Software Engineering 2

Requirement Analisys and Specification Document

Version 1.0

Andrei Constantin Scutariu	833370
Carlo Pulvirenti	828459
Sergio Piermario Placanica	916702

POLITECNICO
MILANO 1863

Academic Year 2018-2019

Contents

1 Introduction

1.1 Purpose

> The new model is mobile pervasive computing, and it will change the very nature of the enterprise
>
> *Carl Yankowski - Former CEO, Palm, Inc.*

In the last few years wearable devices are increasingly used. The evolution of wearables in the medical field has made possible to measure vital parameters useful for understanding the state of health of a person. The application for TrackMe uses this feature.

The main service is Data4Help which is a software system capable of sending and receiving informations about our customers, collected through the use of wearable devices, as accurately as the currently widely available technology permits. Such information includes current location, in terms of geographical position, and body parameters, such as heartbeat rate and blood pressure. This data has great value so the goal is to make it available for other companies to use for their purposes; they will only need to file a request to the system that, when approved, will guarantee access to information, currently collected and yet to be, regarding a specific customer or to a group of people satisfying certain criteria.

Moreover, the software will offer two other services that will make use of this data base.

The first one is AutomatedSOS, which will target mainly elderly people. The software will constantly monitor their health status, and in case of emergency will promptly dispatch an ambulance to their current location, notifying the customer of its arrival.

The second one is Track4Run, which will allow the organizers of a running event to define the path for it, the athletes to enroll in the event, and the spectators to track the position of the runners on a map in real time.

1.1.1 Goals

Data4Help

G1 Third parties shall be able to make requests for accessing single customer data.

G2 Third parties shall be able to make requests for accessing aggregate anonymous data specifiying filters.

G3 Third parties shall be able to access stored data, for which a request has been approved.

G4 Third parties shall be able to subscribe to new data, for which a request has been approved.

G5 Users shall be able to accept or refuse requests for their data from third parties.

G6 Access to customers data from a third party, for which a request does not exist, or has been rejected, shall be denied.

AutomatedSOS

G7 From the time a customer's parameters indicate a health emergency status, an ambulance shall be dispatched to his location in less than 5 seconds.

G8 When an ambulance is dispatched, the customer shall be notified of its arrival.

Track4Run

G9 Organizers shall be able to create a new running event.

G10 Runners shall be able to view and enroll in available runs.

G11 Spectators shall be able to see the position of participants on the map during a run.

1.2 Scope

The main purpose of Data4Help is creating a bridge between companies and individuals health status and location. As such, the main beneficiaries will be companies such as life insurances, hospitals and clinics, and universities for research purposes. The focus of our two other services, AutomatedSOS and Track4Run, is instead on the single customer, mainly elderly people and runners respectively.

The software will collect customer's data through wearable devices; they'll need to either have GPS and an active internet connection per se, or a paired smartphone through which the customer will be localized and data will be sent and received via internet. Companies on the other hand will be able to access our stored data through a REST web API.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

Third party company A company that wants to use our services.

Wearable (device) Electronic device meant to be worn on human body.

Backend The part of a program that carries out the tasks the program is designed to perform as opposed to the front end of the program that allows the user to instruct and interact with the back end.

1.3.2 Acronyms

RASD Requirement Analysis and Specification Document.

API Application Programming Interface.

SSN Social Security Number.

FC Fiscal Code, SSN used in Italy.

UI user interface.

UX user experience.

RAID Redundant Array of Independent Disks, data storage virtualization technology that combines multiple physical disk drive components into one or more logical units.

MTTR Mean Time To Failure, the length of time a device or other product is expected to last in operation

1.3.3 Abbreviations

Third party Third party company.

Company Third party company.

1.4 Revision history

Version	Release Date	Description
1.0	11/10/2017	Initial Release

1.5 Referenced documents

BEEP channel - Mandatory Project Assignment

The world & the machine - M. Jackson, P. Zave

developers.google.com - Google Fit API overview (https://developers.google.com/fit/overview)

developer.apple.com - watchOS API overview (https://developer.apple.com/documentation/watchkit)

statista.com - Smartwatch market share of overall wearable market worldwide in 2018 and 2022, by operating system (https://www.statista.com/statistics/750328/worldwide-smartwatch-market-share-by-platform/)

statista.com - Preferred time of the day to run worldwide (https://www.statista.com/statistics/933781/united-states-preferred-time-of-day-to-run-as-a-sport/)

statista.com - Android versions distribution (https://www.statista.com/chart/10761/adoption-of-latest-android-versions/)

1.6 Document structure

This document is structured in four main chapters, that are as follows:

1. The first chapter serves as an introduction and an overview to the project, describing the main reasons for its development, what are its goals and giving a brief informat description of it.

2. The second chapter serves as a more formal description of the project: it includes class diagrams, state machine diagrams, and it gives details on the shared phenomena and domain models. Class diagrams give a big picture description on how the system should be structured, while state machine diagrams focus on the more relevant entities of the model. Here are also presented all the requirements and domain assumptions the system in project must fulfill and take into considerations, in order to achieve the goals; they are presented each one after the goal it is relevant to.

3. In the third chapter are presented the specific requirements, use cases described through the use of natural language and diagrams such as sequence/activity diagram, and the design constraints the system must satisfy. A mockup is also shown as a general idea of how the end product should be, in terms of design and functionalities offered to the end user.

4. The fourth and last chapter is a formal analysis of the model, made through the use of the open source Alloy language and analyzer, including a graphic representation of it obtained from Alloy Tool.

2 Overall description

We can use "The World & The Machine" model by M. Jackson and P. Zave to make an initial domain analysis. This allows us to understand which are the entities and the phenomena the machine cannot directly observe ("The World"), which are the ones the external world cannot directly see ("The Machine"), and the ones that are shared, i.e. the mean of interaction between the two.

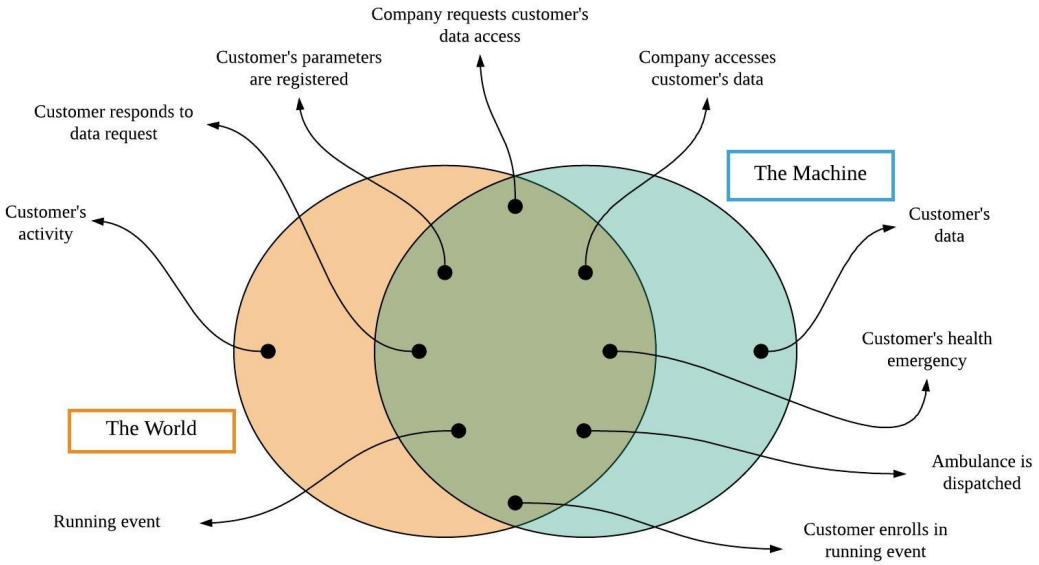


Figure 1: Program analysis as per "The World & The Machine" model

2.1 Product perspective

2.1.1 Class diagrams

Following there are the class diagrams that give an overall description of our main three services through the use of a conceptual model. Only the relevant parts of the system are present in every one of them for the sake of clarity, but it is to be considered as a single model.

