web API that allow third parties to directly download data from accepted request using their request id and their password.

### III.A.4. Communication interfaces

Data4Help uses communication channels:

- Smartphone to smartwatch: Bluetooth connection.
- Smartphone to server: The smartphone application uses https request to receive and send data to the data server. The data server manages and stores all sensitive data. The application only accesses sensitive data using the https API.
- The Data4Help website also sent and request data from the https API.

## III.B. [Requirements](#)

### III.B.1. Functional requirements

In this section the requirements of the Data4Help service are described. They are classified according to the goals they achieve.

The first requirements [R. 1] to [R. 7] concern the ability for third parties and individuals to register, thus they are necessary for the achievement of all the goals.

- [R. 1]** The system must allow users to register as third parties.
- [R. 2]** The system must enforce the third parties to provide all necessary information when signup.
- [R. 3]** The system must allow users to register as individuals.
- [R. 4]** The system must enforce individuals to provide all necessary information when signup.
- [R. 5]** The system must acquire health data of individuals who register to the service
- [R. 6]** The system must be able to acquire health data from a smartwatch and to send it to a data server.
- [R. 7]** The system must be able to store sensitive data securely

- **[G 1] : Third parties must be able to request to access to the data of specific individuals or to anonymized groups of individuals.**

**[R. 8]** The system must allow third parties to make requests about specific individuals identified by their fiscal code.

**[R. 9]** The system must allow third parties to make requests about groups of individuals.

- **[G 2] : At any time, third party should never have access to data of specific individuals without their agreement.**

**[R. 10]** In the case of a request on a specific individual, the system must check for the individual agreement before making the data available for the third party.

[R.10.1] The system must send a notification to the specific individuals whose data are requested and make him choose between accepting or refusing the request

[R.10.2] If a third party made a subscription to a request and the individuals cancels his agreement to share his data, the system must cancel the subscription.

**[R. 11]** In the case of a request on a group of persons, the system must anonymize the data.

[R.11.1] The system must refuse any request on a group of persons for which the number of individuals whose data satisfy the request is lower than 1000.

[R.11.2] If a third party made a subscription to a request on a group of persons and at some point, the number of individuals whose data satisfy the request become lower than 1000, the system must cancel the subscription.

- **[G3] : Third parties must have the possibility to subscribe to new data if their request is accepted.**

**[R. 12]** The system must allow third parties to subscribe to the requests that have been accepted

[R.12.1] If a third party subscribed to a request, the system must make the data corresponding to the request available for the third party as soon as they are produced.

- **[G 4] : Individuals must be able to consult their data and accept/refuse requests**

**[R. 13]** The system must provide an interface for individual that allows them to consult their health data

### III.B.2. Nonfunctional requirements

- **Easy to use:** For the individuals application the app should be so simple and easy to understand and use. It is not necessary to know a lot about using an application, the font should be large and clear also using images as indication of the functions is preferable.
- **Customization:** As everyone is different, so everyone should be able to customize his/her application as they wish from changing the colours of the app or any other modification related to the graphical user interface.
- **Cloud backup:** The user should be able to back up the data on the cloud , in case of changing mobile phone using this backup , he/she can find all the data without any difficulty.
- **Help function:** As this app may be used by old people with a poor knowledge of using mobile apps , a help function should exist in every phase of the app starting from the registration to using the app , explaining every single detail .

### III.C. Scenarios

#### III.C.1. Create user Scenario

Luca opens the mobile application and click on Sign Up then write his email address. If Luca is already registered to the service, the application will give a warning that this email address is used before and that he can reset the password. If Luca is registering for the first time, the application requests him to choose a password which must be secure enough for the system (it must contain an upper-case character and at least one special symbol), also Luca must fill all the mandatory fields (name, surname, fiscal code, date of birth, gender, home address, phone number). Then Luca reads the terms and conditions and accepts them. Finally, he clicks on “create user” and he receive a message informing him that his account is created and that now he can use the application.

#### III.C.2. Third party request Scenario

The company HealthMilano is registered as a third party company and want to make a request .

HealthMilano can make 2 types of requests

- **Individual request**

HealthMilano requests information from Luca using his fiscal code, Luca can accept this request or not. If the request is accepted it will notify HealthMilano that Luca accepted the request, then HealthMilano will be able to use this data and to subscribe to the request. If Luca refused the request, HealthMilano will not be able to access Luca’s data. HealthMilano may make an error while entering the fiscal code of Luca this will give a warning message saying that this user doesn’t exist on the system.

- **Anonymised request**

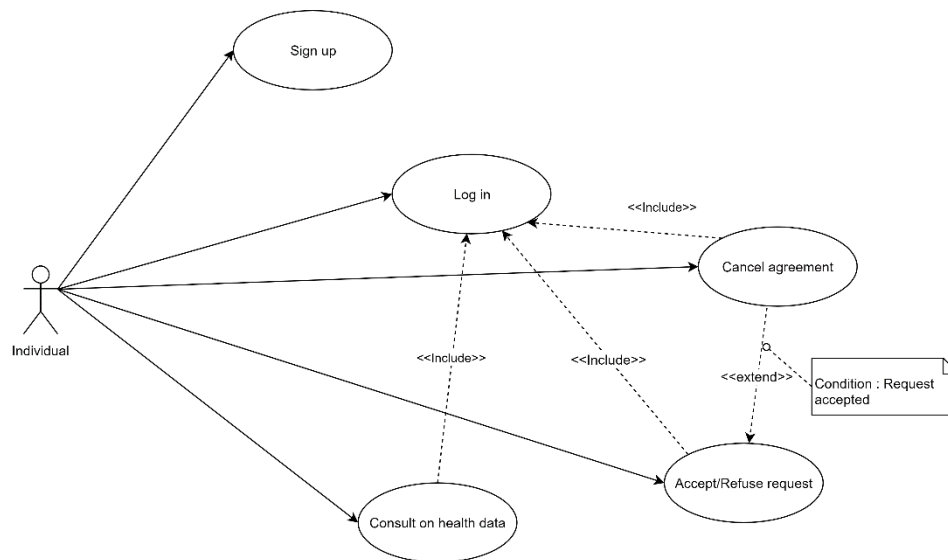
An anonymised request can be made for a specific attribute like(age , gender , city , weight) in this case HealthMilano requests the data from TrackMe. If there are 1000 or more individuals satisfying this attribute or a mixture between two or more attributes, TrackMe will accept the request and HealthMilano will have access to the data resulting from the request and will have the possibility to subscribe to the request. In the other case that the number of individuals whose data satisfy the request are less than 1000, the request is automatically refused.

#### III.C.3. Individual response Scenario

As specified before in the third party request scenario Luca will be able to accept or refuse the request, in case of acceptance he will agree that its data will be used for monitoring his health also his data may be used for other reasons from the third party (HealthMilano), taking in consideration that also the HealthMilano agree that these data will be secured and not used in an inappropriate way.

### III.D. Uses cases

In this section two use cases diagrams are presented, one for the individuals and one for the third parties.



#### LOG IN

ACTOR	Individual
ENTRY CONDITIONS	The individual is already registered to the Data4Help service and has the application installed on his/her device The system is ready
EVENTS FLOW	INDIVIDUAL STEPS
	SYSTEM STEPS
	1. The individual starts the app 2. The system starts the login phase 3. In the homepage of the app, the individual enters his E-mail address and his password 4. In the homepage of the app, the individual presses the "Log in" button
EXIT CONDITIONS	The Individual logs in the app.
EXCEPTIONS	- If the connection is lost, the app page is reloaded and the individual has to fill it again. - If the email is not in the database, the app page is reloaded with an error message. - If the password does not correspond to the email in the database, the app page is reloaded with an error message.

### SIGN UP

ACTOR	Individual	
ENTRY CONDITIONS	The individual has installed the application on his/her device	
EVENTS FLOW	INDIVIDUAL STEPS	SYSTEM STEPS
	1. The individual presses the “Sign up” button on the home page of the app	2. The system starts the registration phase
	3. The individual writes his E-mail address, his name, his surname and his password twice.	
	4. The individual presses the button “Sign up for personal usage”	
	5. The individual fill all the mandatory fields and provide the necessary information	
	6. The individual reads the terms and conditions and accepts it.	
		7. The system saves the data
EXIT CONDITIONS	The Individual logs in the app.	
EXCEPTIONS	<ul style="list-style-type: none"><li>- If the connection is lost, the app page is reloaded and the individual has to fill it again.</li><li>- If the E-mail address is already in the database, the app page is reloaded with an error message</li></ul>	

### CONSULT HEALTH DATA

ACTOR	Individual	
ENTRY CONDITIONS	The Individual is logged in the app	
EVENTS FLOW	INDIVIDUAL STEPS	SYSTEM STEPS
	1. The individual presses the “My data” tab on the app	2. The system update the last data
EXIT CONDITIONS	The individual can visualize his data	
EXCEPTIONS	- if the system cannot acquire data from the external device (the smartwatch), the new data is not printed on the screen and the individual receive an error notification.	

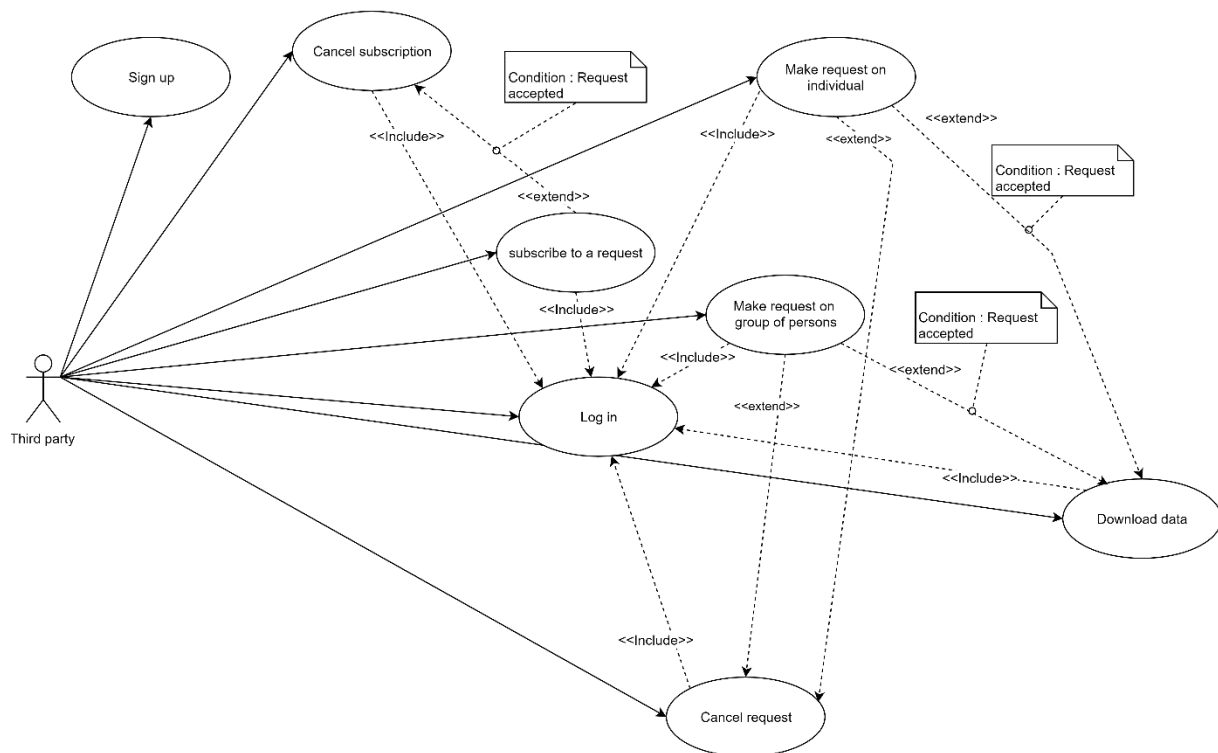
### ACCEPT/REFUSE REQUEST

ACTOR	Individual														
ENTRY CONDITIONS	The Individual is logged in the app and has received a request from a third party.														
EVENTS FLOW	<table border="1"> <thead> <tr> <th>INDIVIDUAL STEPS</th><th>SYSTEM STEPS</th></tr> </thead> <tbody> <tr> <td>1. The individual presses the “My requests” tab on the app</td><td></td></tr> <tr> <td>2. The individual consults the requests he has received</td><td></td></tr> <tr> <td>3. The individual presses the button “Accept” or “Refuse”</td><td></td></tr> <tr> <td></td><td>4. The system saves the answer of the individual</td></tr> <tr> <td></td><td>5. The system updates the status of the request</td></tr> <tr> <td></td><td>6. The system inform the third party of the new status of the request</td></tr> </tbody> </table>	INDIVIDUAL STEPS	SYSTEM STEPS	1. The individual presses the “My requests” tab on the app		2. The individual consults the requests he has received		3. The individual presses the button “Accept” or “Refuse”			4. The system saves the answer of the individual		5. The system updates the status of the request		6. The system inform the third party of the new status of the request
INDIVIDUAL STEPS	SYSTEM STEPS														
1. The individual presses the “My requests” tab on the app															
2. The individual consults the requests he has received															
3. The individual presses the button “Accept” or “Refuse”															
	4. The system saves the answer of the individual														
	5. The system updates the status of the request														
	6. The system inform the third party of the new status of the request														
EXIT CONDITIONS	The status of the request is updated														
EXCEPTIONS	- If the local application cannot send the individual’s answer to the server then the local application sends an error message to the individual asking him to answer the request another time.														



