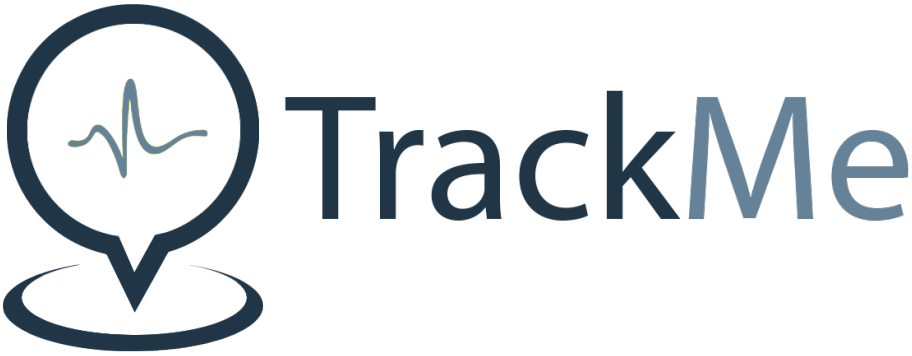
POLITECNICO
MILANO 1863

M.Sc. Computer Science and Engineering
Software Engineering 2 Project



Design Document

Gargano Jacopo Pio, Giannetti Cristian, Haag Federico

10 December 2018

GitHub Repository: https://github.com/federicohaag/GarganoGiannettiHaag

Version 1.0

Contents

Chapter 1

Introduction

1.1 Purpose

TrackMe wants to offer a service named "Data4Help" on top of which two services, named "AutomatedSOS" and "Track4Run", will be built.

Data4Help will be a system collecting data of *Users* through a *Smart wearable* connected to their smartphone. This data may be sent to *Third parties* after *User* consent. AutomatedSOS will constantly monitor *User data* to allow for immediate assistance. Track4Run will allow individuals to organize running competitions, to enroll in or watch one.

More details may be found in section 1.1 of [3].

In this document, the design of the system to be will be explained and analyzed in detail. This includes the identification of all the components and the design decisions regarding the structure and the patterns to be implemented. Moreover, implementation and testing will be discussed and planned.

1.2 Scope

TrackMe proposes to offer its system in a world in continuous technological evolution, where almost everyone owns a smartphone and is always connected to the Internet. By using Data4Help, *Users* will integrate their everyday activities with constant collection of their health data and position, through sensors of their *Smart wearable* and smartphone GPS. By adding *Third party Services* they may benefit of data monitoring and analysis with useful insights on their daily activities.

Users may enhance data collection by adding AutomatedSOS to their *Services*. This *Service*, by constantly monitoring their data, will allow for *Anomalous data*identification and immediate assistance for the *User in need*. This can

be crucial for individuals with health issues and elders living alone. People fond of running have the possibility of enrolling in a *Run* listed in Track4Run. Their data, including position and health data, will be shown to *Spectators*, who can watch the *Run*. The process of organizing running competitions will be carried out by *Organizers* through Track4Run.

More details may be found in section 1.2 of [3], including a detailed analysis of shared phenomena.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

User: registered individual of Data4Help who agreed on the acquisition and processing of their data (see *User data*).

User data: *User*'s health data and location acquired by Data4Help.

Third party: a company that is willing to access *User data* stored in TrackMe's database.

Service: application available for Data4Help *Users*, generally offered by a *Third party*.

Group data: set of *User data* acquired by Data4Help. The set of *Users* is determined by specific characteristics and constraints defined by the *Third party* requesting the data. When sent to the *Third party*, this data is anonymized.

Smart wearable: smart devices that can be worn on the body as accessories. These devices are required to have specific sensors for data acquisition. They must be connected to an external device, such as a smartphone.

Anomalous data: health data that is outside certain intervals identifying a *User* normal health condition.

User in need: registered user of AutomatedSOS in need of assistance since their health data is *anomalous*.

Run: running competition registered on Track4Run.

Organizer: person organizing a *Run*.

Spectator: person participating as spectator of a *Run*.

Participant: *User* who added Track4Run to their services, participating in a *Run*.

1.3.2 Acronyms

GPS: Global Positioning Service

1.3.3 Abbreviations

G_n: n^{th} goal

R_n: n^{th} requirement

1.4 Revision History

1. Version 1.0 - 10^{th} December 2018

1.5 Reference Documents

- WORK IN PROGRESS

1.6 Document Structure

This document is divided into seven chapters.

First chapter In this chapter is summarized the project. This has been already introduced and described in RASD so here it is shown briefly only in terms of purpose and scope. Moreover, the chapter contains the definitions, acronyms and abbreviations needed to properly understand the sections following. All the documents used during the development of this project are listed at the end of the chapter.