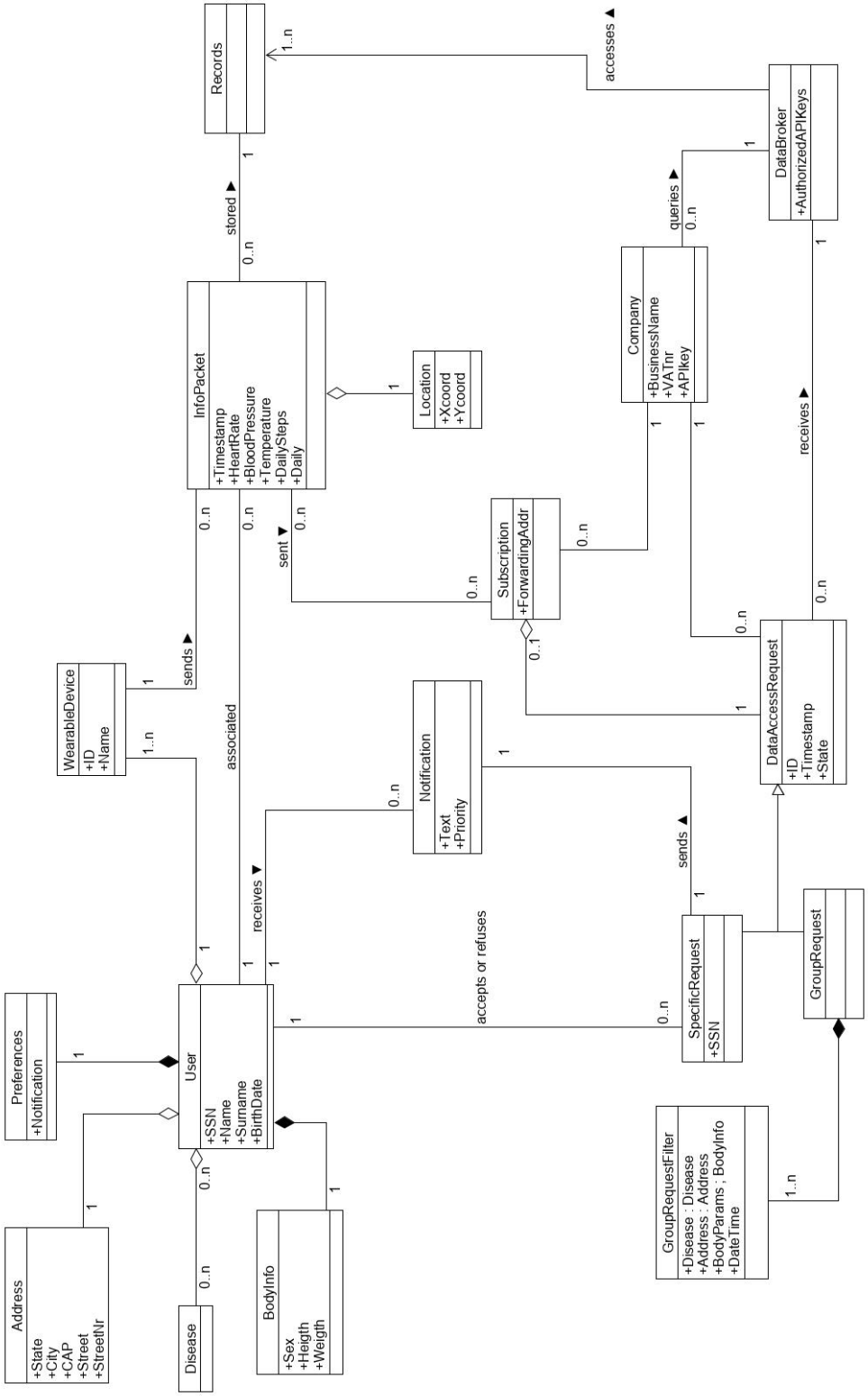


Figure 2: Class diagram for the Data4Help service

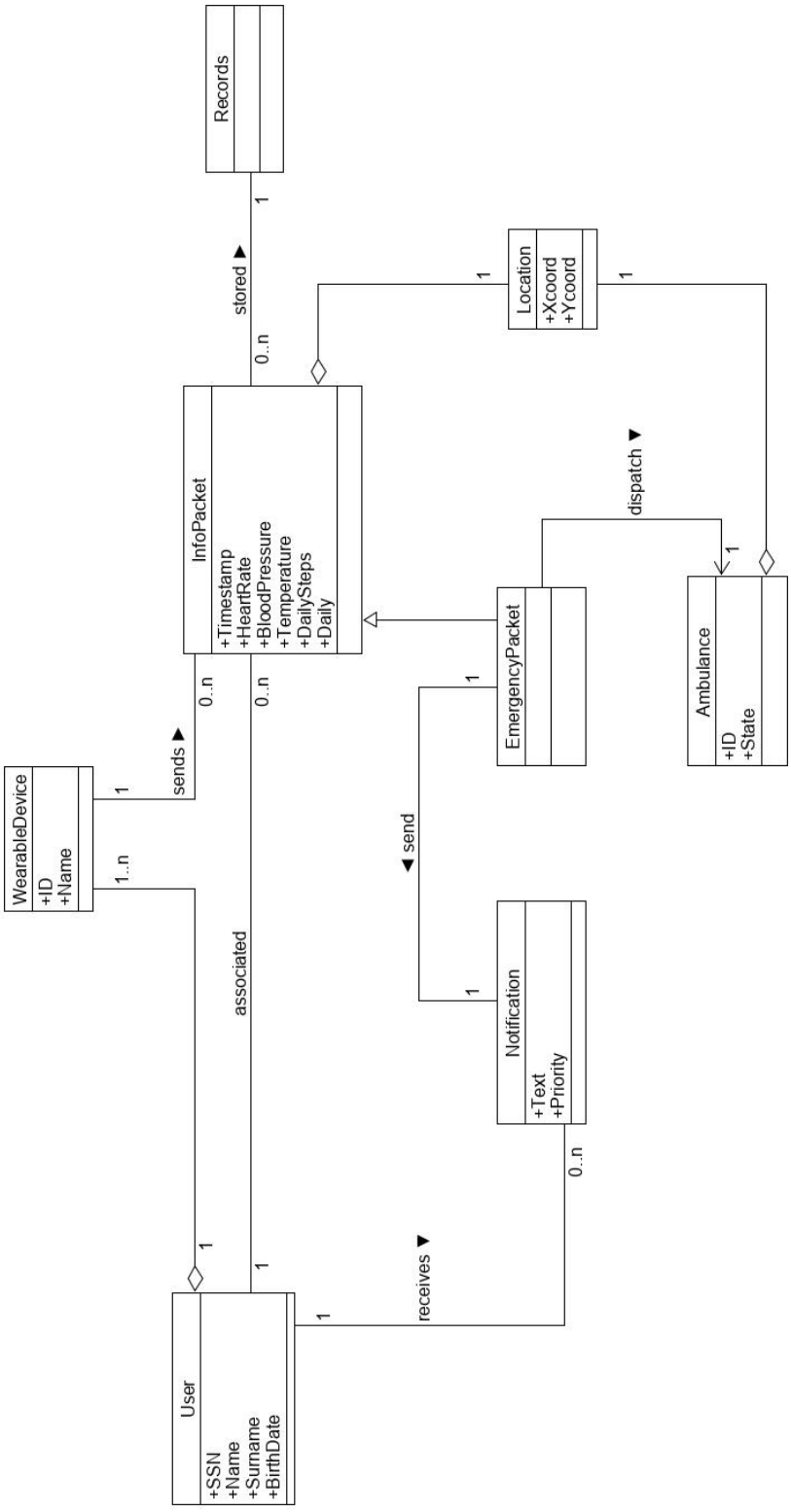


Figure 3: Class diagram for the AutomatedSOS service

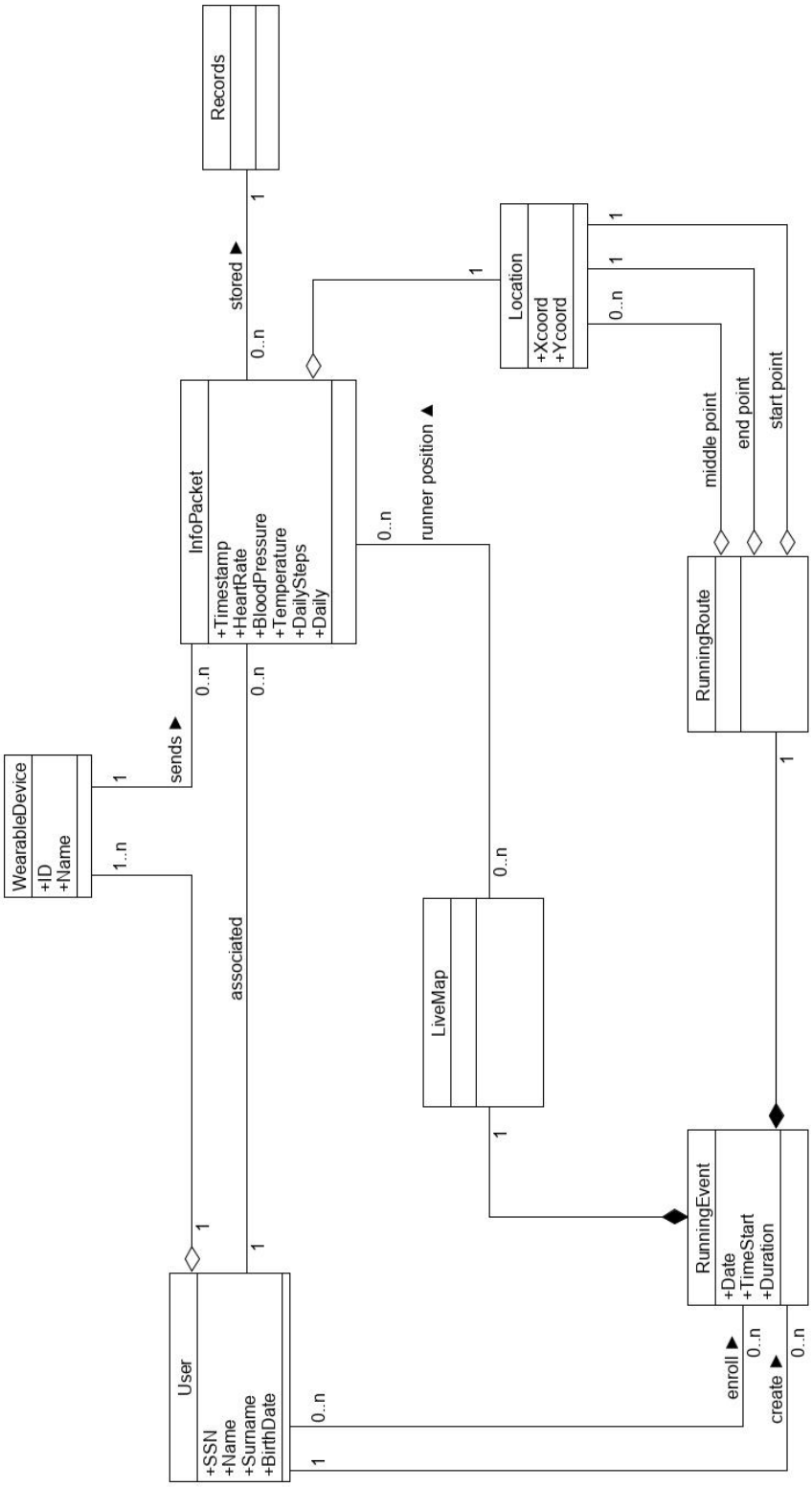


Figure 4: Class diagram for the Track4Run service

2.1.2 State machine diagrams

Through the use of the state machine UML model, it's desribed the evolution of the states for the main components of the system.

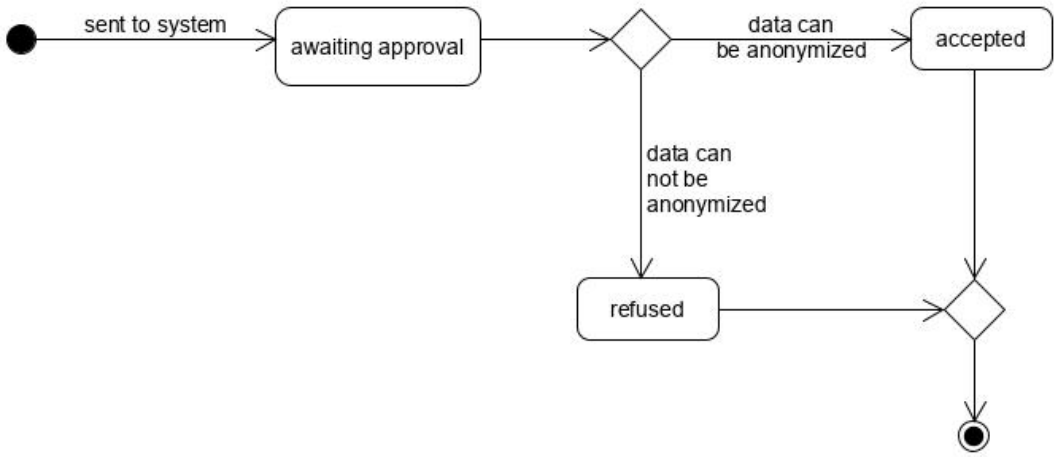


Figure 5: Statechart of a request for a group of cutsomers data

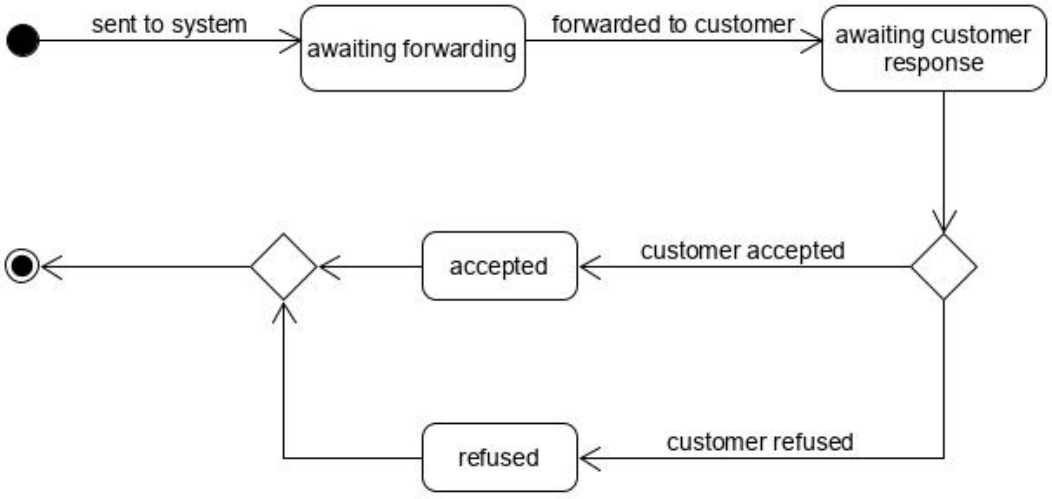


Figure 6: Statechart of a request for a single cutsomer's data

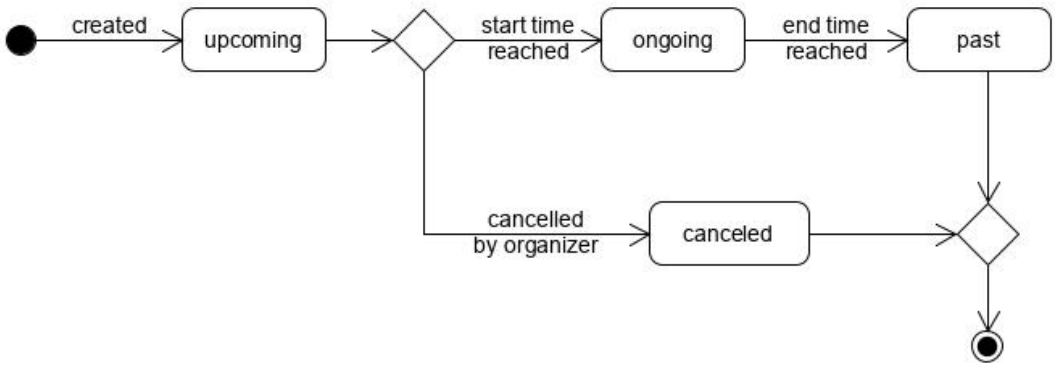


Figure 7: Statechart of a running event

2.2 Product functions

Following are the requirements our system has to satisfy in order to provide the main functionalities, each associated with its proper goal.

Data4Help

R1 The system must forward any request from a company to single customer's data to the corresponding user.

G1

R2 The system must reject any request for data regarding a group of customers when it cannot guarantee the anonimity of its components.

G2

R3 The system must periodically collect and store customer's data.

