POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

Data4Help
AutomatedSOS
Track4Run

Manuel Trivilino, Davide Salaorni, Luca Terracciano

Document version: 2.0
December 10, 2018

Indice

1 Introduction

1.1 Purpose

The purpose of this document is to describe the software-based solutions that TrackMe wants to offer to its customers. First of all, the system is analyzed in terms of description of the problem and analysis of goals, requirements and constraints. After that, a solution is modeled in terms of software and hardware constraints paying attention also to some implementation details. This document is addressed to the developers who will implement this solution.

1.2 Scope

1.2.1 Description of the given problem

TrackMe is a company that develops software-based services for third parties and for consumers. The main service is called Data4Help.
Data4Help is a service that allows third parties to monitor the location and the health status of individual, it handles a policy of permissions for each user and collects individual's data from their personal devices. The service supports the registration of individuals who, by registering, agree that TrackMe acquires their data. After registration, these third parties can request:

- Access to the data of some specific individuals (we can assume, for instance, that they know an individual by his/her social security number or fiscal code in Italy). In this case, TrackMe passes the request to the specific individuals who can accept or refuse it.
- Access to anonymized data of groups of individuals (for instance, all those living in a certain geographical area, all those of a specific age range, etc.). These requests are handled directly by Data4Help that approves them if it is able to properly anonymize the requested 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. Then, to avoid this risk and the possibility of a misuse of data, Data4Help will not accept the request.
  For simplicity, we assume that Data4Help will accept any request for which the number of individuals whose data satisfy the request is higher than 1000.

As soon as a request for data is approved, TrackMe makes the previously saved data available to the third party. Also, it allows the third party to subscribe to new data and to receive them as soon as they are produced.
TrackMe develops itself two third-party services: AutomatedSOS and Track4Run.

AutomatedSOS is a project created by TrackMe and placed on top of Data4Help, which aims to offer a personalized and non-intrusive SOS service to people that are worried about their own safeness and want to have an health

inspector always with them.
AutomatedSOS, exploiting data provided from Data4Help, monitors the health status of the subscribed customers and, when such parameters are below certain thresholds, sends to the location of the customer an ambulance, guaranteeing a reaction time of less than 5 seconds from the time the parameters are below the threshold.

Track4Run is another service offered by TrackMe and conceived for athletes participating in a run, which should allow organizers to define the path for the run, competitors to enroll to the run, and spectators to see on a map the position of all runners during the run.
Of course, also in this case, Track4Run will exploit the features offered by Data4Help.

1.2.2 Goals

G.1 Allow customers' registration.

- G.1.1 Registration can be done as individual user.
- G.1.2 Registration can be done as third party.

G.2 Third parties can request for anonymized data.

G.3 Third parties can subscribe to the service in order to periodically receive new data.

G.4 Third parties can receive for a individual user's data.

G.5 Users' health status is continuously checked in AutomatedSOS and in case of it overcomes a certain threshold values an ambulance is called within 5 seconds.

G.6 Users can organize a run.

G.7 Users can enroll to a run.

G.8 Users can be spectators of a run seeing the partecipants' position on a map.

1.3 Definitions, Acronyms, Abbreviation

1.3.1 Definitions

- *Customer:* generic person not registered in any of the services provided by TrackMe.
- *User:* person registered in at least one of the services provided by TrackMe.
- *Third Party:* person or company registered in Data4Help that wants to access to the service in order to acquire data.
- *Individual Data:* data that belong to a specific user, in particular personal information and data acquired from their devices (i.e. location, health status, etc.).
- *Anonymized Data:* aggregated data acquired from a group of users with specific characteristics without keeping track of who they belong to.
- *Anonymized Data Subscription:* request that allows third parties to receive new anonymized data as soon as they are produced.
- *Ambulance Dispatcher:* the contact center that handles the health care service and the ambulance allocation.
- *Google Maps:* web mapping service developed by Google offering satellite imagery, street maps etc.
- *Push Notification Service:* web service that offers a fast and secure channel to notify and exchange messages on the web.

1.3.2 Acronyms

- *RASD:* Requirement, Analysis and Specification Document
- *GPS:* Global Position System
- *API:* Application Programming Interface

1.3.3 Abbreviations

- *G.n* Goal number n
- *D.n* Domain assumption number n
- *R.n* Requirement number n

1.4 Document Structure

The RASD is composed by:

1. Explanation of the given problem, the analysis in terms of goals and it gives some information in order to better understand all the other document's parts.

2. Describes the solution structure identifying it's general organization, the users, the operating environment and the major design and implementation constraints.

3. The software implementation is analyzed paying attention to the software and hardware details adding also a description in terms of functional requirements, scenarios and use cases.

4. The system is analyzed both in its static aspects and the dynamic ones using a specific formal modeling language, Alloy, in which the most important features are analyzed in order to verify if every constraint of the system is not violated.

2 Overall Description

2.1 Product Perspective

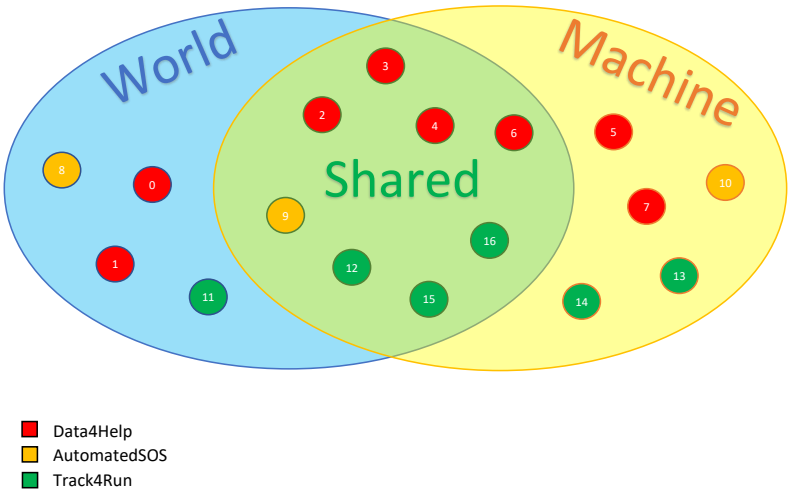


Figura 1: World, machine and shared phenomena.

0. User's location.
1. User's health.
2. Collecting data from smart devices.
3. User's registration (agreement to share anonymized data).
4. Data request from Third Party.
5. Checking the anonymized data threshold.
6. User's agreement for individual data share.
7. Keeping track of connection between user and Third Party.
8. Ambulance.
9. Ambulance notification.
10. Health status monitoring.
11. Run event.
12. Run creation and path definition.
13. Run management and organization.
14. Runner's registered position.
15. View of the run event.
16. Enrollment to the run.

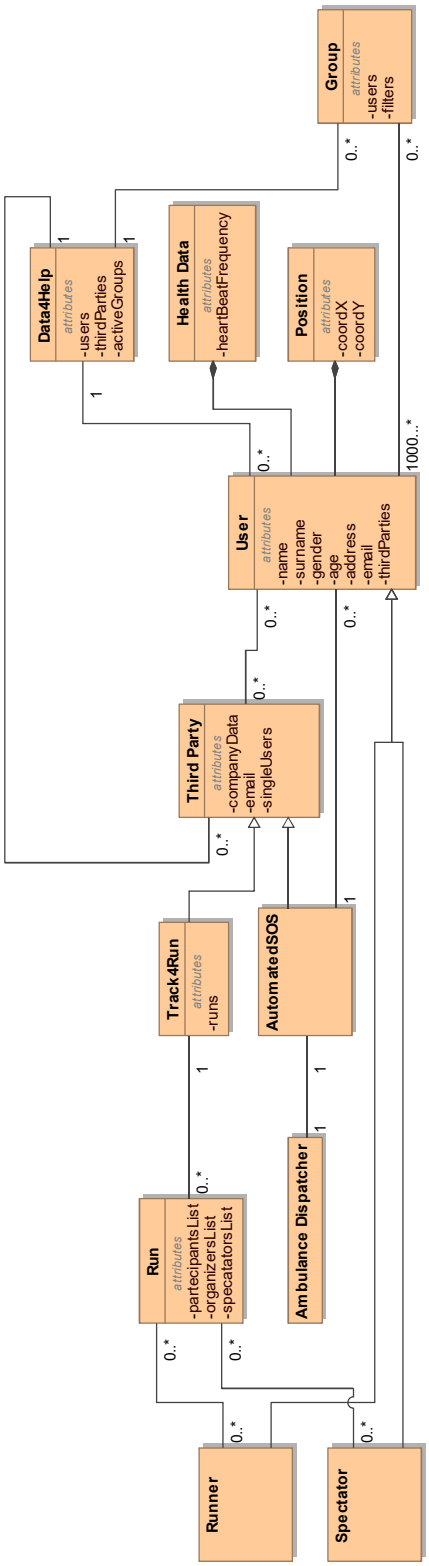


Figura 2: Class Diagram of the whole system.

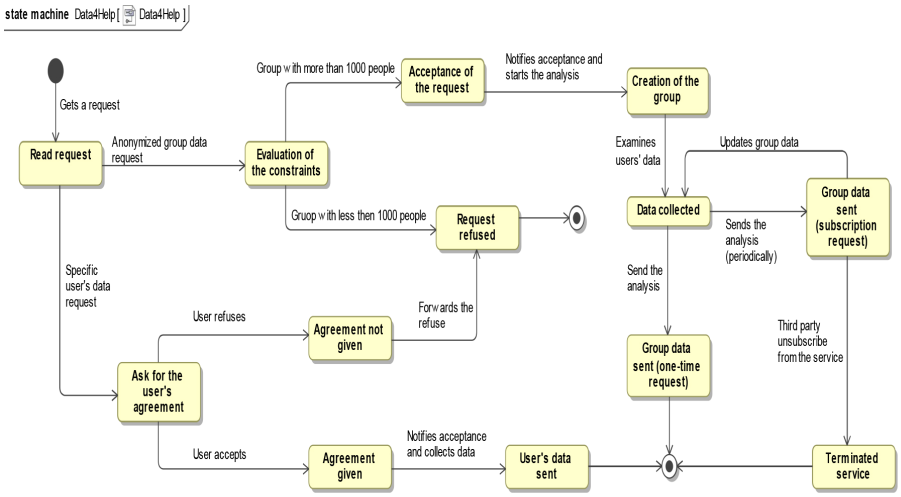


Figura 3: Data4Help state diagram.

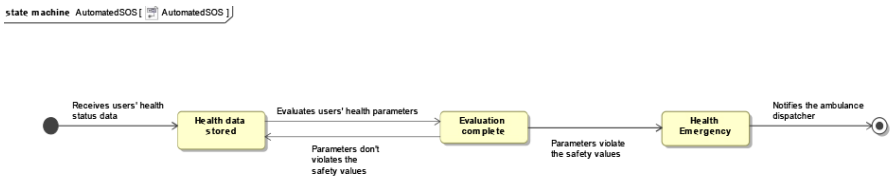


Figura 4: AutomatedSOS state diagram.

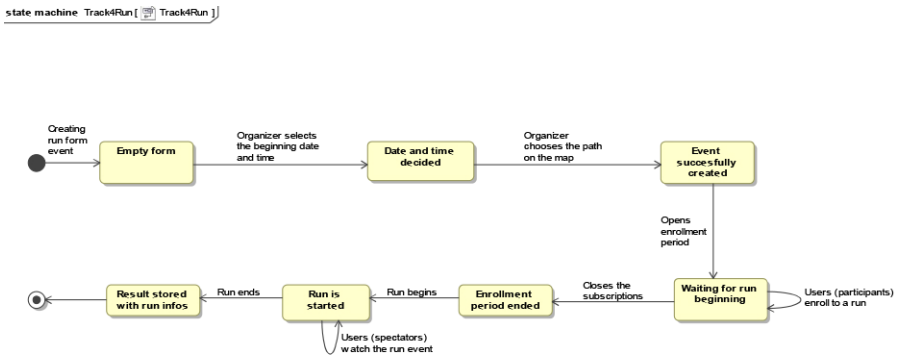


Figura 5: Track4Run state diagram.

2.2 Product Functions

The main function that the system must perform are divided in three section, one for each software components.

- Data4Help
    1. Allow users to share their personal data with Data4Help and with the third parties who require them.
    2. Allow third parties to require users' data in anonymous form by using filters and offers the possibility to make a subscription to the request.
    3. Third parties can request for a specific user's data.
    4. Users can decide whether accept or not a data request from a third party.
- AutomatedSOS
    1. Offers to the registered users to have their health status be continuously monitored.
    2. If the system detects an emergency situation it notifies an ambulance.
- Track4Run
    1. Offers to the users the possibility to organize runs deciding the entire path, the day and the hour at which the event begins.
    2. Users can search available runs that will take place in the city that they have selected previously and enroll to one of them.
    3. Track4Run offers also to the users the possibility to watch a run event from their smartphones using a map representing the path and the real time runners' position.