Second chapter In this chapter is described a proposal of architecture able to deliver an implementation of the system to be described in the RASD. The architecture takes into consideration all the goals and requirements listed in the RASD. A particular attention is dedicated to the satisfaction of non functional requirements (e.g. reliability, performances). The architecture is described following a top-down approach: from overview to detailed components descriptions and interaction flows.

Third chapter In this chapter are listed and showed all the user interfaces of the system to be. The description is given with a particular attention for details and interaction between different interfaces.

Fourth chapter In this chapter requirements defined in the RASD are mapped to the design elements defined in in Chapter 2.

Fifth chapter In this chapter is presented the list of activities needed to implement the system to be, according to the architecture showed in Chapter 2 and according the user interfaces showed in Chapter 3. Activities are analyzed to identify eventual constraints that force certain activities to be executed before or after other else. A rough estimation and estimation is given.

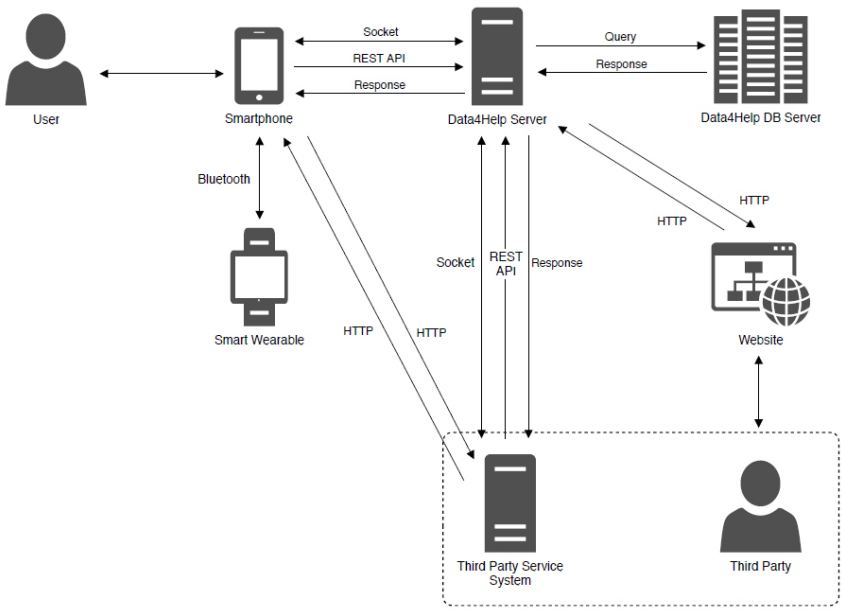
Sixth chapter Effort spent by all team members is shown as the list of all activities done during the realization of this document.

Seventh chapter References of documents that this project was developed upon.