G3

R3 The system must periodically collect and store customer's data.

G3

R4 The system must periodically update the accessible data with the newly collected one

G4

R5 The system must notify the user of the request and permit him to accept or refuse it.

G5

R6 The system must update the request status with the answer provided by the user.

G5

R7 The system must be able to identify which data can be accessed for each third party company.

G6

AutomatedSOS

R8 The system must continuously check the data read from customers subscribed to AutomatedSOS.

G7

R9 In case the data indicate an emergency for a customer, the system must dispatch the closest ambulance to his location.

G7

R10 After an ambulance is dispatched, the system must notify the customer of its arrival.

G8

Track4Run

R11 The system must allow users to define the route, the date and the time of the event.

G9

R12 Two events can't overlap in the same place, date and time.

G9

R13 The system must show open events to the users.

G10

R14 The system must allow users to sort and filter events by date and location.

G10

R15 The system must collect and provide in real time the position of participants on a map.

G11

R16 The system must allow the creator of a running event to cancel the event before it started.

G11

2.3 User characteristics

Following are the actors of the application:

- Companies such as life insurances, hospitals and clinics, universities for research purposes or companies that focus on data analysis.
- People of any age in need for a potentially life saving system. They may have diseases, disabilities, or just be elder.
- Runners who want to organize running events or enroll in them, or people who want to track participants in real time.
- TrackMe's service staff, that will handle third parties registration and assist them in case of need, among other service activities.

2.4 Assumptions, dependencies, constrains

Domain assumptions

D1 Third parties are able to identify specific customers by SSN or FC.

D2 A group is considered anonymous if is composed by more than 1000 people.

D3 A sent request for customer's data is always received by the user.

D4 Health emergency status occurs when a AutomatedSOS customer parameters falls below of or exceeds healthy boundaries.

D5 AutomatedSOS customers always have an active and working internet connection.

D6 An ambulance is always available for dispatching and can reach the customer position.

D7 Organizers choose a valid path for a running event.

D8 A running event is legally autorized by the authorities.

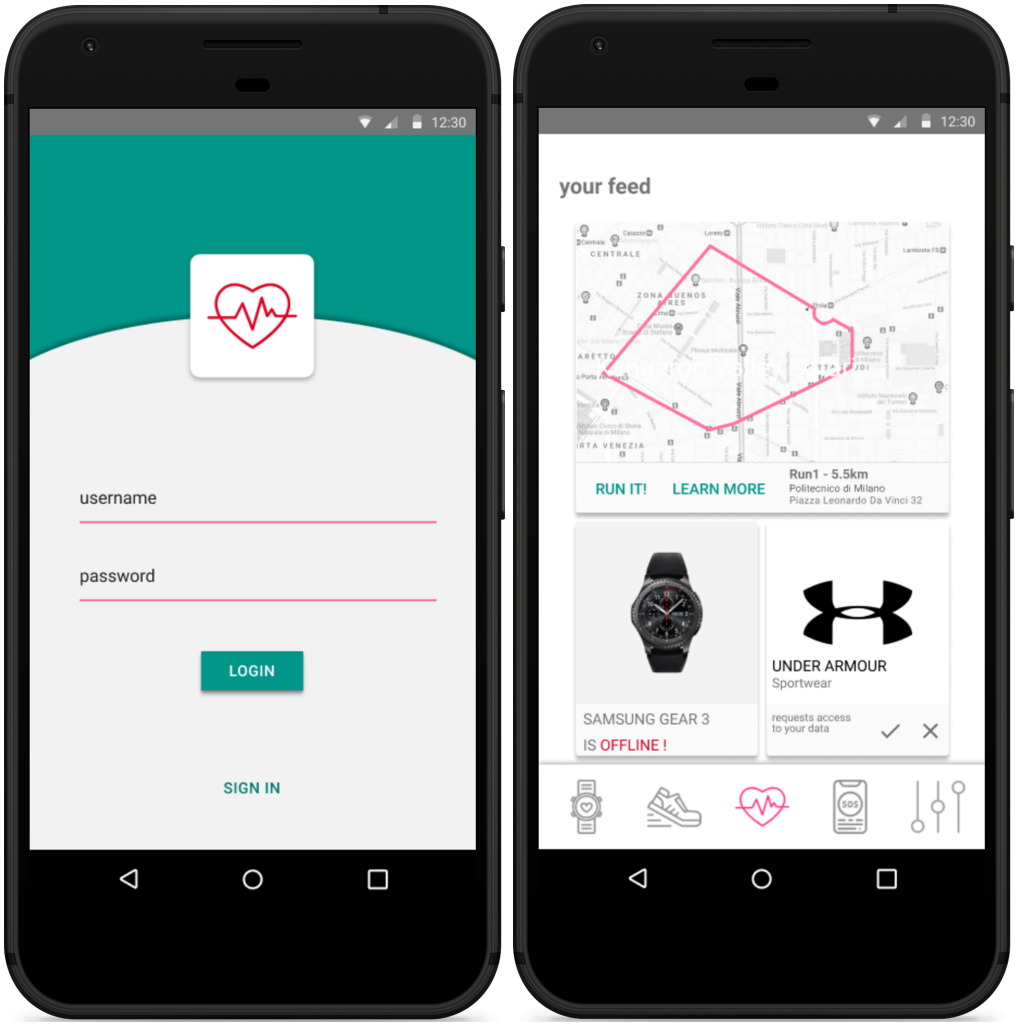
D9 The user is supposed to be at least 18 years old.

3 Specific requirements

3.1 External interfaces requirements

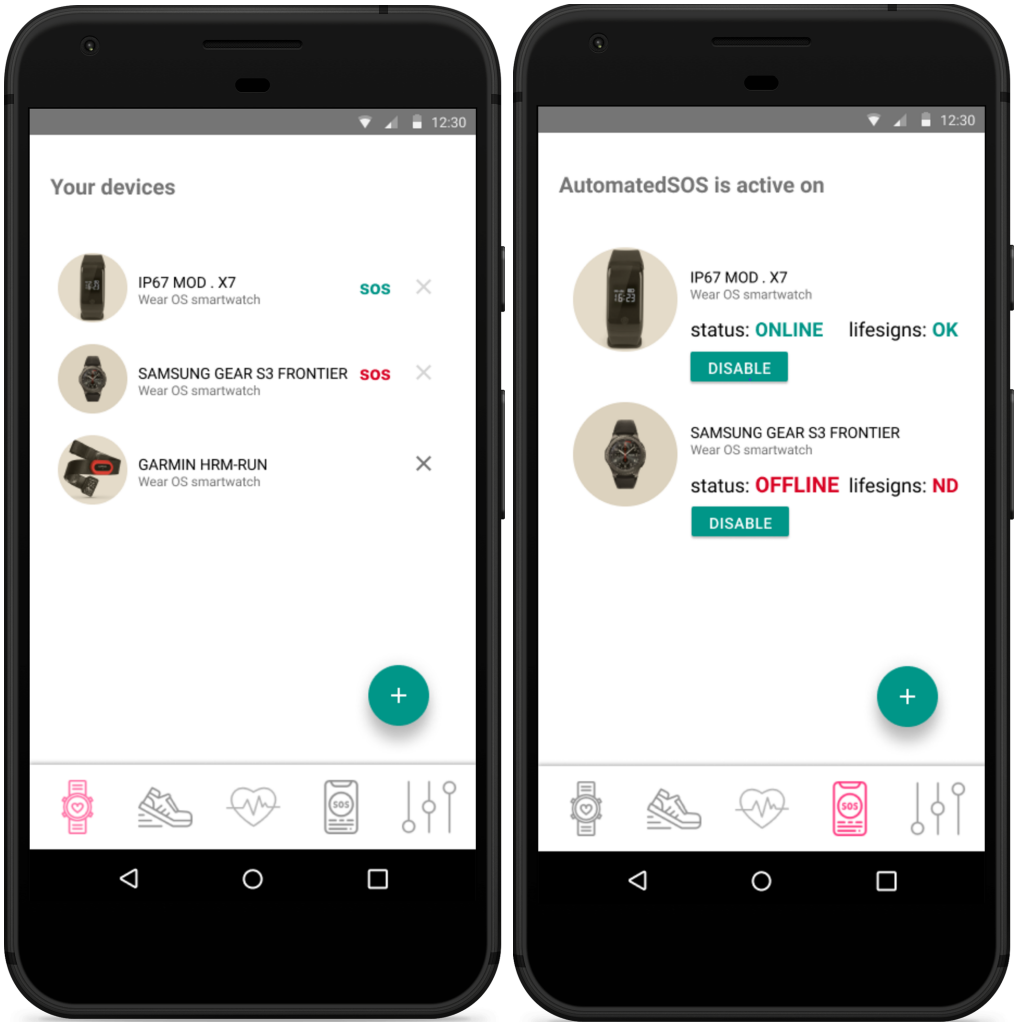
3.1.1 User interfaces

The mobile application was designed taking into account that users can utilize only the core functionality of the app (**Data4Help**) or all services offered (**track4Run** & **AutomatedSOS**). Opening the application, user is presented with the login page (*Fig. 8.a*), if not already registered, he can sign up.
Once logged in the user is presented with his "Main Feed" (Fig. 8.b), in this page all important information (upcoming runs, smartdevices status and company requests) are presented in a scroll page with clickable panels, the objective is to aggregate all three services informations in one screen to simplify the UX. At the bottom of the screen a navigation bar makes possible to navigate application's pages.
The user devices page (*Fig 9.a*) show all user devices, the user can remove a device clicking on the "X" next to the device or add one clicking on the "+" button. It's impossible to remove a device if it is associated with AutomatedSOS, the user need to disable **AutomatedSOS** on that device first.
The **AutomatedSOS** page (*Fig. 9.b*) show the status of the devices and lifesign readings. In the example one device is offline and marked with a warning red color. Here user can add or disable **AutomatedSOS** on his owned devices.
By clicking on Sharing in the settings screen (*Fig 10.a*) the user is presented with the sharing page (*Fig 10.b*), here he can visualize approved and pending requests from companies.
The run page (*Fig 11.a*) shows all available runs in the users vicinity. It also shows the subscribed runs. The user can click on a specific run and visualize details about it (*Fig 11.b*) or add a run clicking on "+". In the Running page the user can visualizes all the informations about the run and enroll himself to the run by clicking "Run It!". In the add a run page, the user can add a run after filling the forms. (*Fig 12.a*)