## CANCEL AGREEMENT

ACTOR	Individual														
ENTRY CONDITIONS	The Individual is logged in the app and has accepted a request from a third party														
EVENTS FLOW	<table border="1"> <thead> <tr> <th>INDIVIDUALS STEPS</th><th>SYSTEM STEPS</th></tr> </thead> <tbody> <tr> <td>1. The individual presses the “My requests” tab on the app</td><td></td></tr> <tr> <td>2. The individual consults the requests he has accepted</td><td></td></tr> <tr> <td>3. The individual presses the button “cancels”</td><td></td></tr> <tr> <td></td><td>4. The system saves the modification</td></tr> <tr> <td></td><td>5. The system updates the status of the request</td></tr> <tr> <td></td><td>6. The system inform the third party of the new status of the request</td></tr> </tbody> </table>	INDIVIDUALS STEPS	SYSTEM STEPS	1. The individual presses the “My requests” tab on the app		2. The individual consults the requests he has accepted		3. The individual presses the button “cancels”			4. The system saves the modification		5. The system updates the status of the request		6. The system inform the third party of the new status of the request
INDIVIDUALS STEPS	SYSTEM STEPS														
1. The individual presses the “My requests” tab on the app															
2. The individual consults the requests he has accepted															
3. The individual presses the button “cancels”															
	4. The system saves the modification														
	5. The system updates the status of the request														
	6. The system inform the third party of the new status of the request														
EXIT CONDITIONS	The status of the request is updated														
EXCEPTIONS	- If the local application cannot send the update to the server then the local application sends an error message to the individual asking him to repeat the process another time.														



### LOG IN

ACTOR	Third party	
ENTRY CONDITIONS	The third party is already registered to the Data4Help service	
EVENTS FLOW	THIRD PARTY STEPS	SYSTEM STEPS
	1. The third party opens the Data4Help website  2. the third party clicks on the “log in” button  3. In the login page, the third party enters his E-mail address and his password  4. In the login page, the third party presses the “log in” button	3. The system starts the login phase
EXIT CONDITIONS	The third party is logged in the website	
EXCEPTIONS	- If the connection is lost, the website page is reloaded and the third party has to fill it again. - If the email is not in the database, the page is reloaded with an error message. - If the password does not correspond to the email in the database, the page is reloaded with an error message.	

## SIGN UP

ACTOR	Third party
ENTRY CONDITIONS	The third party has opened the Data4Help webpage on his browser
EVENTS FLOW	<div>THIRD PARTY STEPS</div> <div>SYSTEM STEPS</div> <div> <p>1. The third party clicks on the button “create an account” in the website home page</p> <p>3. The third party writes his E-mail address, his name, his surname and his password twice.</p> <p>4. The third party clicks the button “Sign up for data request services”</p> <p>5. The third party specify his user type (company or other)</p> <p>6. The third party fill all the mandatory fields and provide the necessary information</p> <p>7. The third party reads the terms and conditions and accepts it.</p> </div> <div>3. The system starts the registration phase</div>
	6. The system saves the data
EXIT CONDITIONS	The third party logs in the app.
EXCEPTIONS	<p>- If the connection is lost, the page is reloaded and the third party has to fill it again.</p> <p>- If the E-mail address is already in the database, the page is reloaded with an error message.</p>

### MAKE REQUEST ON INDIVIDUAL

ACTOR	Third party	
ENTRY CONDITIONS	The third party is logged in the web site	
EVENTS FLOW	THIRD PARTY STEPS	SYSTEM STEPS
	1. The third party clicks on the button "Make a new request"	
		2. The system displays the "New request" window
	3. The third party selects the type "specific individual" by clicking on the corresponding button	
	4. The third party specify the fiscal code of the individual and the motive of the request	
		5. The system add the new request to the request history with the status "waiting for acceptance"
EXIT CONDITIONS		6. The system send the request to the individual
	The request is added to the request history with the status "waiting for acceptance"	
EXCEPTIONS	-If the fiscal code specified by the third party is not valid or if the field has not been filled the page is reloaded with an error message.	

### MAKE REQUEST ON GROUP OF PERSONS

ACTOR	Third party										
ENTRY CONDITIONS	The third party is logged in the web site. The third party is on the request history page										
EVENTS FLOW	<table> <tr> <th>THIRD PARTY STEPS</th><th>SYSTEM STEPS</th></tr> <tr> <td>1. The third party clicks on the button "Make a new request"</td><td>2. The system displays the "New request" window</td></tr> <tr> <td>3. The third party selects the type "anonymized group" by clicking on the corresponding button</td><td></td></tr> <tr> <td>4. The third party build his request selecting among the various criterion available.</td><td>5. The system verify that the number of people whose data satisfy the request is higher than 1000</td></tr> <tr> <td></td><td>6. The system add the new request to the request history with the status "Accepted"</td></tr> </table>	THIRD PARTY STEPS	SYSTEM STEPS	1. The third party clicks on the button "Make a new request"	2. The system displays the "New request" window	3. The third party selects the type "anonymized group" by clicking on the corresponding button		4. The third party build his request selecting among the various criterion available.	5. The system verify that the number of people whose data satisfy the request is higher than 1000		6. The system add the new request to the request history with the status "Accepted"
THIRD PARTY STEPS	SYSTEM STEPS										
1. The third party clicks on the button "Make a new request"	2. The system displays the "New request" window										
3. The third party selects the type "anonymized group" by clicking on the corresponding button											
4. The third party build his request selecting among the various criterion available.	5. The system verify that the number of people whose data satisfy the request is higher than 1000										
	6. The system add the new request to the request history with the status "Accepted"										
EXIT CONDITIONS	The new request is present in the request history with the status "Accepted – waiting for result"										
EXCEPTIONS	- If the number of people whose data satisfy the request is lower than 1000, the request is refused. In this case the system adds the new request to the request history with the status "Refused: less than 1000 individuals".										

### SUBSCRIBE TO A REQUEST

ACTOR	Third party				
ENTRY CONDITIONS	The third party is logged in the web site and at least one of his requests has been accepted. The third party is on the request history page				
EVENTS FLOW	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">THIRD PARTY STEPS</th><th style="width: 50%;">SYSTEM STEPS</th></tr> </thead> <tbody> <tr> <td>1. The third party search in his request history for the request to which he wants to subscribe and clicks on “subscribe”</td><td>2. The system updates the status of the request</td></tr> </tbody> </table>	THIRD PARTY STEPS	SYSTEM STEPS	1. The third party search in his request history for the request to which he wants to subscribe and clicks on “subscribe”	2. The system updates the status of the request
THIRD PARTY STEPS	SYSTEM STEPS				
1. The third party search in his request history for the request to which he wants to subscribe and clicks on “subscribe”	2. The system updates the status of the request				
EXIT CONDITIONS	The third party is subscribed to the request.				
EXCEPTIONS	<p>-If the third party want to subscribe to a request about an individual and the individual does not use the Data4Help service anymore, the system does not update the status of the request and print an error message.</p> <p>- If the third party want to subscribe to a request about an individual and the individual has cancelled his agreement, the system does not update the status of the request and print an error message.</p> <p>- If the third party want to subscribe to a request on a group of persons and the number of people whose data satisfy the request is lower than 1000, the system does not update the status of the request and print an error message.</p>				

### CANCEL SUBSCRIPTION

ACTOR	Third party				
ENTRY CONDITIONS	The third party is logged in the web site and has at least one subscription. The third party is on the request history page				
EVENTS FLOW	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">THIRD PARTY STEPS</th><th style="width: 50%;">SYSTEM STEPS</th></tr> </thead> <tbody> <tr> <td>1. The third party search in his request history for the request to which he wants to unsubscribe and clicks on “unsubscribe”</td><td>2. The system update the status of the request</td></tr> </tbody> </table>	THIRD PARTY STEPS	SYSTEM STEPS	1. The third party search in his request history for the request to which he wants to unsubscribe and clicks on “unsubscribe”	2. The system update the status of the request
THIRD PARTY STEPS	SYSTEM STEPS				
1. The third party search in his request history for the request to which he wants to unsubscribe and clicks on “unsubscribe”	2. The system update the status of the request				
EXIT CONDITIONS	The third party is not subscribed to the request.				
EXCEPTIONS					

### CANCEL REQUEST

ACTOR	Third party				
ENTRY CONDITIONS	The third party is logged in the web site. The third party is on the request history page				
EVENTS FLOW	<table><tr><th>THIRD PARTY STEPS</th><th>SYSTEM STEPS</th></tr><tr><td>1. The third party search in his request history for the request to which he wants to cancel and clicks on "cancel request"</td><td>2. The system update the status of the request</td></tr></table>	THIRD PARTY STEPS	SYSTEM STEPS	1. The third party search in his request history for the request to which he wants to cancel and clicks on "cancel request"	2. The system update the status of the request
THIRD PARTY STEPS	SYSTEM STEPS				
1. The third party search in his request history for the request to which he wants to cancel and clicks on "cancel request"	2. The system update the status of the request				
EXIT CONDITIONS	The status of the request is updated				
EXCEPTIONS					

### DOWNLOAD DATA

ACTOR	Third party				
ENTRY CONDITIONS	The third party is logged in the web site and has at least one accepted request. The third party is on the request history page				
EVENTS FLOW	<table><tr><th>THIRD PARTY STEPS</th><th>SYSTEM STEPS</th></tr><tr><td>1. The third party search in his request history for the request to which he wants to download data and clicks on "download data"</td><td>2. The system send the data file to the user's browser</td></tr></table>	THIRD PARTY STEPS	SYSTEM STEPS	1. The third party search in his request history for the request to which he wants to download data and clicks on "download data"	2. The system send the data file to the user's browser
THIRD PARTY STEPS	SYSTEM STEPS				
1. The third party search in his request history for the request to which he wants to download data and clicks on "download data"	2. The system send the data file to the user's browser				
EXIT CONDITIONS	The third party has received the data				
EXCEPTIONS	If the user interrupts the download, the webpage prints an error message with a button retry.				

### III.E. Sequence diagrams

This section contains the sequence diagrams description of four of the most important functions of the Data4Help service which are making individual and anonymized requests and subscribing to them.

