POLITECNICO
MILANO 1863

TrackMe

Software Engineering II - Prof. Elisabetta Di Nitto

Design Document

Michele Gatti, Federica Gianotti, Mathyas Giudici

Document version: 1.0
December 5, 2018

Deliverable:	DD
Title:	Design Document
Authors:	Michele Gatti, Federica Gianotti, Mathyas Giudici
Version:	1.0
Date:	December 5, 2018
Download page:	https://github.com/MathyasGiudici/GattiGianottiGiudici
Copyright:	

Contents

Section 1

Introduction

1.1 Purpose

The goal of the Design Document (DD) is to provide a more technical functional description of the system-to-be, in particular it wants to describe the main architectural components, their communication interfaces and their interactions. Moreover it will also present the implementation, integration and testing plan. This type of document is mainly addressed to developers because it provides an accurate vision of all parts of the software which can be taken as a guide during the developement process.

1.2 Scope

The *TrackMe* project, as explaind in the RASD, has three different but connected goals to achieve:

- **Data4Help**: a service that allows third parties to monitor the location and health status of individuals. Through this service third parties can request the access both to the data of some specific individuals, who can accept or refuse sharing their information , and to anonymized data of group of individuals, which will be given only if the number of the members of the group is higher than 1000, according to privacy rules.
- **AutomatedSOS**: a service addressed to elderly people which monitors the health status of the subscribed customers and, when such parameters are below a certain threshold (personalized for every user using the data from Data4Help), sends to the location of the customer an ambulance, guaranteeing a reaction time less than 5 second from the time the parameters are below the threshold.

- **Track4Run**: a service to track athletes participating in a run. It allows organizers to define the path for a run, participants to enrol to a run and spectators to see on a map the position of all runners during the run. This service will exploit the features offered by Data4Help.

The system-to-be is structured in a four-layered architecture, which will be described in depth in this document and which is designed with the purpose of being maintainable and extensible.

1.3 Definitions, Acronyms, Abbreviations

ACID: Atomicity, Consistency, Isolation and Durability (Set of properties of database transactions):

API: Application Programming Interface;

DB: Database;

DMBS: Database Management System;

DD: Design Document;

GPS: Global Positioning System;

GUI: Graphical User Interface;

RASD: Requirement Analysis and Specification Document;

UML: Unified Modeling Language;

UX: User eXperience;

1.4 Revision History

1.5 Reference Documents

1.6 Document Structure

This document is structured as follows:

Section 1: Introduction. A general introduction and overview of the Design Document. It aims giving general but exaustive information about what this document is going explain.

Section 2: Architectural Design. This section contains an overview of the high level components of the system-to-be and then a more detailed description of three architecture views: component view, deployment view and runtime view. Finally it shows the chosen architecture styles and patterns.

Section 3: User Interface Design. This section refers to the mockups already preseneted in the RASD.

Section 4: Requirements Traceability. This section explains how the requirements defined in the RASD map to the design elements defined in this document.

Section 5: implementation, Integration and Test Plan. This section identifies the order in which it is planned to implement the subcomponents of the system, the integration of such subcomponents and test the integration.

Section 6: Effort Spent. A summary of the worked time by each member of the group.

At the end there are an **Appendix** and a **Bibliography**.

Section 2

Architectural Design

2.1 Overview

The main high-level components of the system that will be taken into account are structured in four layers, as shown in Figure 2.1 below.

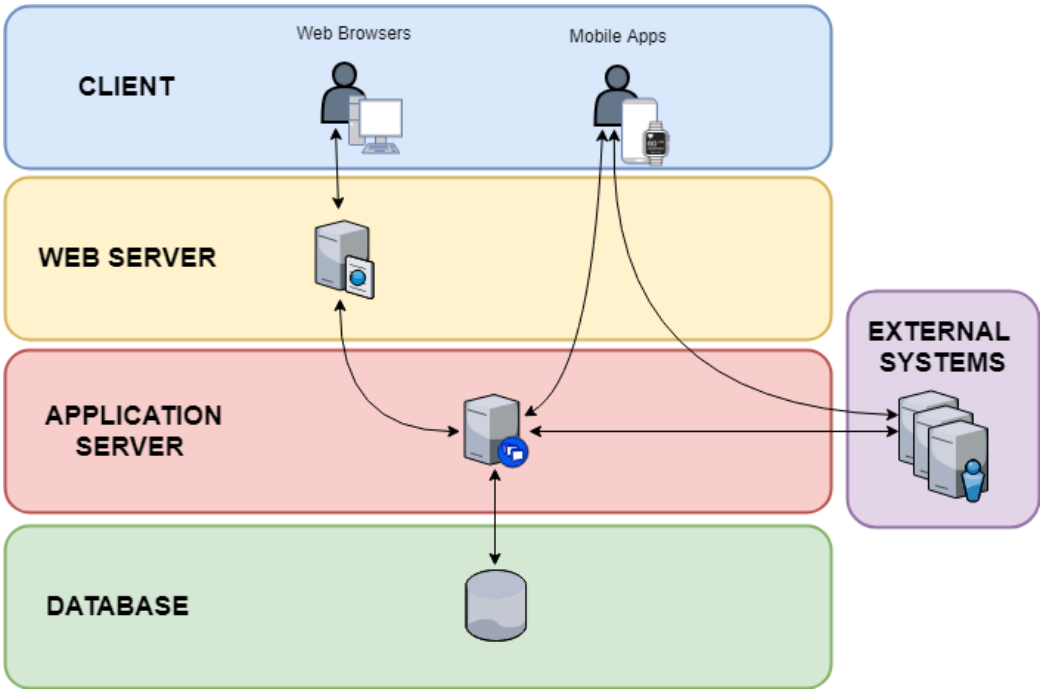

Figure 2.1: *Layered Structure* of the system.

The considered high-level components are:

Mobile Applications: The *Presentation Layer* dedicated to mobile devices; it communicates with the Application Server.

Web Browsers: The *Presentation Layer* dedicated to web browsers; it communicates directly with the Web Server.

Web Server: This is the layer that provides web-pages for the web-based applications; it communicates with the Application Server and with the Web Browsers.

Application Server: This is the layer in which is contained all the *logic* for the application; it communicates with the Web Server and with the Database. Moreover it manages the communication with External Services.

Database: The *Data Layer* of the system; it includes all structures and entities responsible for data storage and management. It communicates with the Application Server.

It has been taken the decision to separate the Application and Web Server in order to allow greater scalability. In the figure above it is also shown the interaction between the Application Server and External Systems. A more detailed description of the intereactions between the described system components is shown in Figure 2.2.

