

DD

Design Document

Data4Help, AutomatedSOS and Track4Run

Irene Nizzoli, Isabella Piacentini, Elio Salvini

10483798 10508831 10490058

POLITECNICO DI MILANO

Sommario

[1 Introduction 4](#_Toc531631118)

[1.1 Purpose 4](#_Toc531631119)

[1.2 Scope 4](#_Toc531631120)

[1.3 Acronyms, Abbreviations 4](#_Toc531631121)

[1.3.1 Acronyms 4](#_Toc531631122)

[1.3.2 Abbreviations 5](#_Toc531631123)

[1.4 Document Structure 5](#_Toc531631124)

[2. Architectural Design 6](#_Toc531631125)

[2.1 Overview 6](#_Toc531631126)

[2.2 Component View 7](#_Toc531631127)

[2.3 Deployment view 9](#_Toc531631130)

# Introduction

## Purpose

The purpose of this document is to give a more detailed description of the architecture of Data4Help system. It will include the illustration of specific components and design choices that will guide the developers during implementation, integration and testing.

Overall this document outlines these elements:

* The high-level architecture
* The main components and their respective interfaces
* The runtime behaviour
* The design patterns
* The algorithm design of the most critical parts of the application
* Implementation plan
* Integration plan
* Testing plan

## Scope

The aim of TrackMe is to provide a service to either companies in need of data for business researches or individuals for more personal reasons. The main functions of Data4Help are managing requests from different users and saving and protecting a great quantity of data. A registration will be needed to provide clients a personalized experience, both by giving them the results of their requests and by showing monitored user their private health status data. The project is extended by AutomatedSOS that monitors the data of subscribed users and contacts medical services in case of need. The target of this system are elderly people who lives alone or are simply worried about their health conditions. This will need a 24/7 reliability of the application. Finally, Track4Run allows run organizers to create new races, tracks runners and show their position on the map to all possible spectators.

## 1.3 Acronyms, Abbreviations

### 1.3.1 Acronyms

* RASD: Requirement Analysis and Specification Document
* API: Application Programming Interface
* GPS: Global Positioning System
* DAD: Data Acquisition Device
* CF: “Codice Fiscale”
* SSN: Social Security Number
* DD: Design Document
* MVC: Model View Controller
* GUI: Graphical User Interface
* DB: Database
* DBMS: Database Management System

### 1.3.2 Abbreviations

* [Gn]: nth goal
* [Rn]: nth functional requirement

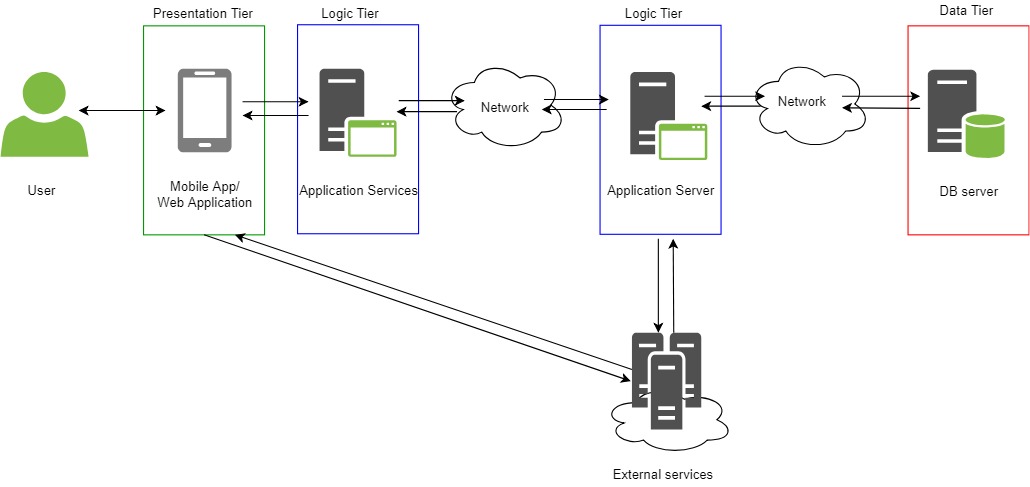
## 1.4 Document Structure

1. **Introduction:** this chapter contains the purpose and the scope of the design document. There’s also a list of the acronyms and the abbreviation that will be used in the document in order to make it more comprehensible.
2. **Architectural Design:** this sectiongives a general idea of the architecture including the three most important views: component, deployment and runtime. The interaction of the component interfaces and some architectural styles and patterns are also contained here.
3. **User Interface Design:** this chapter presents a reference to the mock-ups previously presented in the RASD document.
4. **Requirements traceability:** clarifies how the requirements that have been defined in the RASD map to the design elements that are defined in this document.
5. **Implementation, integration and test plan:** reveal the order in which it is intended to implement the subcomponents of the system and the order in which it is planned to integrate such subcomponents and test the integration.
6. **Effort spent:** shows the number of hours each member of the group spent for every chapter of the document.
7. **References:** presents the external documents used in the construction of the DD document.

