**Politecnico di Milano**
**AA 2018/2019**

POLITECNICO
MILANO 1863

Software Engineering 2

**TrackMe**

# Design Document


Stefano Pecchia
921201

Edoardo Peretti
921286

| | |
|---|---|
| **Deliverable:** | DD |
| **Title:** | Design Document |
| **Authors:** | Stefano Pecchia and Edoardo Peretti |
| **Version:** | 1.0 |
| **Date:** | December 10, 2018 |
| **Download:** | https://github.com/Speck1996/PecchiaPeretti |
| **Copyright:** | Copyright © 2018, S. Pecchia and E. Peretti – All rights reserved |

# Contents



# Chapter 1

# Introduction

## 1.1 Purpose

Data4Help is a service whose aim is to collect data from a first typology of user, the indivudal, and distribute the collected data to a second typology of user, the Third Party.
These features will be the key drivers for building the whole system, from the user applications to the database that manages the data.

These two typologies of user will have their respective application in order to access to Data4Help, as stated in the RASD.
In addition to the these main features, Data4Help will also provide a service called AutomatedSOS, whose aim is to call an ambulance if a vital parameter of an individual goes below a certain threshold.
This document will serve as guideline for building Data4Help underlying software and hardware structure, describing the following aspects:

- General overview of the system;
- Description of the software components and the interactions between them;
- Implementation, integration and testing plan.

Before continuing the reading of this document, it is strongly suggested to read the RASD first, so that it will be easier to understand the decisions taken for Data4Help architectural design.

## 1.2 Scope

The critical points in order to build the Data4Help system are:

- Gather a large amount of eHealth data from individuals;
- Store and distribute the collected data to third parties.

The first point is achieved through a smartphone application from which the individuals will be able to :

- Register and log in;
- Pair a compatible smartwatch;
- Upload and visualize data collected by the paired smartwatch;

- Accept or deny data access requests from third parties;

Additionally, if the paired smartwatch is capable to measure at least a vital parameter (heart rate or blood pressure), individuals can also activate from the app the Automated-SOS service.
The second point is instead achieved through a web application from which third parties will be able to:

- Register and log in;
- Make a request for anonymized data;
- Make a request to access data from a specific individual;
- Visualize the data of accepted requests.

The two critical aspects for building a suitable system are:

- Smartwatch pairing;
- Data handling: from the uploading, to the storing and the distribution.

These aspects are fundamental for the services provided, and both the software and architectural decisions are made to guarantee a reliable functioning of them.

### 1.2.1 Acronyms

- **DBMS**: Database Management System
- **DB**: Database
- **RASD**: Requirements Analysis and Specifications Document

### 1.2.2 Abbreviations

Rn : n-th functional requirement

## 1.3 Document Structure

Chapter 1 presents a quick introduction to the Data4Help system, the definitions and the abbreviations needed to understand the document.

Chapter 2 describes all the architectural decisions made to build Data4Help system. First, a general overview of the system is presented: in particular a general overview of the system, the component view, the deployment view are shown in details.
Then, after presenting all the component interfaces and their associated operations, Data4Help features are analyzed through the use of sequence diagrams. In the end, the selected architectural decisions and chosen design patters are briefly analyzed.

Chapter 3 mainly refers to the section 3.1.1 of the RASD, related to the UI of the applications.

Chapter 4 shows how the requirements presented in the RASD are mapped through the component designed in chapter 2.

Chapter 5 describes how the various software components will be implemented, tested and integrated.

Chapter 6 presents a table with the effort spent by each document author in order to build this document.

Chapter 7 presents the references used in the document.

# Chapter 2

# Architectural Design

## 2.1 Overview

At the highest level of abstraction, the architecture of Data4Help and AutomatedSOS system is a common three-tier client/server paradigm, as shown in Figure 2.1.

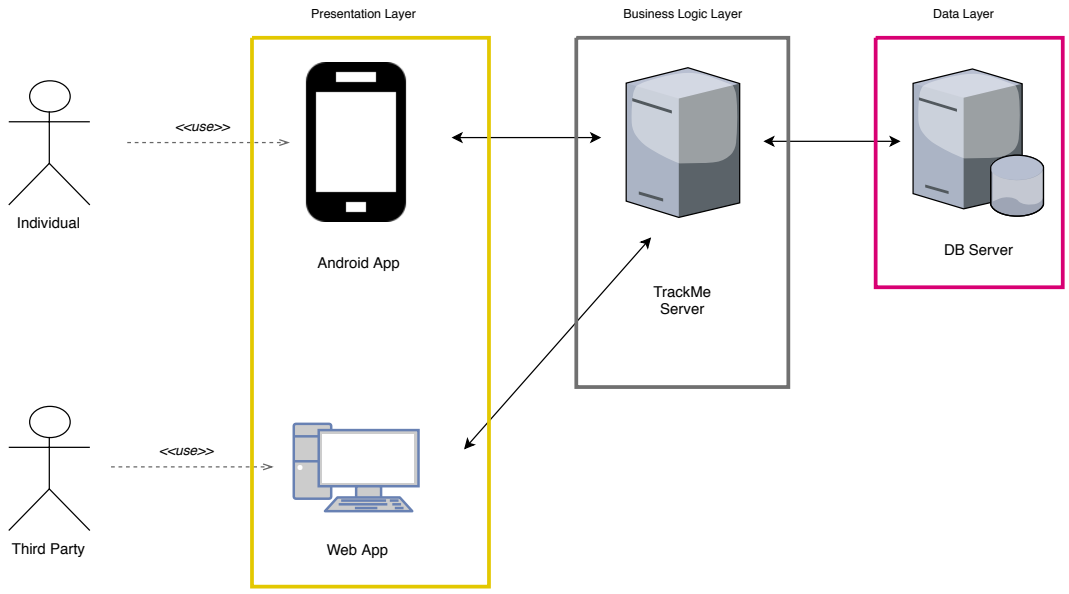



Figure 2.1: The 3 tier architecture of the system

The three layers are the following:

- **Presentation layer**: this layer is hosted on the users' devices, that are the smartphone for the individuals and a computer with a browser for third parties. The GUIs are deployed in this layer.
- **Business logic layer**: this layer contains the main components of the systems that carry out the operations needed for Data4Help and AutomatedSOS services.
- **Data layer**: this layer is responsible for managing all the data of the system. In particular, it stores the data on persistent devices and makes them available when requested by the TrackMe server.

The structure displayed in Figure 2.1 is not to be understood as a rigid constraint on the implementation choices, but it represents where most of the three three logic layers are deployed. For examples, it is possible that part of the data is (temporarily) stored on the clients in order to reduce latency or that some presentation logic is deployed on the TrackMe Server in order to achieve a more dynamic GUI.

## 2.2 Component view

The diagram in Figure 2.2 examines in more detail the subcomponents composing the TrackMe server. Some components requires external services, this is shown by means of pending required interfaces. All the four components defined below have access to the database server interface in order to store and retrieve the data necessary for their correct operation.