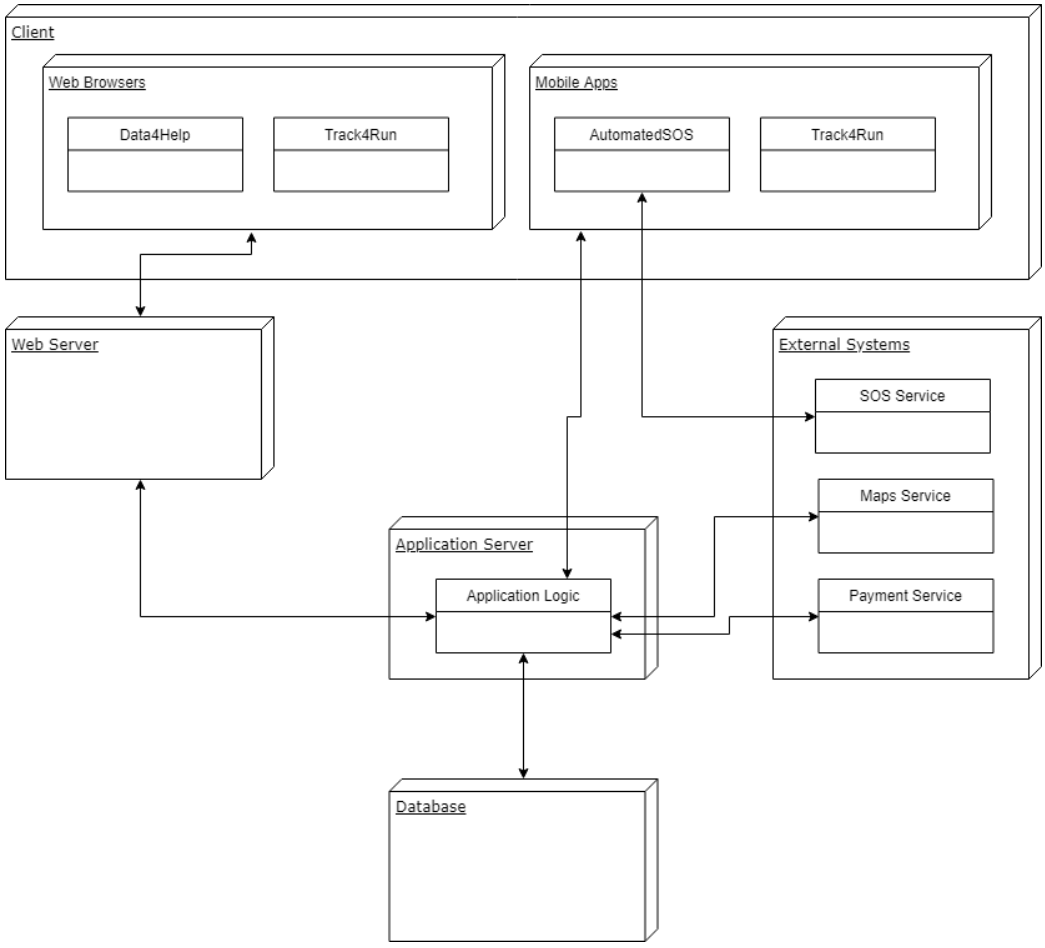

Figure 2.2: *High Level Components* of the system.

2.2 Component View

2.2.1 Database

The application database will be managed using a Relational DBMS. It allows the reading of data, ensuring users the ability to log in and access the applications of interest and check the stored data. It is also used for data manipulation (insertion, modification and deletion). The use of a Relational DBMS guarantees the fundamental properties for a database of this type:

- *Atomicity*: no partial executions of operations.

- *Consistency*: the database is always in a consistent state.
- *Isolation*: each transaction is executed in an isolated and independent way.
- *Durability / Persistence*: changes made are not lost.

The database will offer to the Application Server an interface that it can use to interact with the database. The data stored in the database must be considered personal and confidential, therefore, procedures must be implemented to safeguard the stored information.
Particular attention must be paid to the reading permissions granted to users and to the encryption of passwords used to access the services offered. Below is the designed E-R diagram.(Figure 2.3)

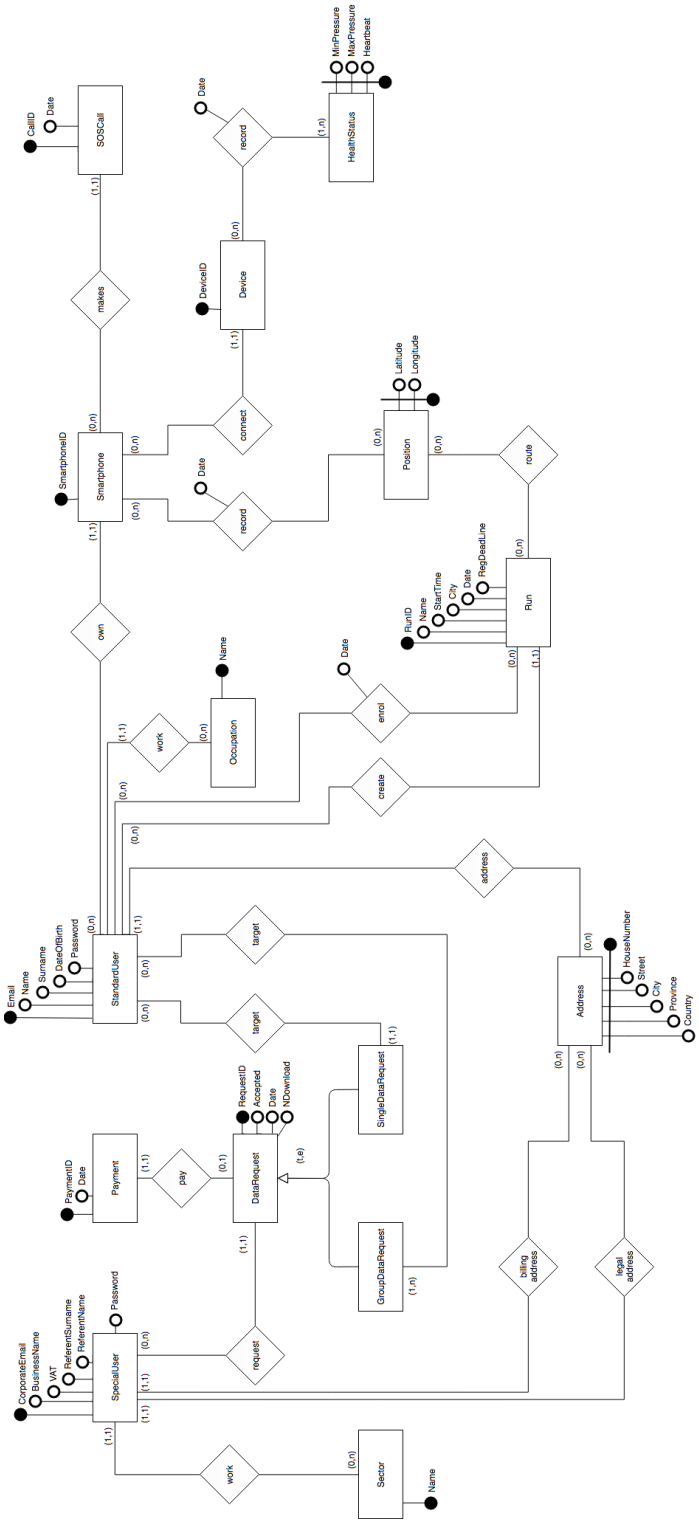

Figure 2.3: *E-R* Diagram

2.2.2 Application Server

This is the crucial layer of the system to be. The main feature of the *Application Server* is to describe rules and work-flows of all the functionalities provided by the application.
The *Application Server* must have intefaces to communicate with the *Web Server* and the *Mobile Apps*, it has also to communicate through interfaces with all the *External Services* (Maps Service, SOS Service and Payment Service).
Moreover, the *Application Server* is the only entity of the system that is granted to communicate with the DBMS. Following this brief introduction there are the logic modules and their descriptions, moreover all connections among components could be seen in the *Global Component View* in Figure 2.4.