Chapter 2

Architectural Design

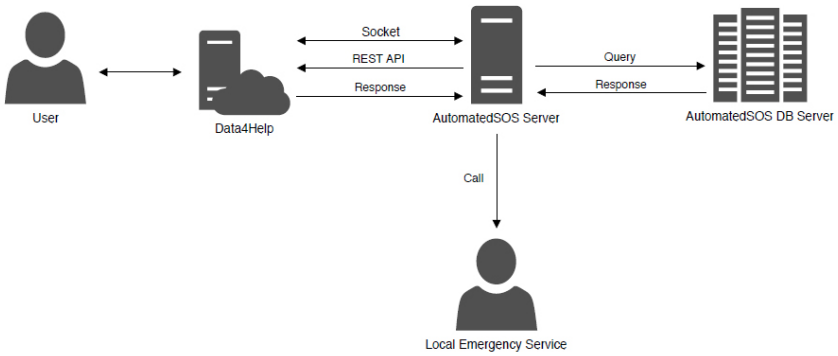
2.1 Overview



Data4Help Overview

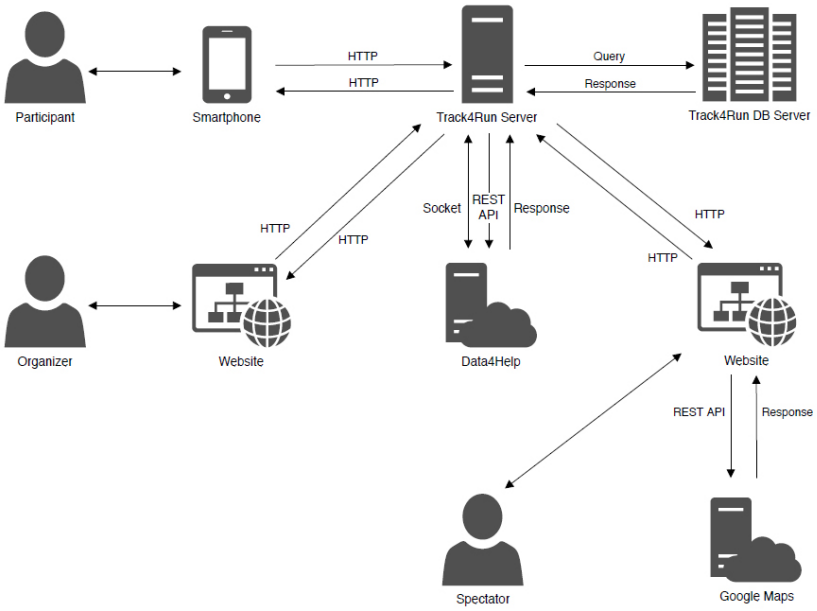
This diagram is a general overview of the system to be.
The *User* interacts with their smartphone performing several actions, including but not limited to managing their services, sending their consent, and using a service. The smartphone is constantly connected to the Data4Help Server both calling it through its REST APIs and through a Socket connection so as to send new data as soon as it is collected(SPIEGARE MEGLIO SOCKET).

The smartphone is also connected to the *User's Smart wearable*, allowing for continuous data collection.
The Data4Help Server is the core of the system to be. It manages all *User* requests, collects its data and sends them to the database - DB Server through completare. It also sends through a Socket connection newly collected *User data* only to those *Services Users* are subscribed to. It allows the forwarding of *Third Party User data* requests to *Users* and supports the interaction of *Third Parties* with the system through their dedicated website. Lastly, the Third Party Service System communicates with the *User's* smartphone via HTTP. SPIEGARE



AutomatedSOS Overview

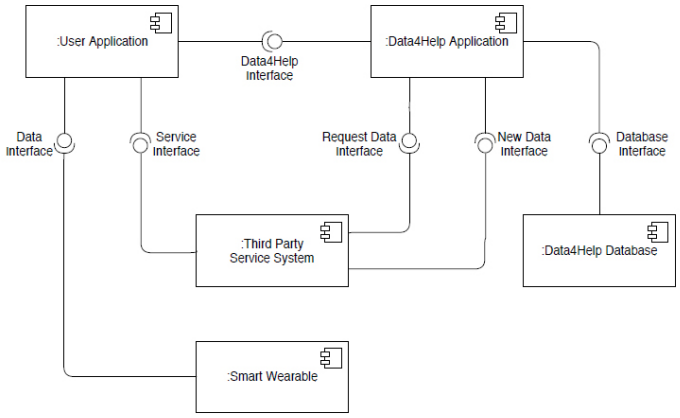
AutomatedSOS is based on a central server which receives *User data* as soon as it is produced and constantly analyzes it. The data is sent from Data4Help to AutomatedSOS via a Socket connection always active. When AutomatedSOS recognizes a *User* as a *User in need*, it calls Local Emergency Services and store in the AutomatedSOS database all the relevant information, including which which Local Emergency Services are in charge of assisting a specific *User in need*.



Track4Run Overview

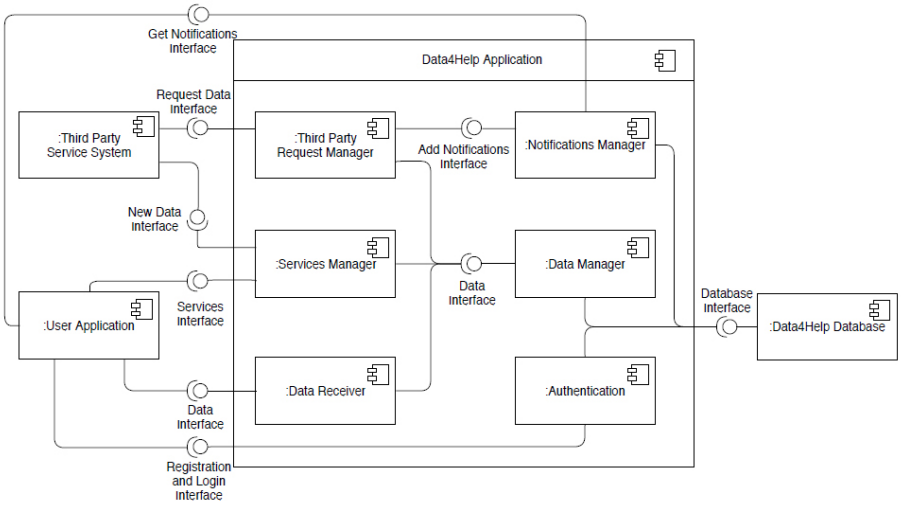
The overview of Track4Run gives a general idea of the several components that comprise this *Service*.
The Track4Run Server manages all the actions that can be performed by the authors in this context. First of all, it allows *Organizers* to create a *Run* and manage it through a dedicated website. Moreover, it communicates with Data4Help in order to allow *Users* to enroll in a *Run* as *Participants* and acquire their real time data while running. This data will be displayed in the dedicated *Spectators* website, together with the real time position of *Participants*, retrieving the real world map of the *Run* through Google Maps. All data that needs to be stored, is passed to and retrieved from a database - DB Server. This data is mainly made up of *Run* information.