- Make an individual Request

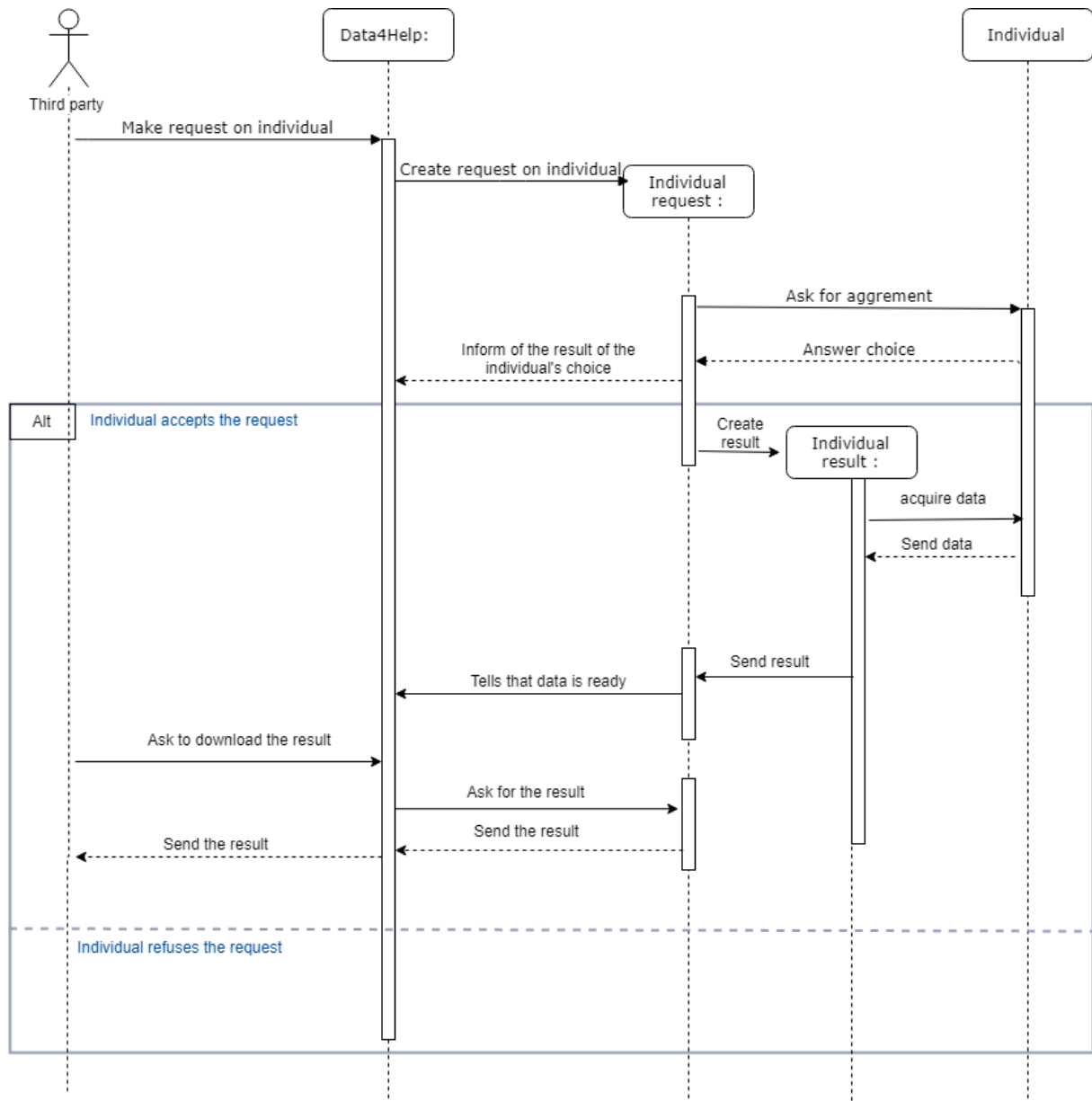


Figure 17 : sequence diagram : make an individual request



- Make an anonymized request

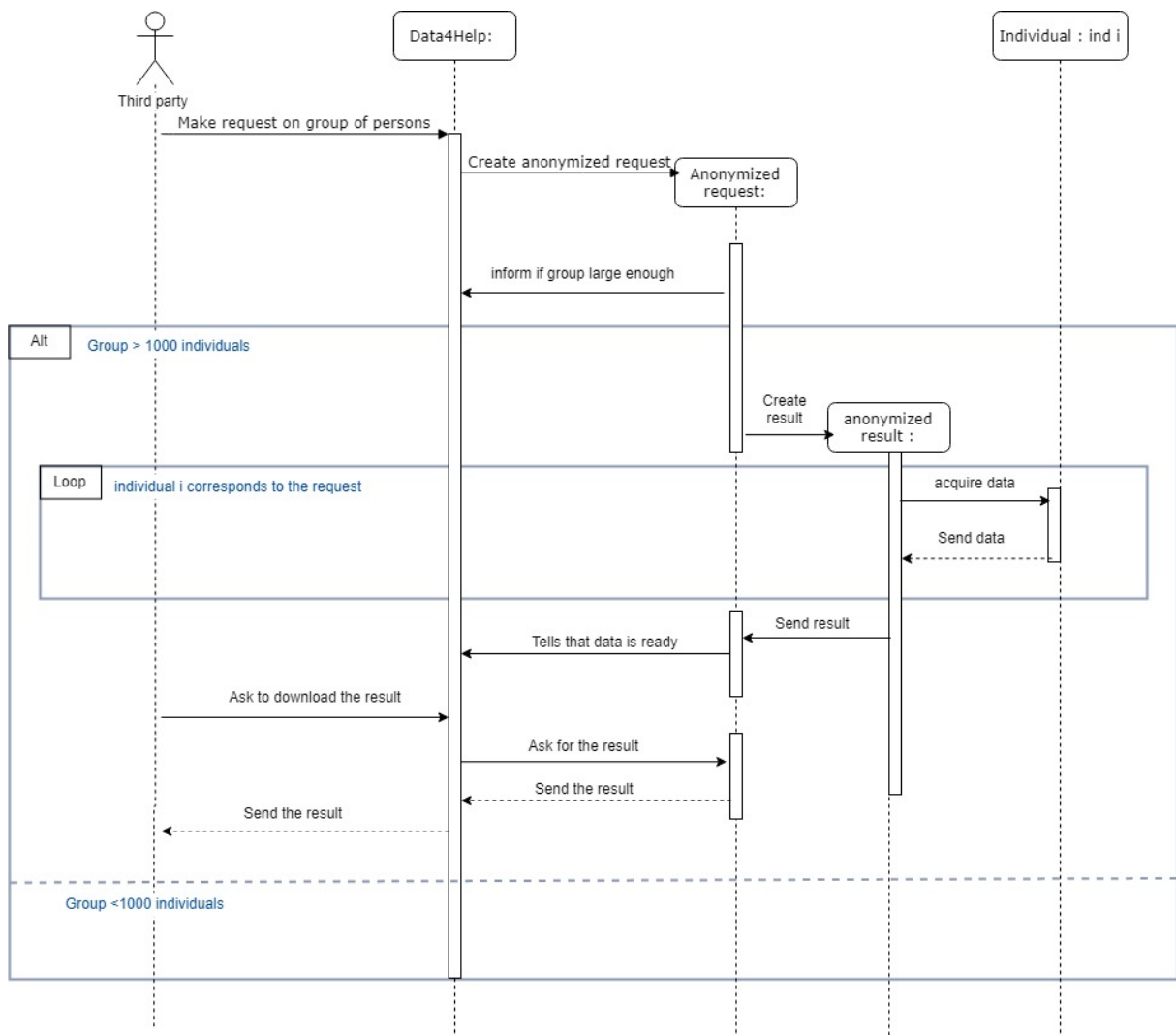


Figure 18 Sequence diagram request anonymized group

- Subscribe to an anonymized request

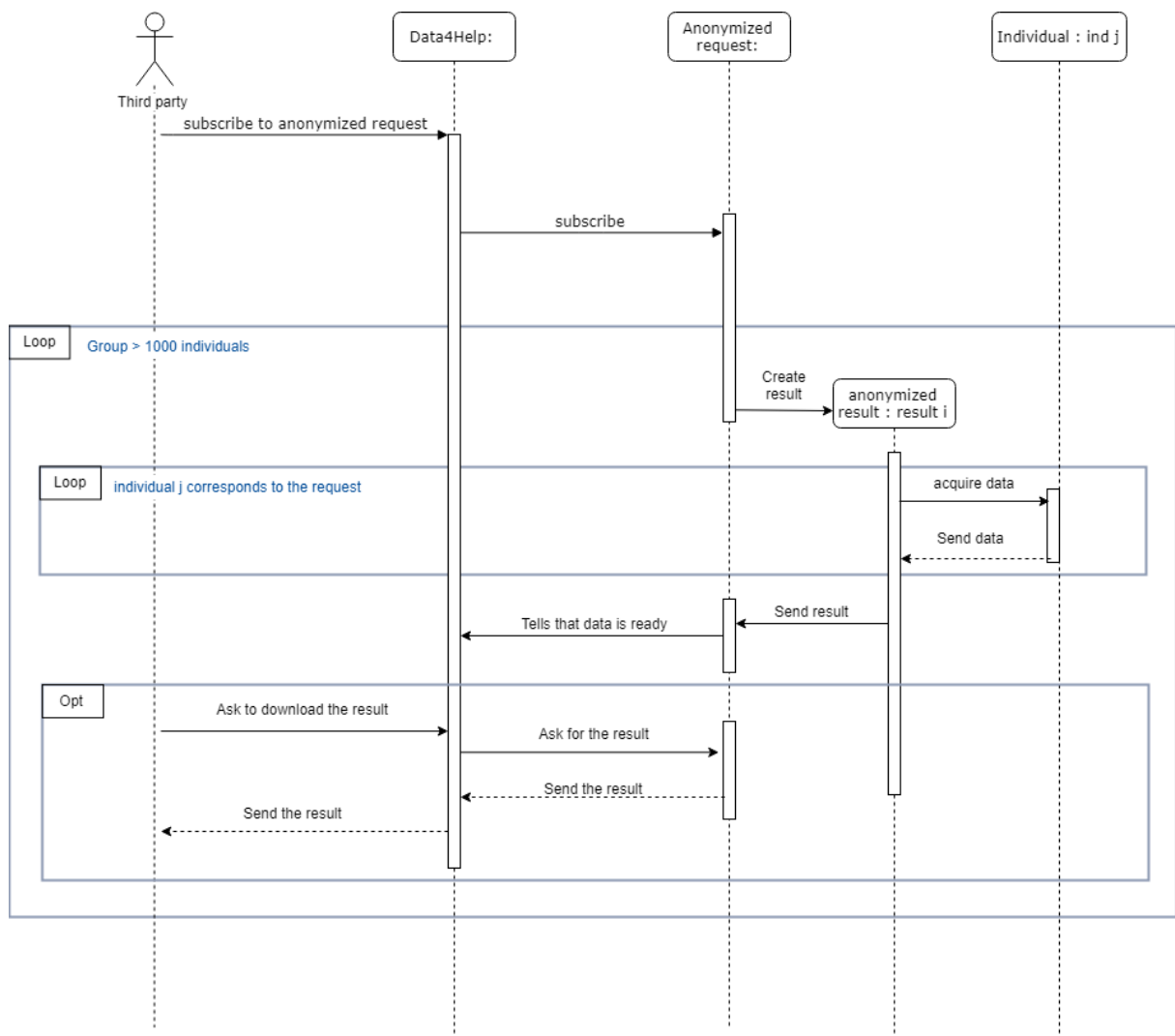


Figure 19 Sequence diagram subscription to anonymized request

- Subscribe to an individual request

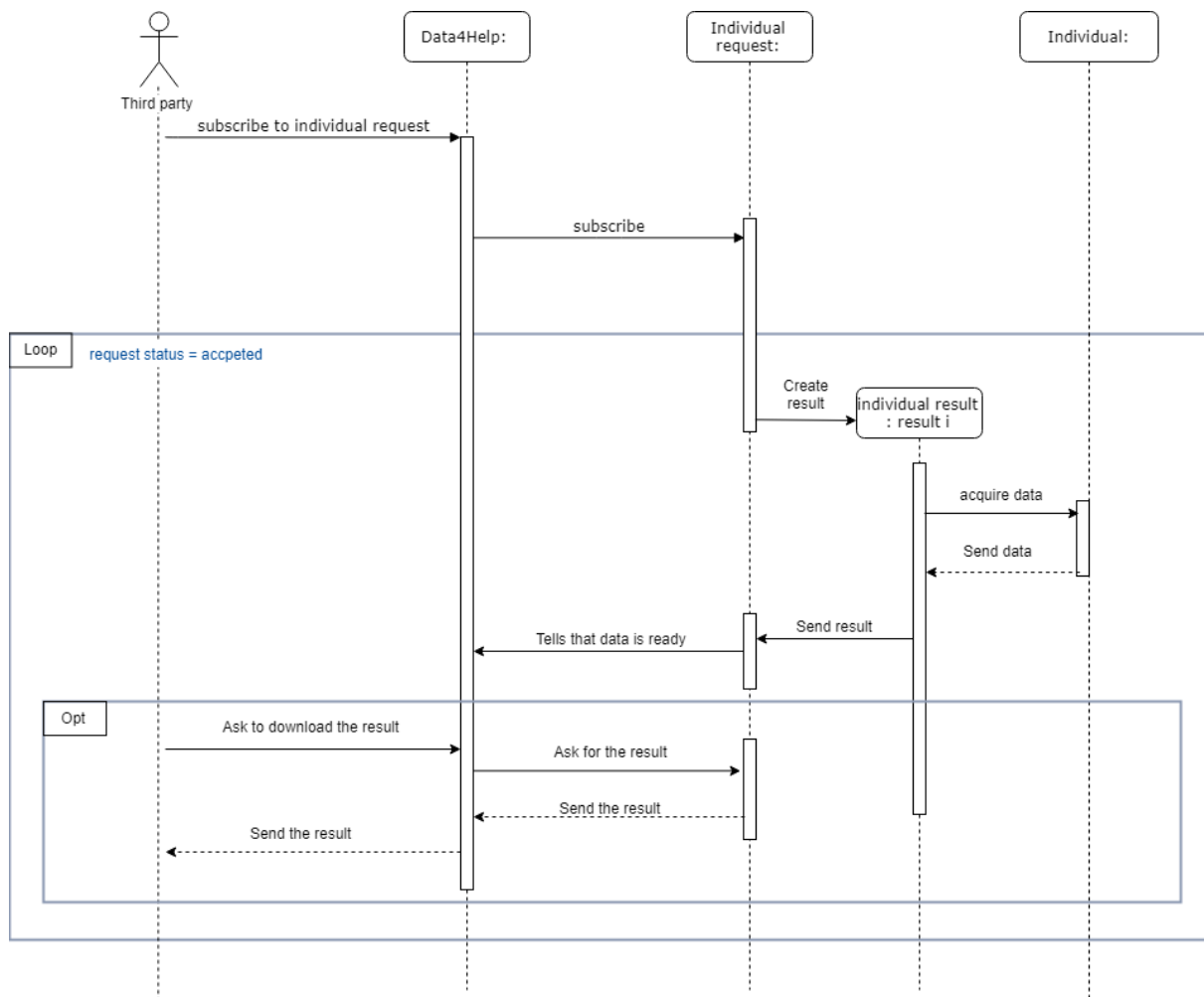


Figure 20 sequence diagram subscription to individual request

### III.F. Performance requirements

#### III.F.1. Performance Characteristics

- From the moment new data is received, all the actions that the system must perform must have a total duration strictly under five seconds.

The quality of performance depends on:

- System should automatically update after every transaction.
- More than five attempts at login and failure will produce a red flag to system administrator.
- Data should be secured and backed up every quarter hour.
- Power supply should have a backup and a disaster recovery plan.

### III.F.2. Error Handling and Extreme Conditions

In a real-life situation, the system input is influenced by the environment in which it operates. The following requirements are specified to provide a certain level of robustness when the system is dealing with errors and invalid input.

- The system must be able to recognize abnormal input.
- The system must activate the alarm when third parties' companies are using data in an inappropriate way.

### III.G. Design constraints

#### III.G.1. Standards compliance

The Data4Help service offers a web interface for the third parties. To ensure interoperability, the web site must be compliant with the web standards of the World Wide Web Consortium (W3C)

#### III.G.2. Hardware limitations

The computer that will run the system has to meet the following requirements:

- Memory: minimum 4 GB RAM
- Processor: minimum 1.8 GHz Intel or an equivalent

Also there will be a mobile application for the users and in order to run this application these are the minimum hardware limitations in order to use the application correctly, also these are the minimum limitation for the smart watch if it will be used and connected to the mobile phone.

- Memory: 2 GB RAM or higher
- Pedometer sensor
- Heart rate and oxygen sensor
- Global positioning system (GPS)

### III.H. Software system attributes

#### III.H.1. Reliability

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment, and this attribute is very important for this software as it is related to the health of people. In some cases, more than one request will be sent to be sure that these requests reached the correct destination and they started taking an action.

#### III.H.2. Availability

Availability is defined as the probability that the system is operating properly when it is requested for use. As described before in the Reliability section (III.H.1), our system will be able to monitor health conditions. So, our system will be available 24 hours 7 days a week.

#### III.H.3. Security

Access to Data4Help is restricted by passwords with the support of User Manager. You can define the roles to control any third party. Besides, sensitive information can be viewed, printed or changed only by using an admin password.

#### III.H.4. Maintainability

Maintainability incorporates such concepts as changeability, modularity, understandability, testability, and reusability. As the system may need a lot of modification through the time because of the changing in the people life and attitudes, all our software is commented in a very simple way that can be understood easily, also a good documentation is delivered explaining every function and its job clearly.