Data Collector Service This module offers the a service to the *Standard User Manager* in order to maange user's data and store it into the *Data4Help* database. Data storage is done through *Stored Data Manager*.

Group Request Manager This module manages the third party requests of *Group Data Requirement*.

Mail Manager This module is a tool for other modules that allow the system to send e-mail, for instance in the registration and enrolment phase.

Maps Manager This module is an handler providing an interface to the external service of maps.

Payment Handler This module is an handler providing an interface to the external service of payment.

Persistence Unit This module is the unique interface to the database's DBMS.

Run Manager This module manages *Runs* in all their functionalities, like creation, deletion and enrolment.
In order to manage guest users that visit a Run the system this module provides an access point to the *Application Server* for the *Web Server* and the *Mobile Apps*.

Single Request Manager This module manages the third party requests of *Single Data Requirement*.

Special User Manager This module provides all functionalities related to the *Special User*.

Standard User Manager This module provides all functionalities related to the *Standard User*.

Stored Data Manager This module provides data to *Group Request Manage* and *Single Request Manage*. It has also to manage all privacy checks and provide data without any type of personal information of the users for *Group Data Requirement*.

User Access Point This module provides an access point to the *Application Server* for the *Web Server* and the *Mobile Apps*. It routes different users in the correct *User Manger* module: Special User and Standard User.

2.2.3 Mobile App

The *Mobile App* must communicate to the *Application Server* through APIs that have to be defined in order to describe the interactions between the two layers and they must be indipendent from the implementation both side.
The App UI must be designed user friendly and it has to follow the guidelines provided by the Android and iOS producer. The application must provide a software module that manages the GPS connection of the device and keeps track of locations data, it has also to manage the Health data from the devices connected to the phone.
The application must provide all collected data to the *Application Server* in order to process them.

2.2.4 AutomatedSOS App

In this Section we want to specify in detail the modules that compose the *AutomatedSOS* application in order to better explain the important activity of detection of possible *Critical Situation*.
All connections among components could be seen in the *AutomatedSOS Component View* in Figure 2.5.

Application Server Handler This module is an handler providing an interface to the *Application Server*.

Background Health Monitor This module checks status of health data that it receives. If an SOS call must be done it call *SOS Handler*.

GUI This module is the *Graphical* interface to the smartphone.

Health Status Service This module manage data detected and produce statistic for the *User*.

SOS Handler This module is an handler providing an interface to the external service of SOS.

2.2.5 Web Server

The *Web Server* must communicate to the *Application Server* through HTTPS protocol.
The *GUI* of the Web Server must be designed user friendly and it has to follow the guidelines provided by W3C standard (using HTML5, CSS and JS).
The *Control Unit* manages all actions played by a user and it must communicate to the *Application Server* through APIs, that are already explained for the *Mobile App*.