2.2 Component View



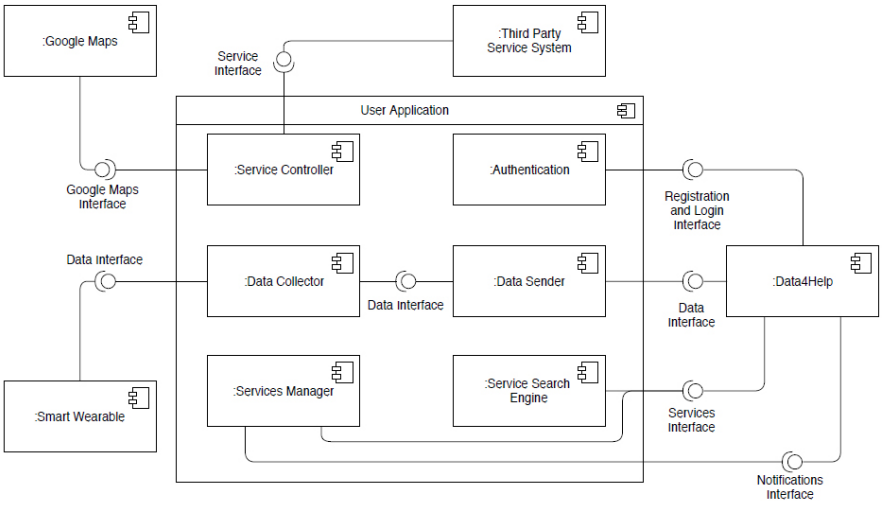
Data4Help Component Diagram Overview

A general view of the main components of the system to be is given. The core components are the Data4Help User Application and the server side Application. They communicate through three different interfaces offered by Data4Help. These will be further explained in the following two diagrams. Here they are simplified under the Data4Help Interface. The User Application receives data from the Smart Wearable through the Data Interface and pushes them to Data4Help. It finally uses the Service Interface to retrieve the *Service* website and interact with it. The server side Application provides an interface for the Third Party Service System to send data requests. It also sends newly collected data to the Third Party Service System through the New Data Interface. It finally communicates with the database through the offered Database Interface.



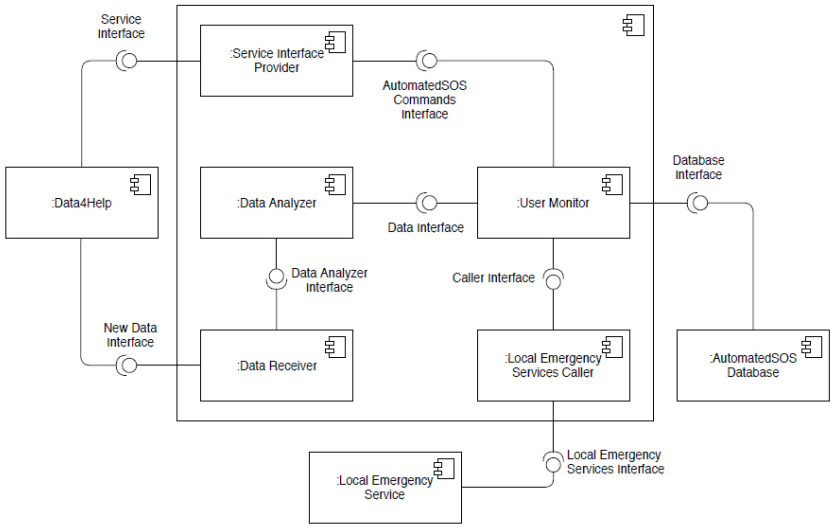
Data4Help Component Diagram

These are the internal components and interfaces of the Data4Help server side Application, together with the interfaces offered to external components. The Authentication component is needed to authenticate the *User* through the User Application. It needs to access the database where all login information is stored through the Database Interface. The Data Receiver is needed to receive data from the User Application as soon as it is collected. It then sends it over to the Data Manager through the Data Interface, whose function is to manage all *User data* stored by Data4Help. In fact, it uses the Database Interface to query it. The Data Manager has a crucial role when it comes to *Third party* requests. A *Third party* may send a request through the Request Data Interface. The request is processed by the Third Party Request Manager, whose role is further explained in section 2.4. Furthermore, the Notifications Manager is needed to provide notifications to the User Application. In fact, the application constantly queries the Data4Help Server for new notifications through the Notifications Interface. The Notifications Manager interacts with the database to retrieve all notifications for a given *User*. Finally, the Services Manager implements the Services Interface, which allows *Users* to add new *Services* or manage the ones they already have through the User Application. Moreover, it uses the New Data Interface offered by the Third Party Service System when it needs to send new *User data* or *Group data* to *Third parties*.