2.3 User Characteristics

The following actors are the users of the services offered by TrackMe.

- User: a person that is successfully registered to TrackMe as consumer and allow to acquire anonymous data, eventually a client of third party services (i.e. AutomatedSOS, Track4Run) that allows even personal data.
- Third Parties: a company or a person who is registered to TrackMe as "Third party" that access to anonymous and can require to access to individual data.
- Administrator: a superuser, responsible for the maintenance of the system (not relevant for the analysis of the requirements).

2.4 Assumption, Dependencies and Constraints

2.4.1 Domain Assumptions

[D.1] The data collected from the devices (position and health status) are reliable, accurate and in real time.

[D.2] The personal information given by the user (age, address, gender...) are correct and the account is not used by other people.

[D.3] Ambulance call and information about patients' status and position are correctly dispatched to the ambulance contact center.

[D.4] Information and data (places, maps, paths, etc.) received from external services are correct are reliable.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The users can access to the services provided by TrackMe in several ways. Data4Help offers a web-based interface to register and manage the account, the users can easily connect to the web site with a browser to access to the services. Furthermore third parties can integrate the Data4Help services in their own apps, in this way the users can manage the permission from the third party's app. AutomatedSOS and Track4Run provide to the users a mobile app available for the most popular devices (Android or iOS smartphone or smartwatch) to access to the services.

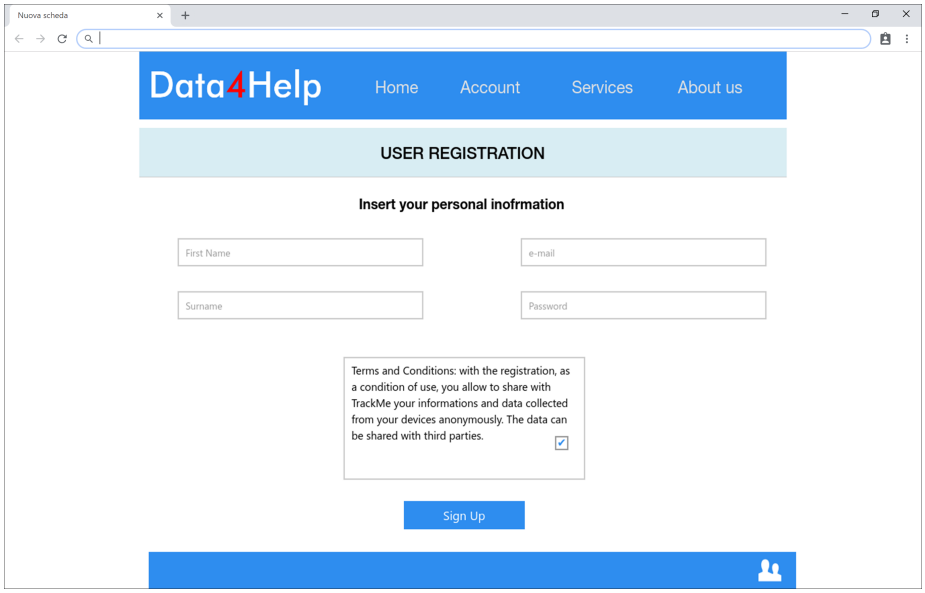


Figura 6: User's sign up.

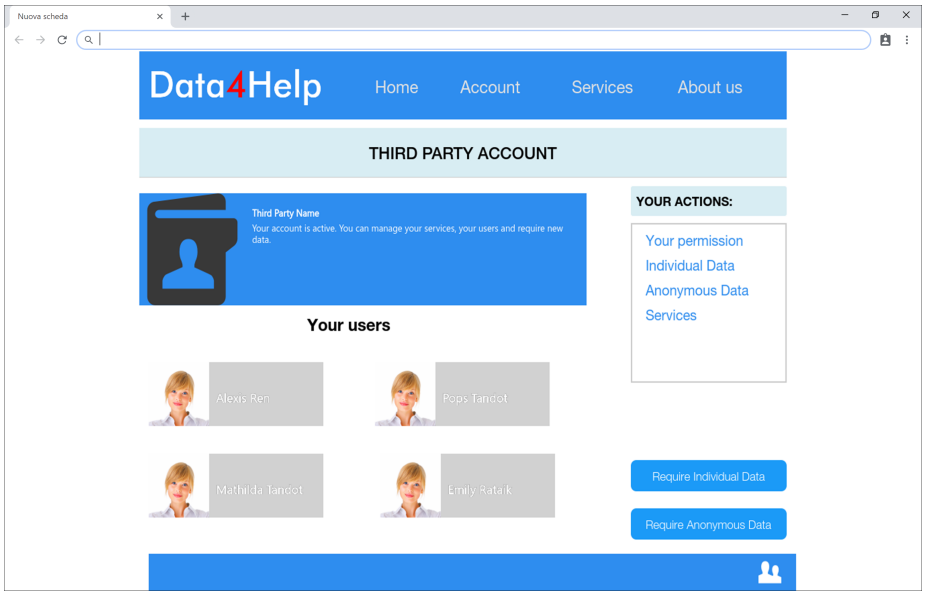


Figura 7: Personal area in third party account.

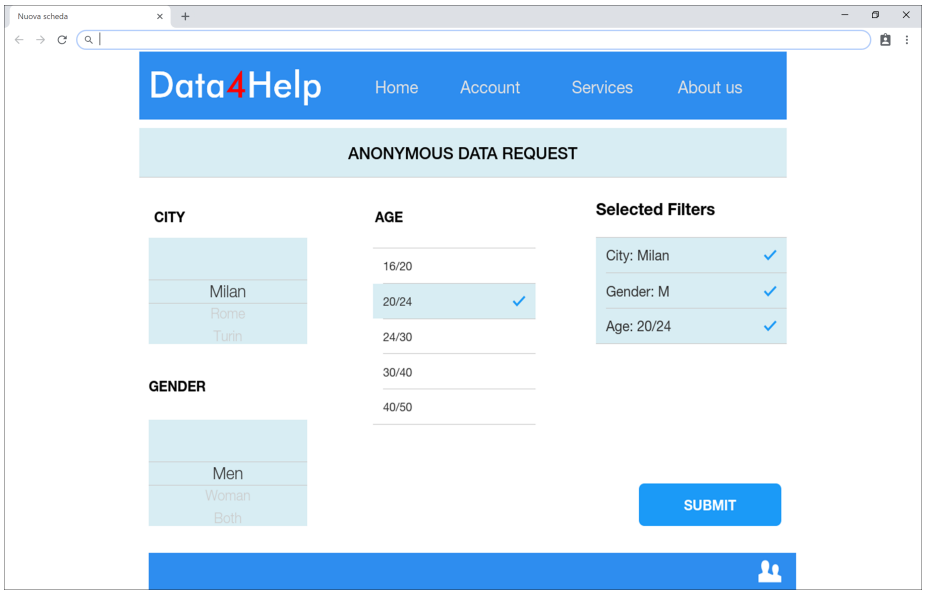
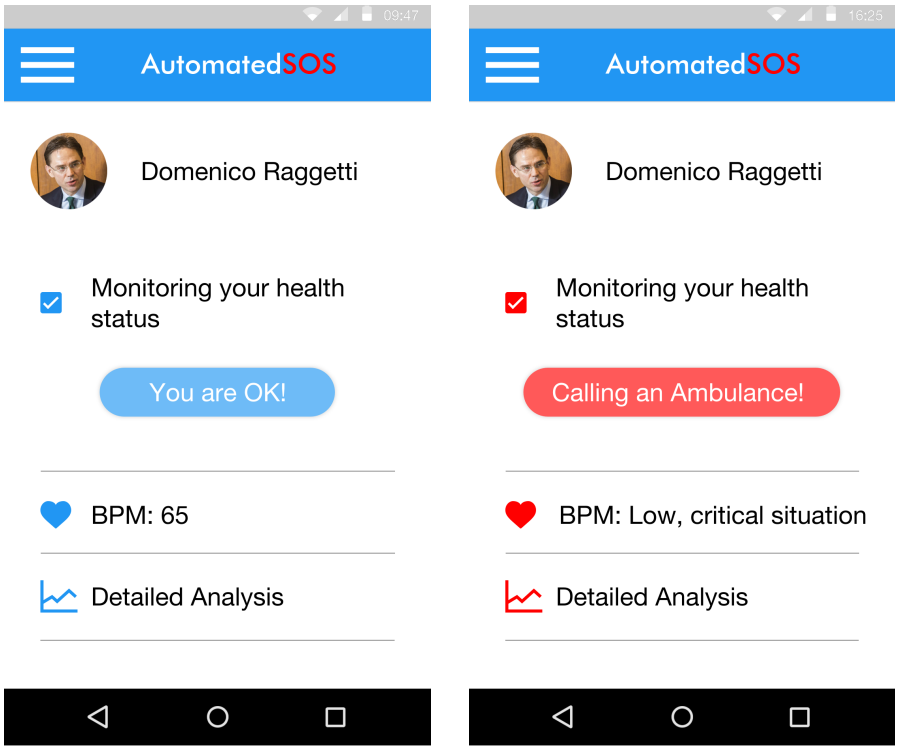


Figura 8: One-time request interface.



(a) Regular health notification. (b) Emergency notification.

Figura 9: Smartphone interfaces of AutomatedSOS.

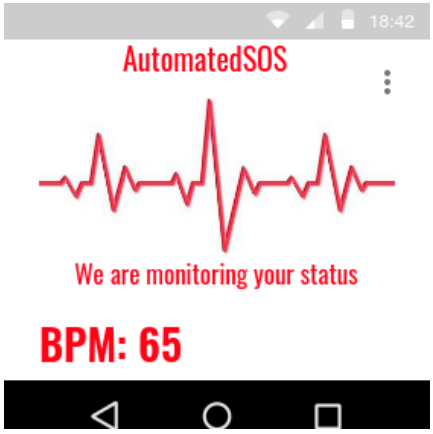
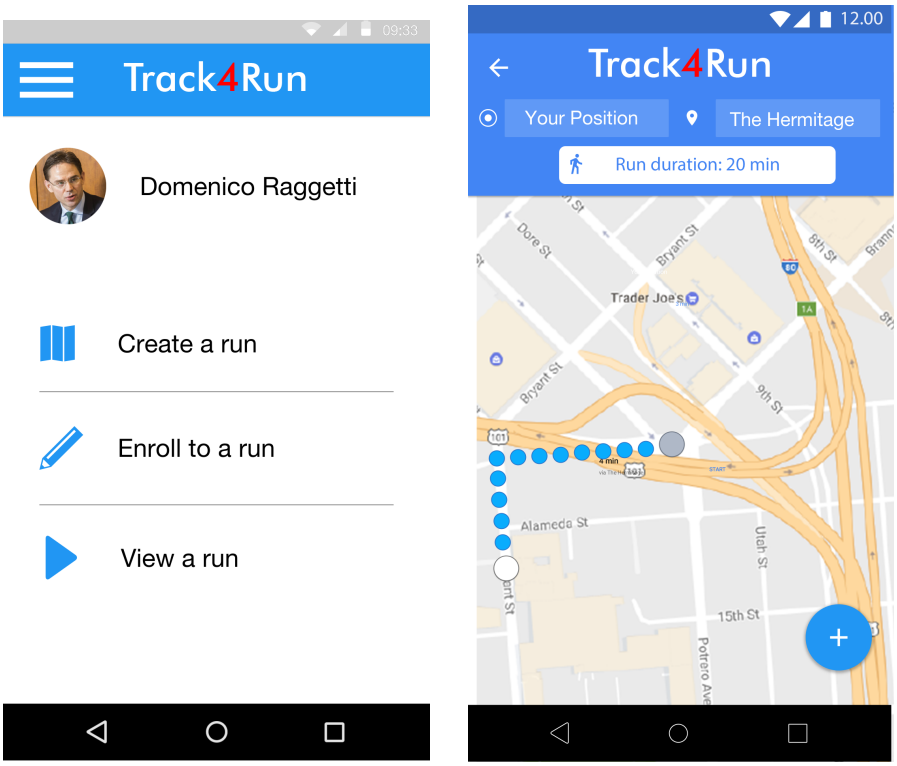


Figura 10: Smartwatch interface of AutomatedSOS.



