Software Engineering 2

Design Document

Version 1.0

Andrei Constantin Scutariu	833370
Carlo Pulvirenti	828459
Sergio Piermario Placanica	916702

POLITECNICO
MILANO 1863

Academic Year 2018-2019

Contents

1 Introduction

1.1 Purpose

This document provides a specification on the architecture of TrackMe's system. It is complementary to the Requirements Analysis and Specification Document already presented, and it provides further description on its components, their interactions, and the implementation, integration and testing plan.

1.2 Scope

The document will focus on illustrating the following components of the application and its development process:

- System architecture
- UX
- Implementation, integration and testing plan

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Gated Commit a software integration pattern that reduces the chances for breaking a build (and often its associated tests) by committing changes into the main branch of version control.

Verigreen Verigreen is a lightweight, server side solution for verification of git commits. It is a gated check-in process which will not allow any failed CI commit to go into an integration/release/any protected branch.

Jenkins Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery

1.3.2 Acronyms

UX User experience

RASD Requirements Analysis and Specification Document

CIA triad Confidentiality, Integrity, Availability; a model designed to guide policies for information security within an organization

API Application Programming Interface

CI Continous Integration

CRUD Create, Read, Update, Delete; basic functions of persistent storage, also translates to proper HTTP verbs

REST Representational State Transfer

1.4 Revision history

Version	Release Date	Description
1.0	10/12/2018	Initial Release

1.5 Reference Documents

- Mandatory Project Assignment AY 2018/2019

1.6 Document Structure

The structure of this document is the following:

1. The first chapter serves as an introduction to this document.
2. The second chapters details the architecture of TrackMe's system: its components, the way they interact with each other, and how are grouped together. It also comprises sequence diagrams that show in more detail the interactions of such components for some selected goal to be achieved. Futhermore, and the end of this chapter, can be found the architectural and design patterns that are used in the system, their purpose and the reason for choosing them, along with other design decisions.
3. The third chapter futher describes the indications on the user interfaces and user experience provided in the RASD.
4. In the fourth chapter is provided a map on which components of the system every requirement and goal specified in the RASD is achieved.
5. The fifth chapter illustrates the plan for the implementation, integration and testing of the system to be developed.

2 Architectural design

2.1 Overview: High-level components and their interaction

TrackMe will be developed on a 3-tier architecture (*Fig. 1*), the rationale behind application servers separation is to mantain Data4Help service aimed at companies, independent from AutomatedSOS and Track4Run functionalities, aimed at users. Application servers will interact with a database server. Adding the tier 2 will provide services separation and will make possible to handle failures and updates without the whole system going down.