#### III.H.5. Portability

This version of application will work on android phones, but the third party can use the web site from any device, in a next version we will implement a version for apple products .

# AutomatedSOS

#### IV. AutomatedSOS : Overall description

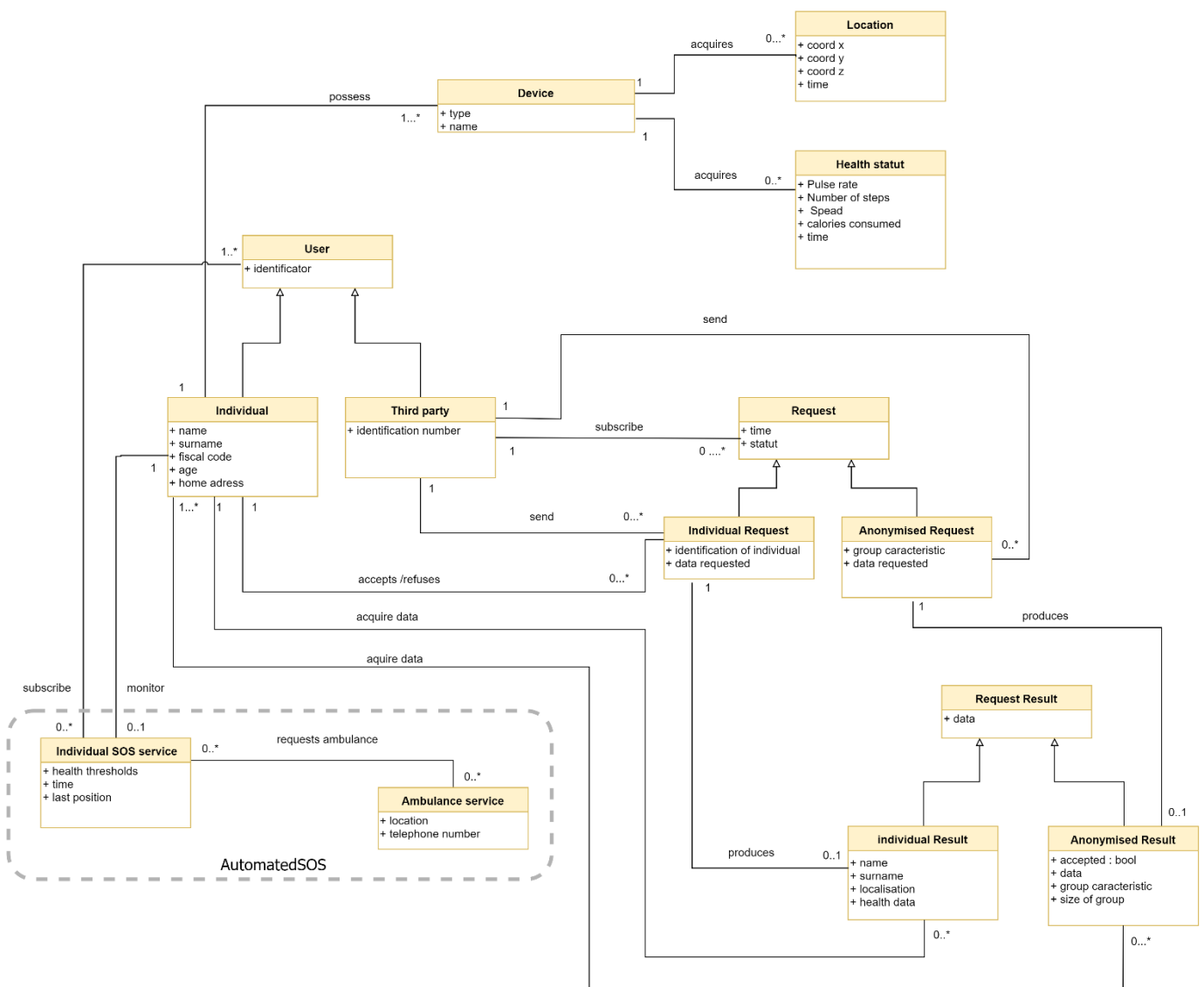
#### IV.A. Product perspective

The AutomatedSOS service is built on top of Data4Help, therefore all the functionality present in Data4Help are also present in the AutomatedSOS service .

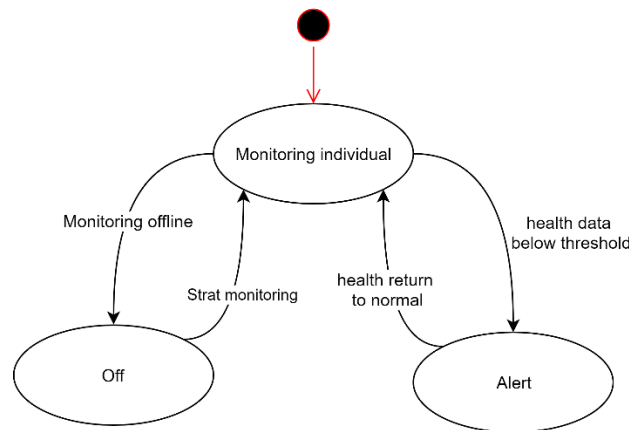
An individual who subscribed to Data4Help can then subscribe to AutomatedSOS. Once he subscribes to AutomatedSOS, he is not required to do any specific action as his thresholds will be automatically defined depending on his characteristics (age, weight, etc..)

To satisfy the goals of the AutomatedSOS service, two additional class are needed: Individual SOS service and ambulance service.

A global overview of the major component of the AutomatedSOS service is given by the class diagram



The following state diagram focuses on changes occurring in the class Individual SOS service :



When an individual subscribes to AutomatedSOS, the monitoring starts. He the individual removes his smartwatch; the data acquisition is deactivated and the monitoring stops. The monitoring will automatically start again when the individual puts his watch back. If the health parameters of the individual go below threshold, the Individual SOS service enters in Alert mode and an SOS is sent.

#### IV.B. [Product functions](#)

The main function that AutomatedSOS service should offer to individuals is the possibility to send an automatic request for an ambulance in the emergency cases and this is defined based on some data collected using a smart watch that is connected to the mobile phone application, in case of extreme changing in health condition for example(low or high pulse rate) a request to an ambulance is sent automatically containing the information of the patient and the position to be easy to the ambulance to reach the place, so the main function of this service is to offer immediate automatic help to people who have particular health conditions , also normal people can request this service , but the main targeted people are aged people and people with particular health conditions.

#### IV.C. [User characteristics](#)

The AutomatedSOS service is intended for elderly people who may have major health problems requiring emergency response. They can access the functionalities: register, login, logout, consult their health data, be notified when an SOS is sent.

#### IV.D. [Domain constraints](#)

- [D.1] The individuals registering to AutomatedSOS own a Smartwatch.
- [D.2] The smartwatch can detect if it is not on the wrist
- [D.3] If the Smartwatch is removed, the data acquisition automatically stops.
- [D.3] The ambulance services can receive SOS sent through SMS.
- [D.4] A data base referencing all the ambulance services is available

## V. AutomatedSOS : Specific requirements

### V.A. External interfaces requirements

#### V.A.1. User interfaces

The AutomatedSOS service is built on top of Data4Help, therefore the interface is mainly the same. But, as the individuals using AutomatedSOS might not be used to smartphones because of their old age, a special smartwatch application has been created for the AutomatedSOS service. The installation of this application is not mandatory for the data acquisition (synchronizing the smartwatch to be phone by Bluetooth is enough) but it can ease the use of the service for elderly people.

On the smartphone application When an individual subscribes to AutomatedSOS, this add a new window to the app : “My thresholds” where the individual can consult the thresholds below which the SOS will be sent. The rest of the app is unchanged.