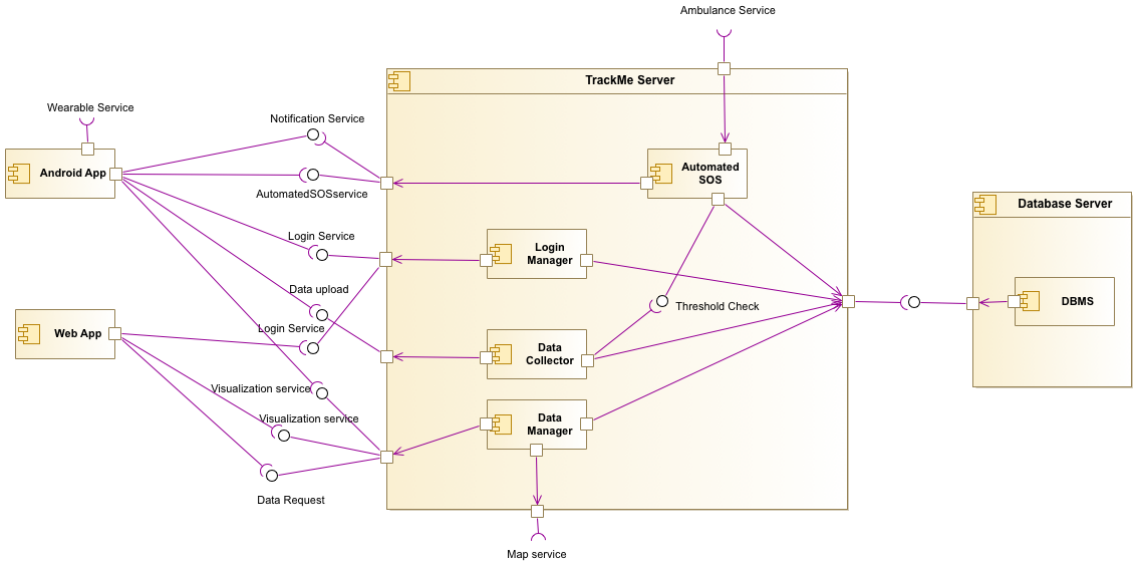



Figure 2.2: Component Diagram

As shown, there are four main components in the business logic server:

- *Login Manager*: is responsible for the login and registration of the users, both individuals and third parties.
- *Data Collector*: is responsible of collecting data from the Android App. It also forwards the proper data to the Automated SOS component.
- *Data Manager*: receives and handles requests from third parties and push to them new data when available and the subscription is valid. For certain functionalities, for example for geographically filter the data, the Data Manager component requires an external API to a map service. It also makes possible for the individuals to visualize their data.
- *Automated SOS*: handles the subscriptions for the SOS service. Every time that it receives new data from the Data Collector, the component checks whether the thresholds are crossed, in this case it immediately calls an ambulance through an external API.

In addition, Figure 2.2 shows that the Android App component requires an interface towards the wearable device, from which it will gather the eHealth Data.

## 2.3 Deployment view

The diagram in Figure 2.3 shows the physical deployment of system's artifacts on various nodes. In particular, the nodes are:

- *Smartphone*: this device is used by an individual, then the Android Application is deployed here. This node interacts with the Business Server.
- *Personal Computer*: this device is used by a third party, only a modern browser compatible with HTML5 and CSS3 is required. Actually, there are no artifacts of our system deployed on this node, because the interaction is carried out with a Web App directly towards the Business Server.
- *Business Server*: this is the core node of our system. All the Business Logic is deployed here, possibly by means of an Application Server like GlassFish. This is the only node that interact with the Database node.
- *Data Server*: on this node are deployed all the artifacts necessary to manage the data.

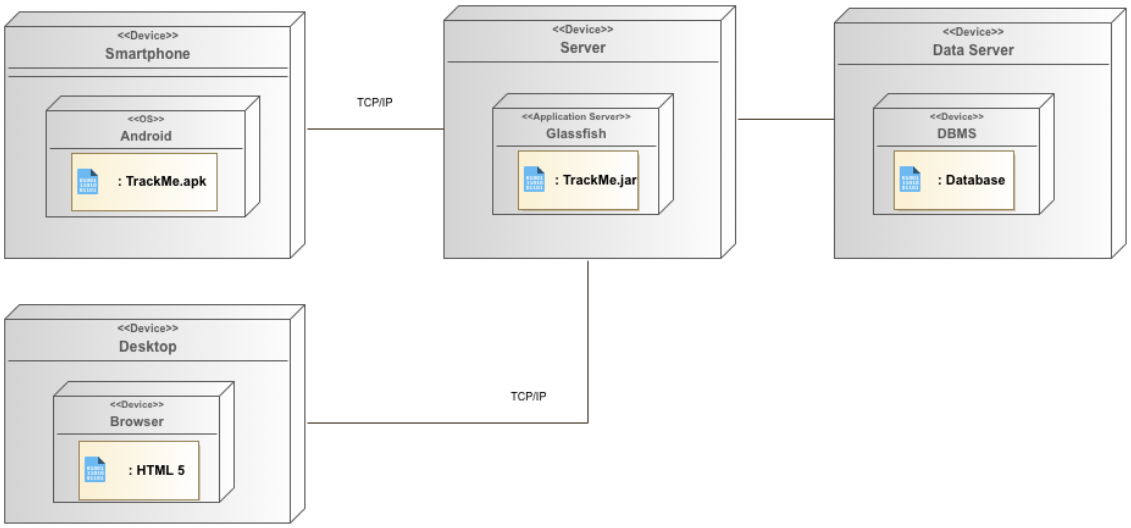



Figure 2.3: Deployment Diagram

## 2.4 Component Interfaces

The following diagram shows what features each component must provide to guarantee the correct functioning of Data4Help features, highlighting the dependencies among the various components. To make the diagram more clear, the crossing arrows have different colors and the external components are highlighted in orange. Following the diagram, each operation is defined through a brief description.

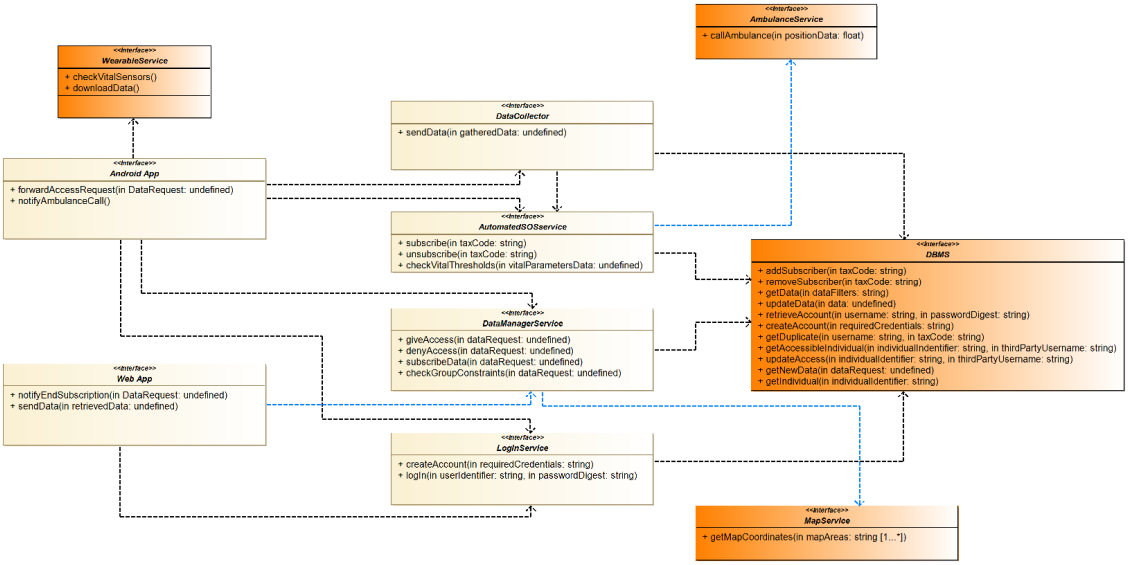



Figure 2.4: Interface Diagram

- Android App
  - **forwardAccessRequest**: forwards the third party data access request to the individual's application. The data request is taken as an input parameter because it is needed to retrieve all the information about the request: who is the requester, in which data he is interested in.
  - **notifyAmbulanceCall**: notifies that an ambulance has been successfully called.
- Wearable Service
  - **downloadData**: downloads the data stored in the paired wearable.
  - **checkVitalSensors**: checks if the wearable is capable of measuring at least one vital parameter, either the blood pressure or the heart rate.
- Web App
  - **notifyEndSubscription**: sends a notification to the Web App about the ended subscription of the data request passed as an input parameter.
  - **sendData**: sends the given data to the Web App.
- Data Collector
  - **sendData**: sends the gathered data from the Android App to the Data Collector.
- Map Service
  - **getMapCoordinates**: searches the coordinates of the border associated to the given map zone. These coordinates will be then used by the DBMS in order to apply filters to the stored data based on geographical zones.
- Data Manager

  - **checkGroupCostraints**: operation used to check if the group data found by the database applying the data request filters specified by the third party counts at least 1001 individuals.
  - **giveAccess**: allows the individual data gathering from the given third party request to be processed.
  - **denyAccess**: denies the individual data gathering from the given third party request to be processed.
  - **subscribeData**: activates the data subscription for the given data request.