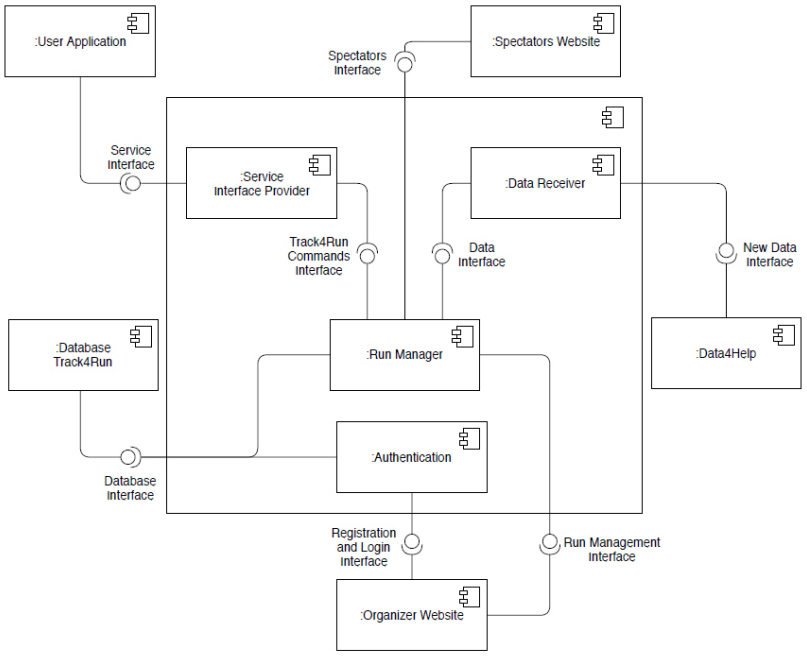
Data4Help User Application Component Diagram

The User Application is used by *Users* to interact with Data4Help and the *Services* they added. They can do this through several components and interfaces. They must be authenticated in order to use the Application. In order to do so, through the Authentication component, they can sign up and login into their Data4Help account. Moreover, the User Application will use the Registration and Login Interface offered by the server side Application to authenticate into it. The User Application collects *User data* from both the smartphone and the *Smart wearable* through the Data Interface of the Data Collector. This forwards the data to Data4Help through the Data Interface offered by the Data Sender which sends them to Data4Help. The Services Manager allows *Users* to add and manage their *Services*. Together with the Service Search Engine, they exploit the Services Interface offered by Data4Help to accomplish their main functions. Moreover, the Services Manager is the component in charge of constantly asking Data4Help for new notifications through the Notifications Interface. Furthermore, the purpose of the Service Controller is to allow the usage of a *Service* on the User Application through the Service Interface. In fact, the Third Party System will offer this interface for the User Application to connect and interact with. Finally, the Service Controller connects to Google Maps to obtain maps information for *Services* like Track4Run.



AutomatedSOS Component Diagram

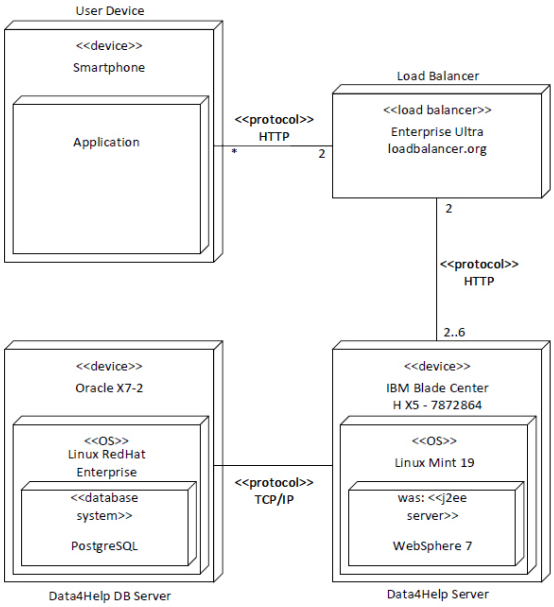
The main component of AutomatedSOS is the User Monitor. It offers the AutomatedSOS Commands Interface for the Service Interface Provider to forward commands received from the interaction with the User Application through the Service Interface. In particular, it allows *Users* to reactivate their data monitoring. This component also uses the Database Interface to store which Local Emergency Services are in charge of assisting a specific *User in need*. When new data is received from Data4Help, through the New Data Interface, it is passed to the Data Analyzer through the Data Analyzer Interface. The Data Analyzer compares the *User data* against certain thresholds, possibly identifying it as *Anomalous data*. When a *User* is identified as a *User in need*, the User Monitor sends a call request through the Caller Interface to the Local Emergency Services Caller, which calls the Local Emergency Services.