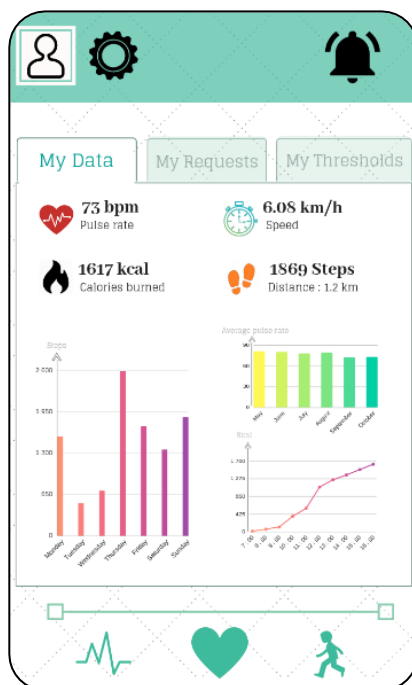


Figure 21 : AutomatedSOS for individuals (data visualization)

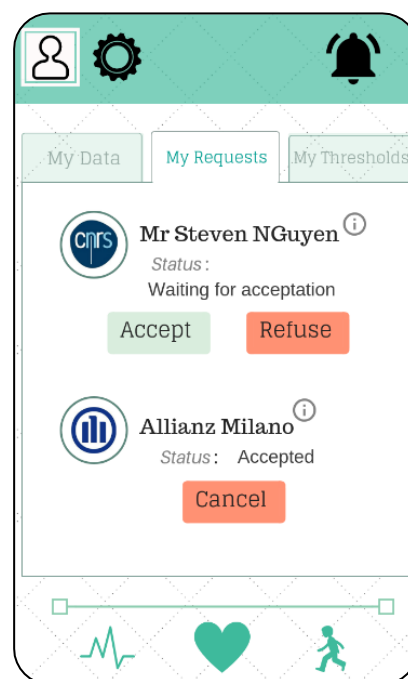


Figure 22 : AutomatedSOS for individuals (Requests)



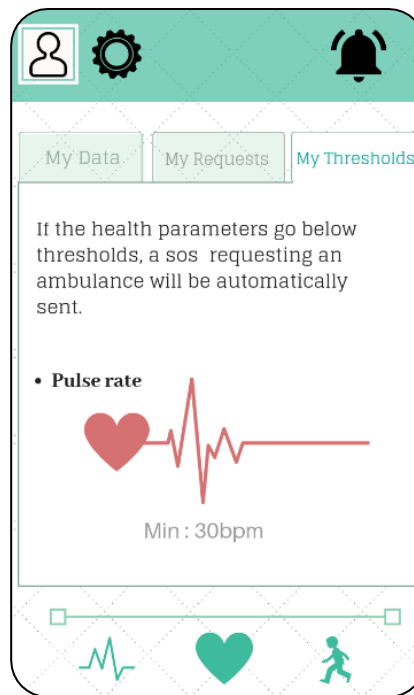


Figure 23 : AutomatedSOS for individuals (my thresholds)

Individuals who subscribe to AutomatedSOS have the possibility to also download an application on their smartwatch. This is not mandatory for the data acquisition and for the sending for the SOS. This application allows the user to see his health status and to be notified instantly when an SOS is sent.



Figure 24 : Smartwatch normal health status



Figure 25 : Smartwatch : health status below thresholds

#### V.A.2. Hardware interfaces

AutomatedSOS is a software-based service application that do not use any hardware interfaces.

#### V.A.3. Software interfaces

The software interfaces offered by AutomatedSOS are the one offered by Data4Help because automated SOS is built on top of Data4Help

#### V.A.4. Communication interfaces

AutomatedSOS uses communication channels:

- Smartphone to smartwatch: Bluetooth connection.
- Smartphone to server: The smartphone application uses https request to receive and send data to the data server. The data server manages and stores all sensitive data. The application only accesses sensitive data using the https API.
- Smartphone to ambulance service: the smartphone application sends a SMS message to the ambulance services.

#### V.B. Functional requirements

All the functional requirements listed in the section Data4Help are also requirements of AutomatedSOS. In this section, only the requirements specific to the AutomatedSOS service are listed.

**[R. 14]** The Data4Help service must provide an interface for allowing individuals to register to the new AutomatedSOS service .

**[R. 15]** The AutomatedSOS service must allow subscribers to cancel their subscription

- **[G 5] : An ambulance is requested to the location of the individual with a reaction time below 5 seconds from the time the parameters are below threshold**

**[R. 16]** The system must detect when the health data of subscribers are below threshold

**[R. 17]** If the health data of a subscribed individual is below thresholds, the system must send an SOS to the ambulance service indicating the position, the phone number and the health parameters of the individual.

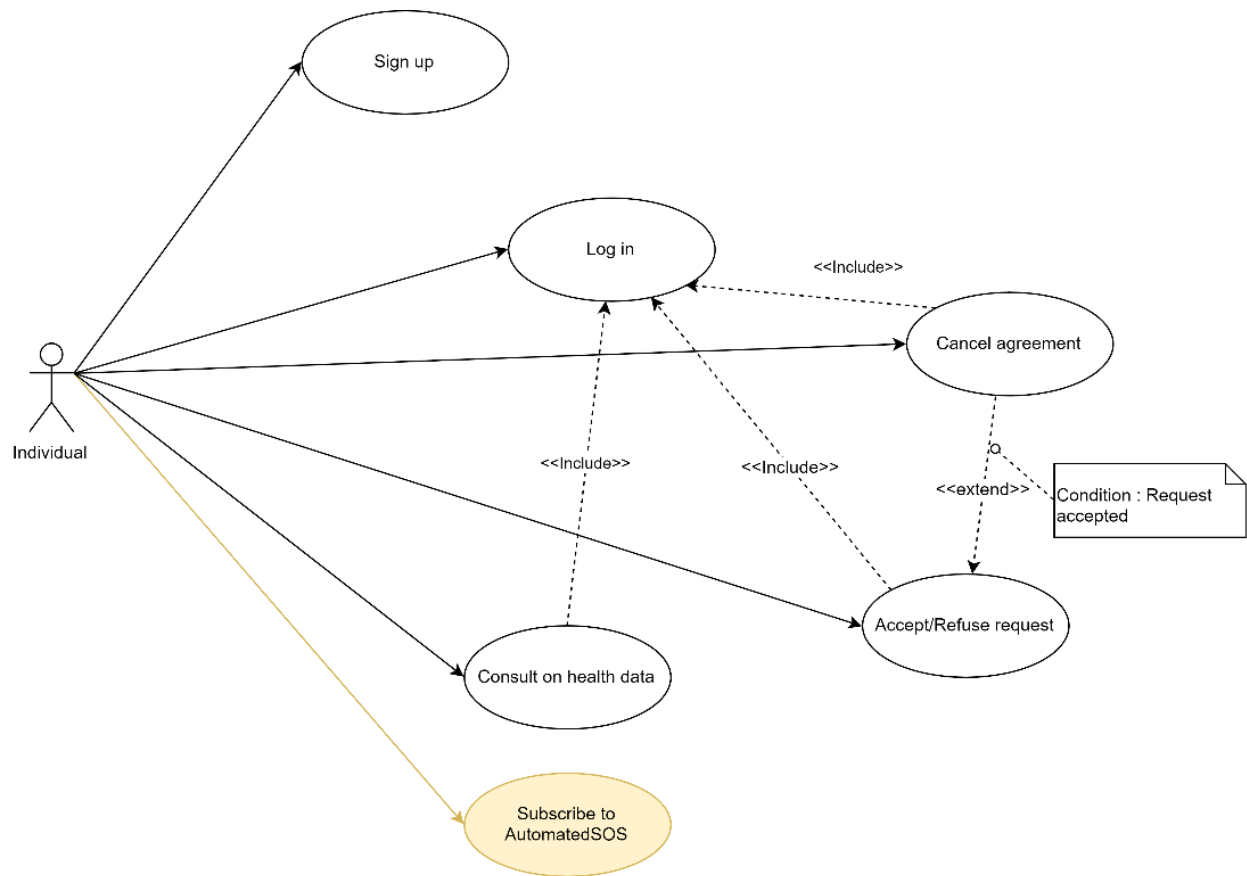
#### I.B. Scenarios

- SOS Scenario

Anna is 72 years old and she subscribed to Automated SOS service, in this case she will have to use a specific device (Smart Watch) and a smartphone, in case of emergency for example (very low heart rate pulse) that will be measured with the smart watch, an automatic request for the ambulance will be sent containing the data of the Anna (phone number, position, age ) in base of this info the ambulance can make a call to Anna to make sure that it is an emergency, if yes the ambulance will start moving to the position and take her to the hospital if needed .

## V.C. Uses cases

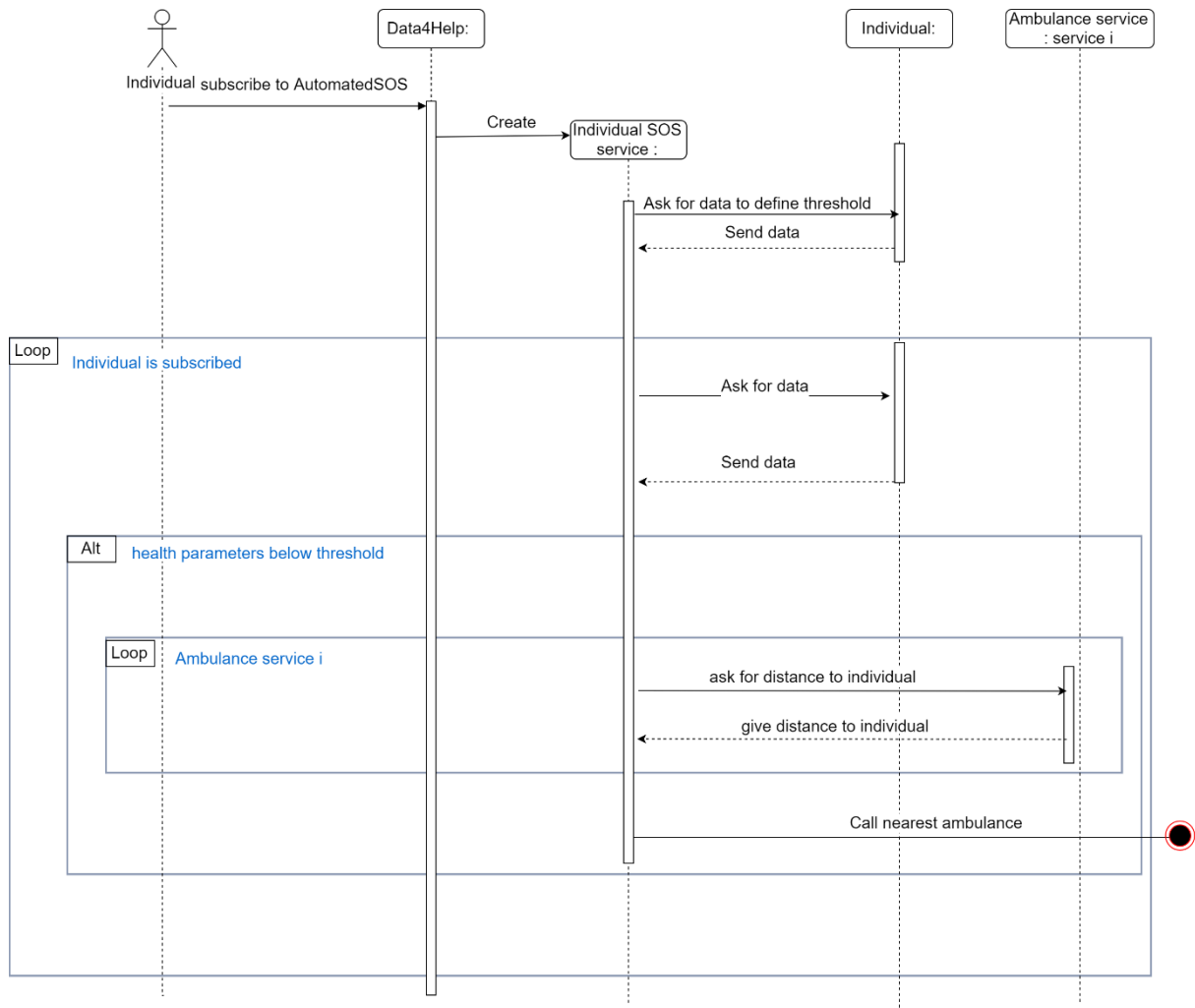
An individual who subscribe to Data4Help can also subscribe to AutomatedSOS.