- Login Manager
  - **createAccount**: uses the given credentials to create a Data4Help account.
  - **logIn**: allows users to access to Data4Help through their username (or tax code in case it is an individual) and password digest.

- Automated SOS
  - **subscribe**: activates the AutomatedSOS service for the individual associated to the given tax code.
  - **unsubscribe**: stops the AutomatedSOS for the individual associated to the given tax code.
  - **checkVitalThresholds**: checks if the data of the vital parameters have crossed the critical thresholds.

- Ambulance Service
  - **callAmbulance**: calls an ambulance, using the input coordinates to specify the destination.

- DBMS
  - **addSubscriber**: promotes the individual account associated to the given tax code as an AutomatedSOS account.
  - **removeSubscriber**: removes the AutomatedSOS subscription from the account bound to the tax code.
  - **getData**: uses the given data filters to retrieve the desired data.
  - **uploadData**: adds the given data to the database.
  - **retrieveAccount**: checks if the given credentials are in the database.
  - **createAccount**: adds a new user account to the database using the given credentials.
  - **getDuplicate**: retrieves a tuple matching the given username or tax code.
  - **updateAccess**: updates the list of all third parties that can have access to the individual data.
  - **getAccessibleIndividual**: finds the tuple containing the given third party username and individual identifier, from the table containing all the list of the third parties and the individuals of which they have data access.
  - **getNewData**: collects all the new data related to a data request.
  - **getIndividual**: retrieve the given individual from the DB.

Note that the operations associated to the external services are fictitious, the real names or input parameters may vary from the one listed above, but the expected functionalities must be the same, even if they are mapped on more than one real function.
In particulare the DBMS operations are just queries to the DB.

In this document the model of the application will not be provided: this makes the development phase more flexible.
However, to better coordinate the interactions between the component, a general description of the input provided in each operation listed before is here presented:

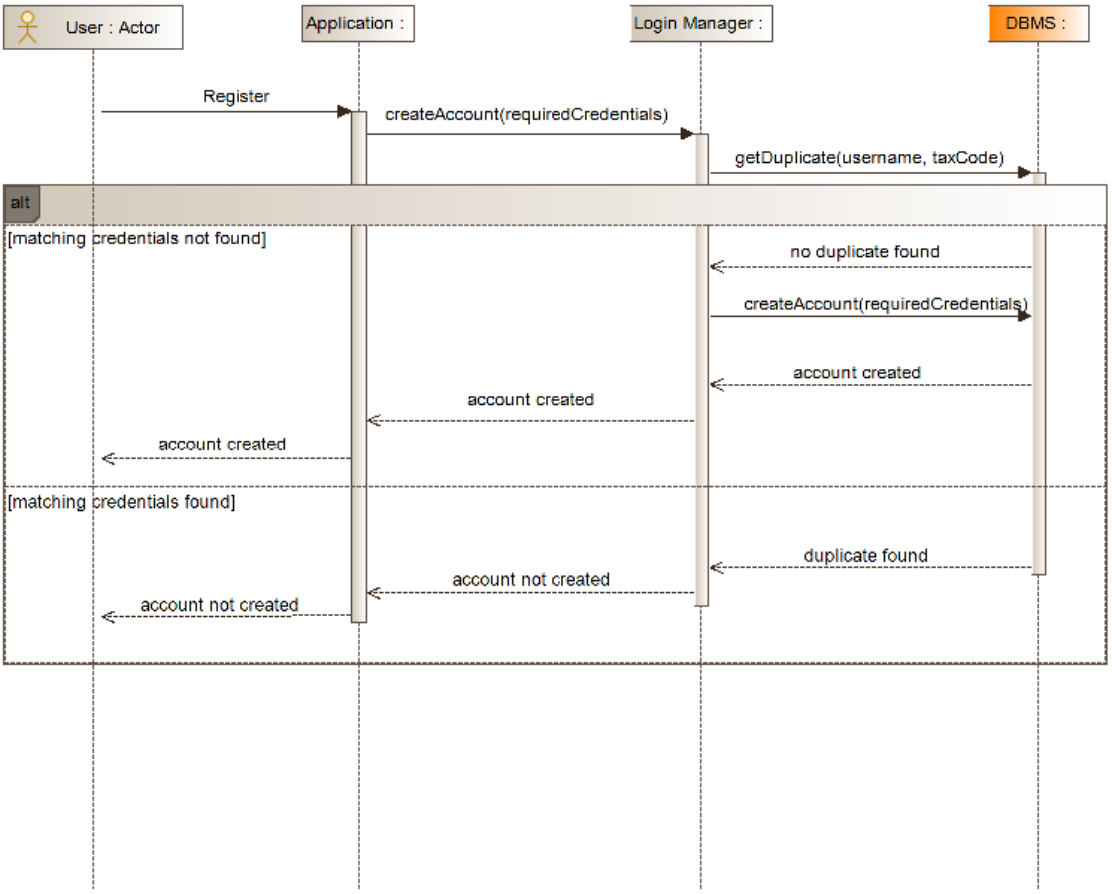
- requiredCredentials: this will be a data structure containing multiple strings such as the taxCode, the username or the passwordDigest.
- userIdentifier: this is a generic string that could be either the individual username or tax code, or the third party username.
- individualIdentifier: this is a generic string that could be either the individual username or tax code.
- dataRequest: this will be the data structure that represents the data request from the third parties. It will contain all the filters applied for the data request, an Id, a typology identifier for the request itself and the sender username.
- mapAreas: list of strings representing the geographical filter applied by the third party.
- dataFilters: this data structure contains all the filters that can be extrapolated from the dataRequest.
- retrievedData: data structure containing all the data retrieved from the DB.
- retrievedIndividual: tuple containing individual's indentifiers.
- gatheredData: data structure containing all the data downloaded from the individual's smartwatch.
- vitalParameterData: data structure containing all the individual data that is needed for the AutomatedSOS service.

## 2.5 Runtime view

The following diagrams shows the interactions between the interface components defined in the previous section in order to provide Data4Help functionalities. Remember that, since Data4Help is a three-tier application based on a client/server architecture it is fundamental that the user applications and the server are connected to the internet.