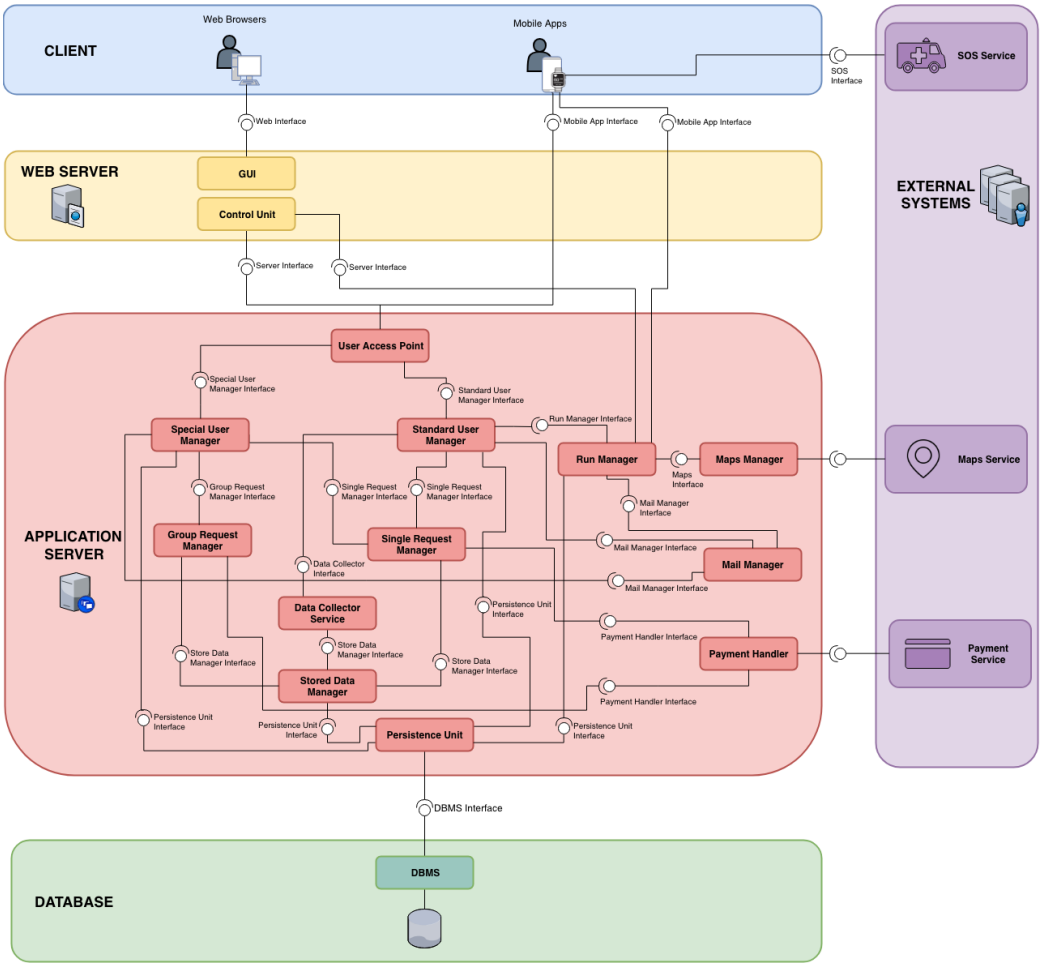

Figure 2.4: *Global Component View*

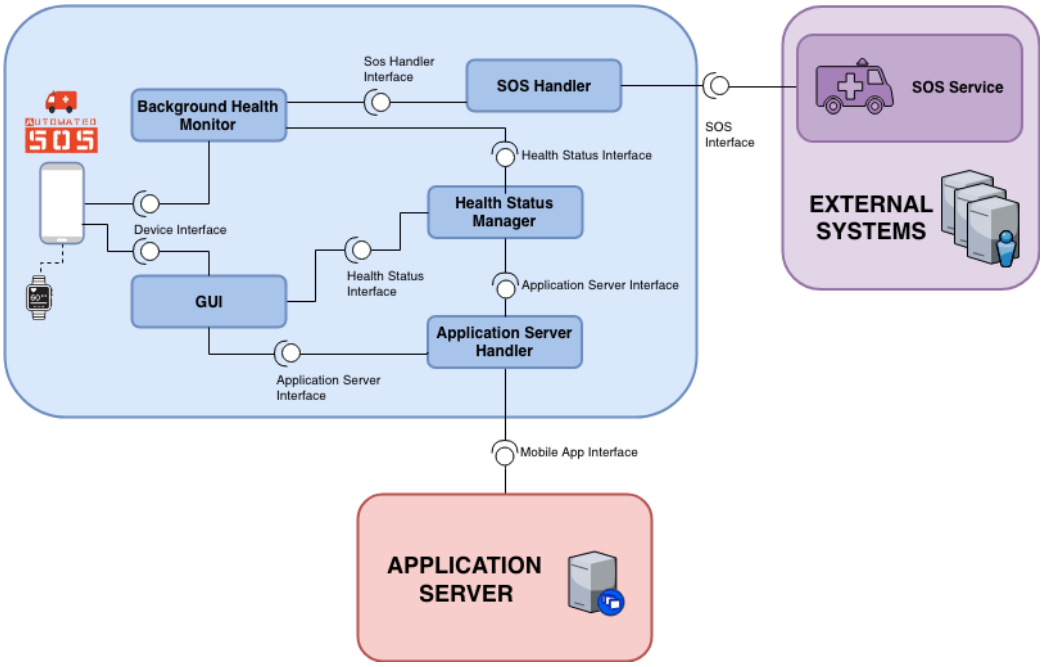

Figure 2.5: *AutomatedSOS Component View*

2.3 Deployment View

Below is the deployment diagram of the system to be (Figure 2.6).

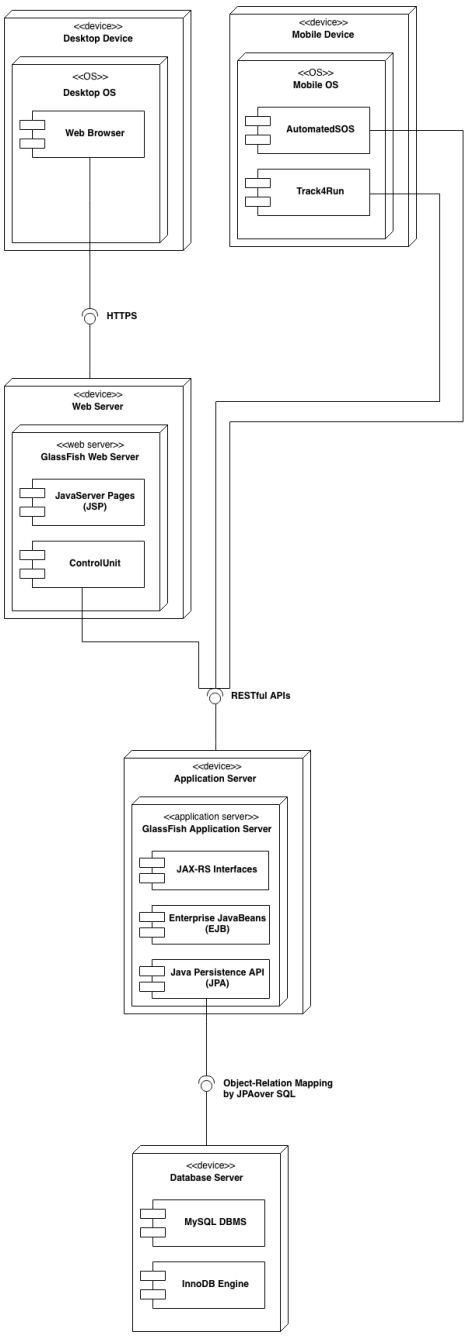

Figure 2.6: Deployment Diagram of the system to be

2.4 Runtime View

In this Section we want to specify the behaviour of our system. Some relevant cases are selected and explained using Sequence Diagrams.

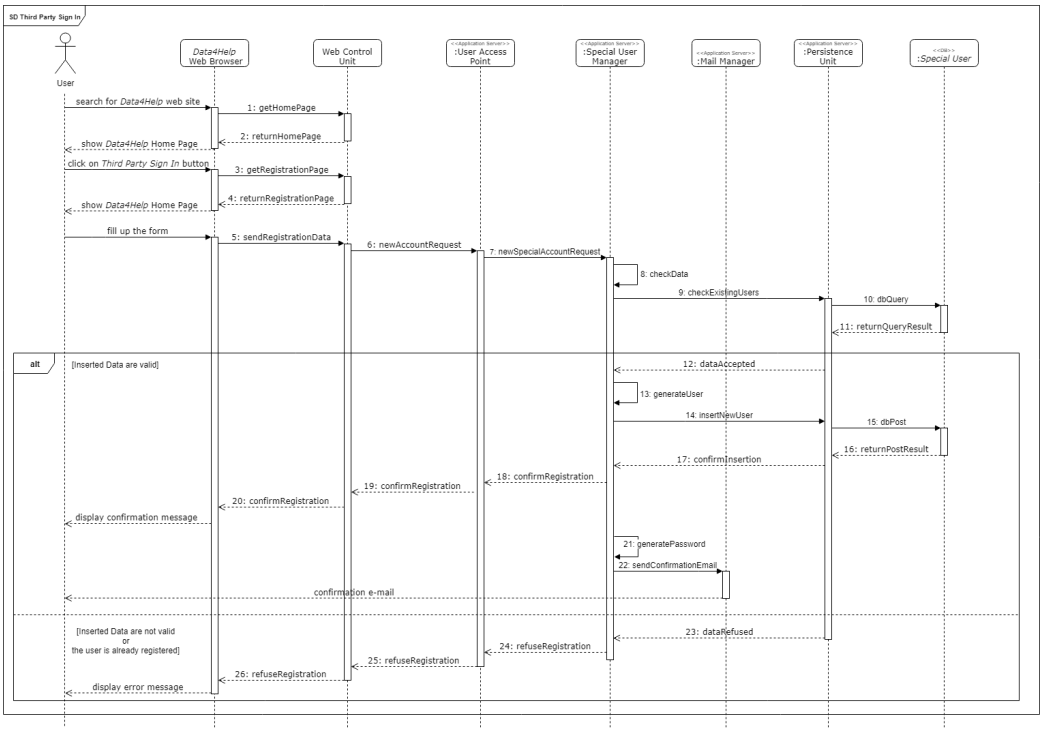

Figure 2.7: *Third Party Sign In* sequence diagram: it is described the process through which a third party can register itself to *Data4Help* services, obviously using *Data4Help*'s web site, since it is the only available platform for third parties.