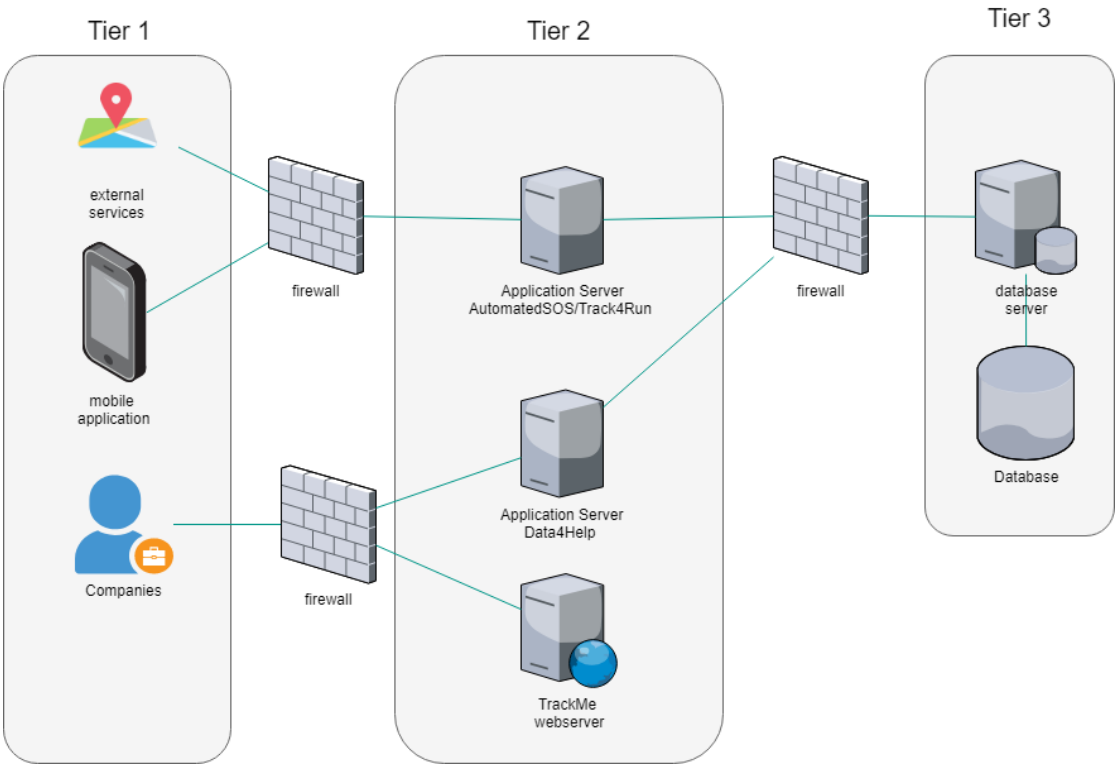

Figure 1: TrackMe architecture

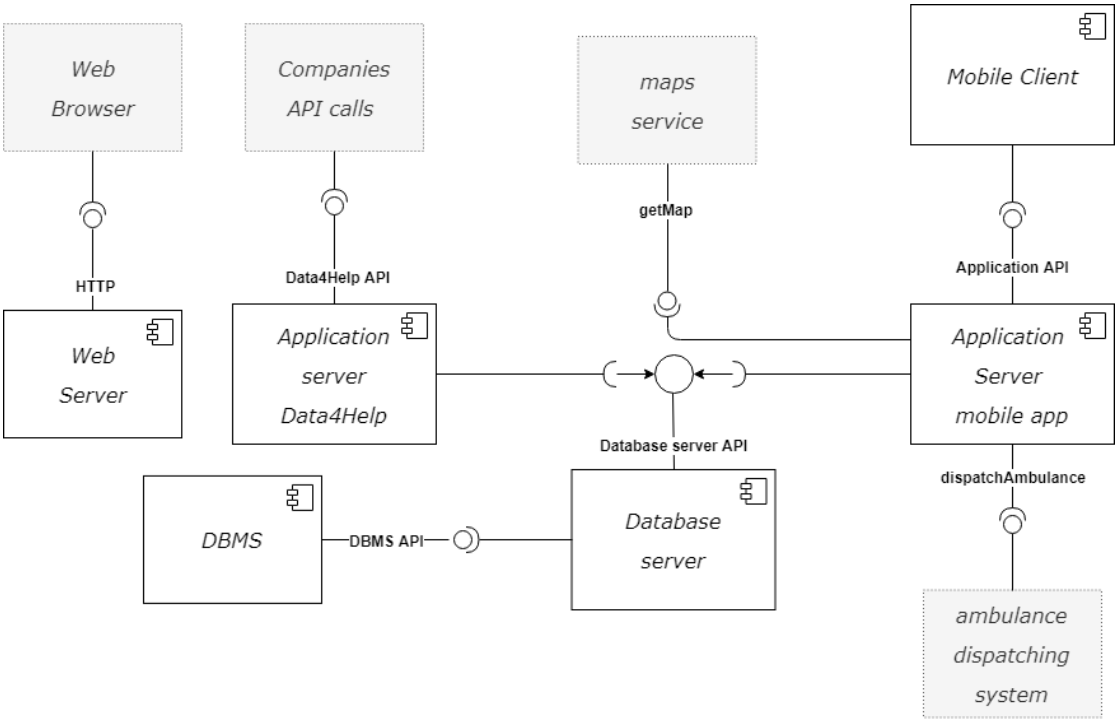