(a) Main page with all options.

(b) Selection of the path during the event creation.

Figura 11: Smartphone interfaces of Track4Run.

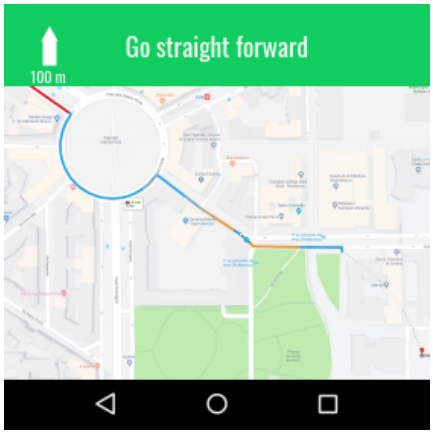


Figura 12: Smartwatch interface of Track4Run.

3.1.2 Hardware Interfaces

Data4Help: The system should be a sever-based system made up of:

- A main server (possibly distributed to guarantee better reliability and scalability) with a server OS to handle the main functions: permission and request managing, data managing, etc.
- A database to store the collected data (possibly in-source to preserve data sensibility).
- A DBMS to manage the database.

The Data4Help services are accessible from the users through a web browser or through mobile device applications of third parties. The system, to collect data, exploits the devices of the users which use third parties services, these devices must integrate sensors in order to allow Data4Help to acquire data.

AutomatedSOS: The system is based on a Client-Server architecture. It is composed by:

- A server to provide the main functions: users' data managing and handle the emergency status and notify the ambulance dispatcher through the push notification external service.
- A mobile device with a supported OS (Android OS, iOS, Android Wear OS, etc.) where it is possible to install the AutomateSOS app. The device (Smartphone, smartwatch, smartwears, etc.) must respect the constraints that Data4Help requires.

Track4Run Also this system is based on a Client-Server architecture. It is composed by:

- A server to provide the main functions: runs managing, runners tracking, streaming for the spectators during the run.
- A mobile device with a supported OS (Android OS, iOS, Android Wear OS, etc.) where it is possible to install the Track4Run app. The device (Smartphone, smartwatch, smartwears, etc.) must respect the constraints that Data4Help requires.

3.1.3 Software Interfaces

Data4Help: The interfaces needed from the system are:

- A server OS to manage all the back-end functions. It has to implement REST API to communicate and transfer data with the mobile devices, the third parties' servers.
- A DBMS to manage the database.

Data4Help furthermore gives to the third parties a set of client-side APIs that provide an interface in order to connect their own client-side application with Data4Help services.

AutomatedSOS: the system needs to manage the communication with Data4Help, to analyze the data and to communicate with the ambulance dispatcher and the communication between the app and the server. The software interfaces needed are:

- Android OS, Android Wear OS or iOS for the client-side application.
- Data4Help API to access to services and receive data with Data4Help.
- REST API to communicate and transfer data on the web between the client app and the server.
- Interface with the Push Notification service to send reliable messages to the ambulance dispatcher.

Track4Run: the system needs to manage the run events with the respective runners and spectators and let the spectators watch the run events on the map. Otherwise has to manage the communication with Data4Help and between the client app and the server. The interfaces needed are:

- Android OS, android Wear OS or iOS for the client-side application.
- Data4Help API to access to the services and receive data with Data4Help.
- REST API to communicate and transfer data on the web between the app and the server.
- interface with the Google Maps API to manage the map for the runs.

3.1.4 Communication Interfaces

The system is based on a Client-Server architecture, in particular the communication between the different actors and elements of the system:

- Data4Help Server and Data4Help API Client
- Third Parties/User and Data4Help
- AutomatedSOS Server and AutomatedSOS Client
- Track4Run Server and Track4Run Client.

The communication is based on:

- the REST API to make requests
- the JSON API to transfer data.

The REST standard is recommended because is largely used on the web from the most of the devices and it guarantees better portability, maintainability and compatibility.
Data4Help collects data from the users through its Client-side APIs which must be integrated in the third party software. Then the acquired data are provided to the Third Parties. Data4Help also takes on the responsibility of managing requests and permissions.

This is a possible configuration of the system:

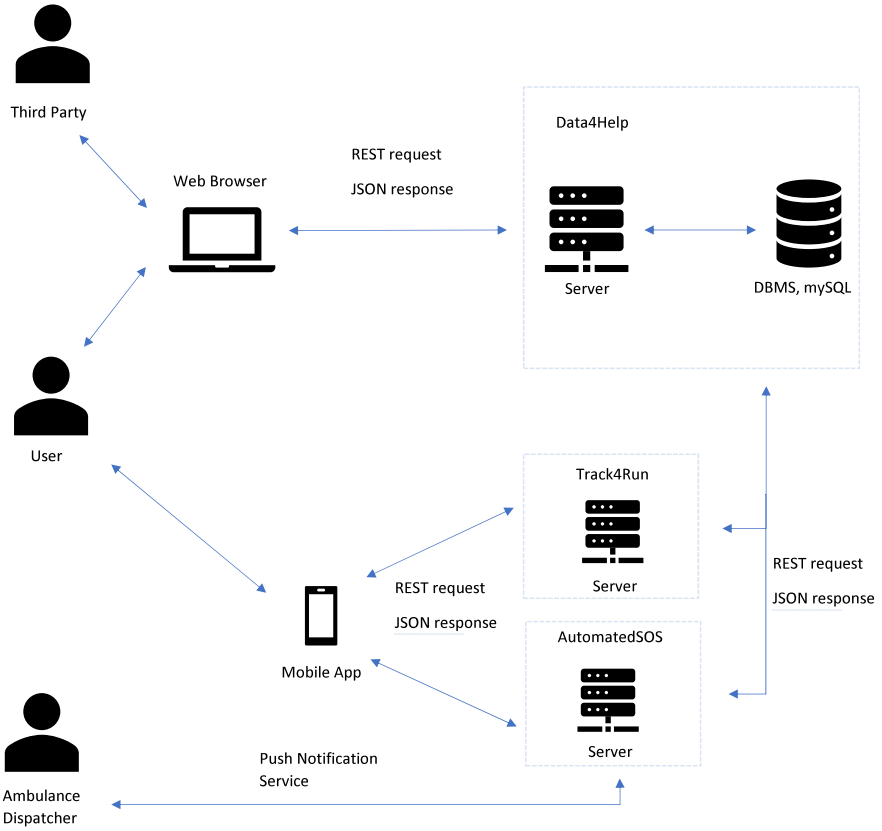


Figura 13: Communication schema of the whole system.

This is the schema of the communication by which data are collected and dispatched among third parties app, Data4help API, third parties server and Data4Help server services:

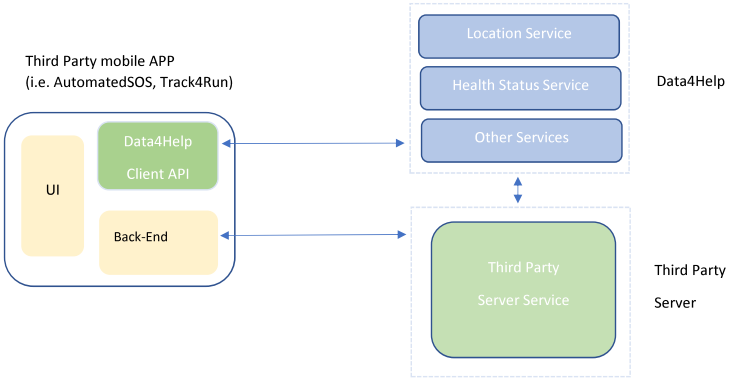


Figura 14: Data collection and trade of Data4Help.

3.2 Scenarios

3.2.1 Bran and the sign up

Bran, a 17-years old guy, likes athletics and, in particular, running, which usually performs with his friend Rickon.
During the last training, Rickon has shown to him Runnister, a new Android app, which explains some exercises for the warm up and monitors the users' health status to warn about possible damaging workouts. The app leans on a system which collects data, called Data4Help, and to which is necessary to be registered to.
Bran, enthusiastic, runs home and turns on the laptop, with the purpose to register to Data4Help. Enters in the homepage of the site and clicks the "Sign Up" button. Then completes the fields with his personal information, providing an email and the password. He also checks the terms of conditions, giving the consensus to the treatment of personal anonymized data. Finally, he finishes the registration and comes back automatically to the homepage.

3.2.2 Runnister's businnes plan

The Runnister company, owner of the namesake application for mobile devices, wants to enlarge its range of users and start to cover a wider area. For this reason needs Data4Help service which can provide information about health status parameters of its users in the Dothraki District.
First of all, the Runnister's agent accesses to the web site and logs in (Runnister has already used the service in the past and doesn't need to register anymore). He clicks on "Data Request" button and selects the one-time typology. Then selects the filter that must be applied to obtain data coming from a certain region (in this case "Dothraki District"). After that, he waits the confirmation to the request from Data4Help.
Unfortunately, users of Data4Help residing in "Dothraki District" aren't enough to satisfy the threshold which preserves the anonymity. The request is refused and Runnister's agent receives the notification of the refusal.

3.2.3 Breakfast with Starkbucks

Starkbucks is a company that makes statistics on the variations of health parameters generated from a non-healthy nutrition. For the next winter it has planned a analysis on the effects of the coffee on adult people's heartbeat and, to conduct this research, it needs to periodically acquire relevant data. Hence, it decides to ask Data4Help to get them.
After the company registration on the web site, the operator of Starkbucks, logged in the personal area, clicks on "Data Request" button and selects the acquirement of data through the subscription way. Then he chooses to pick users' data which have been collected only in the time slot between 7:00 and 9:00, i.e. the breakfast period. Then he waits for the confirmation. After some hour the operator receives the notification of confirmation and Starkbucks starts to periodically obtain the required data.
In March, at the end of the winter, Starkbucks has already collected enough data, so decides to unsubscribe from the service and stops to receive updates from Data4Help.

3.2.4 The worries of Brienne

Hospitarly is an institute which conducts researches on diseases and cancer, offering to its patients the possibility to remain always updated on their own health condition. To integrate user's information, the company takes more precise data from Data4Help, but it must have the consensus of the user.
Brienne is a middle-aged woman which has passed some sad moments for her health problems. For this reason, she wants to keep herself monitored and decides to use Hospitarly service. At the moment, she has already registered to Data4Help.
During the sign up to the service, she receives the proposal to allow Data4Help to share her personal data with Hospitarly, in order to permit a richer and more updated analysis. She accepts the request.
From this point, Hospitarly can ask whenever it needs information to Data4Help about Brienne's status and provide a more accurate service to its user.

3.2.5 The walk of uncle Jamie

Uncle Jamie has recently gone through a bad period, caused by his heart illness. To prevent another relapse, the doctor has advised him to get some walk everyday and install in his smartwatch an application, called AutomatedSOS, that provides an immediate rescue if his health parameters decrease under a given threshold. He listen to his doctor' advise and downloads the application.
After the registration to AutomatedSOS and the previous sign up to Data4Help, uncle Jamie decides to try the new software during a walk. He wears his smartwatch and look at the AutomatedSOS interface, which reports a regular health status, with a normal heartbeat. Pleased with that, he starts his walk.
Suddenly, in the middle of the walk, the smartwatch starts to flash and emit an alert noise. Uncle Jamie watches his clock and understands that he is having some health issue. Immedialtely, he begins to have blurry vision and he collapses on the ground.
The last sound he hears is a notification from his smartwatch that an ambulance has been sent to rescue him.

3.2.6 Forrest, the run planner

Forrest is a young man which loves running. He recently learns about a new application, called Track4Run, with the purpose to allow its users to organize run event. Excited for the new discovery, he invites a lot of friend to download the application and tries to set a run event for the next week.
After the registration to Track4Run and the previous sign up to Data4Help, he launches the app. First of all, he clicks on "Create a run" button, choosing starting and ending point and the path in the middle from the map. Then, he gives the day and the hour at which the run will start. Finally, he receives the notification of the event creation and the opening of the subscription period.
Now Forrest has only to enroll in his run event as a runner and wait until the day of the run.

3.2.7 Jon and his training

Jon is an autonomous athlete that has never joined in a sport association. To test himself he uses to access to Track4Run to find out some run events in which freely enroll. Now he wants to look for an event for the next week in South Carolina.
First of all, he launches Track4Run application from his phone and selects the "Enroll in a run" button. Then Jon selects the city, in this case Beaufort, and chooses the run event that fits better with his appointments. He finds the Forrest's run and decides to enroll himself in that run.
Jon is eventually registered as a runner in the event database and he will receive a notification just before the beginning of the event.