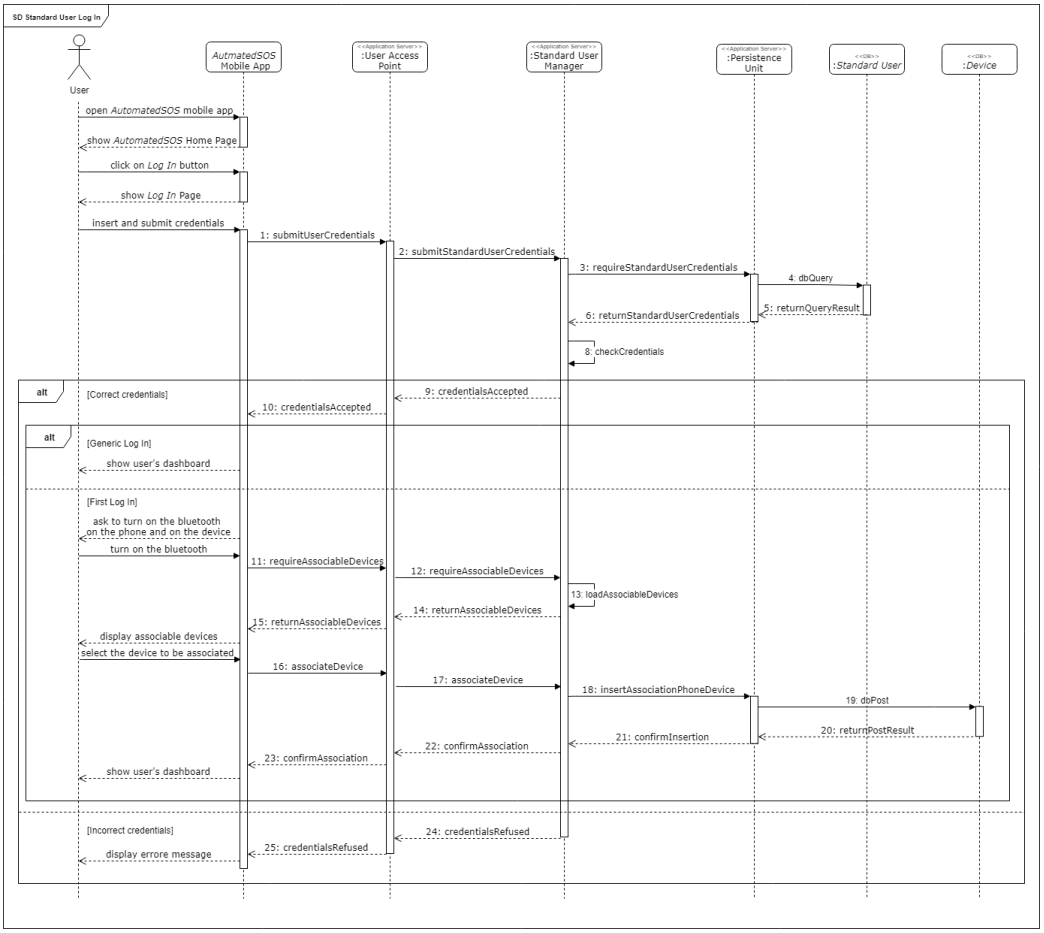

Figure 2.8: *Standard User Log In* sequence diagram: it is described the process through which a standard user can access to *AutomatedSOS* or *Track4Run* applications.

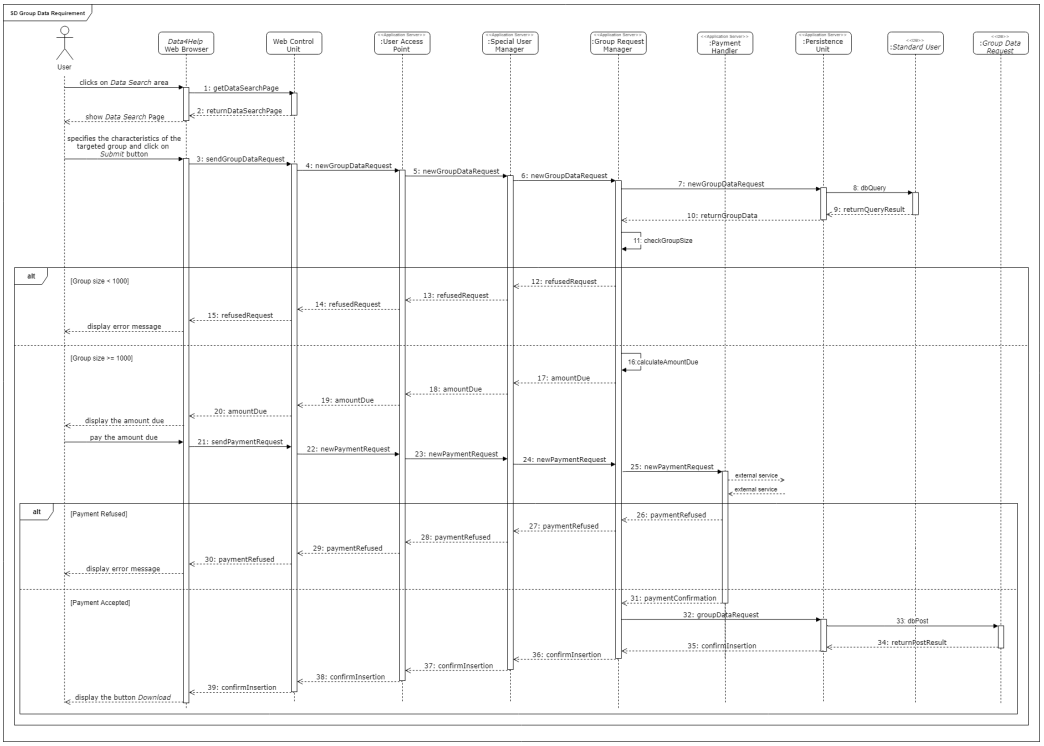

Figure 2.9: *Group Data Request* sequence diagram: it is described the process through which a third party can require data of a group of people and then pay them.

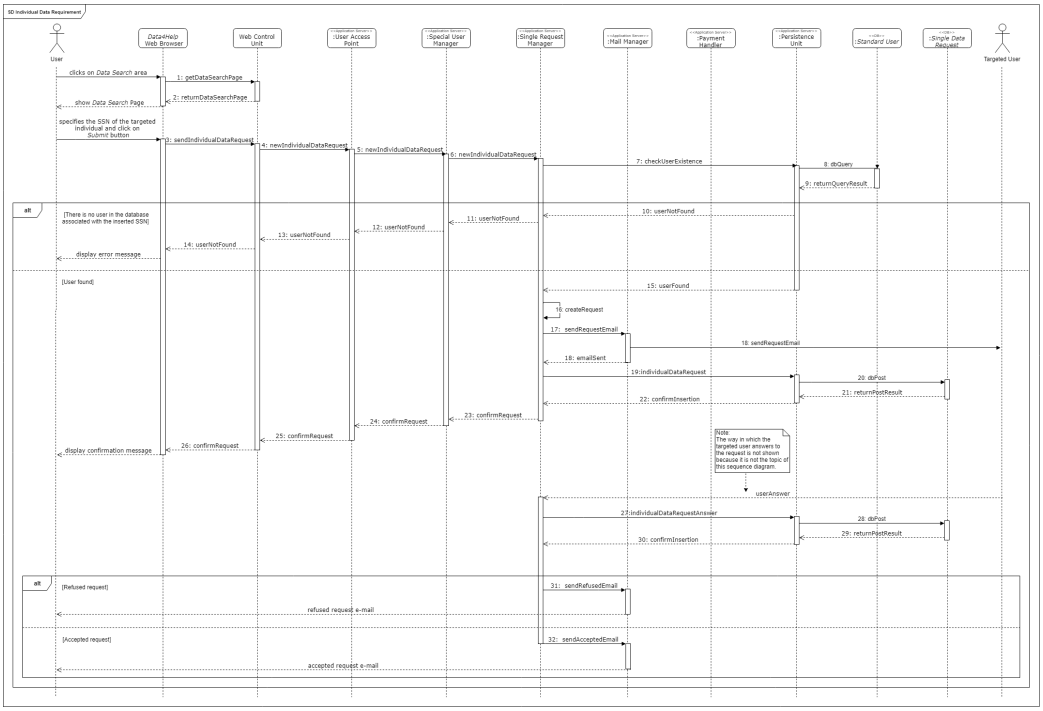

Figure 2.10: *Individual Data Request* sequence diagram: it is described the process through which a third party can require data of an individual user and wait for a response. It is not described the way in which the targeted user answers to the request because it is not the topic of this sequence diagram. It is not even described the way in which the third party pays and eventually downloads the data because it is not considerable and it has just been explained in the *Group Data Requst* sequence diagram.