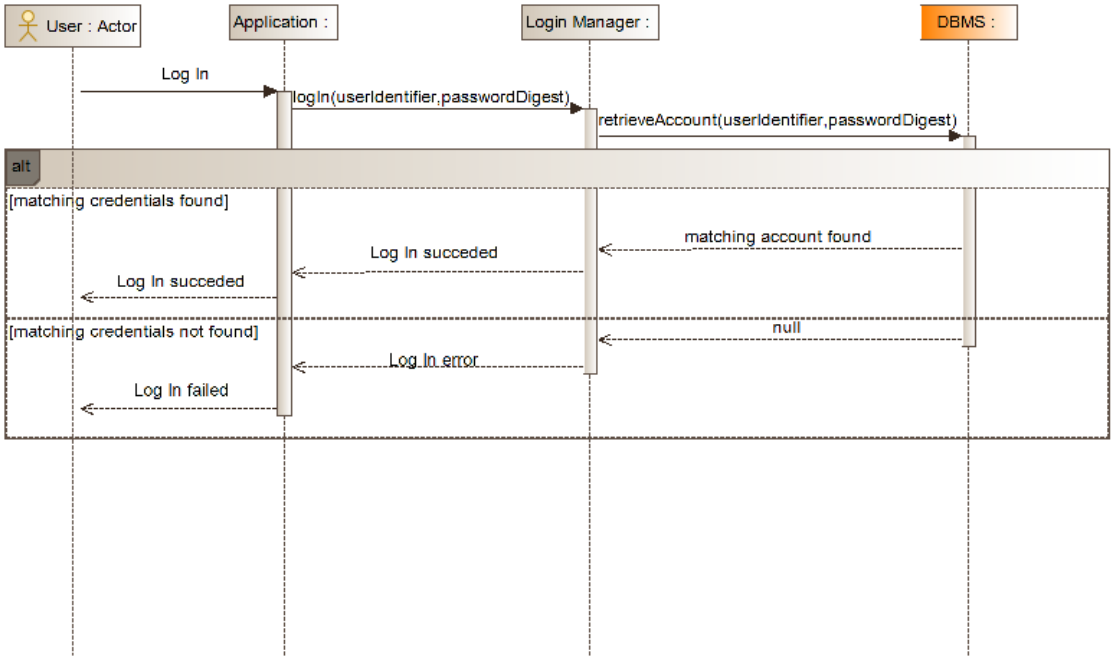
### 2.5.1 User registration




This diagram shows the procedure activated when the user registers to Data4Help.
The precondition of the registration is that the user has filled all the mandatory fields of the Sign Up form.
When the user presses the register button, the content of the compiled fields is sent to the Login Manager component to create an account.
Before creating the account, the LoginService checks through the DBMS services if the provided username and tax code are already present in the DB.
If the provided username or tax code is already taken, the procedure terminates sending an error notification to the client, otherwise the Login Manager adds the new account credentials in the DB through the DBMS.
After the account is created a notification is sent to the user and the procedure ends.

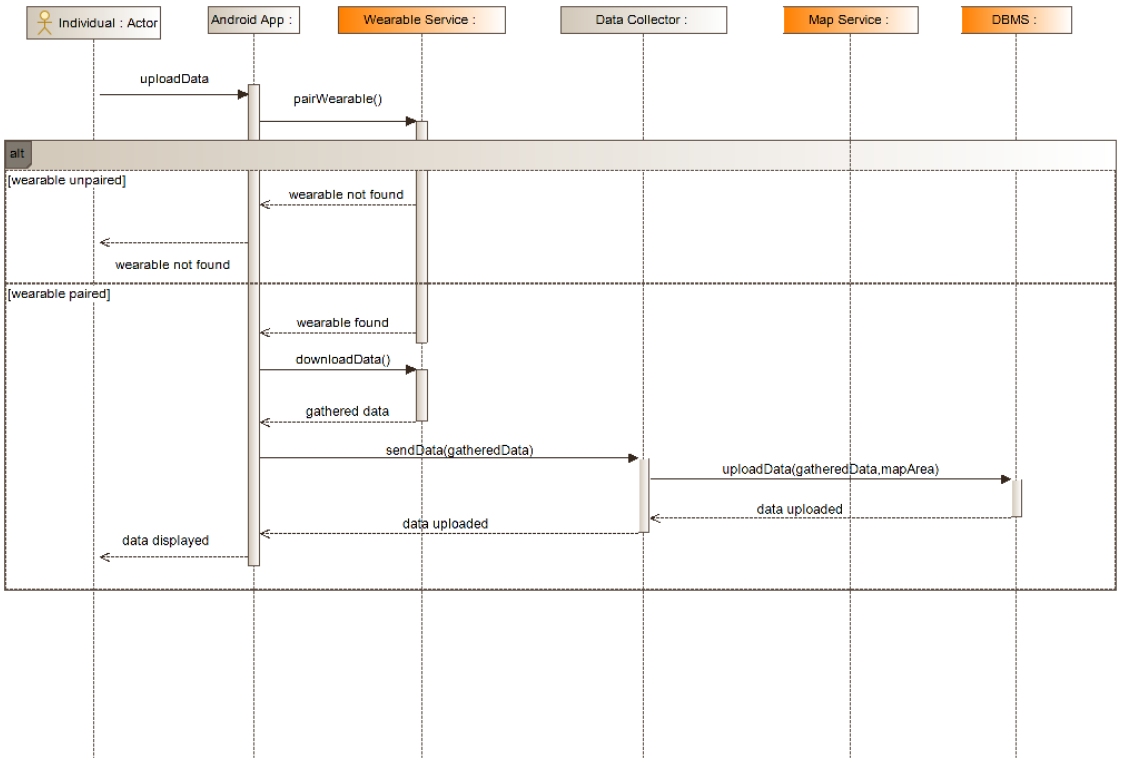
### 2.5.2 User log in




In this diagram it's presented the Log In procedure.
Once the user has pressed the Log In button, the username and the password digest are sent to the Login Manager.
This component checks through the DBMS if there exists an account matching the credentials provided: in the positive case the user is logged in and directed to the respective data visualization screen, otherwise an error notification is sent to the user.
Notice that, in order to maintain user privacy the password is encrypted immediately with an hash function, and only its digest is used during this procedure.

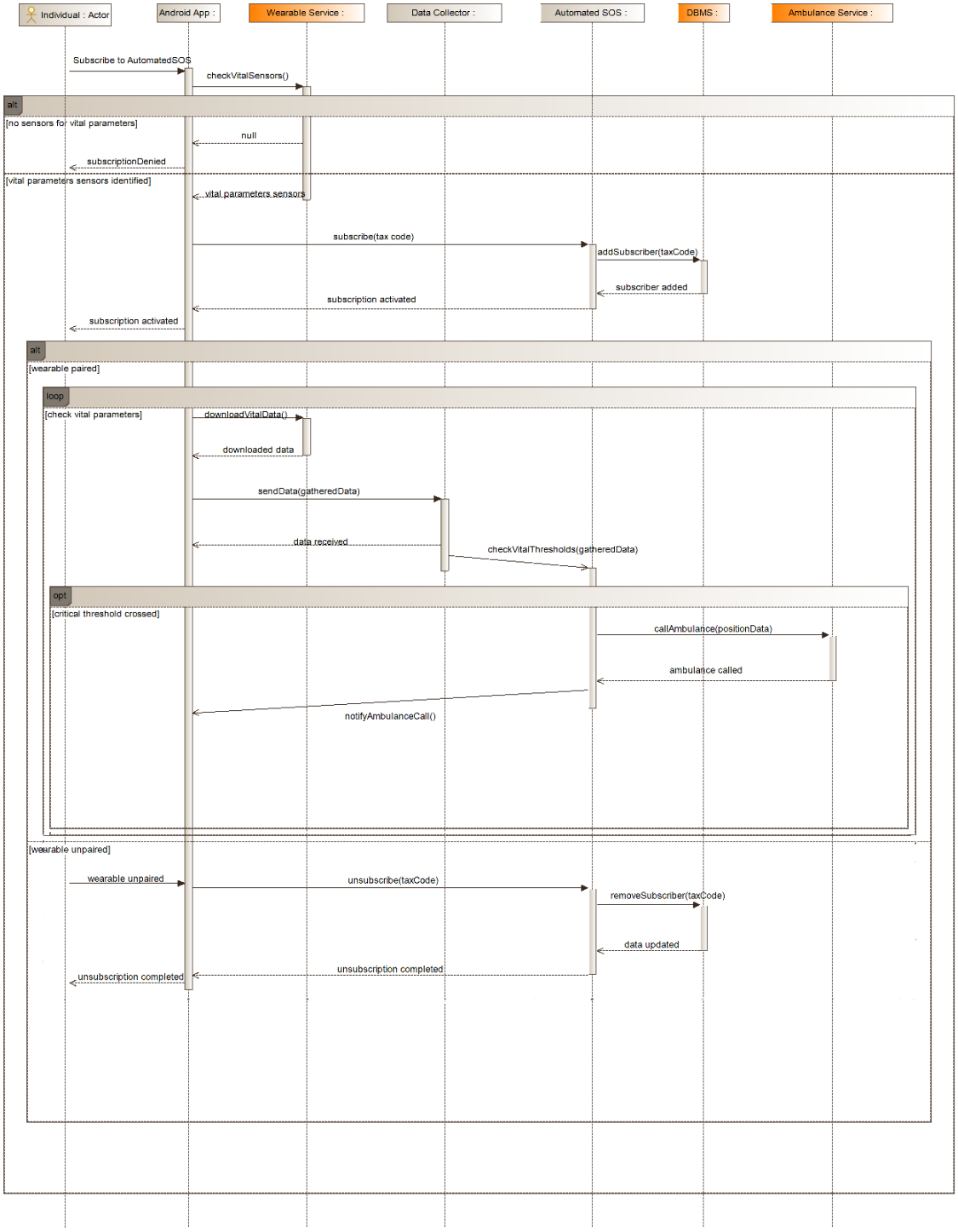
### 2.5.3 Individual data upload




This diagram shows the data upload procedure, including the wearable pairing phase.
Obviously the individual must be logged in, in order to start the data upload.
When the individual goes to the data visualization screen the wearable pairing, using the external wearable service, starts.
If the wearable is not found the procedure terminates sending an error notification to the application. In the other case, once the wearable is paired, the application downloads the data gathered from the smartphone through the wearable services.
The downloaded data is then sent to the server through the Data Collector, that uploads the data into the DB through the DBMS.
After the data is uploaded, a notification is sent to the Android App which now displays the new data.