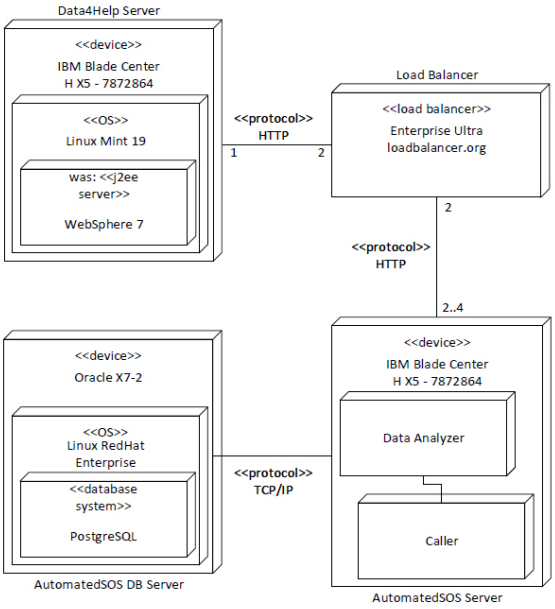
Track4Run Component Diagram

At the core of Track4Run lies the Run Manager. It is responsible for everything that concerns a *Run*. It offers several interfaces. First of all, it offers the Track4Run Commands Interface, for the Service Interface Provider to forward commands received from the interaction with the User Application through the Service Interface. In particular, it gives *Users* the possibility to enroll in a *Run* as *Participants* and to lookup *Runs*. The Run Manager also gives *Organizers* the possibility to create and manage a *Run* through the Run Management Interface from the *Organizers* dedicated website. It uses the Database Interface to store relevant data in a database. Moreover, it offers the Spectators Interface to *Spectators* to watch a *Run* through the *Spectators* dedicated website. The Authentication allows *Organizers* to sign up and login in the dedicated website through the Registration and Login interface. Finally, the Data Receiver offers the New Data Interface, as all *Services* do so as to receive new data from Data4Help. It forwards the just received data to the Run Manager through the Data Interface, which is in charge of displaying it to *Spectators* and of storing it in the database.

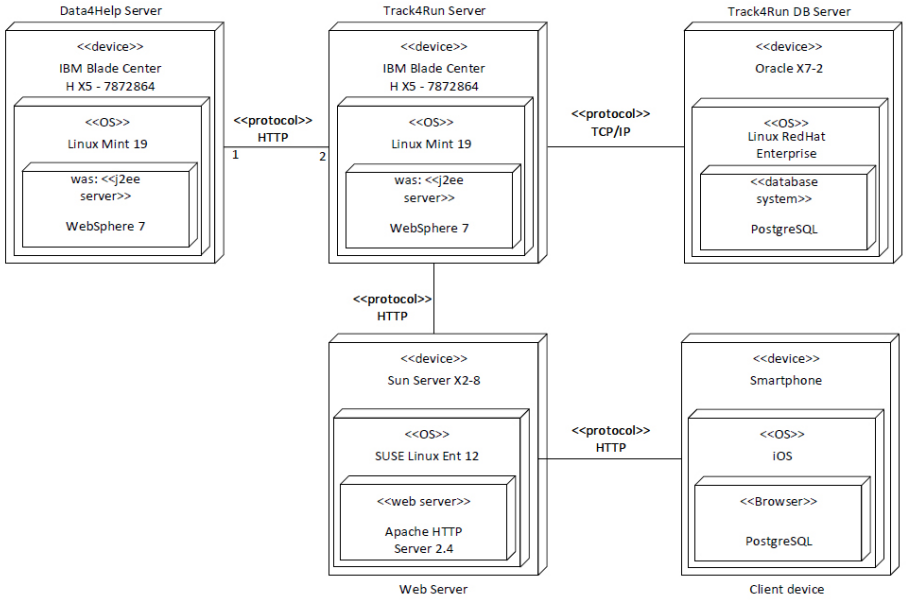
2.3 Deployment View



Data4Help Deployment Diagram



AutomatedSOS Deployment Diagram



Track4Run Deployment Diagram

2.4 Runtime View

In this sequence diagram the *User* login on the User Application is shown. The component involved in the process is the Authentication, which checks the credentials inserted by the *User*.