## 2. Architectural Design

### 2.1 Overview

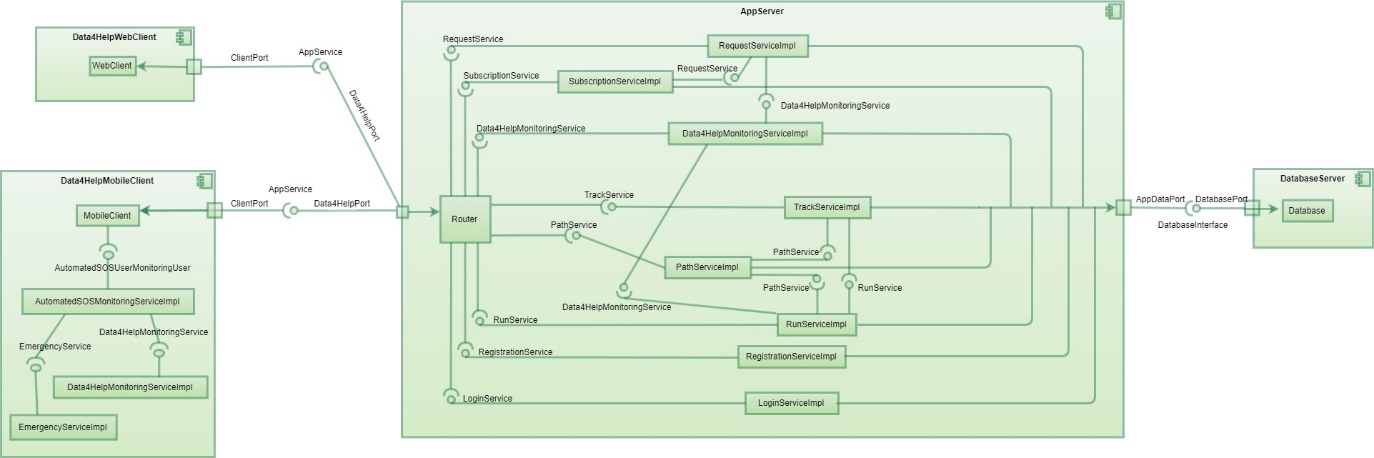
There are three separated layers in the flow in which the process of the application architecture design is executed:



* *Presentation tier*: This is the top-most level of the application. It consists in a user interface (view) and communicates with the application services and the external services. The connection with the external services is useful because the external medical services (for AutomatedSOS) can be directly contacted by the mobile app/web application.
* *Logic tier*: This layer controls the functionality of the application. The logic of the application is mostly stored on the server side, only a small part, such as the logic to analyse data (e.g. an average is calculated every half an hour for the Data4Help service) and detect if the individual’s data are below the threshold (just for the AutomatedSOS service), is located on the client side. The choice of distributed logic architecture allows better performances in contacting the external medical services in case of emergency.
* *Data tier*: The data tier comprises of the database/data storage system and data access layer. Here the information is stored persistently and can be retrieved at any time.

## 2.2 Component View

In the following diagram is represented the interface structure of the system both inside the WebClient, the MobileClient, where only a small part of the application is present, and the AppServer. Using a browser to use TrackMe application the client has access to all the features of the mobile application besides AutomatedSOS functionalities. The external services are not shown because their implementation it is not relevant for our application.



The interfaces portrayed in the diagram above, with their respective implementation identified by the suffix Impl, are the following:

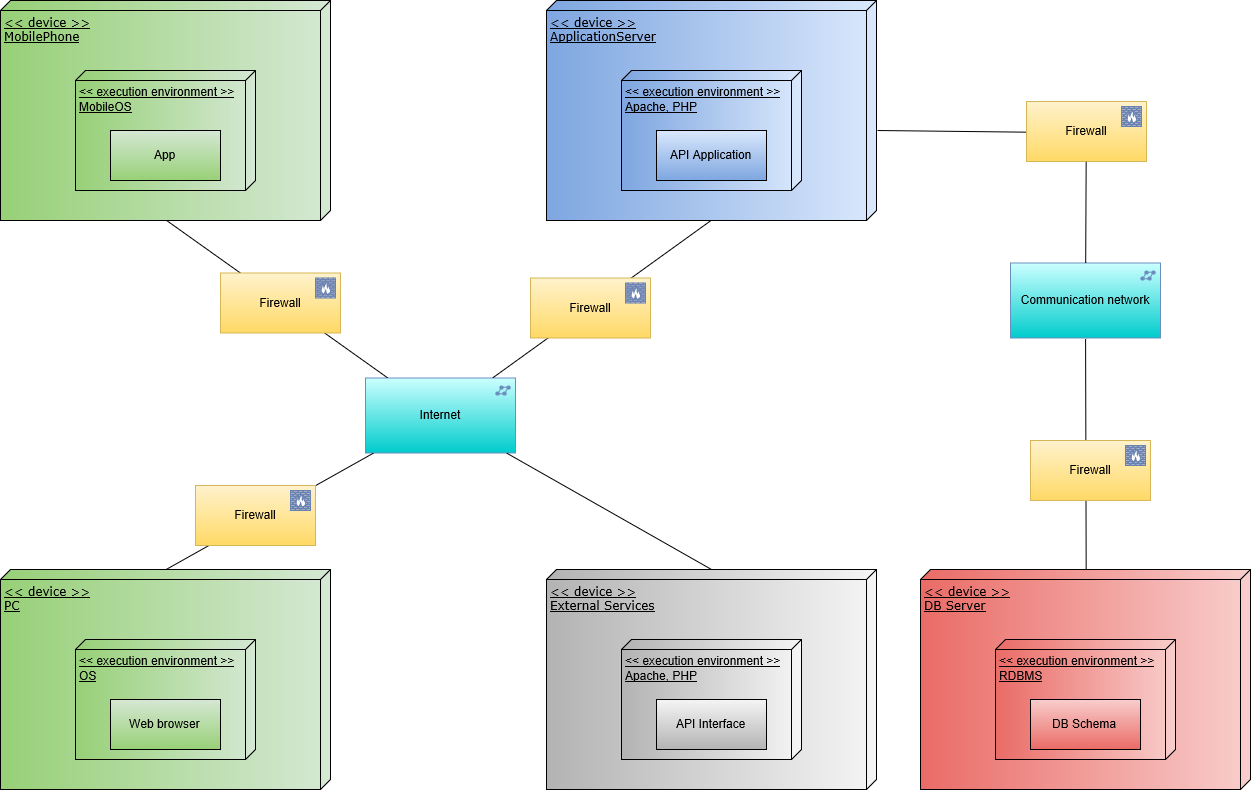
* RegistrationService: allows users to create new personal accounts
* LoginService: deals with the authentication of the users
* Data4HelpMonitoringService: manages the data received from the devices by storing and recovering them from the Database when needed
* RequestService: provides functionalities regarding the creation of new requests or the approval phase
* SubscriptionRequest: responsible for the managing of subscriptions
* AutomatedSOSMonitoringService: provides the mechanisms needed to monitor AutomatedSOSUsers and to continuously check their health status conditions
* EmergencyService: manages the connection with external medical service when needed and awaits confirmations of message received
* PathService: allows organizers to add a new run inside the application, listing place, date, time and other details of the race
* RunService: provides functionalities about the registration of a new runner to a specific run among the ones listed inside the database
* TrackService: deals with the mechanism that shows participants of a specific run on a map inside the application to the spectator who make a request for it

The logic of the application is composed by the interconnection of the services listed above and it works using the connection between each other to exploit different request given by the client. The presence of the router helps to distribute different task more easily inside the appServer and allows a better growth in the future of the application because adding new interfaces will be simpler.

The two diagrams below help giving a more detailed view of the UML constructed before and showing more features of the connection among services.



### 2.3 Deployment view



This deployment diagram shows the architecture of the system from a physical view point. Also, the distribution of software among the different hardware nodes is presented. The main nodes involved in the system are:

* *Client nodes*:
  + *Mobile phone*, users can access to Data4Help services through a mobile application that communicates with TrackMe’s server or directly with other external services (such as the street map service for Track4Run).
  + *PC* or other devices able to access to web services, Data4Help services are also available through its website, accessible via web browser.
* *Application server*, most of application logic belongs to this node. The application server communication with clients is based on the client-server pattern. This node also communicates with the system’s DB that stores all users’ data. The application server exploits external services API to provide a complete service to its clients.
* *DB server*, this node’s aim is to store all users’ data, from general account information to health status and location data. This node is directly accessed only by the application server.
* *External services*, client nodes and application server take advantage of this services to provide all AutomatedSOS and Track4Run functionalities.