### 2.5.4 AutomatedSOS subscription



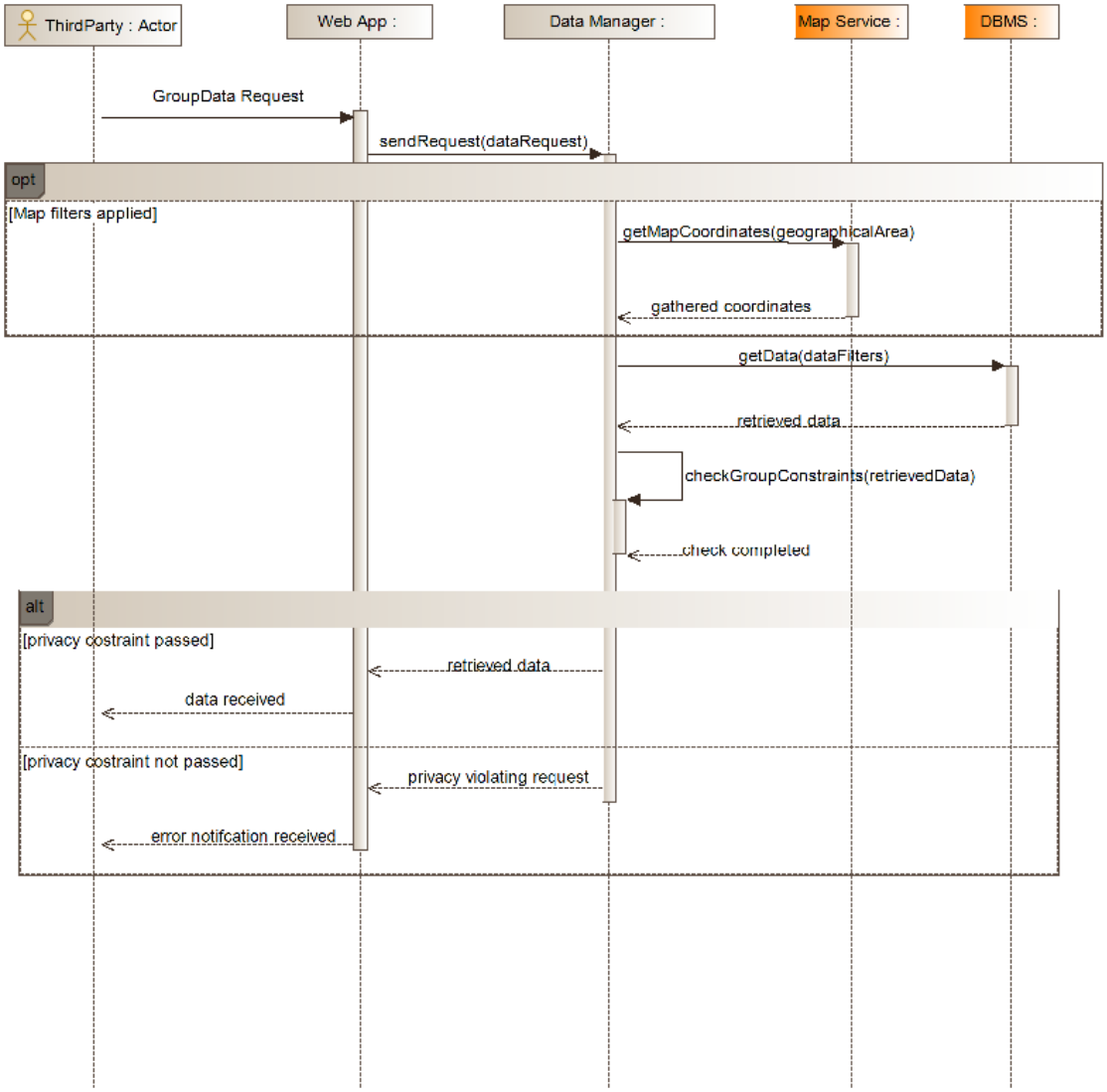

This diagram illustrates the AutomatedSOS subscription procedure that is activated once the individual has paired the wearable and the subscribe button is tapped.
After the subscribe button is tapped, the application checks through the wearable external interface if the wearable is capable of measuring either the blood pressure or the heart rate.
In the negative case the procedure ends sending a notification to the individual. In the

other case, the application starts the subscription by using the Automated SOS component, that updates the DB setting the individual as an AutomatedSOS subscriber. Once the individual is set as a subscriber a notification is sent to the application, notifying the start of the service.
After the service is started, the application regularly downloads the vital parameter data using the wearable external service, enriches it with the individual position data and sends it to the data collector.
The data collector checks if the values of vital data collected have crossed the critical threshold using the Automated SOS component: if the critical threshold of a vital parameter is crossed this component calls an ambulance using the external Ambulance APIs, giving the coordinates to set the destination of the ambulance.
Once the ambulance is called a notification is sent to the user application.
The AutomatedSOS service is active as long as the wearable is paired: when the wearable unpairs the application starts the unsubscribe operation of the Automated SOS, which updates the DB by removing the subscription, and notify the user about the ended subscription.