(a) Login page (b) Main feed

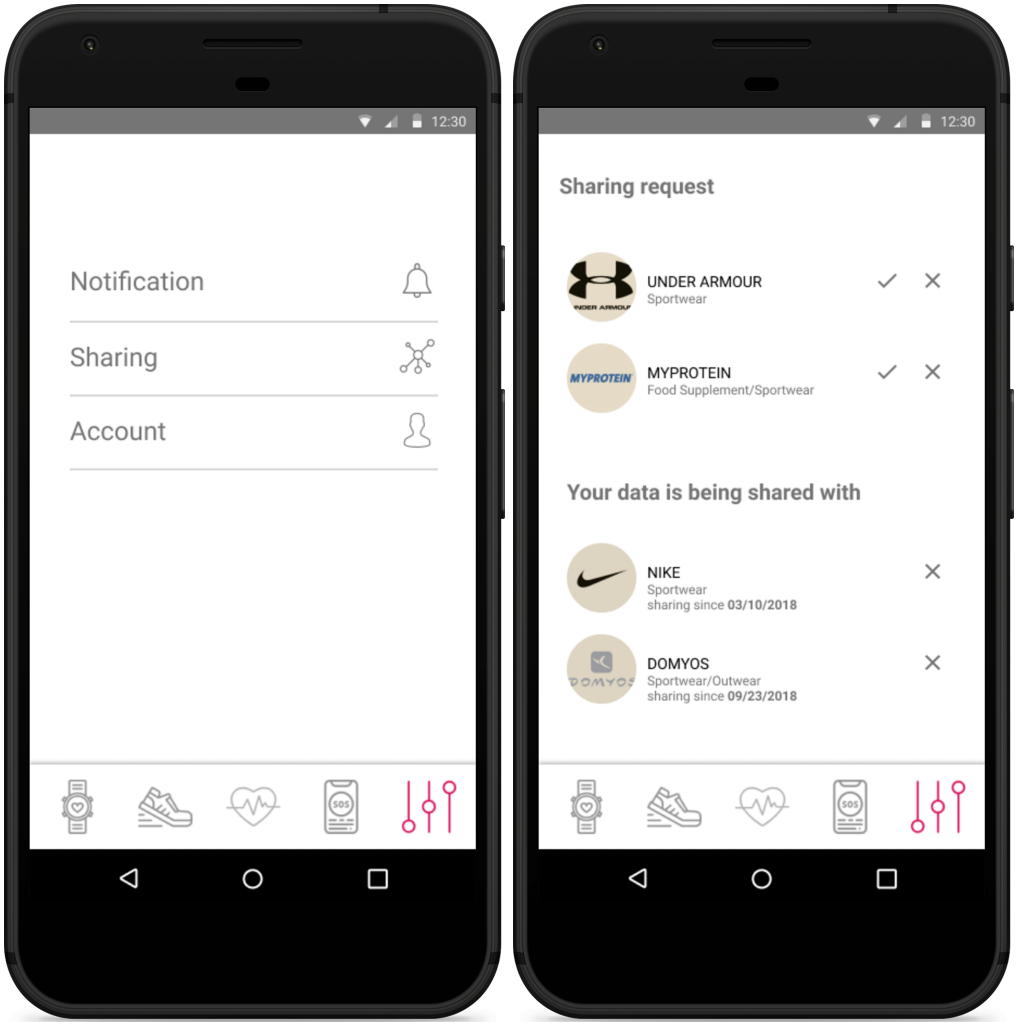
Figure 8



(a) User's devices

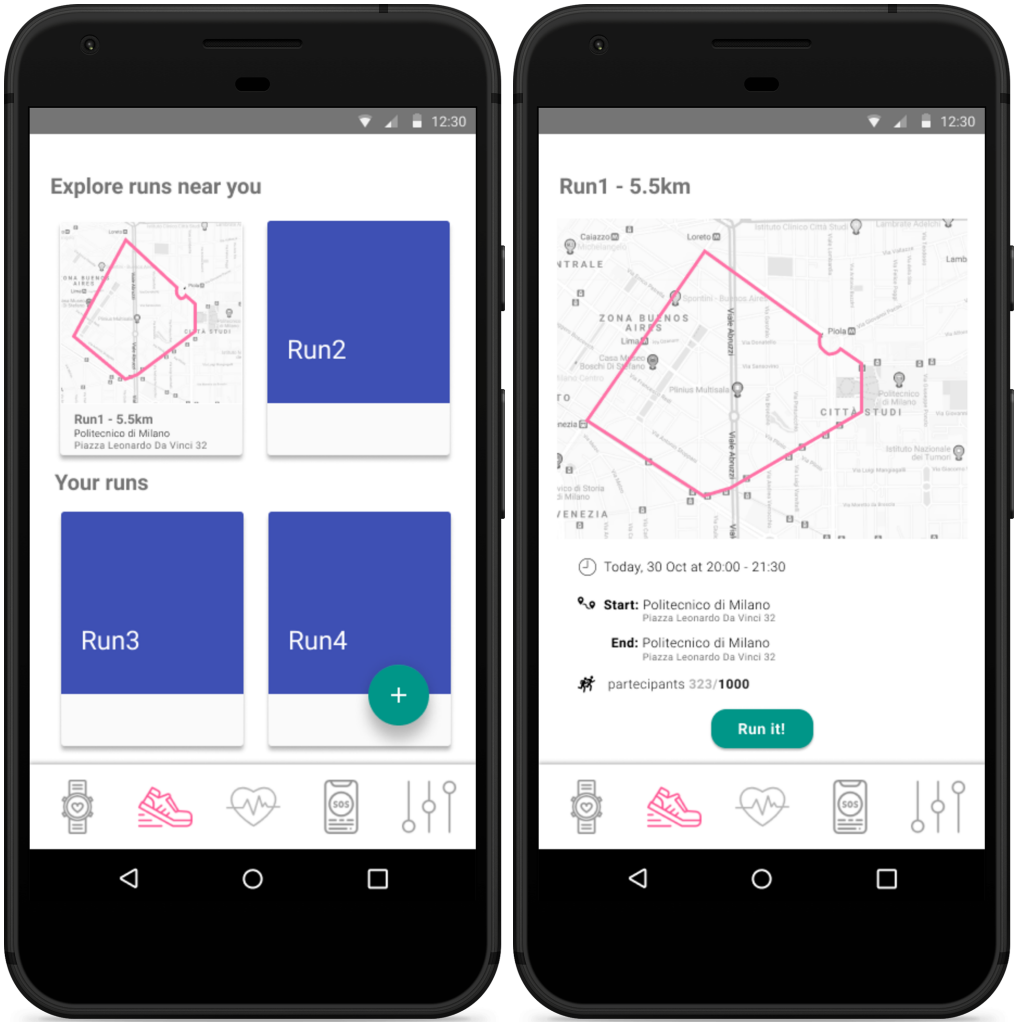
(b) automatedSOS

Figure 9



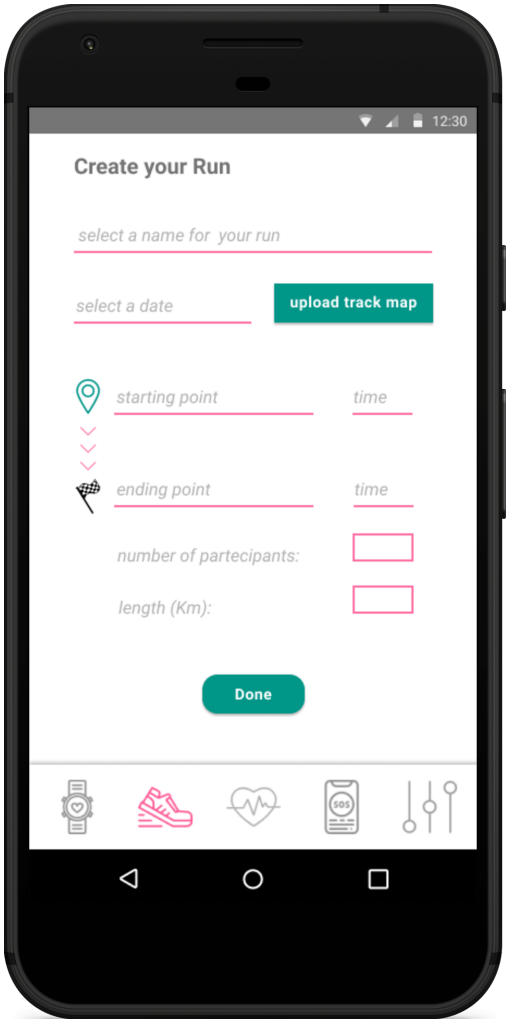
(a) settings (b) sharing center

Figure 10



(a) Running events (b) Running event details

Figure 11



(a) add a run page

Figure 12

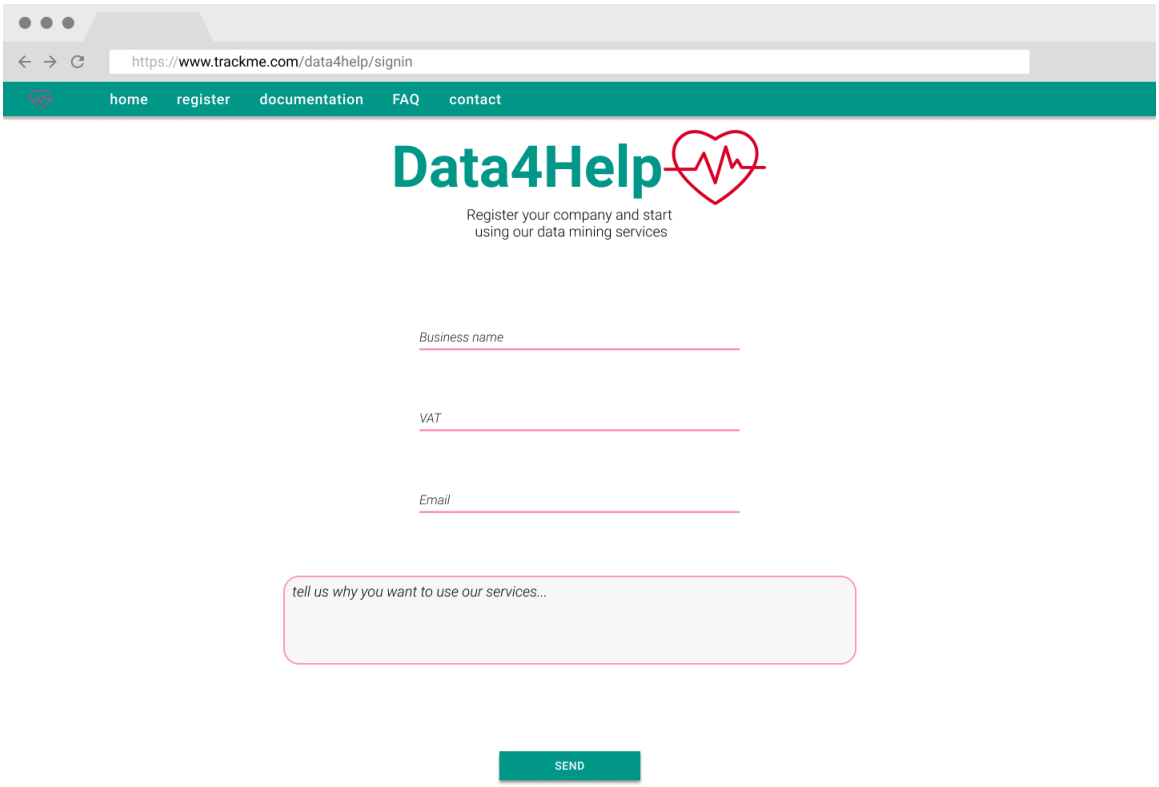


Figure 13: sign in forms for Data4Help

The website of **TrackMe** will offer a simple form to register a company and the API documentations for developers.

3.1.2 Hardware requirements

For using our services, the end customer will need a wearable device with appropriate sensors and interfaces (GPS, heart rate & pressure, etc.) to capture the location and body parameters to our system. A smartphone will also be needed to be paired with such wearable device.

The third party company on the other hand won't need any particular interface, any device capable of communicating data through HTTPS protocol will be fine.

3.1.3 Software interfaces

The application makes use of the following externally developed services:

Google Maps For the Track4Run service, the software uses Google's map service in order to let the spectators view the position of the runners; also, for a running event organizer, it's given the possibility of defining and vieweing the path for the run on the interactive map.

Ambulance dispatching system For the AutomatedSOS service, the system uses external software in order to be able to communicate with the ambulances and to track their position.

Google Fit APIs For communication between smartphone and wearable devices.

watchOS APIs For communication between smartphone and watchOS (Apple) wearable devices.

3.1.4 Communication interfaces

Communication between the customer's wearable device and his/her smartphone will take place through Bluetooth Low Energy protocol, and the one between such smartphone and our system through HTTPS, using a Wi-Fi or 3G/4G internet connection.

Communication between a third party company and our system will go through HTTPS procotol.

3.2 Functional requirements

3.2.1 Use Cases

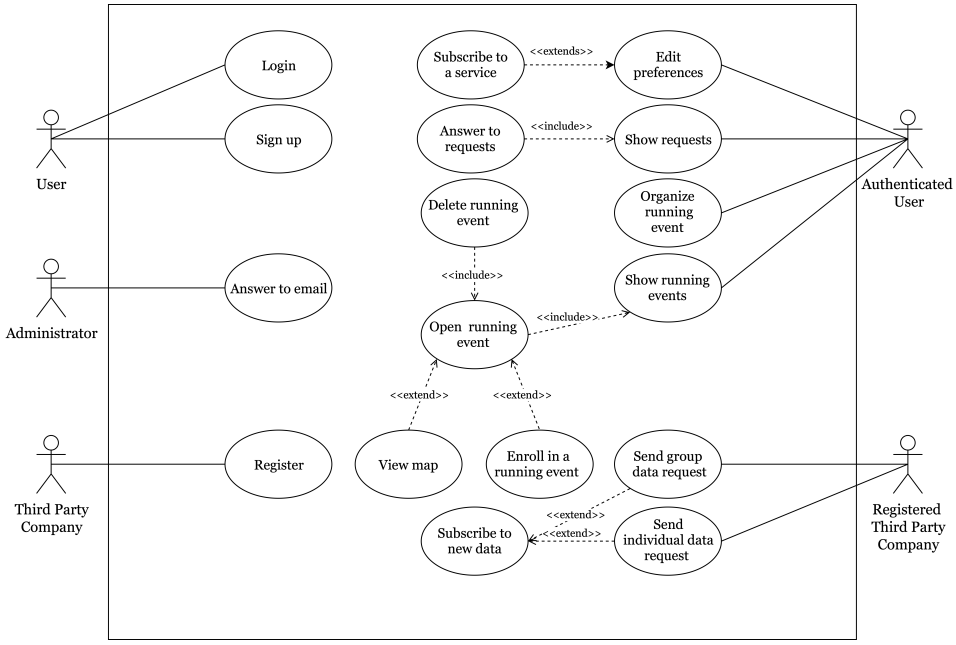


Figure 14: Use Case Diagram

Use Case	Sign up
Actor	User
Entry condition	The user has installed the application on his/her device.
Flow of events	1. The user clicks "Sign Up". 2. The user fills all mandatory fields. 3. The user accepts the privacy policies. 4. The user clicks "Confirm".
Exit condition	The user is registered and now he/she can log in to use the application.
Exception	1. The user is already registered. 2. The user doesn't fill all mandatory fields. 3. The user enters invalid data.
Note	When an exception occurs the user is notified and the application return to previous page.