The User Application is able to receive notifications from Data4Help by querying the server side Application for new notifications. The Services Manager asks the Notifications Manager for new notifications. If there are any new notifications for the *User*, then it returns them, otherwise it tells the Services Manager that there are no new notifications.

When they want to organize a *Run*, *Organizers* login into their dedicated website and send a *Run* creation request to the Run Manager of Track4Run. This last one checks the *Run* data is correct and if it is, it creates the *Run*.

2.5 Component Interfaces

Inserire di seguito testo introduttivo

```
public interface d4h_registerAndLogin {
   boolean isUsernameValid(String username);
   boolean isPasswordValid(String password);
   boolean login(String username, String password);
   boolean userExists(String username);
   boolean signUp(String fullname, Date dateOfBirth, boolean gender,
       String ssn, String email, String password);
}

public interface d4h_services {
   List<Service> lookupServices(String namelike);
   List<Service> getAllServices();
   List<Service> getUserServices(String username); // requires auth
   Service getServiceByID(int serviceID);
   boolean addServiceToUserServices(int serviceID, String username); //
       requires auth
   boolean deleteServiceFromUserServices(int serviceID, String
       username); // requires auth
}

public interface d4h_data {
   boolean receiveData(Data data); // requires auth
   List<Data> getAllUserData(String username); // requires auth
   boolean deleteAllUserData(String username); // requires auth
}

public interface d4h_requestData{
   Data getUserData(UserDataRequest request); // requires auth
   List<Data> getGroupData(GroupDataRequest request); // requires auth
}
```

```
public interface userapp_servicesManagement {
   //TODO: capire con prof. DiNitto se inserire o meno
}

public interface userapp_loginAndRegister {
   //TODO: capire con prof. DiNitto se inserire o meno
}

public interface asos_commands{
   boolean reactivateMonitoring(String username); // requires auth
   boolean subscribeUser(User user);
   boolean removeUser(User user);
}

public interface service_newData{
   boolean receiveData(Data data, String username); // requires auth
}

public interface t4r_spectator{
   Run getRunByID(String ID);
   List<Run> getAllRuns();
}

public interface t4r_runManagement{
   boolean createRun(String name, Date date, DateTime time, Position
       start, int maxNumberOfParticipants, Date enrollmentClosureDate);
   boolean setPath(List<Position> path);
}

public interface t4r_registerAndLogin{
   boolean isUsernameValid(String username);
   boolean isPasswordValid(String password);
   boolean login(String username, String password);
   boolean organizerExists(String username);
   boolean signUp(String fullname, Date dateOfBirth, String email,
       String password);
}

public interface db_queries{
   getUserByUsername(String username);
   insertUser(String fullname, Date dateOfBirth, boolean gender, String
       ssn, String email, String password);
   deleteUser(String username);
   insertData(Data data);
   getUserData(String username);
   deleteAllUserData(String username);
   insertServiceForUserServices(String serviceID, String username);
   deleteServiceFromUserServices(String serviceID, String username);
```

```
    getAllUserServices(String username);
    getAllServices();
    //TODO: capire con prof. DiNitto se fare tutta la lista come qui o
        fare solo doQuery(String query)
}
```

2.6 Selected Architectural Styles

WORK IN PROGRESS

2.7 Other Design Decisions

WORK IN PROGRESS

Chapter 3

User Interface Design

All the relevant mockups relative to user interface are included in the RASD in section inserire quì sezione corretta.
In this chapter are listed the ways to move through different interfaces to give a clear idea of the navigation paths. Inserire schemi di passaggio da un'interfaccia da un altra

Chapter 4

Requirements Traceability

Description fare riferimenti a component diagrams

Data4Help

R_1 Unregistered individuals and companies must not be able to use Data4Help.

1. Authentication (User App and Data4Help)

R_2 At sign up, *User* must provide: first name, last name, SSN, gender, date of birth, email and password.

1. Authentication (User App and Data4Help)

R_3 At sign up, *Third party* must provide a company name.

1. Authentication (Data4Help)

R_4 At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*.

1. Authentication (User App and Data4Help)

R_5 At sign up, *Third party* must accept *Terms and conditions*.

1. Authentication (Data4Help)

R_6 Identify a *User* by their identifier.

1. Authentication (User App and Data4Help)

R_7 Query the database for a *User* by their identifier.

1. Authentication (User App and Data4Help)

R_8 Receive *User Data*.

1. Authentication (User App and Data4Help)

2. Data Collector
3. Data Sender
4. Data Receiver

R_9 Validate *User Data*.

1. Data Receiver

R_{10} Authenticate *User Data*.

1. Data Receiver

R_{11} Store collected *User Data* in a database.

1. Data Receiver
2. Data Manager

R_{12} Retrieve specific *User Data* by database querying based on *User* identification.