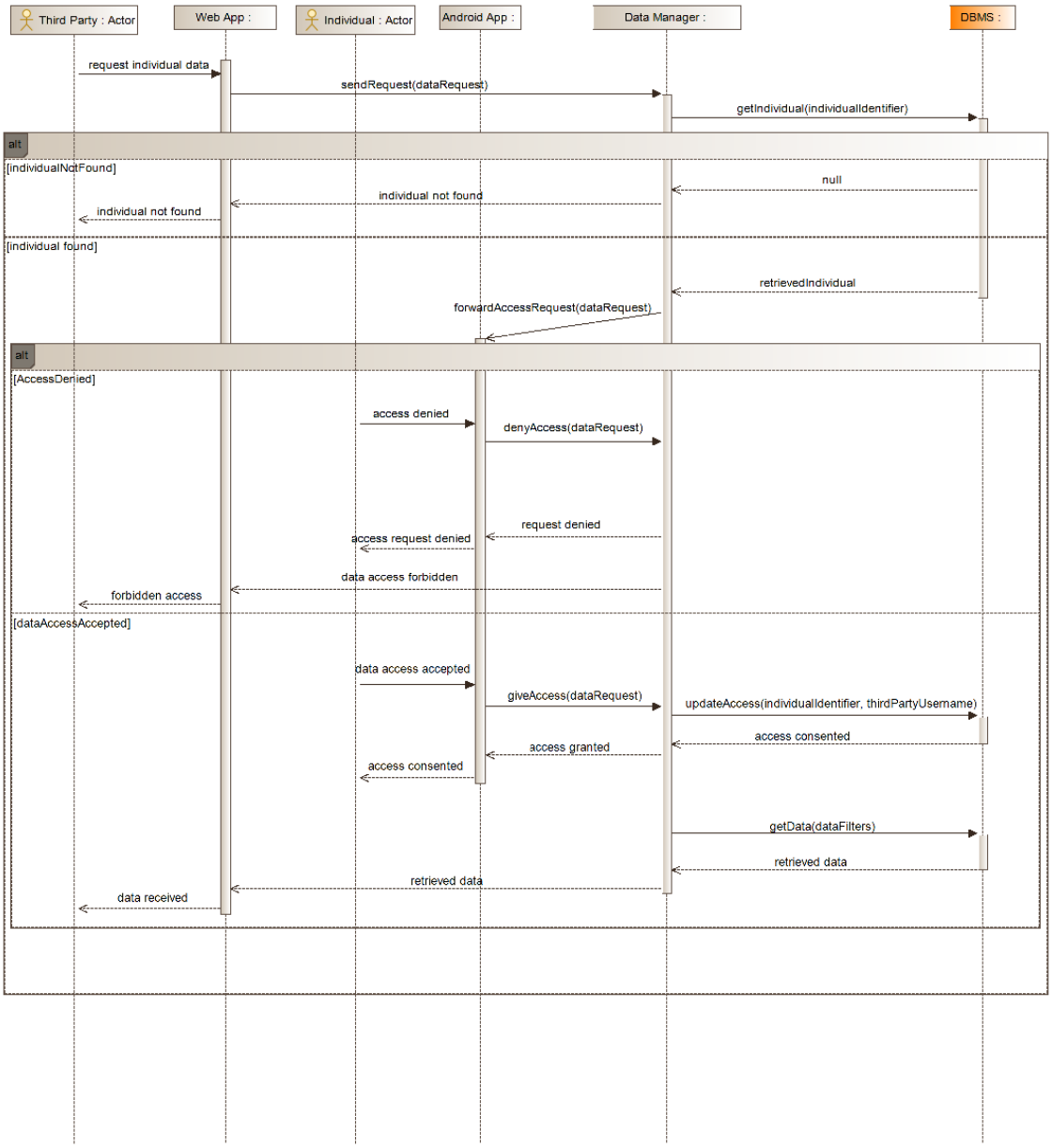
### 2.5.5 Group Data Request




This diagram shows the interactions needed when a group data request from a third party occurs.
When a third party has filled the group data request form and clicks on the send button, the Web App sends the request to the Data Manager component.
This component retrieves the filters from the data request: if a geographical filter is applied, it uses the external map services to transform the given geographical filter in the corresponding border coordinates on the map.
Once the filters are retrieved it then uses the DMBS to gather the requested data.
After the requested data is received, it checks if the privacy constraint is respected, by counting the number of individuals that are part of the selected data group: if it is above 1000 the data is forwarded to the WebApp, otherwise the request is denied and a notification is sent to the Web App.

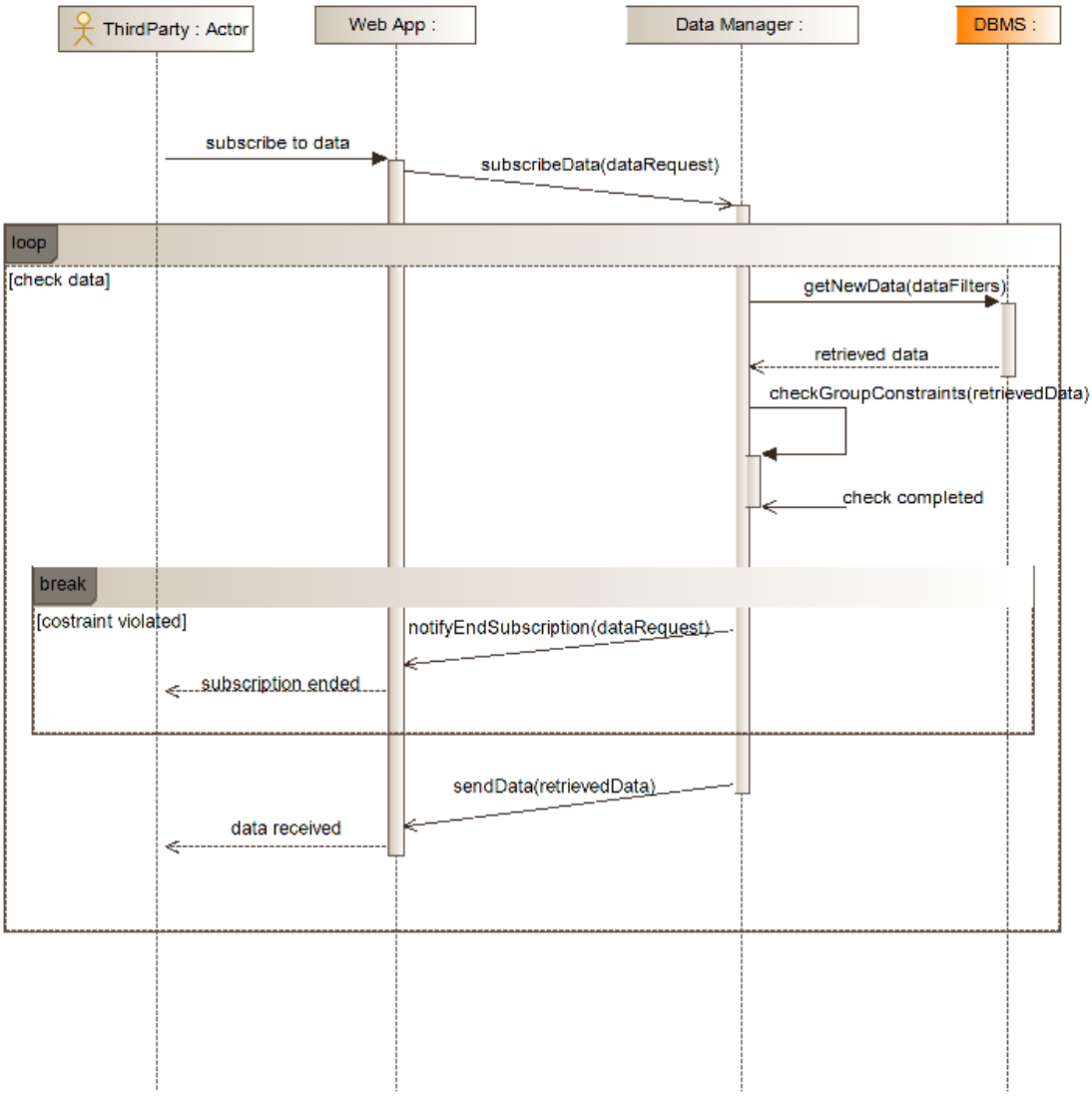
### 2.5.6 Individual Data request




In this diagram the individual data request is presented.
Similarly to what is done with the group data request, once the individual data request form is filled and the send button is clicked, the web application sends the request to the Data Manager component.
The first thing done by the Data Manager is to retrieve the individual requested from the DB: if he is not present a notification error is sent to the web application.
In the other case, the Data Manager forwards the access requests to the individual application. The individual can choose to give the direct access of his data to the third party or deny it through the Android App: if the access request is refused the Android App notifies the Data Manager about the decision, which sends a notification to the third party.
If the individual allows the data access, the Data Manager sends the individual's data, retrieved through the DBMS, to the Web App.