Use Case	Login
Actor	User
Entry condition	The user has installed the application and it is open on his/her device.
Flow of events	1. The user clicks "Log In". 2. The user enters username and password. 3. The user decides if he/she wants to be remembered by the system clicking on the relative checkbox. 4. The user clicks "Confirm". 5. If the user chose to be remembered, next time the use case ends without asking username and password.
Exit condition	The user is on the home page of the application.
Exception	1. The user enters invalid data. 2. The user doesn't fill all fields.
Note	When an exception occurs the user is notified and the application return to previous page.

Use Case	Register
Actor	Third Party Company
Entry condition	The third party company is on TrackMe website.
Flow of events	1. The third party company goes to the registering page. 2. The third party company fills all mandatory fields. 3. The third party company clicks on "Send" button. An email with the informations is sent to TrackMe.
Exit condition	The third party company is waiting for the Data4Help API key to be received.
Exception	1. The third party company enters invalid data. 2. The third party company doesn't fill all fields. 3. Sending the email fails.
Note	When an exception occurs the email is not sent and the third party company is notified.

Use Case	Answer to email
Actor	Administrator
Entry condition	The administrator has received an email from a third party company.
Flow of events	1. The administrator reads the email. 2. If the third party company is legit the administrator sends the API key. 3. If the third party company isn't legit the administrator sends a rejection email.
Exit condition	The third party company receives the API key.
Exception	1. Sending the email fails.
Note	When sending the email fails the administrator is notified.

Use Case	Edit prefernces
Actor	Authenticated User
Entry condition	The authenticated user is on the home page of the application.
Flow of events	1. The authenticated user selects what service he/she wants to activate: AutomatedSOS, Track4Run. 2. The authenticated user can delete the authorizations to give data to the third party companies. 3. The authenticated user can edit personal informations.
Exit condition	The preferences is saved.
Exception	

Use Case	Subscribe to a service.
Actor	Authenticated User
Entry condition	The authenticated user wants to subscribe or unsubscribe to AutomatedSOS and Track4Run.
Flow of events	1. The authenticated user opens the preferences. 2. The authenticated user clicks a box to subscribe himself to a service. 3. If it's required, the authenticated user connects his/her wearable device.
Exit condition	The authenticated user is subriscribed to the desired services.
Exception	1. The authenticated user doesn't connect the wearable device when requested.
Note	When an exception occurs the preferences are not saved.

Use Case	Show requests
Actor	Authenticated User
Entry condition	The authenticated user is on the home page and wants to see the requests.
Flow of events	1. The authenticated user opens the requests page.
Exit condition	The requests page is opened and all the requets are visible.
Exception	

Use Case	Answer to requests
Actor	Authenticated User
Entry condition	The authenticated user is on the requests page.
Flow of events	1. The authenticated user opens a request. 2. The authenticated user clicks on "Accept" or "Refuse". 3. When the user approves a data request of a third party company all user's data becomes available to it.
Exit condition	The answers are saved. The approved requests make the relative third party company able to access currently stored data and to subscribe to new collected data.
Exception	

Use Case	Organize running event
Actor	Authenticated User
Entry condition	The authenticated user is on the events page and wants to organize a running event.
Flow of events	1. The authenticated user clicks "New running event". 2. The authenticated user selects the route, date, time and fills all mandatory fields. 3. The authenticated user clicks "Done".
Exit condition	The running event is organized and now users can view it and enroll to it.
Exception	1. There is another running event in the same place, date and time. 2. The route, date or time selected are not valid.
Note	When an exception occurs the running event is not saved and the user is notified.

Use Case	Show running events
Actor	Authenticated User
Entry condition	The authenticated user is on the home page and wants to view the running events.
Flow of events	1. The authenticated user clicks "Show running events".
Exit condition	All running events are visible to the user.
Exception	

Use Case	Open running event
Actor	Authenticated User
Entry condition	The authenticated user is on the running events page.
Flow of events	1. The authenticated user select a running event. 2. The authenticated user views all details of the running event.
Exit condition	The running event page is shown to the user.
Exception	

Use Case	Delete running event
Actor	Authenticated User
Entry condition	The authenticated user is the creator of the running event and he/she is on the running event page.
Flow of events	1. The authenticated user clicks on "Delete".
Exit condition	The running event is deleted.
Exception	1. The user tries to delete a running event that is in progress.
Note	When an exception occurs the running event is not deleted and the user is notified.

Use Case	Enroll in a running event
Actor	Authenticated User
Entry condition	The authenticated user is on the specific running event page and he/she isn't the creator of it.
Flow of events	1. The authenticated user clicks on "Enroll".
Exit condition	The user is enrolled to the running event selected.
Exception	1. No more participants are allowed in the run.
Note	When an exception occurs the user is not enrolled to the event.

Use Case	View map
Actor	Authenticated User
Entry condition	The authenticated user is on the specific event page.
Flow of events	1. The authenticated user clicks on "Map".
Exit condition	The user views the map of the running event and if it's already started he/she can see the participants location.
Exception	

Use Case	Send group data request
Actor	Registered Third Party Company
Entry condition	Third party company has already received the API key.
Flow of events	1. Registered third party company makes an html request with the proper parameters through TrackMe's web API.
Exit condition	Registered third party company received the requested data.
Exception	1. Some parameters are not valid. 2. The request is rejected because it concerns less then a thousand people.
Note	When an exception occurs the third party company is notified and it doesn't receive requested data.

Use Case	Send individual data request
Actor	Registered Third Party Company
Entry condition	Third party company has already received the API key and it knows the fiscal code or the social security number of the person to whom it's asking for data.
Flow of events	1. Registered third party company make an html request with the proper parameters and the fiscal code of the person through TrackMe's web API.
Exit condition	The request was sent.
Exception	1. Some parameters or the fiscal code are not valid.
Note	When an exception occurs third party company is notified and the request is not sent.

Use Case	Subscribe to new data
Actor	Registered Third Party Company
Entry condition	Third party company has already received the API key and it's sending a group or individual data request.
Flow of events	1. Registered third party company makes an html request through TrackMe's web API including a parameter to subriscribe to receving new data as soon as thay are available.
Exit condition	The request was sent and if it will be accepted new data will be sent to the company as soon as they are available.
Exception	1. Some parameters are not valid.
Note	When an exception occurs third party company is notified and the request is not sent.

3.2.2 Scenarios and Sequence Diagrams

Scenario 1

HealthyResearch is a company who works in the medical research field. It's making a project for which it needs people's data. It accesses this data thanks to TrackMe's Data4Help service specifying for people who live in Italy, 25 to 50 years old and who regularly do some sports and not. The sample is bigger that 1000 users so HealthyResearch receives the requested data shortly after TrackMe's system approved the request. Then HealthyResearch wants the same data but this time referring to the people who are participating to the running event in Milan today. In this case the sample is smaller than 1000 users so the request is rejected.

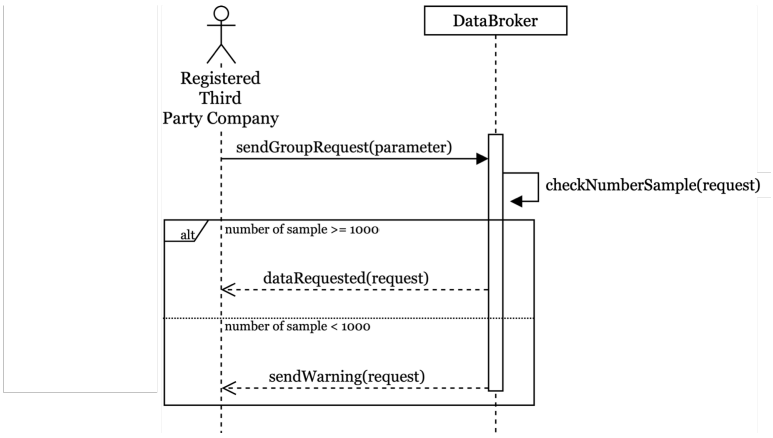


Figure 15: Sequence Diagram Scenario 1

Scenario 2

A hospital wants to monitor a convalescing patient who has been relesed. The patient has signed up on TrackMe's services; the hospital using TrackMe's APIs sends him an individual request for data, and subscribes to it, so it can monitor the patient's parameters and check his status in real time.

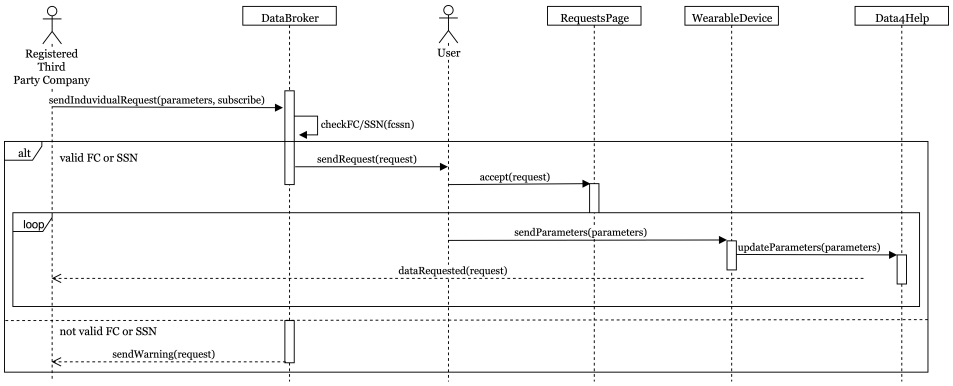


Figure 16: Sequence Diagram Scenario 2

Scenario 3

Mrs. Jane is 75 years old. She always forget to drink water and she eats very poorly. His son buys her a smartwatch and installs the TrackMe's application on the phone subscribing Jane to AutomatedSOS service. One day Jane feels bad due to an hypovolemia so her blood pressure drops dramatically. The smartwatch recognizes that the value is under the minimum, so it automatically calls an ambulance. Jane arrives to the hospital in time.

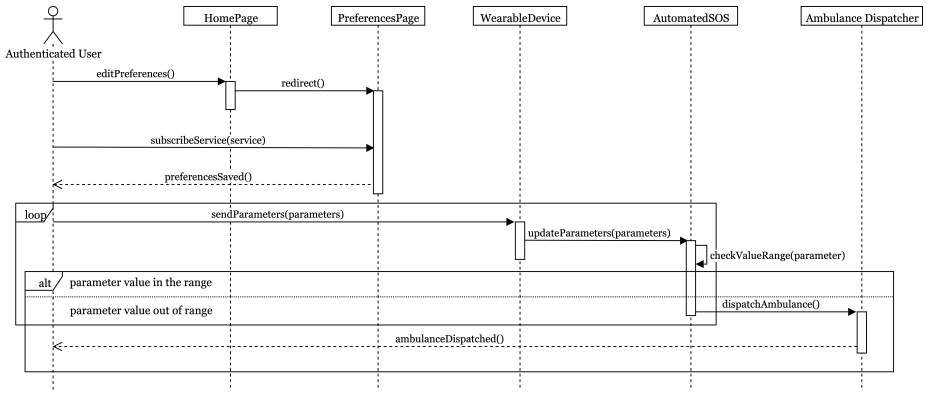


Figure 17: Sequence Diagram Scenario 3

Scenario 4

Mario wants to organize a running event in his city; he is subscribed to Track4Run so he opens the events page and creates a new one. He then selects the route, date and time, and forwards the creation of the event, but it gets rejected because there is an overlapping event at the same place and time. He then decides to create the same event, but a week later, and this time it all goes well. Unfortunately, an unexpected appointment is assigned to Mario the next day right for the day of the run. At this point Mario decides to delete the event he created, postponing it for a time when he'll have more free time.