3.2.8 A runner as brother

To be updated on the placement of his brother Jon, Arya, which has already downloaded Track4Run on is device, decides to watch the run during is progression.
First of all, at the beginning of the run Arya launches Track4Run from her smartphone and selects the "View a run" button. Therefore, she indicates the city in which the run is held and chooses the desired run from the list of that is appeared. After that, on her phone appears the interface of the real-time map with the position of the runners, between which there is also Jon.
Arya watches the event until the end, being witness of his brother defeat.

3.3 Functional Requirements

3.3.1 Specific Requirements

G.1 Allow customers' registration.

D.2 The personal information given by the user (age, address, gender, etc.) and the third parties (name, address, etc.) are correct and the account is not used by other people.

G.1.1 Registration can be done as individual user.

R.1 People can create a user account selecting username, password, giving personal information (age, address,gender) and allow to share their anonymized data.

G.1.2 Registration can be done as third party.

R.2 It is possible to create a third party account selecting username and password and giving the company main information.

G.2 Third parties can receive anonymized data.

R.3 Data4Help allows third parties to request anonymized data acquired from a filtered group of users (by age, gender, address, etc.).

R.4 Data4Help collects data from registered users and gives access to third parties only if the number of individuals whose data satisfy the request is higher than 1000.

D.2 The personal information given by the user (age, address, gender...) and the third parties are correct and the account is not used by other people.

G.3 Third parties can subscribe to the service in order to periodically receive new data.

R.5 Data4Help allows third parties to join the subscription service for an indeterminate period and then, in case, unsubscribe from that.

R.6 Data4Help provides new data, checking each time if groups data satisfy the given constraint (number of individuals not lower than a thousand).

R.7 Data4Help keeps track of the associations between third parties subscriptions and the required group data.

G.4 Third parties can receive specific person's data.

R.8 Data4Help allows third parties to require specific person's data.

R.9 Data4Help forwards requests from third parties to the demanded users which can accept or refuse to share their own personal data.

R.10 Data4Help keeps track of the connections between a specific user and the third parties which can access to his/her data.

R.11 Data4Help allows third parties to have access to demanded users' data each time they need them.

G.5 Users' health status is continuously checked in AutomatedSOS and if it is below a certain threshold values an ambulance is called within 5 seconds and sent to the user's position.

D.1 The data collected from the devices (position and health status) are reliable, accurate and in real time.

R.12 Users' health and position information received from Data4Help data collector process are analyzed and compared with the threshold values.

R.13 In case of emergency (the health status values overcome the threshold) a request for an ambulance is sent to the ambulance dispatcher in 5 seconds, containing the user's information.

D.3 Ambulance call and information about patients' status and position are correctly dispatched to the ambulance contact center.

G.6 Users can organize a run.

R.14 Users select the day, the hour at which the run begins, the starting point, the ending point and the path for the run.

R.15 The run event is stored in the system in order to be managed during its lifecycle.

D.4 Information and data (places, maps, paths, etc.) received from external services are correct and reliable.

G.7 Users can organize a run.

R.16 Users can browse among the available runs and see their information.

R.17 Users can choose a run and register to it as a runners.

R.18 The system saves the enrolled users as runners and keeps track of the association with the run event.

G.8 Users can be spectators of a run seeing the participants' position on a map.

R.14 The system keeps track of all the enrolled runners' position during the run.

R.15 Users who require to watch a run are saved as spectators to the run event.

R.16 Track4Run offers the possibility to see the run live through a map.

D.4 Information and data (places, maps, paths, etc.) received from external services are correct and reliable.

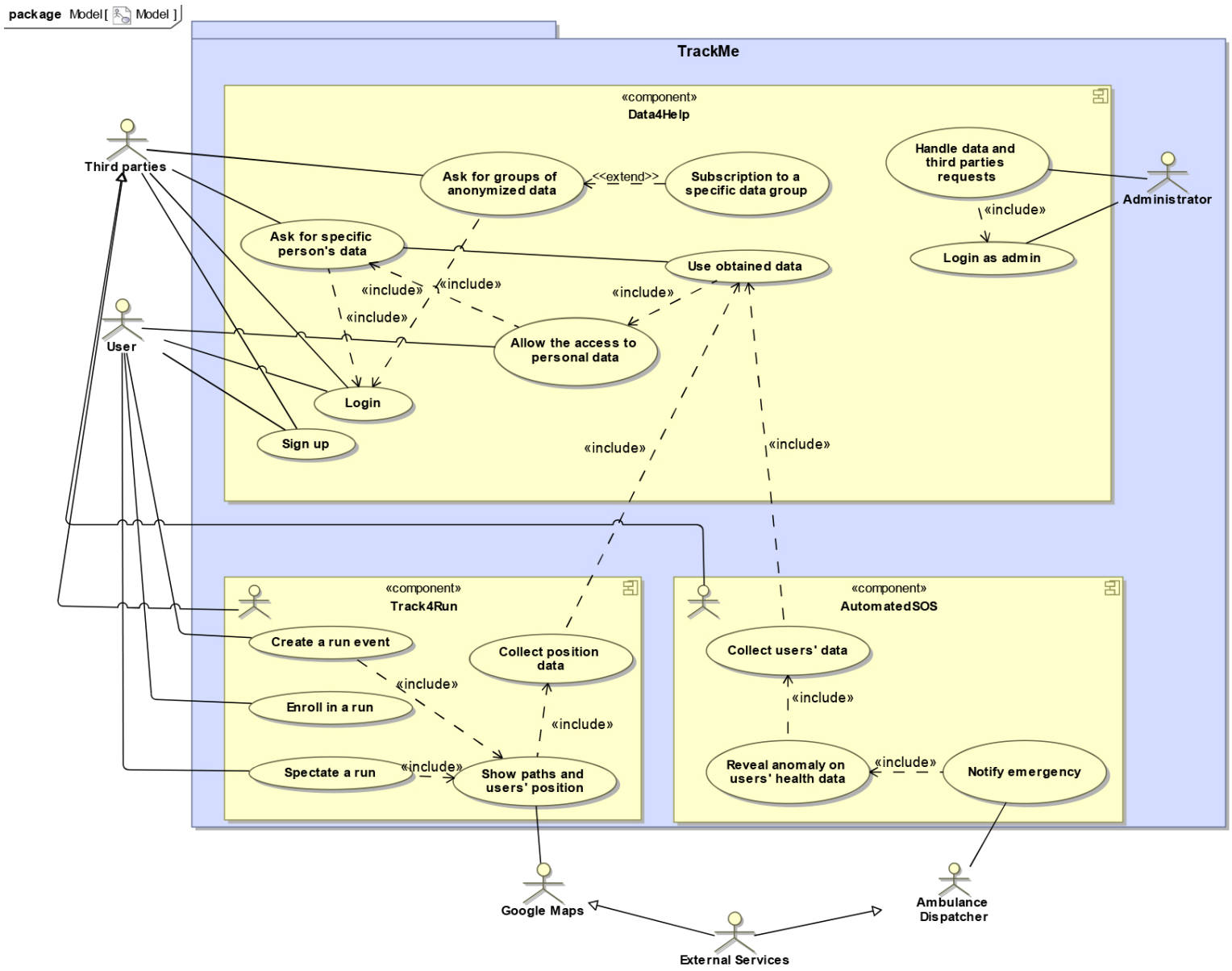


Figura 15: Use Case Diagram of the whole system.

Use Cases

3.3.2 User's sign up

ACTORS	User
ENTRY CONDITIONS	No entry condition needed
EVENTS FLOW	1. User opens Data4Help web page and press "Sign Up" button. 2. User inserts personal information (name, address, age, gender, fiscal code). 3. User selects email and password. 4. User allows terms of conditions to share anonymized data with Data4Help. 5. User completes the registration. 6. The system saves the new account.
EXIT CONDITIONS	The user is successfully registered and his/her data can be collected by Data4Help. He/she is redirected to the homepage.
EXCEPTIONS	1. The user doesn't fill all the mandatory fields with valid data. 2. The inserted email is already used by another user. 3. The user doesn't allow the term of condition and he/she can't complete the registration.

3.3.3 One-time request for anonymous data

ACTORS	Third Party
ENTRY CONDITIONS	This use case starts when the third party is registered, successfully logged in to Data4Help and is connected to the web page.
EVENTS FLOW	1. The Third Party click on "Data Request" button. 2. The Third Party selects to require anonymous data with a one-time request. 3. The Third Party selects the filter that must be applied to gather users. 4. The system checks if the request can be accepted to guarantee the users' anonymity or not. 5. Third party receives the confirmation to the request.
EXIT CONDITIONS	The request is accepted and anonymized data are sent to the Third Party.
EXCEPTIONS	The number of users satisfying the filter is smaller than 1000, then the request is refused.

3.3.4 Subscription to anonymous data set

ACTORS	Third Party
ENTRY CONDITIONS	This use case starts when the third party is registered, successfully logged in to Data4Help and connected to the web page.
EVENTS FLOW	1. The Third Party clicks on "Data Request" button. 2. The Third Party selects to require anonymous data with a subscription request. 3. The Third Party selects the filter that must be applied to gather users. 4. The system checks if the request can be accepted to guarantee the users' anonymity or not. 5. The Third Party receives the confirmation to the request. 6. The Third Party is periodically notified with new anonymous data about the selected users' group.
EXIT CONDITIONS	The Third Party decides to unsubscribe from the service and stops to receives periodical updates.
EXCEPTIONS	1. The number of users satisfying the filter is smaller than 1000, then the request is refused. 2. During the service supply the number of users in the group decreases under 1000 entities, so the service is stopped.

3.3.5 Request for specific user's data

ACTORS	Third Party, User
ENTRY CONDITIONS	This use case starts when the Third Party is registered, successfully logged in to Data4Help and connected to the web page.
EVENTS FLOW	1. The Third Party clicks on "Data Request" button. 2. The Third Party selects the option "Require Individual Data". 3. The system sends a request to the users whose the Third Party needs data. 4. Once the user has allowed to acquire his/her personal data, a confirmation to the request is sent to the Third Party.
EXIT CONDITIONS	The Third Party can require and access to user's personal data.
EXCEPTIONS	The user doesn't allow to share personal data and the system rejects the request of the Third Party.

3.3.6 Alert emergency status

ACTORS	User, Ambulance Dispatcher
ENTRY CONDITIONS	This use case starts when the user is registered and logged in to AutomatedSOS on his/her device, moreover has allowed the permission to share personal data with this third party.
EVENTS FLOW	1. The system receives the user's health data collected from his/her device by Data4Help. 2. The system analyzes the data and notices that the health status is under the safety threshold. 3. The system notifies the emergency to the Ambulance Dispatcher within 5 seconds, reporting the user's health status and position. 4. The system sends a warning to the user's device to notify that the health status is under the threshold and that an ambulance has been called to reach his/her position. 5. The system receives the acknowledgement from the Ambulance Dispatcher which notifies that an ambulance has been sent. 6. The system notices that the health status has became regular again.
EXIT CONDITIONS	The system keeps monitoring the user's health status and position.
EXCEPTIONS	The Ambulance Dispatcher doesn't send any confirmation and the system continues to periodically notify it with the alert status until a confirmation is received.

3.3.7 Creation of a run event

ACTORS	User
ENTRY CONDITIONS	This use case starts when the user is registered and logged in to Track4Run on his/her device, moreover has allowed the permission to share personal data with this third party.
EVENTS FLOW	1. The user selects the "Create a run" button. 2. The user chooses the starting point, the ending point and the path on the map. 3. The user selects the day and the starting time of the run event. 4. The system confirms the run event creation to the user and saves it in the database.
EXIT CONDITIONS	The run event is created and the track4Run users can enroll to the event.
EXCEPTIONS	The selected path or the starting/ending point is unavailable for a run. The system lets the user choose an available one.

3.3.8 Enrollment in a run

ACTORS	User
ENTRY CONDITIONS	This use case starts when the user is registered and logged in to Track4Run on his/her device. Moreover has allowed the permission to share personal data with this third party and at least a run event has been already created.
EVENTS FLOW	1. The user selects the "Enroll in a run" button. 2. The user selects the city that interests him/her. 3. The system shows the list of available runs in the selected city. 4. The user selects the run event he/she wants to enroll. 5. The system saves the user in the run event as a runner. 6. The system notifies the user when the run is going to start.
EXIT CONDITIONS	The user is correctly enrolled to the run event and can take part to it.
EXCEPTIONS	There are no available runs.