### SUBSCRIBE TO AUTOMATEDSOS

ACTOR	Individual										
ENTRY CONDITIONS	<p>The individual is already registered to the Data4Help service and has the application installed on his/her device</p> <p>The system is ready</p>										
EVENTS FLOW	<table border="1"> <thead> <tr> <th>INDIVIDUAL STEPS</th><th>SYSTEM STEPS</th></tr> </thead> <tbody> <tr> <td>1. The individual clicks on the “subscribe to AutomatedSOS” button either on the options page or in the notification window.</td><td></td></tr> <tr> <td></td><td>2. The system starts the registration phase</td></tr> <tr> <td></td><td>3. The system checks if the account of the user is linked to a smartwatch</td></tr> <tr> <td>3. The user read and accept the terms and conditions.</td><td></td></tr> </tbody> </table>	INDIVIDUAL STEPS	SYSTEM STEPS	1. The individual clicks on the “subscribe to AutomatedSOS” button either on the options page or in the notification window.			2. The system starts the registration phase		3. The system checks if the account of the user is linked to a smartwatch	3. The user read and accept the terms and conditions.	
INDIVIDUAL STEPS	SYSTEM STEPS										
1. The individual clicks on the “subscribe to AutomatedSOS” button either on the options page or in the notification window.											
	2. The system starts the registration phase										
	3. The system checks if the account of the user is linked to a smartwatch										
3. The user read and accept the terms and conditions.											
EXIT CONDITIONS	The user is subscribed to AutomatedSOS										
EXCEPTIONS	If the Data4Help account of the user is not linked to a smartwatch, the system send an error message “Smartwatch not found”.										

The sequence diagrams describes all the interactions taking place during the sending of an SOS :



#### V.D. Performance requirements

To achieve to goal:

*An ambulance is requested to the location of the individual with a reaction time below 5 seconds from the time the parameters are below threshold*

The system must respect the following requirements: From the moment new data is received, all the actions that the system must perform (from the data recording to the sending of an emergency SOS) must have a total duration strictly under five seconds

#### V.E. Design Constraints

##### V.E.1. Standards compliance

As the AutomatedSOS service does not provide any web interface, there is no standard compliance.

### V.E.2. Hardware limitations

The hardware limitations of the AutomatedSOS service are the same as the one of the Data4Help service

### V.F. Software system attributes

The Software system attributes of the AutomatedSOS service are the same as the one of the Data4Help service

## VI. Formal analysis using Alloy

### VI. A Data4Help

#### VI.A.1. Purpose

We decided to model the Data4Help service using Alloy. This model will demonstrate that the basic functions of the Data4Help service can be achieved such as the ability for the third parties to make anonymized and individual requests and the possibility for individuals to accept or refuse requests. The purpose of this model is to test if the main goal of the Data4Help service is achieved:

*At any time, third parties should never have access to data of specific individuals without their agreement.*

This means that :

- Third party can access the data of individuals if and only if the individual request has been accepted
- Anonymized requests are accepted if and only if the number of individuals whose data satisfy the request is higher than 1000. As Alloy is not adapted to the use of big numbers, we chose to consider that anonymized requests are accepted if and only if the number of individuals whose data satisfy the request is higher than 2.

To prove that the constraint is respected, we tested the two following assertions:

```
assert No_access_data_if_refused {  
  /*third party can access an individual result if his individual request has been  
  refused*/  
  no th:Third_party | some req:th.Ind_request_issued| req.accepted=False and  
  req.result.data != none}
```

```
assert No_access_data_if_less_than_two_ind {  
  /*third party can access anonymized result if anonymized request concerns less  
  than two individuals*/  
  no th:Third_party | some req:th.Anonym_request_issued|  
  #req.Anonymized_Req_concern<2 and req.result.data != none}
```

For both of the assertions, no counter example was found. This mean that Alloy was **not able** to find :

- A third party for which an individual request had not been accepted but still produced a result.
- A third party for which an anonymized request concerned less than 2 individuals but still produced a result.

This mean that the privacy of the individuals is always guaranteed, and that third parties can never access private data without the individual's agreement which is for obvious reasons very important given the problem at hand.

#### VI.A.2. Worlds obtained

In this sections, three examples of world obtained by this model are presented.

- The first example shows two third parties making anonymized requests. The anonymized request of Third\_party0 concerns 2 individuals therefore it has been accepted and produces a result. At the contrary the request of Third\_party1 concerns only one individual therefore it has been refused.

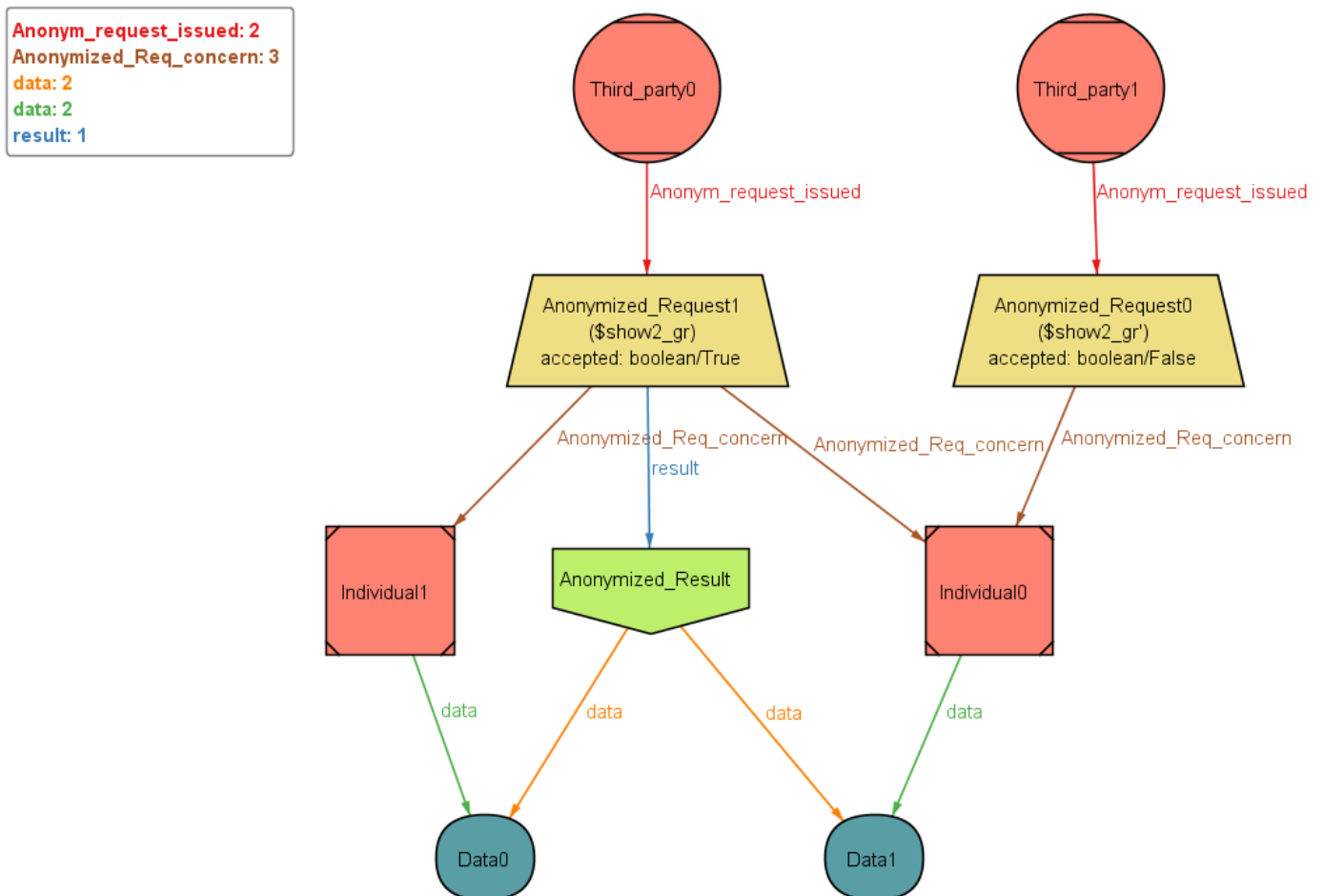


Figure 26 : World obtained by running predicate "show2"

- The second example shows a third party that made different types of requests.

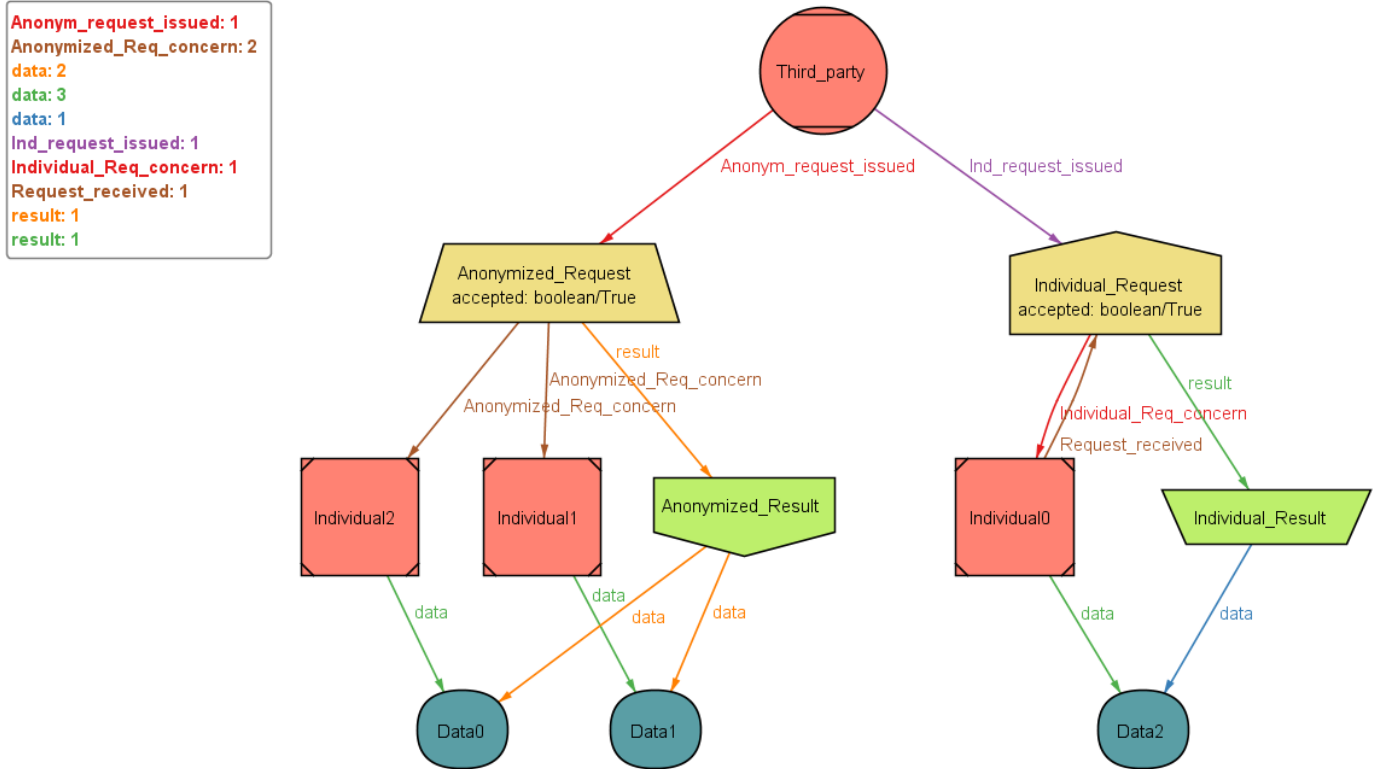


Figure 27: World obtained by running the predicate "show4"

- The third example shows three third parties that made individuals requests. In this example we can see that only the accepted requests produce a result