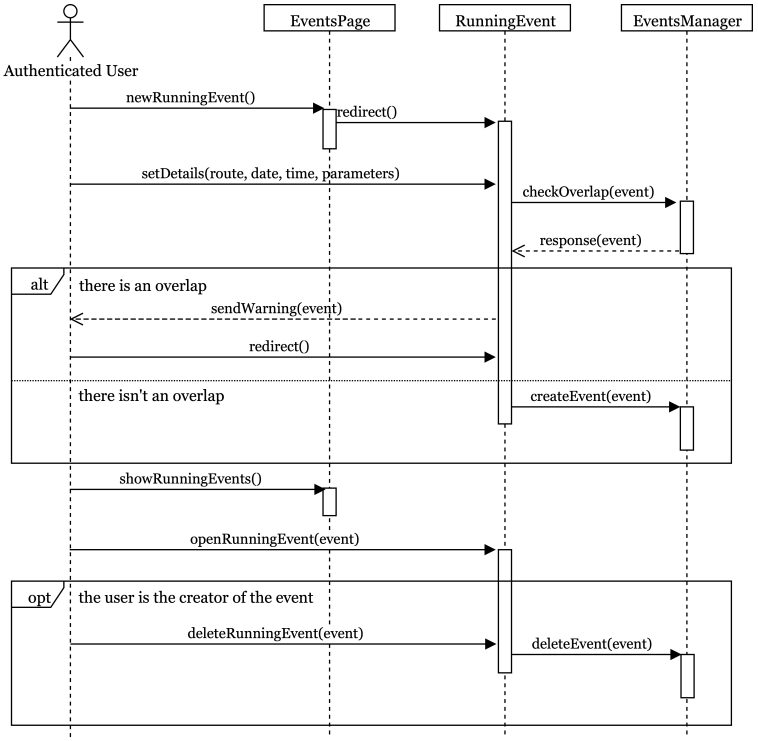


Figure 18: Sequence Diagram Scenario 4

Scenario 5

Anne is a hobbyist runner, she decides to subscribe to Track4Run. She open the events page and she sees one that interests her and her friends so she enrolls into it. She unfortunately breaks her foot before the event date, so she can't participate anymore, but thanks to the possibility of seeing the live map with the position of the participants she will at least be able to see the event progress as a spectator and cheer for her friends.

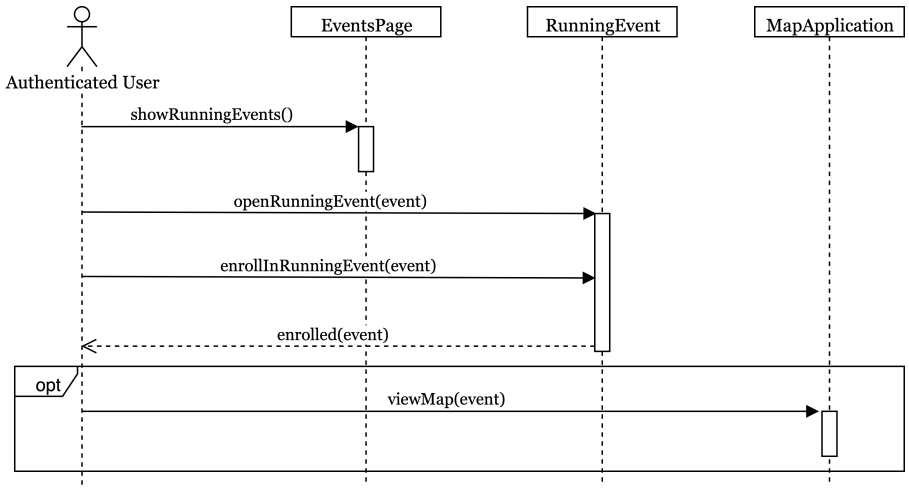


Figure 19: Sequence Diagram Scenario 5

3.3 Performance requirements

Our system needs to permit a large number of third party companies to always operate on our services, and in the same way must promptly operate the incoming data from a large number of customers. It must therefore be scalable on the increasing number of users over time, and ready to serve at least 10.000 customers and 200 companies at launch.

The software on customer's device must be reactive, and must take into account communication protocols unreliability, transmitting all the data that wasn't succesfully sent after an occasional internet or bluetooth connection disservice.

3.4 Design constraints

3.4.1 Standard compliance

As per privacy policies, the system must ask appropriate permissions to be installed on the respective device, and must ask the customer to accept a statement that describes the use of the information that it collects about him/her, at the moment of registering to our services.

3.4.2 Hardware limitations

Following are the hardware limitations required by the end customer for using our services.

Wearable Device

- GPS
- Bluetooth Low Energy
- Heart rate sensor
- Blood pressure sensor

Smartphone

- Android or iOS platform
- Bluetooth Low Energy
- Wi-Fi or 3G/4G technology

3.5 Software system attributes

3.5.1 Reliability

AutomatedSOS it the critical component of the application, thus it's infrastracture should be characterized by the lowest MTTR possible.
A larger MTTR is acceptable for our business to business components, however it's mandatory to prevent "data loss" (ie : using RAID configurations for servers)

3.5.2 Availability

AutomatedSOS service must be up and running 24/7, the user must be quickly notified of any communication issues between smartphone and AutomatedSOS smartdevices or with Data4Help servers.
Track4Run service should be operative only during day hours, as it's expected a low utilization of the service during night hours as shown in the histogram (Fig. 19).
Companies and devices should be able to communicate with TrackMe's servers 24/7.

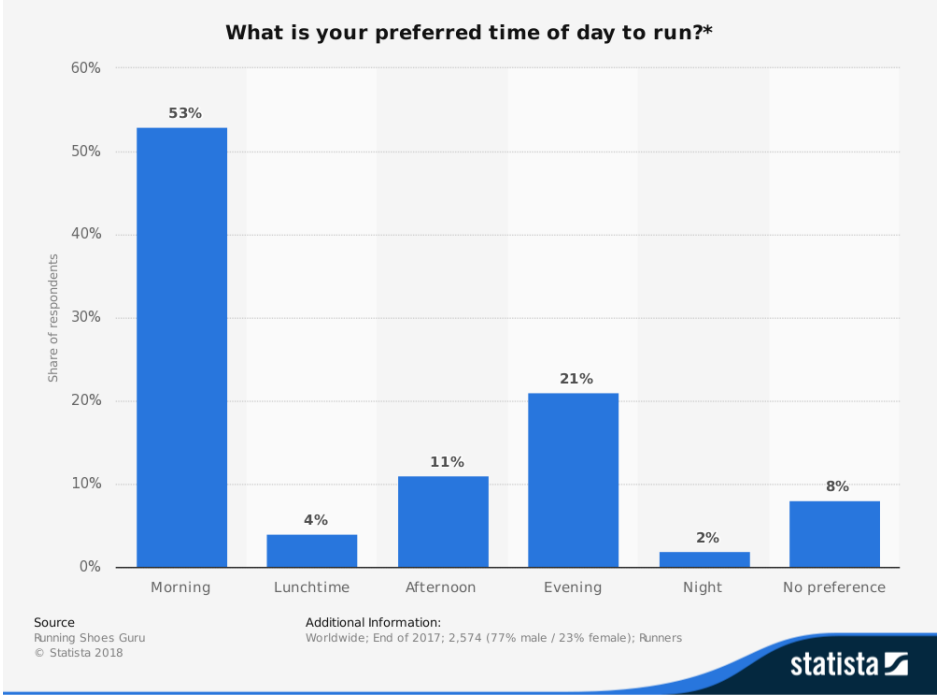


Figure 20: Preferred time of day for running (statista.com)

3.5.3 Security

In order to guarantee the protection of customer's sensible data, all internet traffic must be encrypted with SSL and sent through HTTPS protocol;

3.5.4 Mantainability

The back-end system must be capable of sustaining maintenance operations on a copy, to be then deployed after being tested on all its functions, in order to guarantee no down-time for the critical functions.

Software that's on devices will be updated through the appropriate app store, as needed.

3.5.5 Portability

Our smartphone application must be available for Android and iOS platforms. Cross platform developing tools can be used to achieve this. As shown in the figure below, android market is characterized by a great fragmentation in OS versions. The application will target the Lollipop API (android lollipop 5.0) to ensure compatibility with most devices.
Communication between the application and the wearable device will be held by the Google Fit API for Android and Wear OS API for iOS (see reference in introduction).

Using Google Fit API will ensure compatibility with smart devices running different OS than Android (ie: TizenOS)

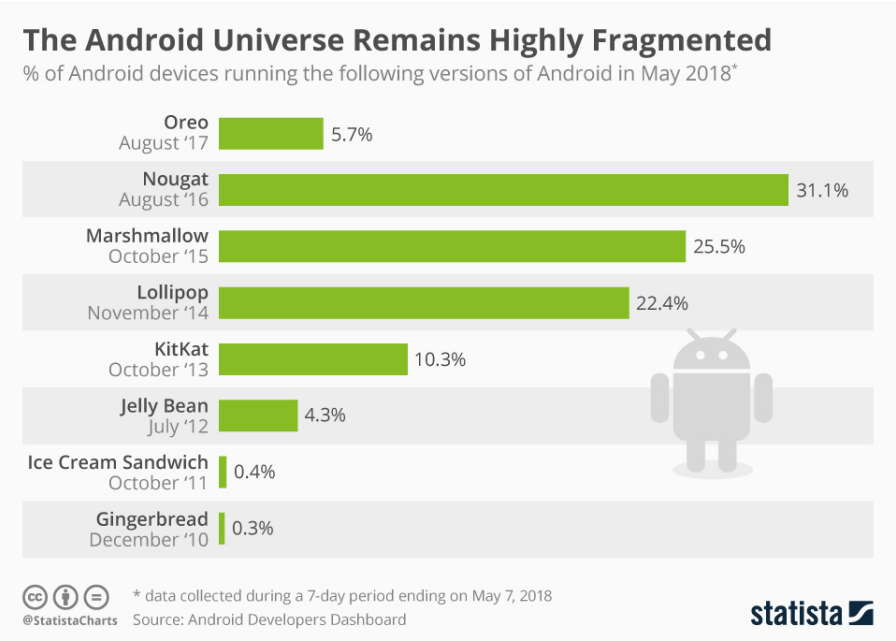


Figure 21: Android OS versions distribution (statista.com)

4 Formal analysis

In this chapter there is a formal representation of the application. Only the interaction with third party companies is modeled, since it is the most easily misinterpreted concept when explained in an informal way and is a crucial part of the application; also, various minor semplifications have been made in this model with respect to the actual expected structure of the application and its class diagrams proposed in the previous chapters, for ease of readability and modelization of it. These include:

- No check for user's anonimity when a group request is validated. However, it is simply a matter of counting how many users satisfy a certain request for user's data, and not authorizing the request if this number is below a certain threshold - here, 1000.
- Users have no SSN/FC; instead, the specific request targets directly the user.
- Only age and city are kept as user's parameters; also, age is used in place of a user's birth date, and city in place of a user's address.
- Age is in the 18-30 range.
- Every interaction between two or more entities is represented as a relation between the two.