1. Data Manager

R_{13} Receive *Third party* data access request.

1. Request Manager

R_{14} Validate *Third party* data access request.

1. Request Manager

R_{15} Authenticate *Third party* data access request.

1. Request Manager

R_{16} Forward *User Data* access request to the specific *User*.

1. Request Manager
2. Authorization
3. Data Manager
4. Services Manager (Data4Help)

R_{17} Receive *User* consent approval or denial.

1. Services Manager (User App and Data4Help)

R_{18} Check if a specific *User* gave consent to a specific *Service*.

1. Authorization
2. Data Manager

R_{19} Send specific *User Data* to the requesting *Third party*.

1. Request Manager
2. Authorization
3. Data Manager

R_{20} Not send specific *User Data* to the requesting *Third party* if the specific *User* denied consent.

1. Request Manager
2. Authorization
3. Data Manager

R_{21} *Third party* must be able to set specific constraints to define a group of *Users*: age, gender, residence.

1. Request Manager

R_{22} Check how many *Users* requested *Group Data* refers to.

1. Anonymizer

R_{23} Properly anonymize *Group Data*.

1. Anonymizer

R_{24} Send *Group Data* to the requesting *Third party*.

1. Anonymizer
2. Request Manager

R_{25} Not send *Group Data* if the group it refers to is made up of less than 1000 *Users*.

1. Anonymizer
2. Request Manager

R_{26} Receive *Third party* subscription request.

1. Request Manager

R_{27} Validate *Third party* subscription request.

1. Request Manager

R_{28} Authenticate *Third party* subscription request.

1. Request Manager

R_{29} Automatically send new data to subscribed authorized *Third parties* as soon as they are produced.

1. Data Receiver

2. Data Manager
3. Services Manager (Data4Help)
4. Authorization

R_{30} Allow *Users* to subscribe to *Services*.

1. Services Manager (Data4Help)
2. Data Manager

R_{31} Allow *Users* to unsubscribe from *Services*.

1. Services Manager (Data4Help)
2. Data Manager

R_{32} Send a specific *User* all their data stored, if requested by them.

1. Services Manager (Data4Help)
2. Data Manager

R_{33} Delete a *User* specific data, if requested by them.

1. Services Manager (Data4Help)
2. Data Manager

R_{34} Allow *Users* to request all their data stored by TrackMe at any time.

1. Services Manager (User App and Data4Help)

R_{35} Allow *Users* to request the deletion of all their data stored by TrackMe at any time.

1. Services Manager (User App and Data4Help)

AutomatedSOS

R_{36} Receive *User Data* from Data4Help.

1. Data Receiver (AutomatedSOS)

R_{37} Compare *User Data* against certain thresholds.

1. Data Analyzer

R_{38} Call local emergency services providing necessary *User Data* of *User in need*.

1. Local Emergency Services Caller

R_{39} *User* must be able to reactivate AutomatedSOS monitoring.

1. Services Manager (User App and Data4Help)
2. User Monitor spiegare interface

Track4Run

R_{40} Receive *User Data* from Data4Help.

1. Data Receiver (Track4Run)

R_{41} At sign up, *Organizers* must provide: first name, last name, email and password.

1. Authentication

R_{42} At sign up, *Organizers* must accept *Terms and conditions*.

1. Authentication

R_{43} Allow *Organizers* to create a *Run*, defining: name, path, date, maximum number of *Participants* and enrollment closure date.

1. Run Manager

R_{44} Provide *Users* enrollment for an existing *Run*.

1. Run Manager

R_{45} Prevent a *User* from enrolling in a *Run* if the maximum number of *Participants* was already reached.

1. Run Manager

R_{46} Prevent a *User* from enrolling in a *Run* if it already started or finished.

1. Run Manager

R_{47} Prevent a *User* from enrolling in a *Run* if enrollment is closed.

1. Run Manager

R_{48} Show a *Run* by displaying the position of *Participants* on a map.

1. Run Manager

R_{49} Identify a *Run* by its identifier.

1. Run Manager

R_{50} Query the database for a *Run* given its identifier.

1. Run Manager

Chapter 5

Implementation, Integration and Test Plan

WORK IN PROGRESS

Chapter 6

Effort Spent

Gargano Jacopo Pio Total hours of work:

- effort spent

Giannetti Cristian Total hours of work:

- effort spent

Haag Federico Total hours of work:

- effort spent

Chapter 7

References

1 E. Di Nitto. *Lecture Slides*. Politecnico di Milano.

2 E. Di Nitto. *Mandatory Project Assignment AY 2018-2019*. Politecnico di Milano.

3 J. Gargano, C. Giannetti, F. Haag. *Requirement Analysis and Specification Document*. Politecnico di Milano.

- references