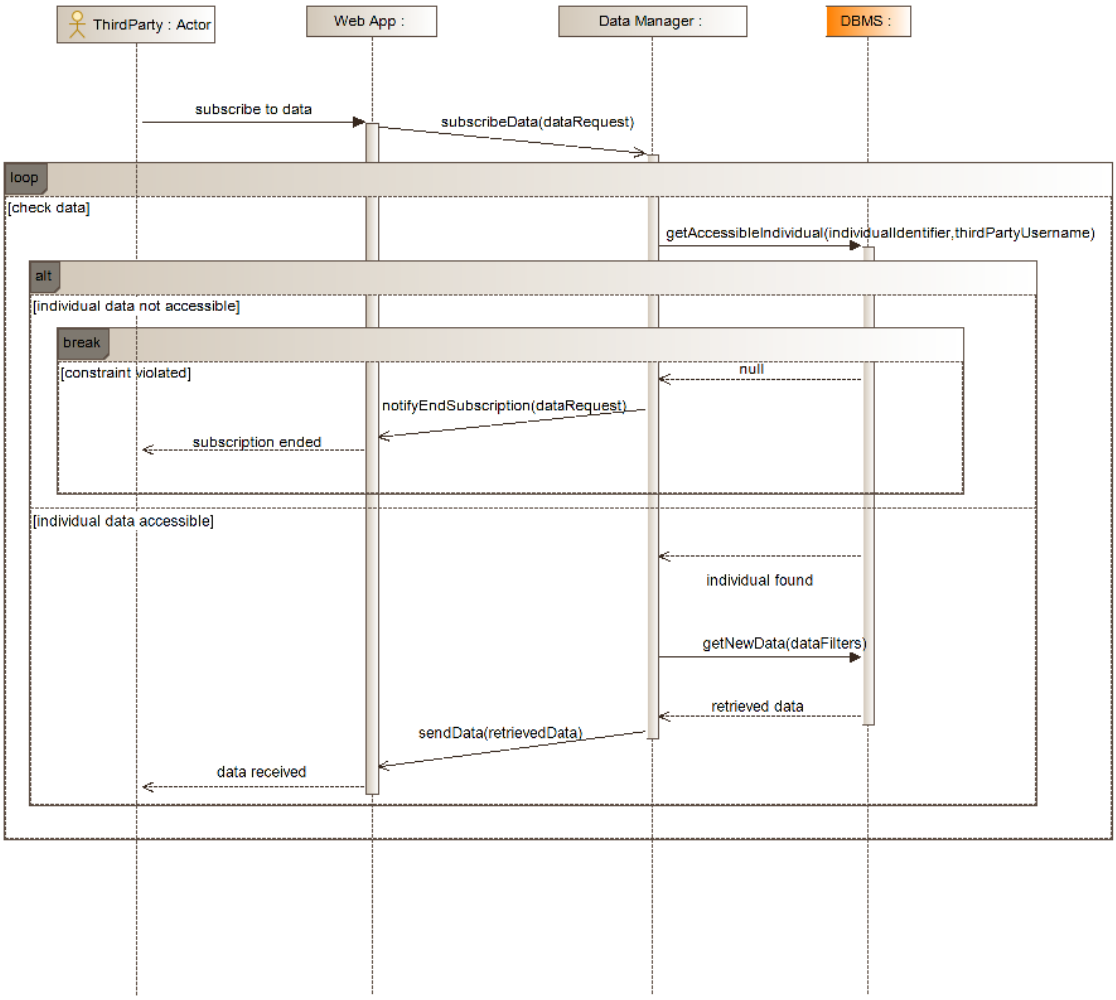
### 2.5.7 Group Data subscription




The above diagram shows the data subscription procedure that is activated when a third party decides to subscribe to a data request.
The Web App starts the subscription through the Data Manager component. Once the subscription is activated the Data Manager regularly checks if there is new data matching the data request to which the third party is subscribed to using the DBMS services.
If new data is found, the Data Manager checks if the privacy constraints are still active (both if the request is a group data request or an individual data request).
If the constraints are respected the new data is sent to the web application, otherwise the subscription is ended and the third party is notified about it.

### 2.5.8 Individual Data subscription




The individual data subscription procedure is very similar to the group one.
The only difference is that the regular check of the privacy constraint is done before retrieving the data, through the DBMS. Indeed, the DB contains the list of all the individuals whose data is accessible by each third party.

## 2.6 Selected architectural styles and patterns

### 2.6.1 Overall Architecture

Data4Help is based on a three tier architecture: almost all the system logic is executed in the server component, with the help of some external services. The client applications mainly focus on presenting the information delivered by the server, however some functionalities may be delegated to it, for example the wearable pairing, which is handled by the individual application in order to give a faster response. The data layer is handled by a single, well organized, DB. This choice was made to avoid data fragmentation among different databases.

Moreover, the three tier architecture allows to make changes to any of the three layer without causing to much trouble to the other components: this is a fundamental aspect for system like Data4Help that continuously expands its functionalities.

As stated in the RASD the availability is expected to be at 95% of the annual time. Indeed there are no backup options in the system, all the component are in series with the server being the bottleneck due expected maintenance.
Obviously this value has to be improved over time, by acquiring new components to put in parallel to the old ones.

### 2.6.2 Design Patterns

The main design pattern used to build Data4Help is the ModelViewController design pattern. This design pattern can be easily mapped to the client/server architecture used for Data4Help: the view part is obviously handled by the client applications, while the controller splits between the server and the client in order to pick the user input from the client and transform it into actions on the model stored in the server.
The use of other design patterns, more related to the implementation, are left to the developers.

### 2.6.3 Other design decisions

The main philosophy about the design process to Data4Help is to keep the system as simple as possible, since it offers few but well defined functionalities. This is useful to have a fast system, that can be developed quickly.
Furthermore, Data4Help strongly relies on external services:

- The Wearable external service is used to handle the pairing and the data download from the individual smartwatch. Initially the application will be build using Google wearOS APIs, but more smartwatches will be made compatible in the future.
- The Map external service is used to translate geographical area names into a range of coordinates. Many options are available, and will be all considered in the development phase.
- The Ambulance external service is used to call the ambulance. This services are provided by the local ambulance associations.

# Chapter 3

# User Interface Design

The general guideline to develop the user interface for the client applications is too keep the interface as simple as possible while providing all the needed functionalities.
The style used to build the user interface mockups is based on the Material Design by Google, however those mockups serve purely as a guideline for the development of the interface. As stated in the RASD, the final product style may differ from them, but the provided functionalities must be the same. To have a better idea of how the user interface will look like when the applications will be finished please refer to the section 3.1.1 of the RASD.

# Chapter 4

# Requirements traceability

The goals of the system was specified in terms of domain assumptions and requirements in the RASD document. In the following, the requirements are mapped to one or more design component specified before.

- **[R1] Users are able to login with the username and the password associated to their account.**
  - Android App
  - Login Manager
  - DBMS
- **[R2] Data4Help is able to pair with the user wearable.**
  - Android App
- **[R3] Data4Help is able to download eHealth data from the individual's wearable.**
  - Android App
- **[R4] Individuals can upload their data through Data4Help app.**
  - Android App
  - Data Collector
- **[R4.1] Data4Help is able to store the data provided by individuals.**
  - Data Collector
  - DBMS
- **[R4.2] Data4Help is able to organize data provided by individuals.**
  - Data Manager
  - DBMS
- **[R5] Third parties can formulate requests to access anonymized data of groups.**
  - Web App

  - Data Manager

- **[R6] Third parties can apply filters on data while formulating their request for data of groups.**
  - Web App
- **[R7] The system checks if the number of individuals in the group detected by the third party request is higher than 1000.**
  - Data Manager
  - DBMS
- **[R7.1] If the groups has 1000 or less individuals the system denies the request.**
  - Data Manager
  - DBMS
- **[R8] The system is able to distribute the requested data to the third party.**
  - Data Manager
  - DBMS
  - Web App
- **[R9] Third parties can formulate requests to access specific individual data, indicating the individual's TC or username.**
  - Web App
  - Data Manager
- **[R10] Third parties can apply filters on the type of data while formulating their request for individual data.**
  - Web App
- **[R11] Data4Help is able to forward the request to the individual specified by the third party.**
  - Data Manager
  - DBMS
  - Android App
- **[R12] The individual to whom the request will be forwarded is able to reply.**
  - Android App
- **[R12.1] The system notifies the third party about the individual reply.**
  - Data Manager
  - Web App

- **[R13] The system can check if the individual has accepted the data access request by the third party.**
  - Data Manager
  - DBMS
- **[R14] Third parties can specify to be updated whenever new data of groups detected by their data requests, or observed individuals is gathered. The time interval between an update and the next one can be set by the third party, in a limit that goes from one hour to 1 year.**
  - Web App
  - Data Manager
- **[R15] The system is able to update third parties with new data, respecting their preferences on the interval timing.**
  - Data Manager
  - DBMS
  - Web App
- **[R16] Individual can subscribe to the AutomatedSOS service.**
  - Android App
  - Automated SOS
  - DBMS
- **[R17] The system is able to recognize if the wearable can measure at least a vital parameter.**
  - Android App
- **[R18] The system is able to continuously monitor individual vital parameters.**
  - Android App
  - Data Collector
  - Automated SOS
- **[R19] If the wearable is disconnected the AutomatedSOS service is suspended, untill the individual connects his wearable again.**
  - Android App
  - Automated SOS
  - DBMS
- **[R20] Whenever a vital parameter cross its respective threshold, the system calls an ambulance in under 5 seconds, giving to the ambulance driver the location of the individual.**
  - Automated SOS