3.3.9 Watch a run

ACTORS	User
ENTRY CONDITIONS	This use case starts when the user is registered and logged in to Track4Run on his/her device, on the main page. Moreover has allowed the permission to share personal data with this third party and at least a run event has been already created.
EVENTS FLOW	1. The user selects the "View a run" button. 2. The user selects the city that interests him/her. 3. The system shows the list of available runs in the selected city. 4. The user selects the run event he/she wants to watch. 5. The system saves the user in the run event as a spectator.
EXIT CONDITIONS	The user can watch on his/her device the map with the path and the runners positions updated in real time.
EXCEPTIONS	

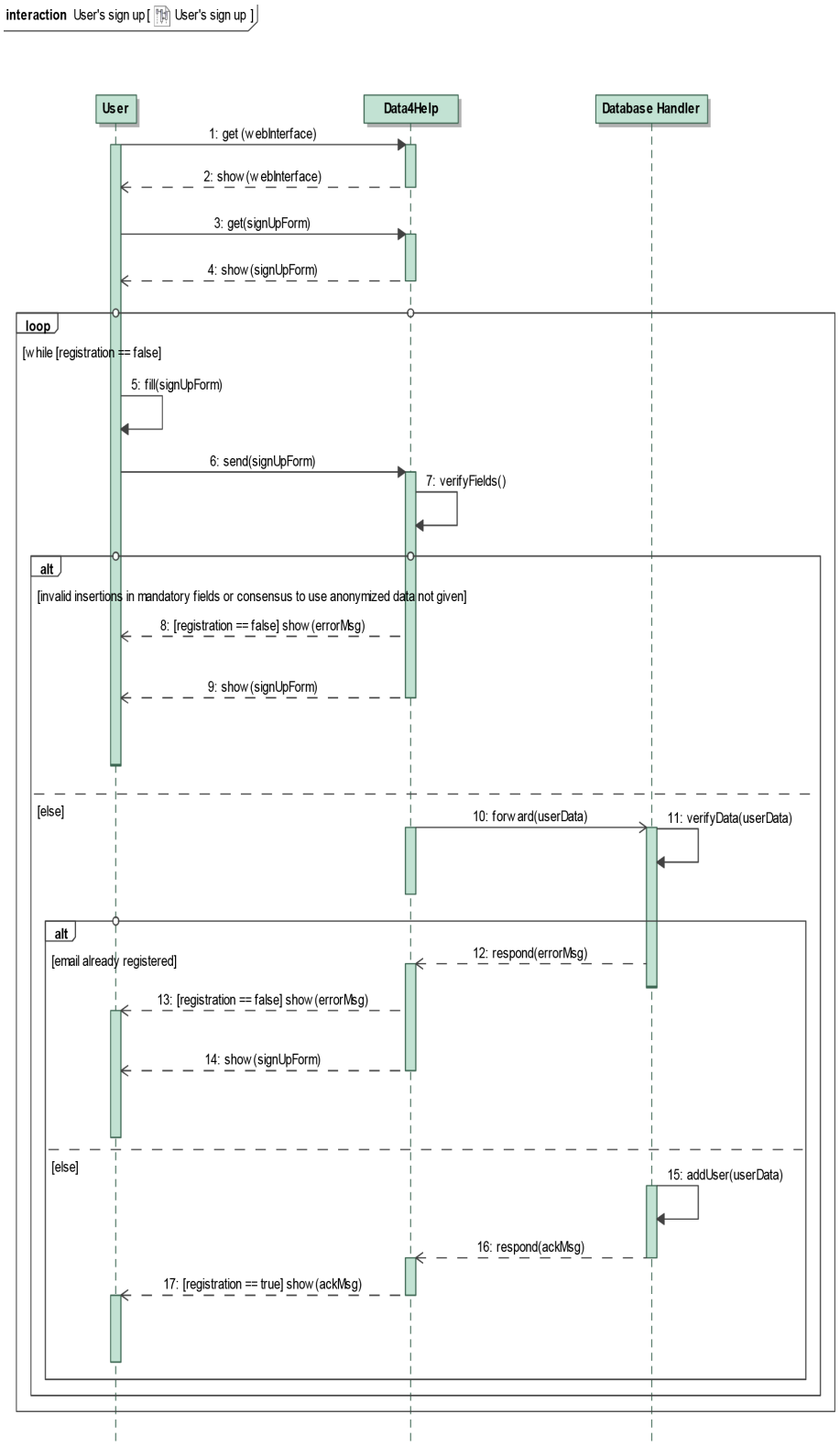


Figura 16: Sequence diagram of a user's sign up.

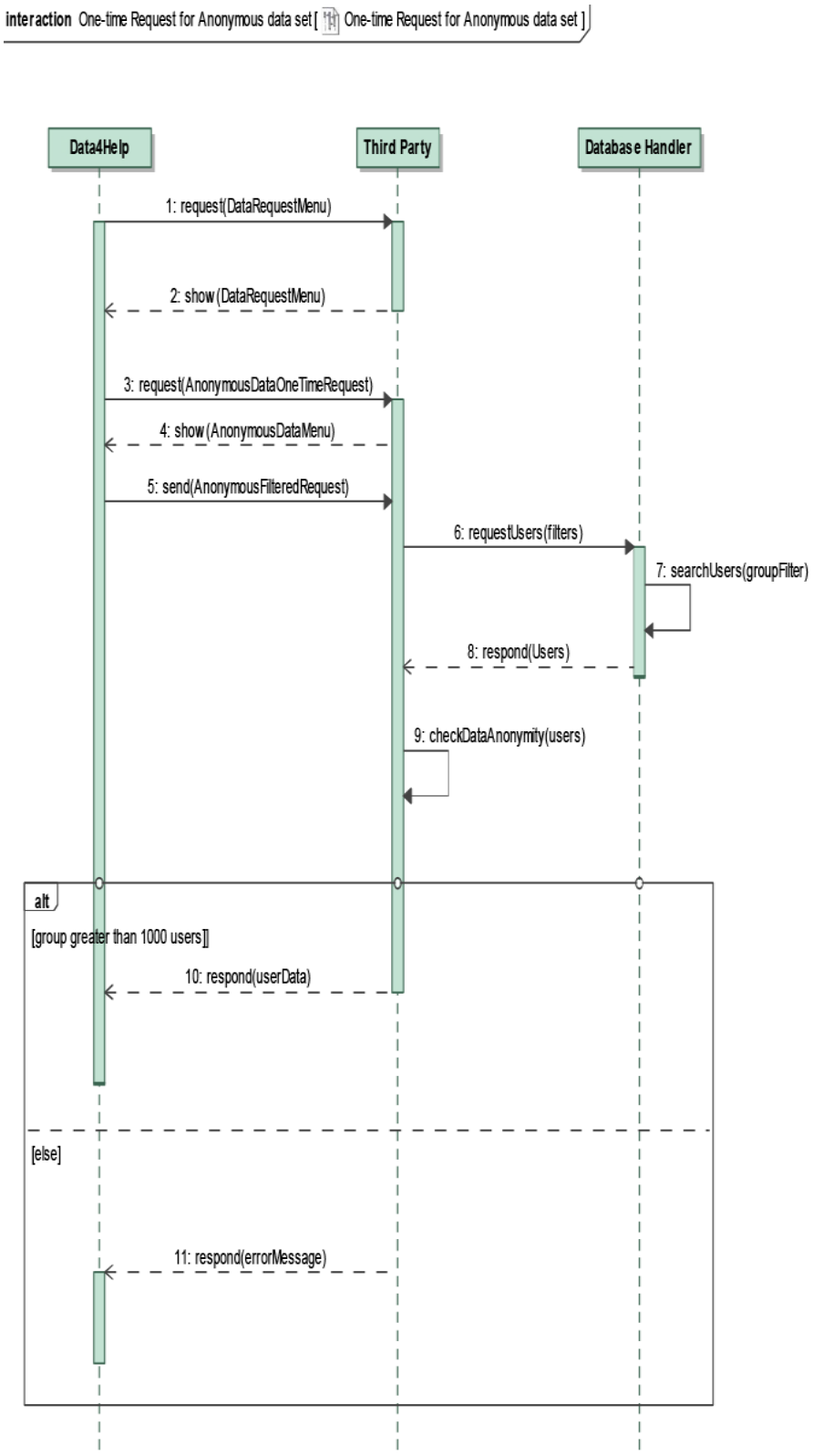


Figura 17: Sequence diagram of one-time anonymous data request.

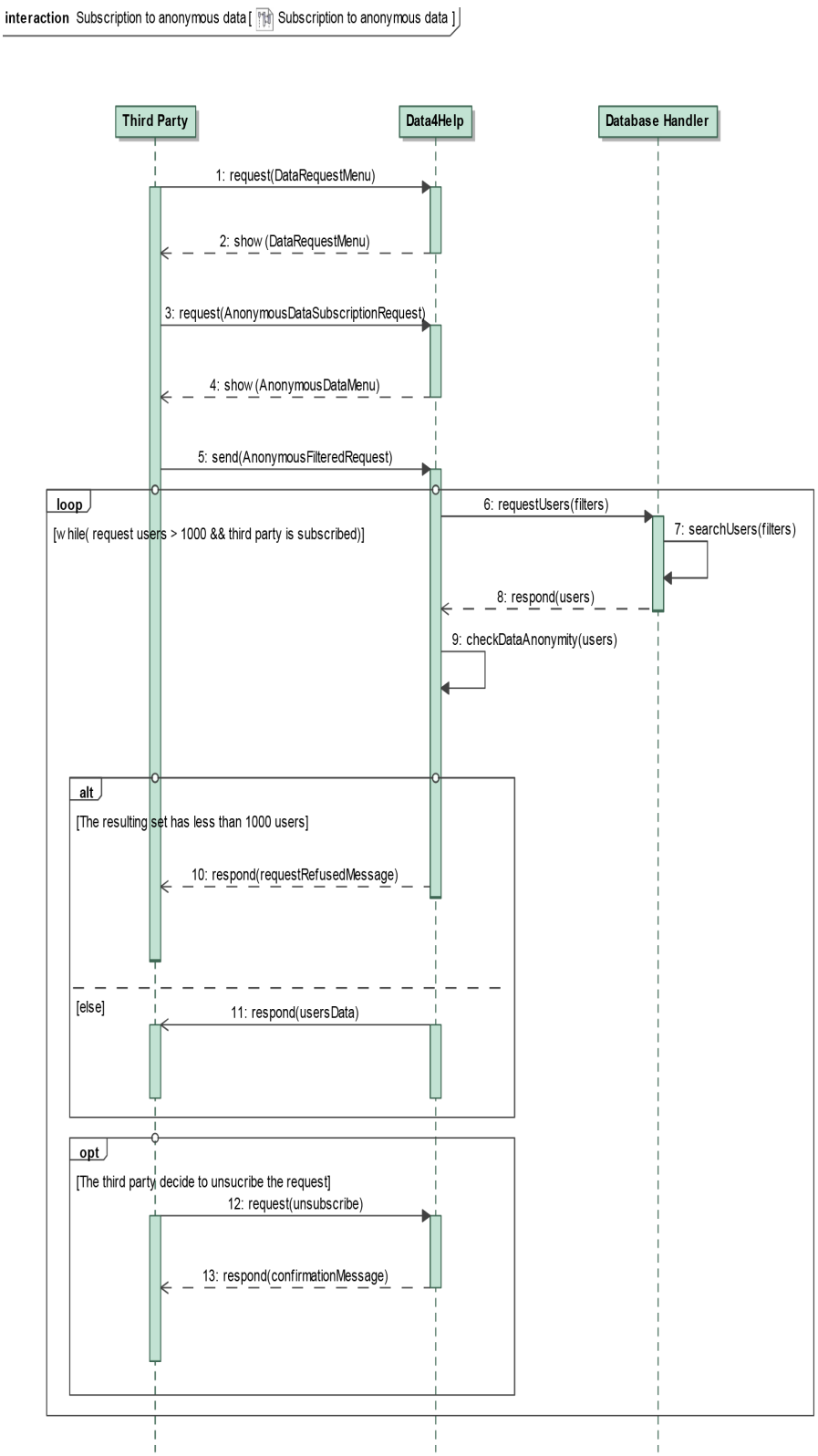


Figura 18: Sequence diagram of a subscription to anonymous group data.