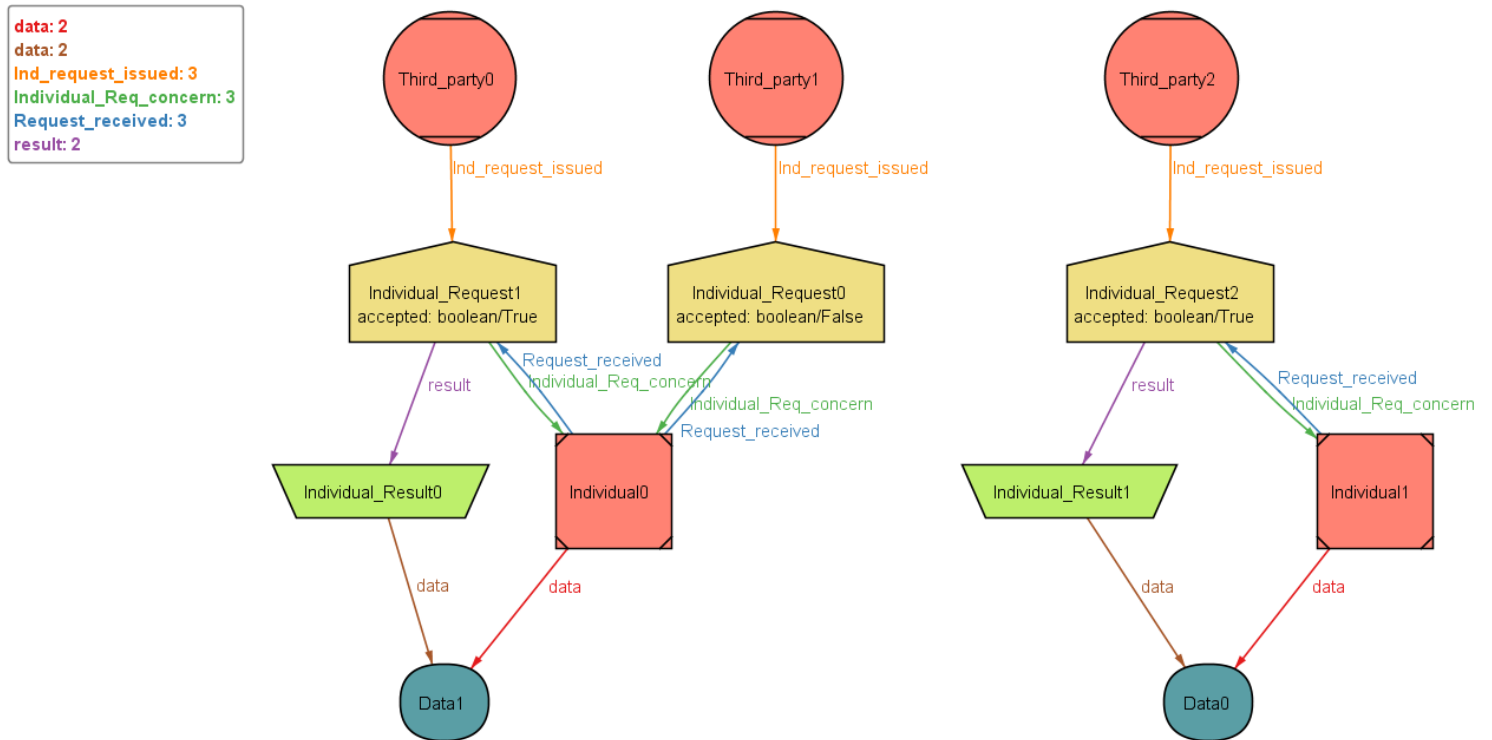


Figure 28 World obtained by running the predicate "show3"



### VI.A.3. Model

```
open util/integer
open util/boolean
```

```

/*****
/*****                               SIGNATURES                               *****/
/*****

sig Request {
  /*There are two types of requests :
    - anonymized request
    - individual request
  */
  accepted : one Bool,}

sig Result {/* the result of a request exists only if the request has been accepted*/}

sig Anonymized_Result extends Result {/* the result of a request exists only if the
request has been accepted*/
  data: some Data, /*an anonymized result contains the data of all the
individuals concerned by the request*/}

sig Individual_Result extends Result {/* the result of a request exists only if the
request has been accepted*/
  data: one Data,/*an individual result contains the data the individual
concerned by the request*/}

sig User {/* a user can be an Individual or a third party*/}

sig Data{/* the attributes of data (which are the Location and health data) are not
specified as it would it would unnecessarily
overload the model */
}

sig Individual_Request extends Request {
  /* An individual request is a request concerning one and only one individuals*/
  Individual_Req_concern: one Individual,
  result:lone Individual_Result}

sig Anonymized_Request extends Request {
  /* An anonymized request is a request concerning a group of individuals*/
  Anonymized_Req_concern: some Individual,
  result:lone Anonymized_Result}

sig Third_party extends User {
  /* A third party is a type of user that is able to make requests.
  A third party can make as much requests as he wants. Those requests can be
  anonymized requests or individual requests.
  */
  Ind_request_issued: set Individual_Request ,
  Anonym_request_issued: set Anonymized_Request
  /*A third party also has a name, an identifier, an address, a phone number etc
  but as the goal of this model is to verify the interactions between the individuals,
  the third parties and the requests,
  we thought that adding these attributes would not be relevant because it would not
  prove anything and
  it would make reading more difficult */}

```

```

sig Individual extends User {
  /* An individual is a type of user that is NOT able to make requests.
  An individual see the requests he receive and has to answering them by accpeting or
  refusing them*/
  Request_received: set Individual_Request,
  data: one Data,
  /*An individual also has a name, a surname, a fiscal code, an age, an adress, a
  phone number etc
  but as the goal of this model is to verify the interactions bettween the individuals,
  the third parties and the requests,
  we thought that adding these attributes would not be relevant because it would not
  prove anything and
  it would make reading more difficult */}

/*****
/*****FACTS *****/
/*****/

/**ABOUT REQUESTS *****/

fact unique_individual_per_ind_request {
  /*two disjoint individuals cannot be receive the same individual request*/
  all ind, ind' : Individual | no ind_r:Individual_Request | ind != ind' and ind_r in
  (ind.Request_received & ind'.Request_received)}

fact reciprocity_for_ind_and_requests {
  --if an individual is linked to an individual_request then the ind_request must
  linked to ind
  all req:Request, ind:Individual | (req in ind.Request_received) iff (ind in
  req.Individual_Req_concern)}

fact request_must_be_linked_to_third_party{
  /* A request must be issued by a third party*/
  (all req:Individual_Request | some th:Third_party| req in th.Ind_request_issued)
  and (all req:Anonymized_Request | some th:Third_party| req in
  th.Anonym_request_issued)}

/**ABOUT THIRD PARTIES*****/

fact unique_third_party_per_request {
  /*two third parties cannot produce the same request.
  Indeed, a request male a unique connection between ONE individual, ONE third party
  and if accepted ONE result*/
  all third, third' : Third_party | no req : Request | third != third' and req in
  (third.Ind_request_issued & third'.Ind_request_issued)}

/** ABOUT RESULTS *****/

fact unique_result_anon {
  /*two disjoint requests cannot produce the same result
  Indeed, any anonymized request produces its own result*/
  all req, req' : Anonymized_Request | no res:Anonymized_Result | req!=req' and res
  in (req.result & req'.result)}

fact unique_result_ind {
  /*two disjoint requests cannot produce the same result

```



```

        no th:Third_party | some req:th.Ind_request_issued| req.accepted=False and
req.result.data != none}

```

```

assert No_access_data_if_less_than_two_ind {
/*third party can acces anonymized result if anonymized request concerns less than
two individuals*/

```

```

        no      th:Third_party      |      some      req:th.Anonym_request_issued|
#req.Anonymized_Req_concern<2 and req.result.data != none}

```

```


/*****
/***** PREDICATES *****/
/*****/

pred show1{}

```

```

pred show2 {/* We want to see two anonymized requests and no individual requests.
The two anonymized requests
must be issued by two different third parties. One of the anonymized request must
be accepted and the other
must be refused*/

```

```

        #Anonymized_Request=2
        #Individual_Request=0
        #Third_party = 2
        (some gr:Anonymized_Request | #gr.Anonymized_Req_concern>=2)
        (some gr':Anonymized_Request | #gr'.Anonymized_Req_concern<2)
        (all th:Third_party | #th.Anonym_request_issued = 1)
        }

```

```

pred show3 {/* We want to see three individual requests and no anonymized requests.
The three individual requests
must be issued by three different third parties. Only one of the individual requests
must be refused.*/

```

```

        #Anonymized_Request=0
        #Individual_Request=3
        #Third_party=3
        #Individual=2
        (one req:Individual_Request | req.accepted=False)
        (all th:Third_party | #th.Ind_request_issued >= 1)
        (all ind:Individual | #ind.Request_received>=1)
        }

```

```

pred show4{/* We want to see three individual requests and no anonymized requests.
The three individual requests
must be issued by three different third parties. Only one of the individual requests
must be refused.*/

```

```

        #Anonymized_Request=1
        # Individual_Request=1
        #Third_party=1
        #Individual=3
        (one req:Individual_Request | req.accepted=True)
        (one req:Anonymized_Request | req.accepted=True)
        }

```

```

--run show2 for 4
--run show3 for 5
run show4 for 5

```

```

--check No_access_data_if_refused
--check No_access_data_if_less_than_two_ind

```

## VI. B AutomatedSOS

### VI.B.1. Purpose

As the AutomatedSOS service is build on top of Data4Help, we based our Alloy model for AutomatedSOS on the one for Data4Help. Therefore, we simply added the classes SOS\_monitoring\_service, SOS, and Ambulance\_service.

The purpose of this model was to check if in the case of a monitored individual having his health parameters below threshold, an SOS would be sent. Which correspond the goal of the AutomatedSOS service : *“An ambulance is requested to the location of the individual with a reaction time below 5 seconds from the time the parameters are below threshold”*.

As Alloy is not adapted to the simulation of time, we chose not to take into account the fact that the SOS need to be sent in less than 5 seconds

To prove that an SOS was always sent when it was necessary, we tested the following assertions:

```
assert SOS_sent_if_bellow_threshold {  
  /* no SOS is sent even when parameters are bellow threshold*/  
  all serv:SOS_monitoring_service |#serv.sos =1 iff  
    serv.monitored_ind_below_threshold=True  
}
```

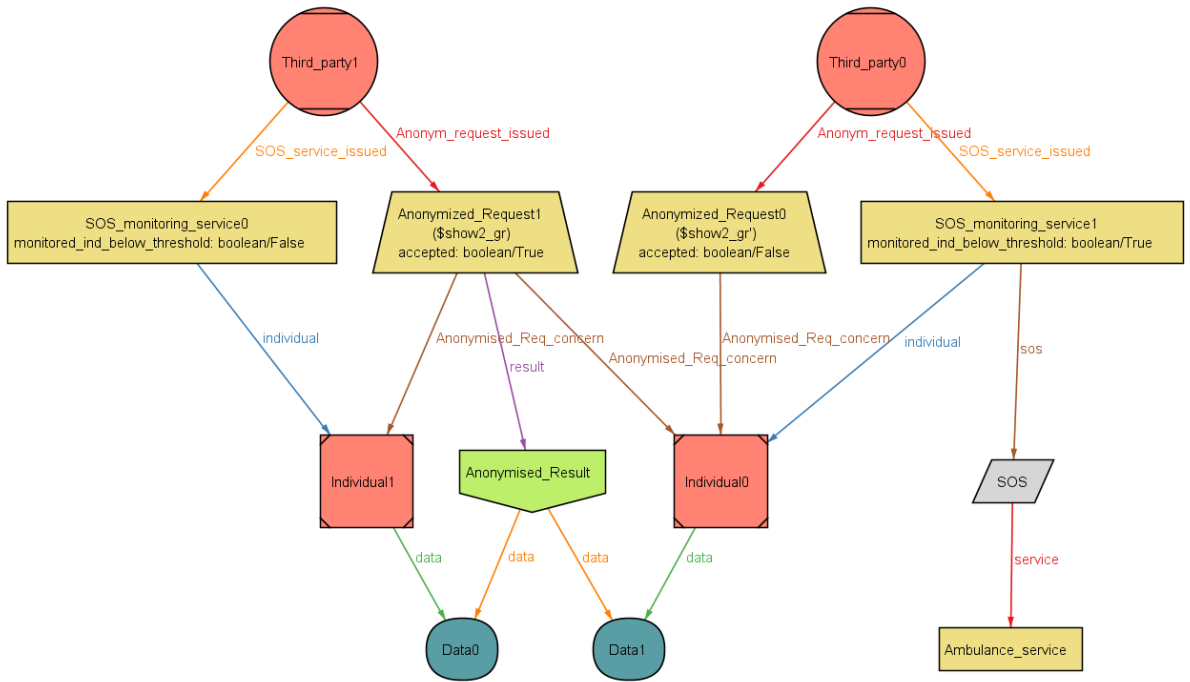
For this assertion, no counter example was found. This mean that Alloy was **not able** to find an SOS\_monitoring\_service that had not sent any SOS even if it was monitoring someone who had his health parameters bellow thresholds.