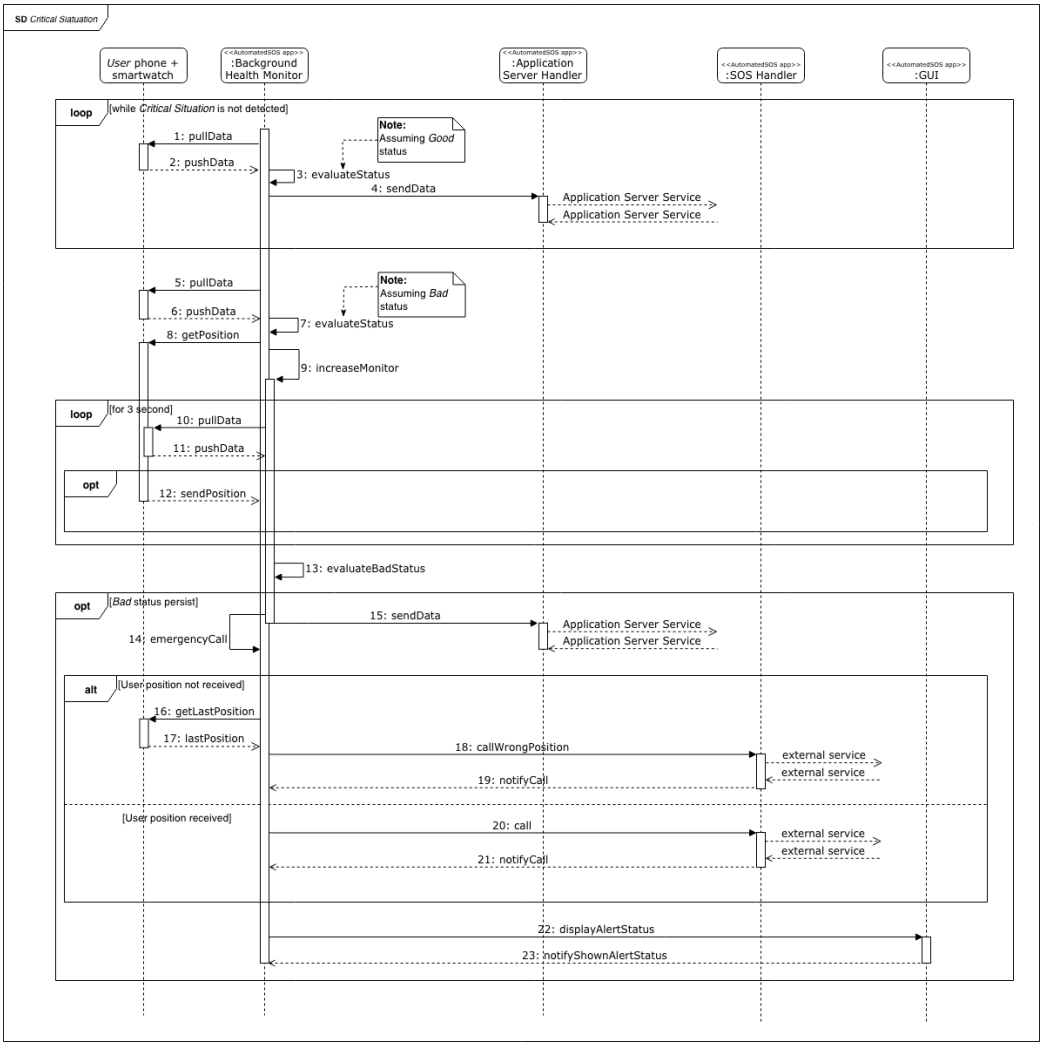

Figure 2.11: *Critical Situation* sequence diagram: it is described the process through which a critical situation is detected by *AutomatedSOS* application + device. It is not described the way in which *AutomatedSOS* application stored the acquired data because it is not considerable for this topic.