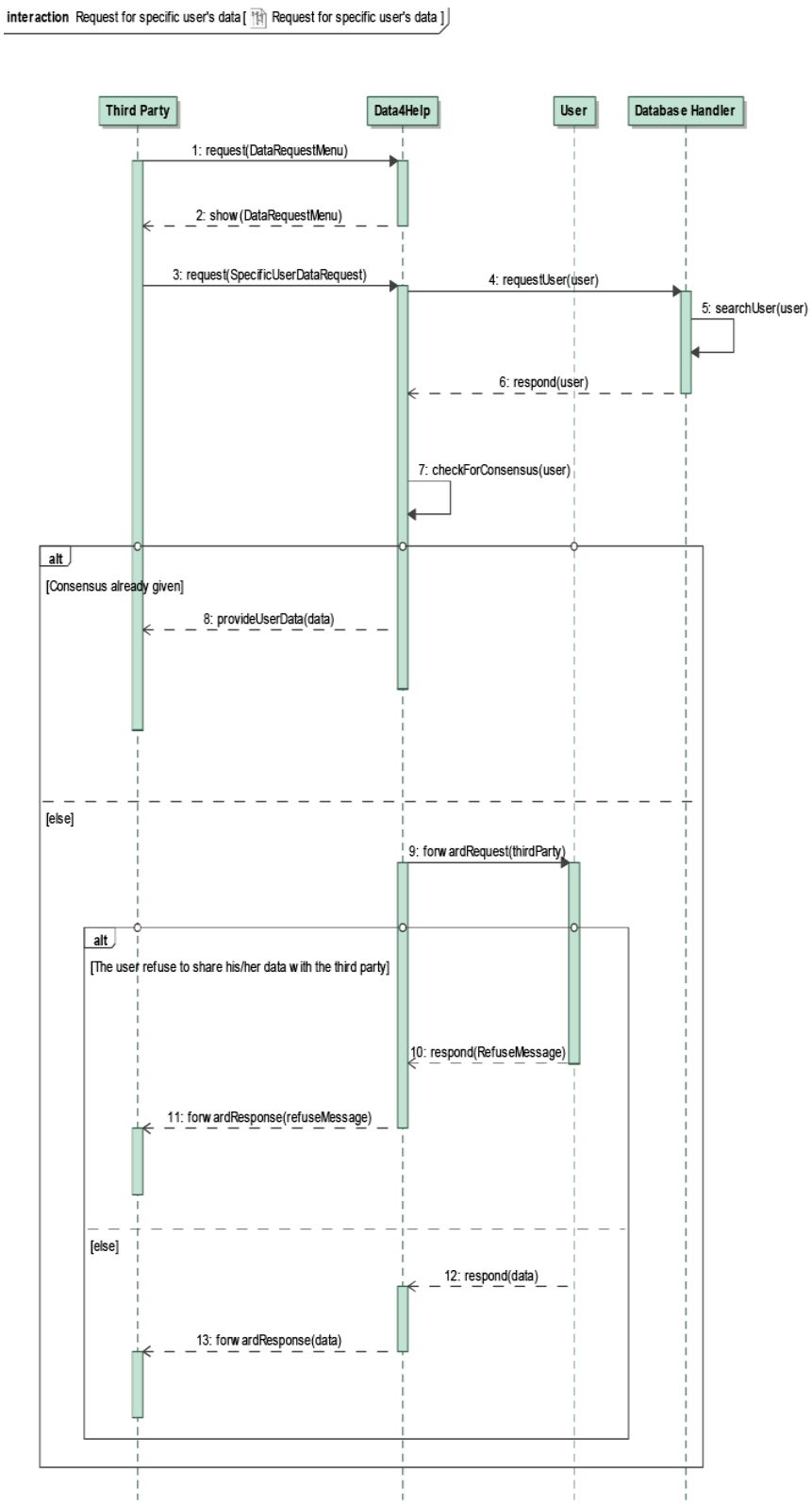


Figura 19: Sequence diagram of a user's personal data request .

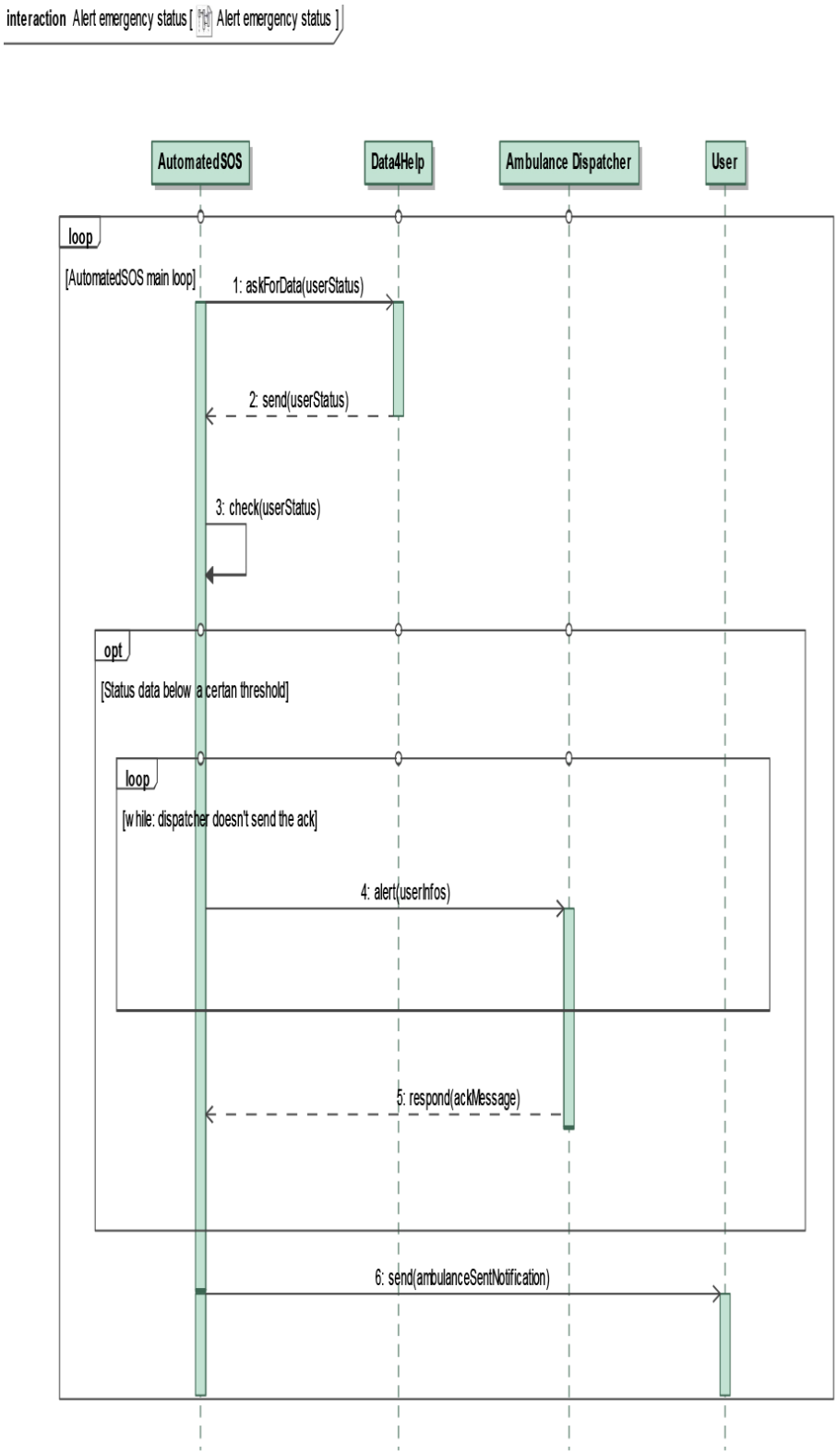


Figura 20: Sequence diagram of an emergency alert event.

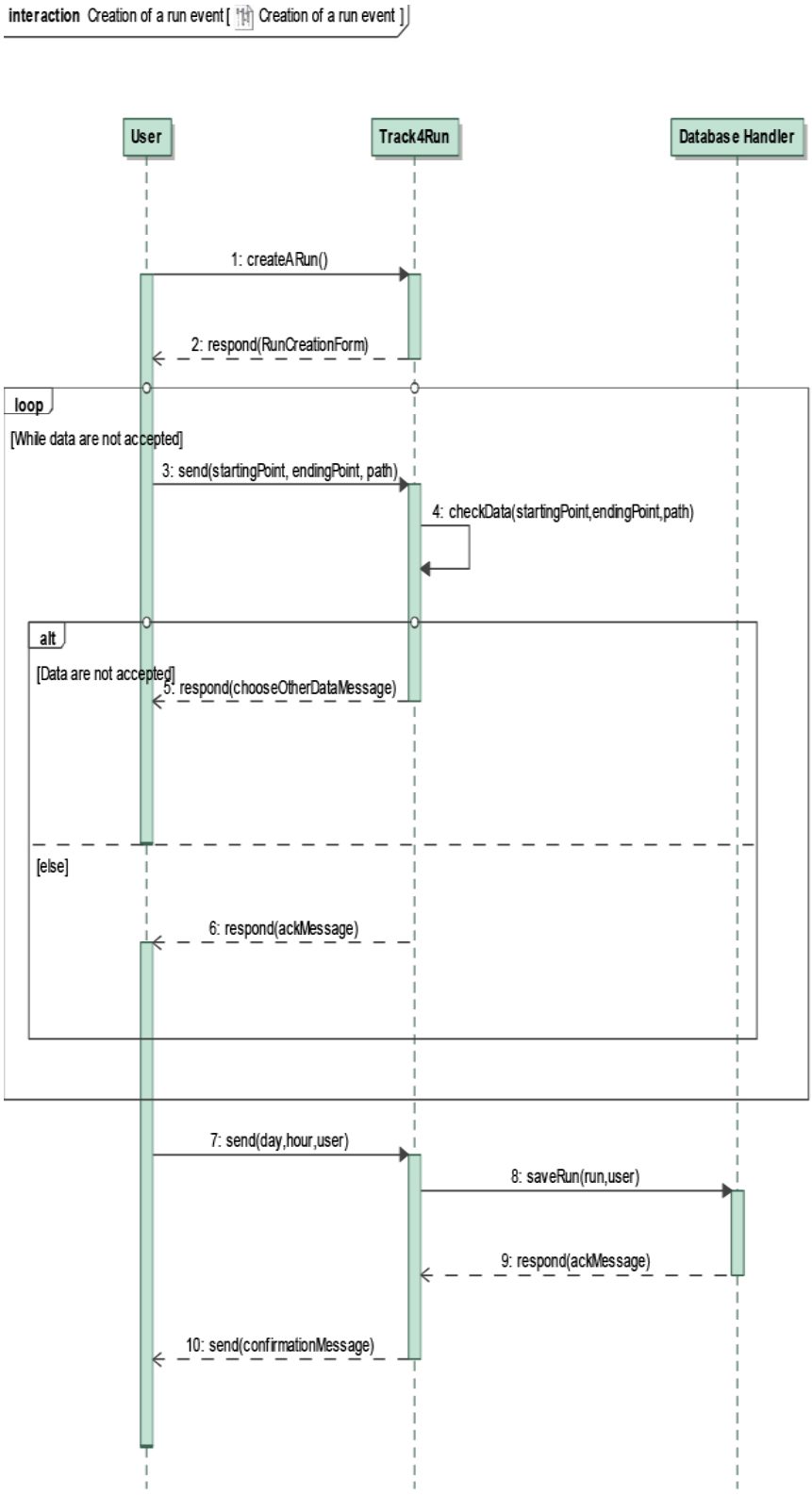


Figura 21: Sequence diagram of the creation of a run event.