In others words, his model prove that when an individual is monitored by AutomatedSOS, if his health parameters are below thresholds, an SOS is sent.

### VI.B.2. Worlds obtained

- The following example shows a third party monitoring an individual. And as the individual's health parameters are below thresholds, an SOS has been sent.

Anonym\_request\_issued: 2  
 Anonymised\_Req\_concern: 3  
 data: 2  
 data: 2  
 individual: 2  
 result: 1  
 service: 1  
 sos: 1  
 SOS\_service\_issued: 2



### VI.B.3. Model

open util/integer  
 open util/boolean

```

/*****
/***** SIGNATURES *****/
/*****/

```

```

sig Request {
  /*There are two types of requests :
  - anonymized request
  - individual request
  */
  accepted : one Bool,}

```

```

sig Result { /* the result of a request exists only if the request has been accepted*/}

```

```

sig Anonymized_Result extends Result { /* the result of a request exists only if the
request has been accepted*/
  data: some Data, /*an anonymized result contains the data of all the
individuals concerned by the request*/}

```

```

sig Individual_Result extends Result { /* the result of a request exists only if the
request has been accepted*/
  data: one Data, /*an individual result contains the data the individual
concerned by the request*/}

```

```

sig User { /* a user can be an Individual or a third party*/}

```

```

sig Data { /* the attributes of data (which are the location and health data) are not

```

```
specified as it would it would unnecessarily
overload the model */
}
```

```
sig Individual_Request extends Request {
  /* An individual request is a request concerning one and only one individuals*/
  Individual_Req_concern: one Individual,
  result: lone Individual_Result}
```

```
sig Anonymized_Request extends Request {  
  /* An anonymized request is a request concerning a group of individuals*/  
  Anonymized_Req_concern: some Individual,  
  result:lone Anonymized_Result}
```

```
sig Third_party extends User {
  /* A third party is a type of user that is able to make requests.
  A third party can make as much requests as he wants. Those requests can be
  anonymized requests or individual requests.
  */
}
```

```

Ind_request_issued: set Individual_Request ,
Anonym_request_issued: set Anonymized_Request,
SOS_service_issued: set SOS_monitoring_service
/*A third party also has a name, an identifier, an address, a phone number etc
but as the goal of this model is to verify the interactions between the individuals,
the third parties and the requests, we thought that adding these attributes would
not be relevant because it would not prove anything and it would make reading more
difficult */}

```

```
sig Individual extends User {
  /* An individual is a type of user that is NOT able to make requests.
  An individual see the requests he receive and has to answering them by accpeting or
  refusing them*/
  Request_received: set Individual_Request,
  data: one Data,
  /*An individual also has a name, a surname, a fiscal code, an age, an adress, a
  phone number etc but as the goal of this model is to verify the interactions bettween
  the individuals, the third parties and the requests, we thought that adding these
  attributes would not be relevant because it would not prove anything and it would
  make reading more difficult */}

```

```
sig SOS_monitoring_service {
  monitored_ind_below_threshold:one Bool,
  sos: lone SOS,
  individual : one Individual}
```

```
sig Ambulance_service{}
```

```
sig SOS {
  service: one Ambulance_service }
```

```

/*****
/*****                               FACTS                               *****/
/*****
/*****/

```

```

/ **ABOUT REQUESTS **** */

```

```
fact unique_individual_per_ind_request {  
  /*two disjoint individuals cannot be receive the same individual request*/
```

```

all ind, ind' : Individual | no ind_r:Individual_Request | ind != ind' and ind_r in
(ind.Request_received & ind'.Request_received)}

fact reciprocity_for_ind_and_requests {
--if an individual is linked to an individual_request then the ind_request must
linked to ind
all req:Request, ind:Individual | (req in ind.Request_received) iff (ind in
req.Individual_Req_concern)}

fact request_must_be_linked_to_third_party{
/* A request must be issued by a third party*/
(all req:Individual_Request | some th:Third_party| req in th.Ind_request_issued)
and (all req:Anonymized_Request | some th:Third_party| req in
th.Anonym_request_issued)}

/**ABOUT THIRD PARTIES***/

fact unique_third_party_per_request {
/*two third parties cannot produce the same request.
Indeed, a request make a unique connection between ONE individual, ONE third party
and if accepted ONE result*/
all third, third' : Third_party | no req : Request | third != third' and req in
(third.Ind_request_issued & third'.Ind_request_issued)}

/** ABOUT RESULTS ***/

fact unique_result_anon {
/*two disjoint requests cannot produce the same result
Indeed, any anonymized request produces its own result*/
all req, req' : Anonymized_Request | no res:Anonymized_Result | req!=req' and res
in (req.result & req'.result)}

fact unique_result_ind {
/*two disjoint requests cannot produce the same result
Indeed, any individual request produces its own result*/
all req, req' : Individual_Request | no res:Individual_Result | req!=req' and res
in (req.result & req'.result)}

fact acceptance_implies_result_anon {
/* (If an anonymized request is accepted THEN the request must produce a result)
AND (if an anonymized request is refused THEN it cannot produce a result)*/
all req : Anonymized_Request | (req.accepted = True iff req.result != none) and
(req.accepted =False iff req.result = none)}

fact acceptance_implies_result_ind {
/* (If an individual request is accepted THEN the request must produce a result)
AND (if an individual request is refused THEN it cannot produce a result)*/
all req :Individual_Request | (req.accepted = True iff req.result != none) and
(req.accepted =False iff req.result = none)}

fact individuals_concerned_by_Anonymized_request {
/* An anonymized request must concern at least 1000 individuals to be accepted.
As Alloy does not allow this type of constraint, we chose to consider that an
anonymized request must concern at least 2 individuals
to be accepted*/
(all req: Anonymized_Request | req.accepted=True iff #req.Anonymized_Req_concern
>=2) }

fact result_produced_by_request{

```



```

/* A result cannot exists on his own, it must be produced by a request*/
(all res:Anonymized_Result | some req:Anonymized_Request| res=req.result ) and (all
res:Individual_Result | some req:Individual_Request| res=req.result )}

/** ABOUT DATA*****/

fact data_unicity{
/* two different individuals cannot have the same data*/
all ind, ind' : Individual | no dat:Data | ind != ind' and dat in (ind.data&
ind'.data)}

fact data_belong_to_someone{
/* the data must be owned by an individual*/
all dat:Data | some ind:Individual| dat=ind.data}

fact data_consistency_ind {
/* The data of an individual_result is the data of the unique individual*/
(all req:Individual_Request | req.accepted=True implies
(req.result.data=req.Individual_Req_concern.data))}

fact data_consistency_anonymous{
/* The data of an anonymized_result is the data of the individuals*/
(all req:Anonymized_Request | all ind:req.Anonymized_Req_concern |ind.data in
(req.result.data) iff req.accepted = True)}

/** ABOUT SOS*****/

fact one_ind_monitored_per_one_service {
/*one individual can be monitored by only one service*/
(all serv, serv' :SOS_monitoring_service | no ind:Individual |serv!=serv' and ind
in (serv.individual & serv'.individual))}

fact SOS_sent_if_bellow_threshold {
/*An SOS is sent if and only if the individual monitored is bellow thresholds*/
(all serv:SOS_monitoring_service | serv.monitored_ind_below_threshold=True iff
#serv.sos=1)}

fact SOS_linked_to_service{
/* A sos can not exist without being linked to an SOS_monitoring_service */
all message:SOS| one serv:SOS_monitoring_service | message=serv.sos}

fact SOS_serive_linked_to_third_party{
/* A sos_service must be linked to a third party */
all serv:SOS_monitoring_service| one th:Third_party | serv in th.SOS_service_issued}

/*****
/***** ASSERTIONS *****/
/*****/

assert No_access_data_if_refused {
/*third party can access an individual result if his individual request has been
refused*/
no th:Third_party | some req:th.Ind_request_issued| req.accepted=False and
req.result.data != none}

```

```

assert No_access_data_if_less_than_two_ind {
/*third party can acces anonymized result if anonymized request concerns less than
two individuals*/
    no    th:Third_party    |    some    req:th.Anonym_request_issued|
#req.Anonymized_Req_concern<2 and req.result.data != none}

assert SOS_sent_if_bellow_threshold {
/* no SOS is sent even when parameters are bellow threshold*/
    all    serv:SOS_monitoring_service    |#serv.sos    =1    iff
serv.monitored_ind_bellow_threshold=True
}

/*****
/***** PREDICATES *****/
/*****/
pred show1{ }

pred show2 { /* We want to see two anonymized requests and no individual requests.
The two anonymized requests must be issued by two different third parties. One of
the anonymized request must be accepted and the other must be refused*/
    #Anonymized_Request=2
    #Individual_Request=0
    #Third_party = 2
    (some gr:Anonymized_Request | #gr.Anonymized_Req_concern>=2)
    (some gr':Anonymized_Request | #gr'.Anonymized_Req_concern<2)
    (all th:Third_party | #th.Anonym_request_issued = 1) }

run show2 for 4

--check SOS_sent_if_bellow_threshold
--run show1

```

## VII. Effort spent

### VII.A. Common work

Date	Work	Hours
12/10/2018	Discussing the subject and assumptions we made, distribution of the different part	3
19/10/2018	Presenting to each other our work and discussing repartition of the remaining work	3
26/10/2018	Presenting to each other our work and discussing repartition of the remaining work	3
3/11/2018	Presenting to each other our work and discussing repartition of the remaining work	3

## VII.B. Mohamed

<i>Date</i>	<i>Work</i>	<i>Hours</i>
11/10/2018	Working on overall understanding of the project	2
20/10/2018	Working on over all description	2
22/10/2018	Over all description correction and modification	1
27/10/2018	Specific requirements (performance requirements) and correcting some mistakes in overall description	2
01/11/2018	Design constraints	1
02/11/2018	Software system attributes (Reliability ,Availability)	2
03/11/2018	Software system attributes (Security,Maintainability,Portability)	2
04/11/2018	Requirements part (Non-Functional)	2
05/11/2018	Checking all previous parts in the rasd and correcting some mistakes	4
06/11/2018	Writing some scenarios and correcting previous mistakes	2
07/11/2018	Understanding and reading alloy part	3
10/11/2018	Reading all the document and checking for corrections	4
11/11/2018	Product function modification and final document modification	4

## VII.C. Emma

<i>Date</i>	<i>Work</i>	<i>Hours</i>
11/10/2018	Working on overall understanding of the project	1
13/10/2018	Working on overall understanding of the project, Scope definition (interface world machine)	3
15/10/2018	Purpose, Scope, Definitions	1
17/10/2018	Purpose, Definitions	1
19/10/2018	Class Diagrams, Purpose	2
20/10/2018	Completing Class Diagrams, States Chart and writing their description	3
24/10/2018	User Interfaces, States Chart	2
25/10/2018	Document Structure, User Interfaces	2
27/10/2018	User Interfaces, Update Definitions	4
28/10/2018	Uses cases	2
30/10/2018	Update previous parts, Requirements	1
01/11/2018	Uses cases, and modifying previous parts	3
03/11/2018	Sequence Diagrams, Uses cases description	2
04/11/2018	Sequence Diagrams and sequence Diagrams description	4
05/11/2018	Improving previous part in report, adding details, checking overall coherence	3
07/11/2018	Implementation of Alloy models	2
08/11/2018	Implementation of Alloy models	1
10/11/2018	Finishing alloy models, Adding to the report Alloy diagrams and diagram description, Improving previous part	4
11/11/2018	Proofreading and adding more details to the report	4

## VIII. References

- [1] SOMMERVILLE, Iam. *Software engineering* 9. International edition.
- [2] Assignment *Mandatory Project: goal schedule, and rules*.
- [3] Les numériques, *COMPARATIF / Quelle montre connectée choisir ?*  
<https://www.lesnumeriques.com/montre-connectee/comparatif-montres-connectees-a1781.html>