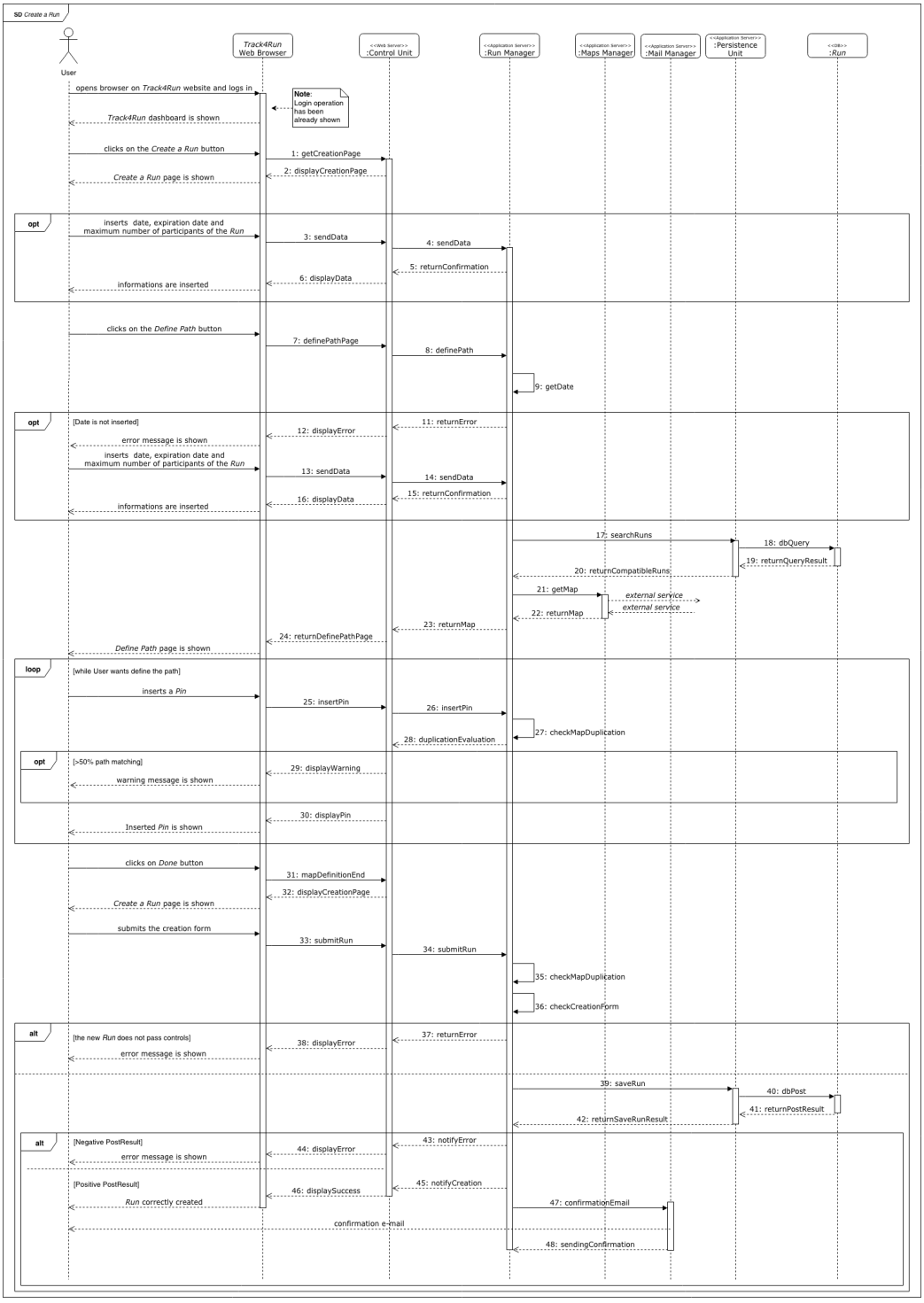

Figure 2.12: *Create a Run* sequence diagram: it is described the process through which a standard user can create a new run.

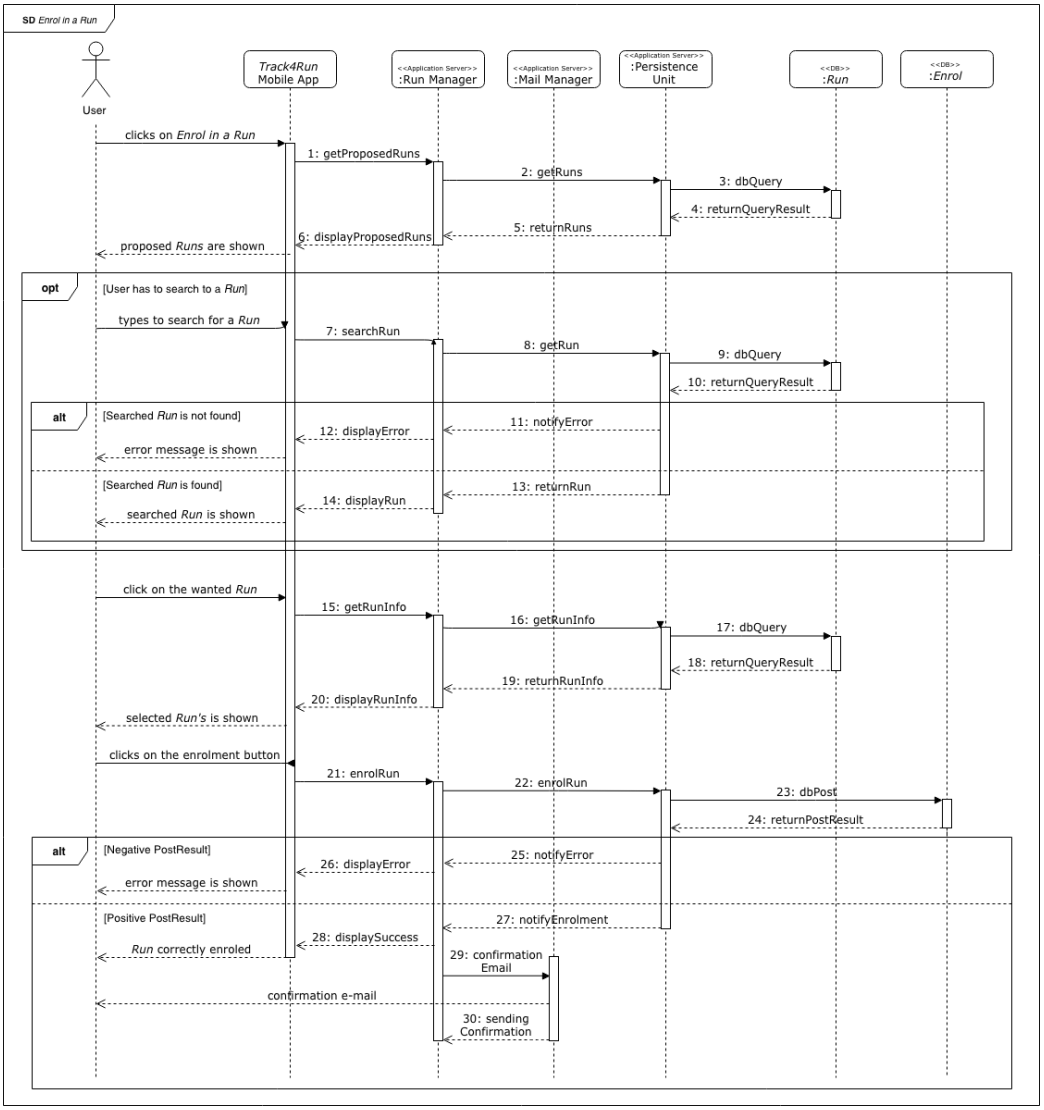

Figure 2.13: *Enrol in a Run* sequence diagram: it is described the process through which a standard user can enrol himself/herself in a run.

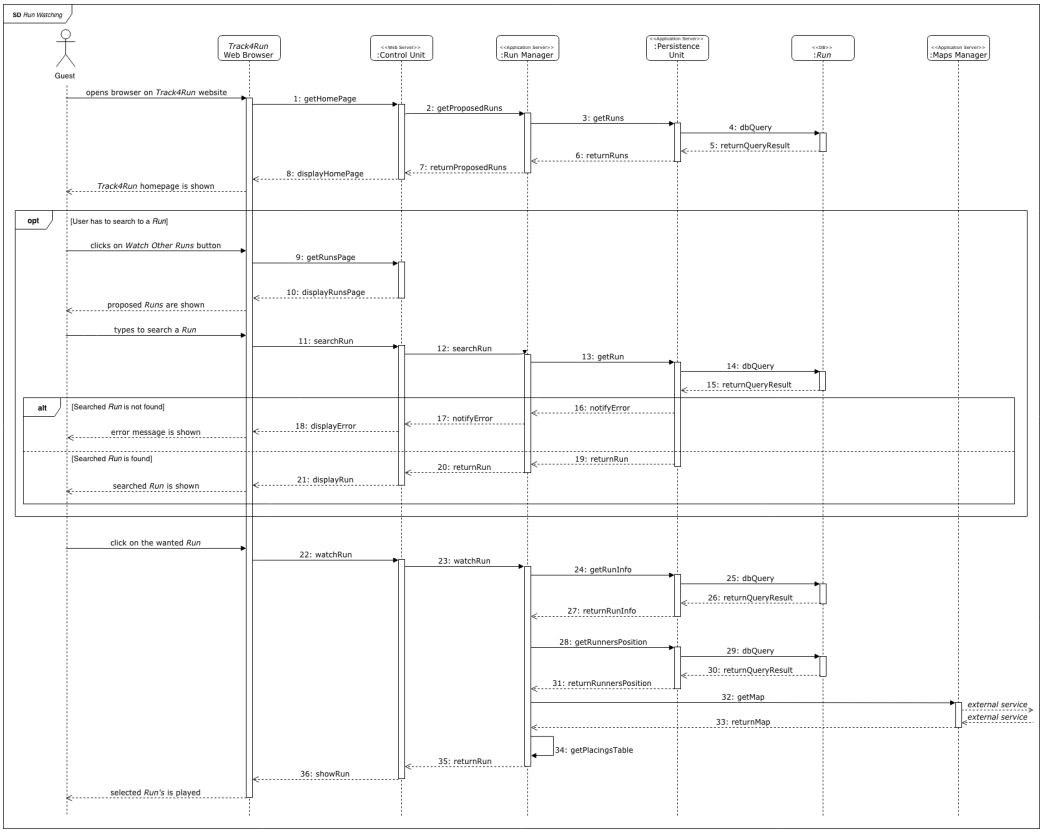

Figure 2.14: *Run Watching* sequence diagram: it is described the process through which a not registered user can watch a run. Notice that he/she can access directly to part of the *Run Manager* services without any log in.