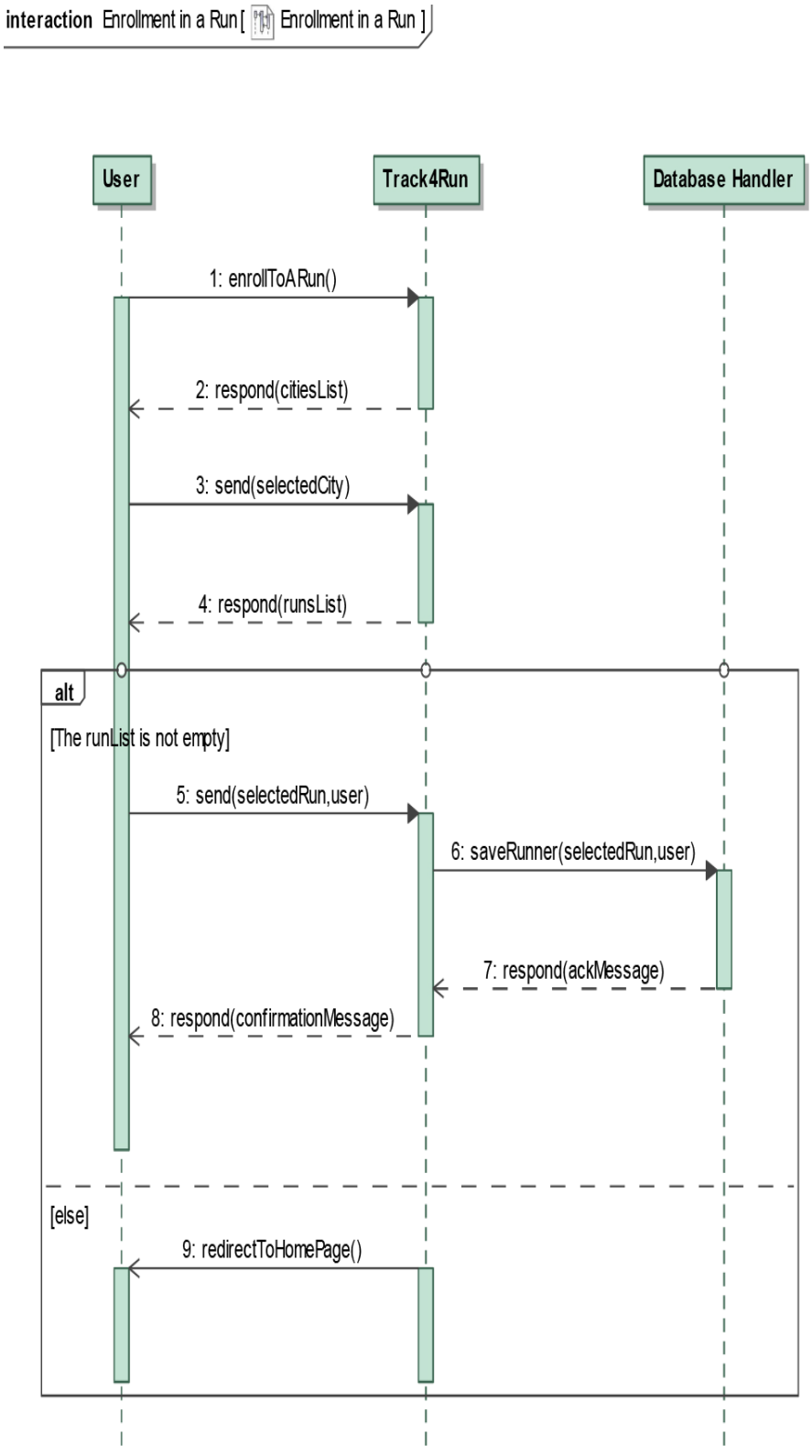


Figura 22: Sequence diagram of the enrollment in a run.

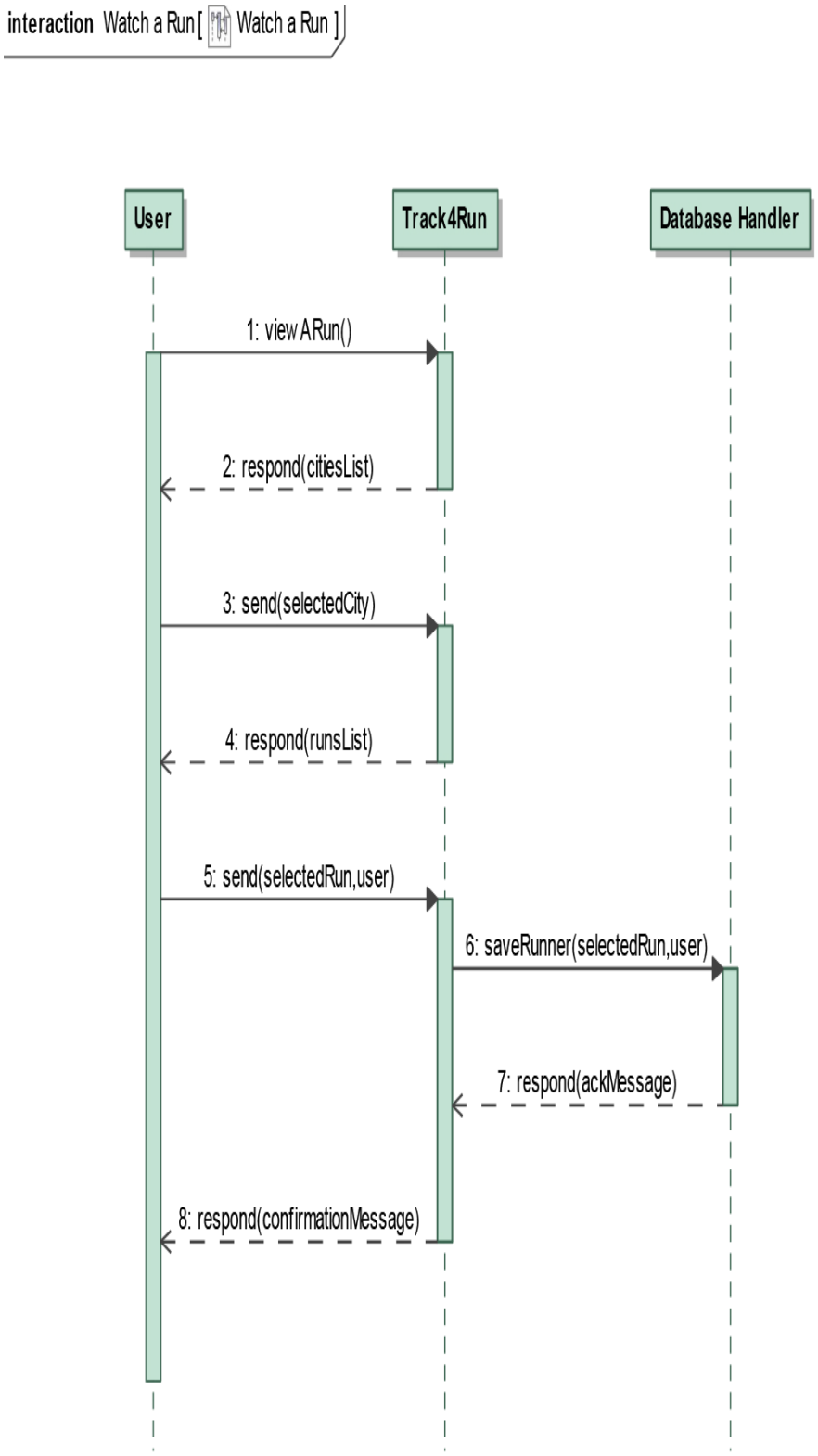


Figura 23: Sequence diagram of the viewing of a run.

3.4 Performance Requirements

Data4Help The system is designed to provide a large number of users and third parties moreover it is necessary to guarantee that the data is acquired, analyzed and made available to third parties in reasonable times. Especially individual data has to be provided to the third parties as soon they require them (periodically or even with real time updates).

AutomatedSOS This is a non-intrusive SOS service that needs to monitor the user health status in every moment. Once user's data are received from Data4Help the system must analyze the values and compare them with safe thresholds in a short time. If the values are under the fixed thresholds the system must warn the ambulance contact center in 5 seconds to send an ambulance to the user location.

Track4Run the system must be able to manage a large number of users enrolled in a single run event (the number of runners for a single run can be between less then 10 and almost 1000), a large number of run events for several cities and it must guarantee that the spectators can watch the runners' position on the map with a real time update.

3.5 Design Constraints

3.5.1 Standards Compliance

Individual people to get access to Data4Help, AutomatedSOS and Track4Run services must sign in to Data4Help. By registering to Data4Help the user must allow the permissions to share anonymazed data with Data4Help and third parties, unless he/she can't complete the registration. AutomatedSOS and Track4Run are third parties actors developed by TrackMe and, as such, they have to require to Data4Help the permission to receive individual data from the users. If an user wants to use AutomatedSOS and Track4Run services must accept the permissions to share individual data with them, unless their data can't be dispatched from Data4Help to the specific third party and the third party service can't be provided.

3.5.2 Hardware Limitations

Data4Help doesn't require specific hardware limitations for third parties. Users should use mobile devices with sensors useful for collecting data (GPS, cardiac pulse sensor, accelerometer, camera, light sensor). Even if they are not all strictly required, there might be restrictions dictated by the service that has to be provided from the third party. The only necessary requirement is that the device must have an internet connection (possibly fast):

- 3G/4G internet connection

The hardware requirements for **AutomatedSOS** users' device, to collect the health information are:

- 3G/4G internet connection
- GPS
- Cardiac pulse sensor, other health sensors

The hardware requirements for **Track4Run** users' devices are:

- 3G/4G internet connection
- GPS
- Device that supports Google maps API

3.6 Software System Attributes

3.6.1 Reliability

Data4Help The system must be able to collect data from the users and guarantee the services to the third parties. Third parties can request users' data with arbitrary frequency, also in "real-time". The system to guarantee a good and trusty service should not have failure and long period of time in which the functions are not provided to customers, specially for individual data requests. Infrequent short periods of time of unavailability may be tolerated (at most a few ongoing minutes).

AutomatedSOS The service must acquire users' data from Data4Help and monitor the health status in real time (24h) to guarantee the health care in case of emergency. The system should not have failure that lasts for long period of time (it can be tolerated at most for a few minutes) and the status analysis, once the data are received from Data4Help, must be performed and completed in very short time to guarantee the time consistency of the health status.

Track4Run The service must be guaranteed to the users during the entire day, letting them to manage and enroll to runs. In particular, in order to guarantee a good quality of service, during the runs it is needed an high level of location accuracy. The system should not have long period of unavailability especially in the day time and in the evening, the periods with the higher number of customers connected.

3.6.2 Availability

Data4Help The system must have an high availability to guarantee all the functions. Using the "hour-per-week" (a week is composed by 168 hours) the system should have an availability greater or equal than 165/168 or using the "nines" metric the system should have an availability greater than 1 nine ($>$ 0.9).

AutomatedSOS Also the AutomatedSOS system must have an high level of availability, to provide a better service is preferable an availability of two nines (0.99) in the "nines" metric.

Track4run The system should offer their service for the runners with an availability at least of 150/168, with the priority of availabilty for the the day time and evening hours.

3.6.3 Security

In all the software offered by TrackMe it is necessary to guarantee an high level of security, in order to respect the sensibility of the users' data. All the communications must be encrypted and the information should be sent on secure channels. Furthermore the data stored in the Data4Help database must be accessible only from entities which have the necessary permissions. Users and third parties can access to the services providing their credentials (username, password).

3.6.4 Maintainability

A good level of maintainability, since it is approximately 75% of the cost related to a project, should be guaranteed for the development of the system, so it has to be corrective, adaptive, perfective, and preventative and must be avoided poor code quality, source code defects, excessive technical complexity and poorly documented code.

3.6.5 Portability

The system should have an high level of portability in order that TrackMe can guarantee its services to a wide range of devices and users. Providing a set of API instead of creating a dedicated app for Data4Help is useful to achieve this goal, in fact the third parties can choose different OS and devices for developing their own services and application, integrating the Data4Help service with them.