4.1 Alloy code

```
open util/integer

// user signatures ------------------------------------------------------------

sig User {
        city : one City,
        age: one Int,  // age instead of birthdate
        devices: some WearableDevice,
        acceptedRequests: set SpecificRequest
} {
        // all users are between 18 and 30 (for simplicity)
        age ≥ 18 and age ≤ 30
        one dB : DataBroker | acceptedRequests in dB.authorizedRequests
}

sig WearableDevice {
        sentData : set InfoPacket
} {
        one dB : DataBroker | sentData in dB.availableData
}

sig City{}


// company signatures ---------------------------------------------------------

sig Company {
        requests: set DataAccessRequest,
        accessibleData: set InfoPacket,
        subscriptions: set Subscription,
}

abstract sig DataAccessRequest {}
```

```
sig SpecificRequest extends DataAccessRequest {
        user: one User
}

sig GroupRequest extends DataAccessRequest {
        filters: some GroupRequestFilter
}

sig GroupRequestFilter {
        ageStart : lone Int,
        ageEnd : lone Int,
        city: lone City
} {
        // consistent with User data
        ageStart ≥ 18 and ageStart ≤ 30 and
        ageEnd ≥ 18 and ageEnd ≤ 30 and
        // it either has one city or one ageStart or one ageEnd
        (one ageStart or one city or one ageEnd) and
                ((one ageStart and one ageEnd) implies
                ageEnd ≥ ageStart)
}

sig Subscription {
        request : one DataAccessRequest,
        company : one Company
} {
        one dB : DataBroker | request in dB.authorizedRequests
}


// general signatures -------------------------------------------------------

sig InfoPacket {}

one sig DataBroker {
        pendingRequests : set DataAccessRequest,
        authorizedRequests: set DataAccessRequest,
        availableData: set InfoPacket
}


// facts --------------------------------------------------------------------

fact userConstraints {
        // every WearableDevice is owned by one user
        all device : WearableDevice | one user : User | device in user.devices

        // every InfoPacket is only sent by one WearableDevice
        all data : InfoPacket | one dev : WearableDevice | data in dev.sentData
}

fact requestConstraints {
        // all requests are made by one company
        all req : DataAccessRequest | some company : Company | req in company.requests

        // no DataAccessRequest is neither in authorizedRequests nor pendingRequests but it
            ↪ has to be in one of the two
        all req: DataAccessRequest | one dB : DataBroker |
                (req in dB.pendingRequests and req not in dB.authorizedRequests) or
                (req in dB.authorizedRequests and req not in dB.pendingRequests)

        // no pending SpecificRequest is in any User's acceptedRequests
        one dB : DataBroker | all req : SpecificRequest |
                req in dB.pendingRequests implies all user : User | req not in user.
                    ↪ acceptedRequests

        // a SpecificRequest can be accepted only by the proper user, and then it goes in
            ↪ authorizedRequests
        one dB : DataBroker | all req : SpecificRequest | all u : User |
                req in u.acceptedRequests iff (req in dB.authorizedRequests and req.user = u)

        // a company can not make two SpecificRequests for the same User
        all company : Company | all disj req1, req2 : SpecificRequest |
```

```
                (req1 in company.requests and req2 in company.requests) implies req1.user ≠
                    ↪ req2.user

        // every GroupRequestFilter is owned by a GroupRequest
        all grf : GroupRequestFilter | one gr : GroupRequest | grf in gr.filters
}

fact subscriptionConstraints {
        // every subscription is generated from a company
        all sub : Subscription | sub in sub.company.subscriptions

        // a company can not have two subscriptions for the same data (no 2 subscriptions
            ↪ with same request)
        all c : Company | all disj s1, s2 : Subscription |
                (s1 in c.subscriptions and s2 in c.subscriptions) implies (s1.request ≠ s2.
                    ↪ request)

        // a company has a subscription only if it made the request it is linked to
        all c : Company | all sub : Subscription | sub.request in c.requests
}

fact dataAccessConstraints {
        // all data from a user is accessible to the company
        // if it has access to even a single packet
        all c : Company | all u : User |
        (some data : InfoPacket | data in u.devices.sentData and data in c.accessibleData)
            ↪ implies
        not (some data : InfoPacket | data in u.devices.sentData and u.devices.sentData not
            ↪ in c.accessibleData)

        // data sent from users device is accessible from a company if and only if
        // the company made a SpecificRequest that the user accepted
        // or it made a GroupRequest that was authorized and the user
        // satifies the criteria in GroupRequest's filters
        all c : Company | all u : User | one dB : DataBroker |
        (some data : InfoPacket | data in u.devices.sentData and data in c.accessibleData)
            ↪ iff
        (
                (some sr : SpecificRequest | sr in c.requests and sr in u.acceptedRequests)
                or
                (some gr : GroupRequest | some grf : GroupRequestFilter | grf in gr.filters
                    ↪ and
                gr in dB.authorizedRequests and gr in c.requests and
                (one grf.ageStart implies u.age ≥ grf.ageStart) and
                (one grf.ageEnd implies u.age ≤ grf.ageEnd) and
                (one grf.city implies grf.city = u.city))
        )
}

// predicates and assertions ------------------------------------------------


pred companyMakeSpecificRequest [disj c1, c2 : Company, u : User] {
        one sr : SpecificRequest | sr not in c1.requests and sr not in u.acceptedRequests
        implies c2.requests = c1.requests + sr
}

run companyMakeSpecificRequest for 2 but 7 Int, exactly 0 GroupRequest,
        exactly 1 InfoPacket, exactly 1 SpecificRequest


pred userAcceptsSpecificRequest [disj u : User, disj sr1, sr2 : SpecificRequest, c : Company]
    ↪  {
        sr1 in c.requests and sr1 not in u.acceptedRequests implies
        sr2 in c.requests and sr2 in u.acceptedRequests
}

run userAcceptsSpecificRequest for 2 but 7 Int, exactly 0 GroupRequest, exactly 2 InfoPacket,
        exactly 2 SpecificRequest, exactly 2 Company, exactly 1 User


pred groupRequestOverview[c : Company, dB : DataBroker] {
        some gr : GroupRequest | gr in c.requests and gr in dB.authorizedRequests
```

```
}

run groupRequestOverview for 2 but 7 Int, exactly 1 GroupRequest


pred overview [disj d1, d2 : InfoPacket, c : Company] {
        d1 in c.accessibleData and d2 not in c.accessibleData
}

run overview for 2 but 7 Int, exactly 1 SpecificRequest, exactly 1 GroupRequest


assert noAuthorizedRequestMeansNoDataAccess {
        some company : Company | one dB : DataBroker |
        #dB.availableData > 0 and
        (#company.requests = 0 or
                all request : DataAccessRequest |
                request in company.requests and request not in dB.authorizedRequests)
        implies #company.accessibleData = 0
}

check noAuthorizedRequestMeansNoDataAccess for 10 but 7 Int, exactly 10 DataAccessRequest


assert accessibleDataMeansSomeAuthorizedRequest {
        all c : Company | one dB : DataBroker |
        #c.accessibleData > 0 implies
        some req : DataAccessRequest | req in c.requests and req in dB.authorizedRequests
}

check accessibleDataMeansSomeAuthorizedRequest for 10 but 7 Int
```

4.2 Alloy graphs

4.2.1 Overview

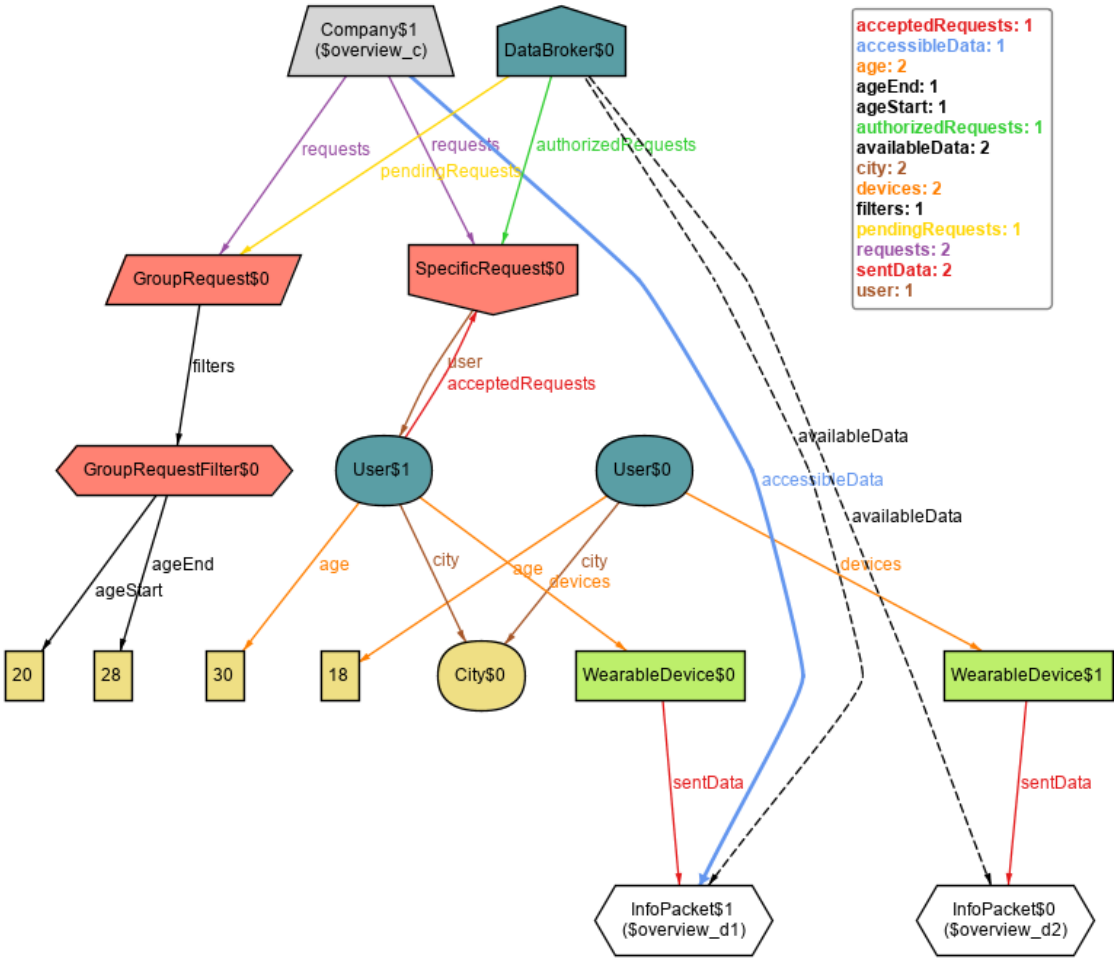


Figure 22: Overview of the Alloy model

An overview of the modeled world can be viewed in the image above. There is a company who made two requests: one GroupRequest that filters for users between 20 and 28 years old, that grants no data access since it has not been authorized by the system - here modeled by the entity DataBroker - as it is still a pendingRequest, and one SpecificRequest for User1 that the user accepted, and as such is represented in a relation of authorizedRequests with DataBroker. This means that the company has access to all the data sent by the devices of User1, represented as InfoPacket, as can be seen by the relation accessibleData between such InfoPackets and the company. The relation availableData between InfoPackets and DataBroker simply means that the system has received and stored such data and is available for companies to request or access it.

4.2.2 Overview with group request

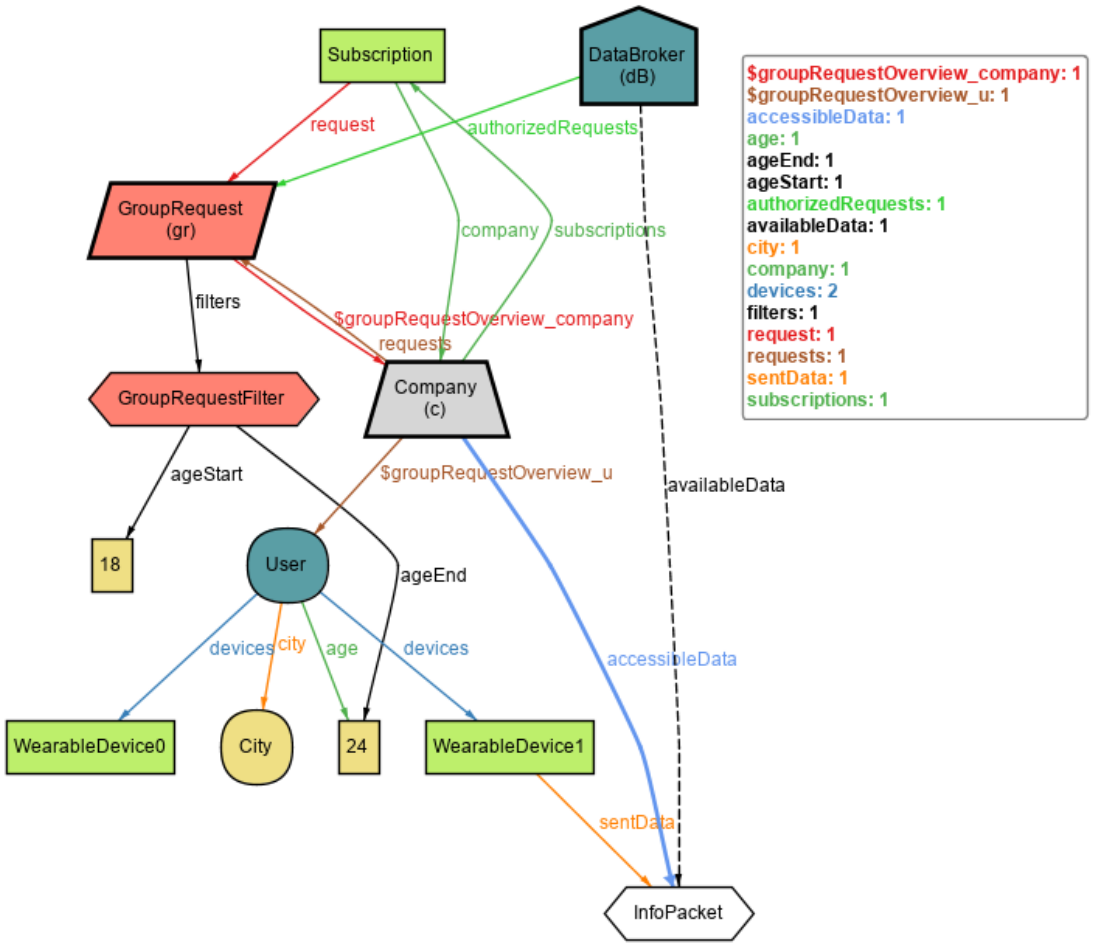


Figure 23: Overview of the Alloy model with a group request for user data

Here is represented a group request that has been authorized. This just filters for users between 18 and 24 years old and grants access to these user's data for the company. Since the only user in the model is 24 years old, all data sent from his devices is accessible. Also, as modeled, the company has subscribed to this request, so it will receive new data as soon as it is produced by the user; subscriptions can be made with SpecificRequests too.