2.5 Component Interfaces

2.6 Selected Architectural Styles and Patterns

Following are the choices made regarding architectural styles and patterns.

2.6.1 Client and Server

A client server architecture was chosen for system implementation. This allows the exchange of information between Internet-connected devices used by users and the centralized server that collects data and offers Data4Help services. In particular:

- The AutomatedSOS and Track4Run applications installed on users' devices are clients connected to the application server that receives and processes data.
- The desktop web browsers used by users (both standard and special) to access the services offered by Data4Help are clients that communicate with the Web Server that provides an HTTPS interface.
- The application server is seen as a server by the web browser and as a client by the application server that allows it to process requests.
- The application server is seen as a server from the web server and applications installed on smartphones and as a client from the database that receives and processes its requests (queries).
- The database is seen as a server by the application server that sends the requests and waits for the answers.

2.6.2 Multi-tiered architecture

A multi-tiered client-server architecture is used to ensure the separation between the MVC model levels (model, view, controller). In this way a greater stability of the system is also guaranteed:

- In case of failure of the application server, the applications installed on users' devices continue to work and collect data. However, AutomatedSOS is able to make an emergency call. This also applies if the device is temporarily without an Internet connection (but with emergency calls available). Once the application server or Internet connection has been restored, the device will reconnect and send the data collected during the period of absence of the service to the server.

- In case of failure of the web server, the application server remains active and the applications installed on users' devices continue to work correctly (and to send data correctly to the server). The services offered through the use of the web browser will not be available.

2.6.3 Thin Client

The services accessible through web browsers are considered thin clients because the software used does not install software belonging to Data4Help. All the necessary information required to use the service is offered by the Web Server through HTTPS, including a graphical interface and application logic. Also, the Track4Run app is considered thin client because the application that provides the graphical user interface is installed on the phone but most of the application logic (and in particular the one used to manage the races) is based on the application server to which it connects.

2.6.4 Thick Client

The Automated SOS app is to be considered as a thick client in fact in the app in addition to the graphical interface and the logic related to the acquisition of user data, there is also the logic that allows the phone to make a call to the external service used for handling emergency calls. Thanks to this the app is able to detect a state of emergency and alert the ambulance even if the connection with the server is not available.

2.6.5 Model-View-Control

The MVC model allows to identify three main components in the management of a client-server application. They are the Model, the View and the Controller. The following diagram shows the management of these three levels according to the use of thin or thick clients (2.15).

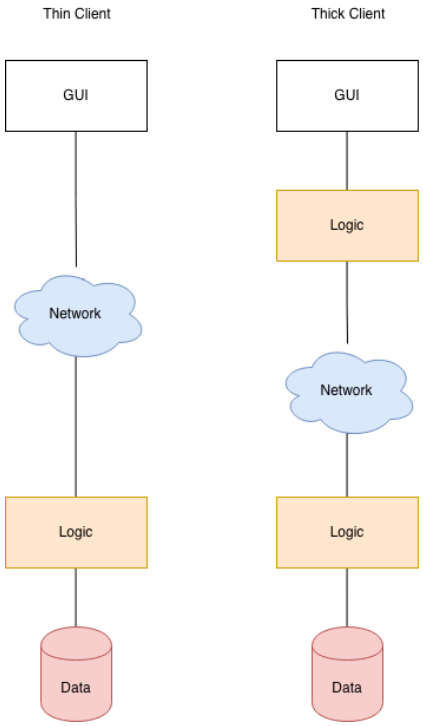

Figure 2.15: Possible management of the client-server connection according to the MVC model. Thin or thick client.

2.7 Other Design Decisions

2.7.1 User authentication

In order to guarantee the confidentiality of user data, access to the system, both for standard users and for special users, takes place by entering a username and password. The username used for access coincides with the email address used during registration.

2.7.2 Password storing

The passwords chosen by the users are saved in the database after being encrypted using a SHA512 algorithm. In this way, the password to be used for accessing Data4Help services is not stored in plaintext in the user database, but it is a transformation from which, even with very powerful computers, it is very difficult to trace the original password. In this way even using the normal Internet connection, if an intruder could pick up the string containing

the password, he would not be able to trace the password used to access the service.

2.7.3 Security in the transfer of information

The transfer of data between the clients and the server is done using the most modern data encryption systems. For example, the Web Server offers an HTTPS interface to the Web Browser.

2.7.4 Maps

In order to manage the rides offered by Track4Run, the external Google Maps service is used. Longitude and latitude are used to locate a position within the map. To ensure a better service to the user who is responsible for creating the race, it is also given the opportunity to create the route using the features provided by Google Maps such as search by address, search for places of public interest, and so on. In the database the race course will still be saved as a sequence of positions in terms of longitude and latitude.

2.7.5 Emergency call

Emergency calls are handled using an external service. This ensures that even in the event that the data network is not available, or the application server is out of service, it is still possible to call for help. In fact, the external service works via the Internet or, in the absence of a connection, via SMS. When the Automated SOS application detects an anomaly, it invokes the external service and sends it the location of the device and a precompiled text containing information related to the health of the user. This information is then sent to the rescue service by the external service.

Section 3

User Interface Design

Section 4

Requirements Traceability

Section 5

Implementation, Integration and Test Plan

Section 6

Effort Spent

6.1 Michele Gatti

Task	**Hours**
Team work	8
E-R Diagram	3
Deployment View	1
Selected Architectural Styles and Patterns	2
Other Design Decisions	2
UX Diagram	1
Total	

6.2 Federica Gianotti

Task	**Hours**
Team work	8
Introduction	1
Overview and High Level Components	2
Sequence Diagrams	11
Total	

6.3 Mathyas Giudici

Task	**Hours**
Team work	8
Global Component Diagram	4
Sequence Diagrams	10
Total	

Appendix A

Appendix

A.1 Software and Tools

- LaTeX used to build this document;
- *GitHub* used to manage the different versions of this document;
- *draw.io* used to draw diagrams;

A.2 Changelog

- **1.0** : First release of this document;

Bibliography

[1] Michele Gatti, Federica Gianotti, Mathyas Giudici *Requirements Analysis and Specification Document*

[2] IEEE Standard 1016:2009 *System design - Software design descriptions*

[3] Elisabetta Di Nitto - Software Engineering 2 Slides (AY 2018/2019) *Project goal, schedule and rules*