4 Formal Analysis Using Alloy

In the Alloy model only the relevant elements for the system behaviour analysis are present. The model focuses, in particularly, on these entities:

- User
- Group of Users (used in anonymized data requests)
- Third Party
- AutomatedSOS
- Track4Run
- Request Status
- Individual permission
- Anonymous permission

In general customer's data and personal information are not considered because they don't affect the system behaviour except only for the user's health status and the time at which a run begins.
The most important aspect of the analysis are the permissions because they represent the relation between users and third parties which is the most sensible part of the service. The subscription to an anonymous data request is not modeled because it must respect the same constraints of the one time anonymous permission, they only are valid for a longer period of time.
In fact, Data4Help aims to collect data, but, most importantly, gives access to them only to the authorized entities.
As concern AutomatedSOS, the most relevant part that requires a formal modelization is the distinction between the emergency status and the safety one.
Instead, Track4Run doesn't present some major constraints that need to be modeled. Indeed, we have only paid attention to the run structure and its limitations.
To model the permission evolution the system uses three states: Accepted, Refused and Undefined.
The request status Undefined is used to represent a permission that has been already created and it has not been approved yet.

```
  open util/relation
open util/integer

sig User {
dataFor:  set ThirdParty,
health:one Int
} { health >=0 and health <= 4}

sig GroupOfUsers {
targetUsers:  set User
}{#targetUsers > 0}

sig ThirdParty {
individualDataFrom:  set User,
anonymizedDataFrom:  set GroupOfUsers
}

one sig AutomatedSOS extends ThirdParty {
emergency:  set User
}

one sig Track4Run extends ThirdParty {
runs:  set Run
}

sig Run{
runners:  some User,
spectators:  set User,
time:one Int
}{(all r:  Run | r in Track4Run.runs) and time >= 0 and time
<= 6 }


abstract sig RequestStatus {}
one sig ACCEPTED extends RequestStatus {}
one sig REFUSED extends RequestStatus {}
one sig UNDEFINED extends RequestStatus{}

sig IndividualPermission {
thirdParty:  one ThirdParty,
user:  one User,
status:  one RequestStatus
}

sig AnonymousPermission {
thirdParty:  one ThirdParty,
group:  one GroupOfUsers,
status:  one RequestStatus
}
```

```
-Data4Help

fact individualDataRequestCondition {
all ip:  IndividualPermission | ip.status = ACCEPTED iff
(ip.user in ip.thirdParty.individualDataFrom and
ip.thirdParty in ip.user.dataFor )
}

fact toAcquireIndividualDataMustExistsAPermission{
all u:  User, tp:  ThirdParty | u in tp.individualDataFrom
iff
( one ip :  IndividualPermission | ip.user = u and
ip.thirdParty = tp and ip.status = ACCEPTED )
}

fact oneToOnePermissionCondition{
all u:  User, tp:  ThirdParty | u in tp.individualDataFrom
iff tp in u.dataFor
}

fact notAcceptedPermissionAvoidDataExchange{
all ip:  IndividualPermission | (ip.status = REFUSED or ip.status
= UNDEFINED) iff !( ip.user in ip.thirdParty.individualDataFrom)
}

fact individualPermissionAreUnique{
no disj ip1, ip2 :  IndividualPermission | ip1.user = ip2.user
and ip1.thirdParty = ip2.thirdParty and !(ip1.status = UNDEFINED
or ip2.status = UNDEFINED)
}

fact anonymazedPermissionCondition {
all ap:  AnonymousPermission | (ap.status = ACCEPTED iff
#ap.group.targetUsers > 1000) and ( ap.status = REFUSED iff
#ap.group.targetUsers <= 1000 ) and !(ap.status = UNDEFINED)
}

fact groupExistsOnlyForAnonymazedPermissions {
all gu:  GroupOfUsers | some ap:  AnonymousPermission | ap.group
= gu
}

fact toAcquireAnonymazedDataMustExistAPermission{
all gu:  GroupOfUsers, tp :  ThirdParty | gu in
tp.anonymizedDataFrom iff ( some ap:  AnonymousPermission |
ap.group = gu and
ap.thirdParty = tp and ap.status = ACCEPTED )
}

fact anonymazedPermissionsAreUnique {
```

```
no disj ap1,ap2:  AnonymousPermission |
ap1.thirdParty = ap2.thirdParty and
ap1.group = ap2.group
}

fact usersInGroupAreUnique {
all gu:  GroupOfUsers | ( no disj u1,u2:  User | u1 in
gu.targetUsers and u2 in gu.targetUsers and u1 = u2 )
}

pred notExistsAnAcceptedIndividualPermission[u:User,tp:ThirdParty]
{
!(u in tp.individualDataFrom)
}

pred addUserToThirdParty[tp,tp':ThirdParty,u:User] {
tp'.individualDataFrom = tp.individualDataFrom + u
}

pred addThirdPartyToUser[u,u':User, tp:ThirdParty] {
u'.dataFor = u.dataFor + tp
}

pred addIndividualPermission[ u,u' :  User, tp,tp' :  ThirdParty]
{

notExistsAnAcceptedIndividualPermission[u, tp]

addUserToThirdParty[tp,tp',u]

one ip:IndividualPermission | ip.status = ACCEPTED and ip.user
= u' and ip.thirdParty = tp'
}

-AutomatedSOS

fact userInEmergencyCondition {
all u:AutomatedSOS.individualDataFrom | (u.health < 3 iff u
in AutomatedSOS.emergency)
}

fact onlyAutomatedSOSUserCouldBeInEmergency {
AutomatedSOS.emergency in AutomatedSOS.individualDataFrom
}

pred isUserSafe[u:User] {
!(u.health < 3)
}

pred userIsNotSafe[u,u':User] {
```

```
u'.health = minus[u.health,2]
}

pred userEnterInEmergencySituation[u,u':User] {
u in AutomatedSOS.individualDataFrom
u' in AutomatedSOS.individualDataFrom
isUserSafe[u]
!(u in AutomatedSOS.emergency)

userIsNotSafe[u,u']
(u' in AutomatedSOS.emergency)
}

-Track4Run

fact noAnonymousPermissionAutomatedTrack4Run{
no ap:AnonymousPermission | ap.thirdParty = AutomatedSOS or
ap.thirdParty = Track4Run
}

fact noSameUserIsRunnerAndSpectatorsInSameRun{
all r:  Run | no u:  User | u in r.runners and u in r.spectators
}

fact onlyTrack4RunUserCanBeRunnerOrSpectators{
all u:  User | (some r:  Run | (u in r.runners or u in
r.spectators ) ) implies u in Track4Run.individualDataFrom
}

fact noUserInTwoRunEventsAt{
no u:User | some disj r1,r2:Run | (u in r1.runners or u in
r1.spectators) and (u in r2.runners or u in r2.spectators)
and r1.time = r2.time
}

pred addRunner[r,r':Run, ru:User]{
r'.runners = r.runners + ru
}

pred addSpectator[r,r':Run, s:User]{
r'.spectators = r.spectators + s
}

pred addSpectatorAndRunnerToARun[r,r':Run, s,s',ru,ru':User]{
!(s in r.spectators or ru in r.runners)
addRunner[r,r',ru]
addSpectator[r,r',s]

!(ru in r'.spectators or s in r'.runners)
}
```

```
assert noRunnerInTwoRunAtSameTime{
all u:User | all disj r1,r2:Run | !(u in r1.runners and u in
r2.runners and r1.time = r2.time)
}


pred show {
some u:User| u in AutomatedSOS.emergency
some ip:IndividualPermission | ip.status = ACCEPTED
some ap:AnonymousPermission |ap.status = ACCEPTED
some r:Run | #r.runners>0 and #r.runners<3 and #r.spectators>0
and #r.spectators<3
}

run show for 5 but 4 AnonymousPermission, 4
IndividualPermission, 3 ThirdParty,5 User
run addIndividualPermission
run userEnterInEmergencySituation
run addSpectatorAndRunnerToARun
check noRunnerInTwoRunAtSameTime for 5
```

Executing "Run show for 5 but 4 AnonymousPermission, 4 IndividualPermission, 3 ThirdParty, 5 User"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
6956 vars. 440 primary vars. 16996 clauses. 16ms.
Instance found. Predicate is consistent. 35ms.

Executing "Run addIndividualPermission"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3440 vars. 247 primary vars. 8289 clauses. 11ms.
Instance found. Predicate is consistent. 20ms.

Executing "Run userEnterInEmergencySituation"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3678 vars. 241 primary vars. 9180 clauses. 19ms.
Instance found. Predicate is consistent. 30ms.

```
Executing "Run addSpectatorAndRunnerToARun"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3461 vars. 253 primary vars. 8265 clauses. 17ms.
  Instance found. Predicate is consistent. 62ms.
```

```
Executing "Check noRunnerInTwoRunAtSameTime for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  8968 vars. 518 primary vars. 21039 clauses. 24ms.
  No counterexample found. Assertion may be valid. 16ms.
```

*In the dynamic analysis the pred *show* is run with the group of user constraints for the anonymous permission set to 2 persons instead of 1000.

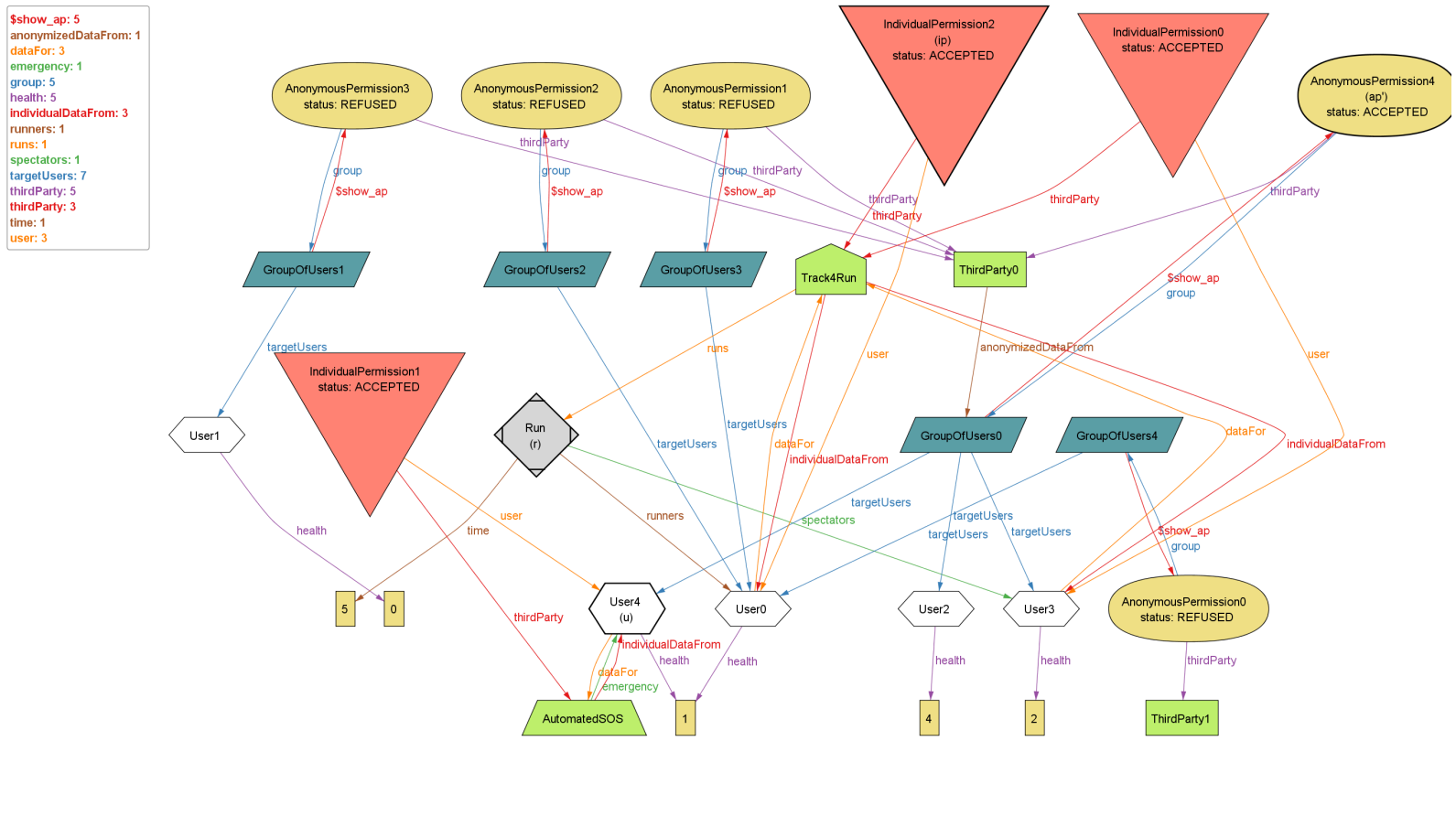


Figura 24: World created.

5 Effort Spent

Davide Salaorni

Purpose, Scope, Definitions	1
Product Perspective and Functions, Assumptions,Scenarios	4
External Interfaces	3
Functional Requirements	9
Design Constraint	3
Software System Attributes	1
Alloy	9

Luca Terracciano

Purpose, Scope, Definitions	2
Product Perspective and Functions, Assumptions,Scenarios	4
External Interfaces	4
Functional Requirements	9
Design Constraint	2
Software System Attributes	3
Alloy	9

Manuel Trivilino

Purpose, Scope, Definitions	2
Product Perspective and Functions, Assumptions,Scenarios	2
External Interfaces	5
Functional Requirements	9
Design Constraint	3
Software System Attributes	3
Alloy	9

6 References

6.0.1 Used Tool

- MagicDraw (diagrams)
- Alloy Analyzer 5.0
- Overleaf LaTex Editor
- gitHub
- Proto.io (mockups)
- Adobe Photoshop
- Visual Studio Code

6.0.2 Documentation and References

- Mandatory Project Assignment AY 2018-2019
- ISO/IEC/IEEE 29148 dated Dec 2011
- Alloy Documentation http://alloytools.org/documentation.html
- Slides: Alloy, Use of Alloy in RE
- LaTex Documentation: https://www.latex-project.org/help/documentation/