4.2.3 Data request

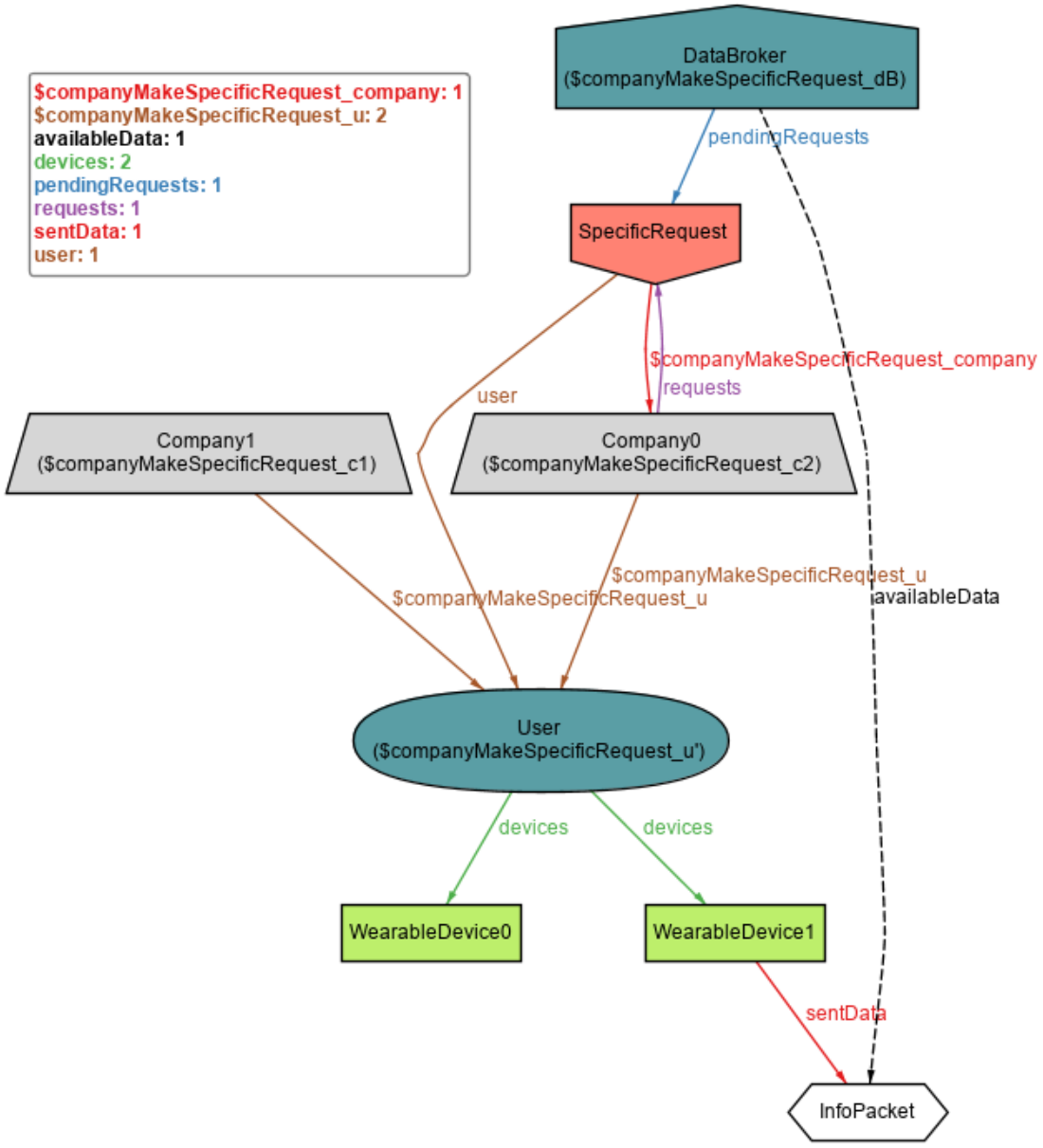


Figure 24: Overview of the Alloy model with a company forwarding a request for a user's data

Here is modeled the forwarding of a request for data access from a company. At the moment of the arrival of such request, Company1 is the company in the past, while Company0 is the company in the present; Company1 has no requests associated to it, while Company0 has one request for User's data, but the user hasn't accepted it yes, and as such his data is not accessible to the company.

4.2.4 Data request approval

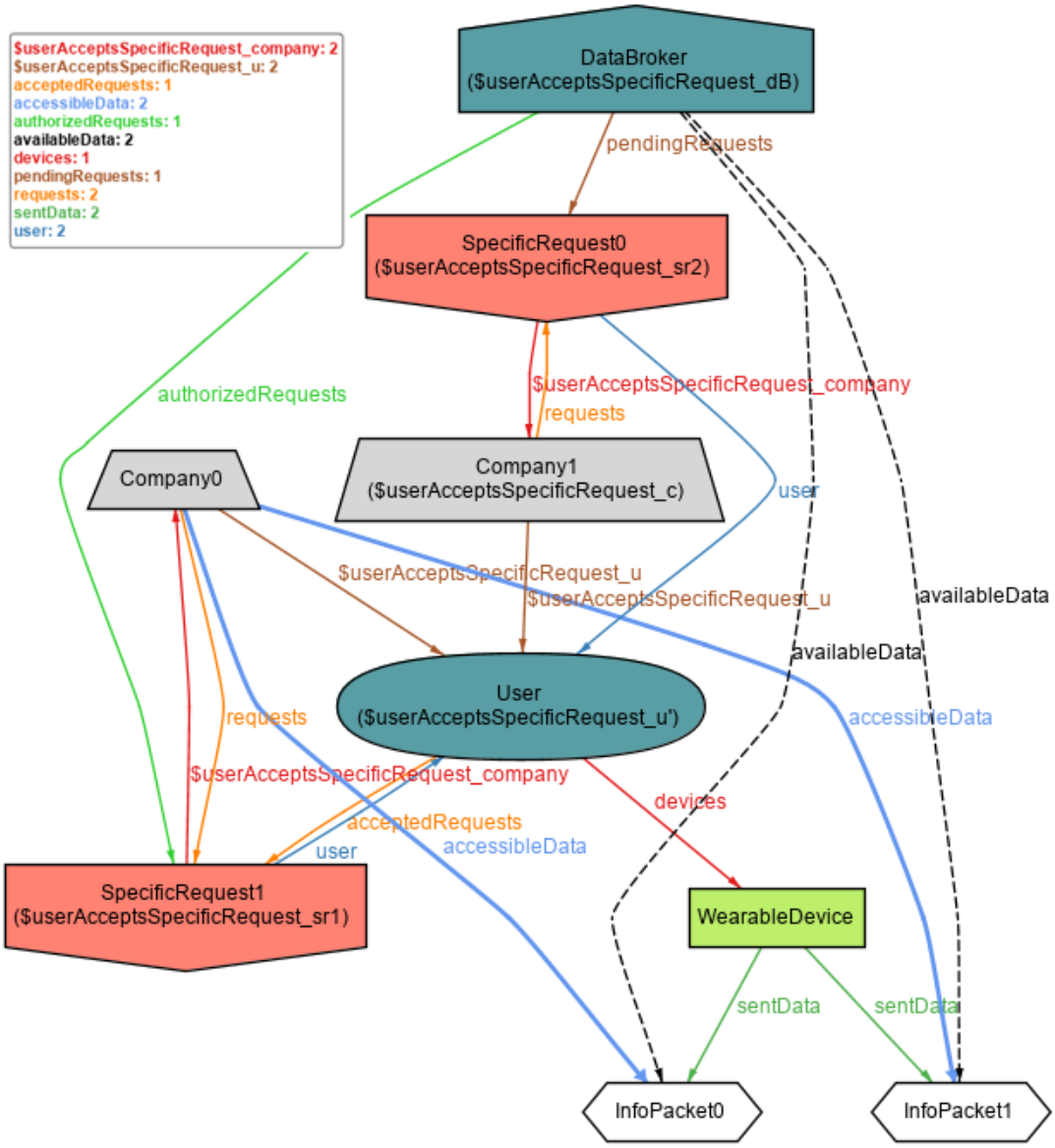


Figure 25: Overview of the Alloy model with a user accepting a request for data

Here is modeled one possible follow up to the situation of the previous section: now that the user has accepted to give his data to the company that made the request, his data is accessible for the company. Company1 and SpecificRequest0 are the entities of the previous section, while Company0 and SpecificRequest0 are the entities after the user's approval.

4.2.5 Final notes on alloy model and results

In the previous sections of overview, some entities that were irrelevant to the analyzed situation have been hidden for clarity. The constraints on the predicates serve only to generate a meaningful and not knotty visual representation; more complex examples of the model can be generated altering these.

Following are the results the Alloy software generated.

Figure 26: Alloy software results

5 Effort spent

Andrei Constantin Scutariu

Date	Hours	Description
17/10/18	2h	Introduction
19/10/18	3h	Goals, Requirements, Domain assumptions
24/10/18	4h	Goals, Requirements, Domain assumptions
26/10/18	1h	Purpose, Scope
29/10/18	2h	UML Class Diagrams
30/10/18	1h	Purpose, Scope
31/10/18	3h	UML Class Diagrams
01/11/18	1h	Use Cases & Scenarios
03/11/18	1h	Mockups, Scenarios
03/11/18	2h	UML Class & State Machine Diagrams
04/11/18	3h	Specific Requirements
05/11/18	1h	Goals, Requirements, Domain assumptions
06/11/18	1h	Mockups
07/11/18	4h	Alloy
08/11/18	5h	Alloy, Revisioning
09/11/18	4h	Alloy
10/11/18	5h	Alloy, Revisioning
11/11/18	4h	Revisioning
total	47h	

Carlo Pulvirenti

Date	Hours	Description
17/10/18	2h	Introduction
19/10/18	3h	Goals, Requirements, Domain assumptions
24/10/18	4h	Goals, Requirements, Domain assumptions
26/10/18	2h	Purpose, Scope
29/10/18	6h	Use Cases
31/10/18	6h	Use Cases, Scenarios
02/11/18	4h	Scenarios
03/11/18	1h	UML Class & State Machine Diagrams
05/11/18	4h	Scenarios, Sequence Diagrams
08/11/18	2h	Sequence Diagramm, Alloy
09/11/18	2h	Sequence Diagram, Revisioning
10/11/18	2h	Revisioning
11/11/18	4h	Revisioning
total	42h	

Sergio Piermario Placanica

Date	Hours	Description
17/10/18	2h	Introduction
19/10/18	3h	Goals, Requirements, Domain assumptions
24/10/18	4h	Goals, Requirements, Domain assumptions
26/10/18	2h	Purpose, Scope
31/10/18	4h	Mockups
03/11/18	4h	Mockups
05/11/18	1h	Scenarios, Sequence Diagram
08/11/18	4h	Mockups, Sequence Diagramm, Alloy
09/11/18	4h	Alloy, Performance Requirements
10/11/18	4h	Performance Requirements, Revisioning
11/11/18	4h	Revisioning
total	36h	