- **[R21] The system doesn't allow third parties to access specific individuals data without first asking for their permission.**
  - Data Manager
  - Android App
- **[R22] The system stops to update third parties about new data whenever observed individuals decide to remove data access permissions, or whenever the number of individuals of an observed groups goes belove 1001.**
  - Android App
  - DBMS
  - Data Manager
  - Web App
- **[R23] Only users that know their username and password can access to their respective accounts**
  - Android App
  - Web App
  - Login Manager
  - DBMS

# Chapter 5

# Implementation, integration and test plan

## 5.1 Implementation plan

The implementation of Data4Help will be done component by component, developing the most critical components first. The implementation order will be the following:

1. Data Manager: this is the most important component of the system, since it is the one that handles the data requests from third parties and translates this requests in queries for the DBMS. This is also the component that creates a direct communication between the third parties Web App and the Android App, when third parties send an individual data request.
   Furthermore, this component is the one that interacts with the Map Service, in order to retrieve the border coordinates associated to the optional geographical filters.

2. Data Collector: this component is the one that uploads the received data to the database. The Data Collector is also critical, all the individual's data will pass through it while using Data4Help. When the individual is subscribed to Automated SOS, this component interacts with the AutomatedSOS component to check the thresholds and, in case of necessity, call the ambulance.

3. Automated SOS: this component is the one that handles all the functionalities linked to the AutomatedSOS service such as the subscription, the vital parameters threshold check and the ambulance call. Of course, to call the ambulance, it has to interact with the ambulance external service.

4. Login Manager: this will be the last server side component to be implemented since it is not related to Data4Help main functionalities.

5. Android App: once the server side is ready, the Android App can be developed. The Android app will be developed before than the Web App because it has some critical aspects like the wearable pairing and the AutomatedSOS subscription.

6. Web App: this will be the last component to be implemented, and will be the one that will let the third parties formulate their data requests.

## 5.2 Integration and testing

### 5.2.1 Entry criteria

The integration and testing plan will follow the implementation plan using a bottom up approach: each component will be fully tested as soon as it is finished and will be integrated with other related components, creating a subcomponent of the whole system that will be again tested to make sure everything works fine.
More precisely:

- Data Manager: this component will be tested by simulating third parties data requests.
- Data Collector: this component will be tested by simulating individual's data upload.
- AutomatedSOS: this component will be tested by checking if the ambulance is called by analyzing simulated vital data. After it is tested, it will be integrated with the Data Collector, and the interaction between the two components will be tested to see if the data delivered to the Data Collector is correctly analyzed by AutomatedSOS.
- Login Service: this component will be tested with simulated account creations.

All the components above development will include the integration with the needed external services, since they strongly rely on them to guarantee their respective services.
Once the server side is ready, the applications will be developed, integrated with the server and then tested to see if they can correctly use the server functionalities.

### 5.2.2 Components to be integrated

In our system, components must be integrated with the DBMS, with external services, among them and finally the clients must be integrated with the business logic server.

#### Components to be integrated with the DBMS

All the components of the server interact with the DBMS, therefore it is necessary perform integration testing between them and the DBMS in order to be sure that the components update correctly the DB and/or their queries are well formulated and the results correctly interpreted.

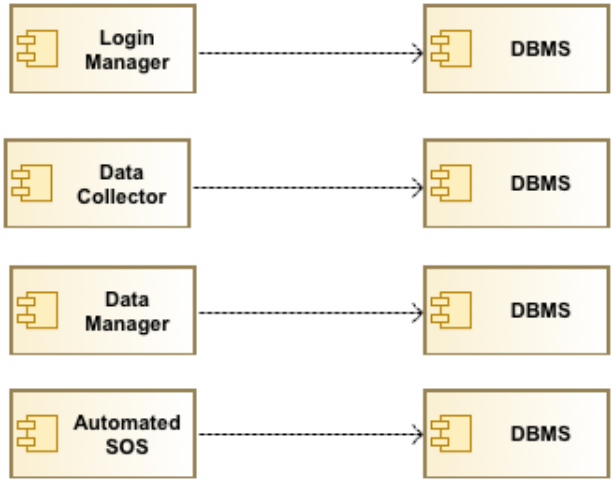



Figure 5.1: Components to be integrated with the DBMS

### Components to be integrated with external services

Our system uses several external service in order to provide its functionalities. In particular, the Automated SOS component must be integrated with Amulance Services, the Data Manager with a Map Service and the Android App with Wearable Services.

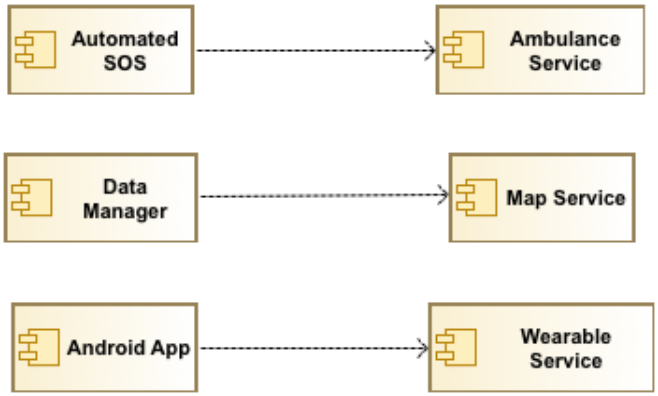



Figure 5.2: Components to be integrated with external services

### Components to be integrated within the server

Only the Data Collector and the Automated SOS components directly interact inside the server. Their integration test aims to guarantee that vital data of the SOS service subscriber, and only that data, is forwarded from the Data Collector to the Automated SOS.

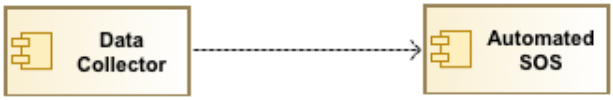



Figure 5.3: Components to be integrated within the server

### Integration clients/server

Finally, the interaction between the clients and the server must be also tested. These tests must assure that the communications are correctly carried on and all the various interfaces must be checked. This will be the last step of the integration testing because the TrackMe server is seen as a single component, therefore its subcomponents must be already integrated and tested.

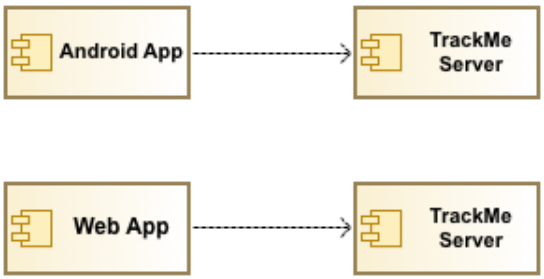



Figure 5.4: Integration clients/server

# Chapter 6

# Effort spent

In the following tables the time spent for each section of the project is presented. For the efforts spent on user interfaces, please refer to the RASD.

**Stefano Pecchia**

| Section | Hours |
|---|---|
| Introduction, Component view, Deployment view, Requirements traceability | 3 |
| Component interfaces, Selected architectural styles and patterns | 8 |
| Runtime view | 12 |
| Implementation, integration and test plan | 4 |

**Edoardo Peretti**

| Task | Hours |
|---|---|
| Introduction, Component view, Deployment view, Requirements traceability | 11 |
| Component interfaces, Selected architectural styles and patterns | 2 |
| Runtime view | 6 |
| Implementation, integration and test plan | 4 |

# Chapter 7

# References


- Specification document "Mandatory Project Assignment AY 2018-2019.pdf"
- Requirements analysis and specification document, version 1.1 "RASD2.pdf"
- Material design by Google: https://material.io/design/
- Tool for uml diagrams: https://www.modelio.org/