Figure 2: High level components view

2.2 Component view

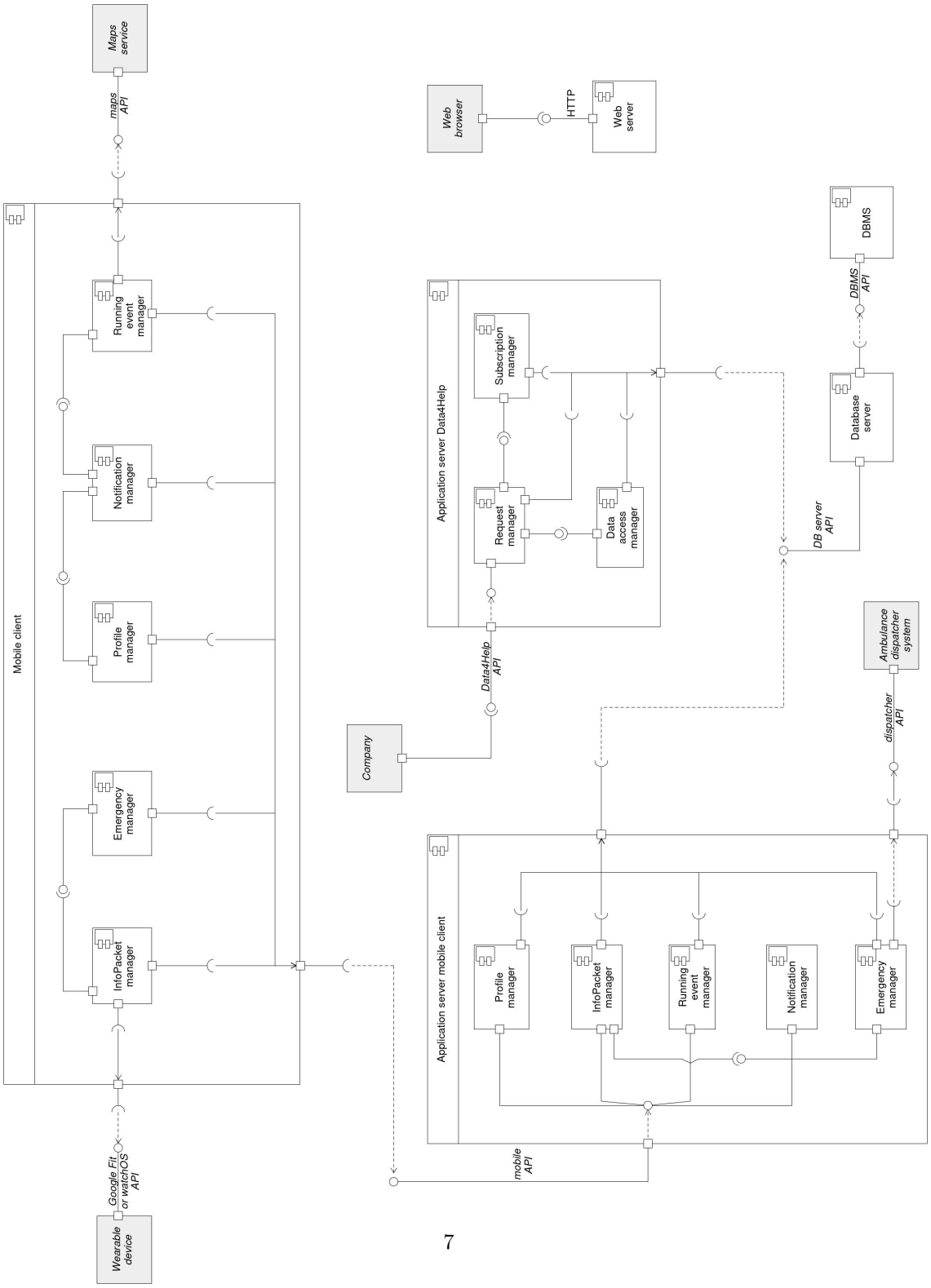

Figure 3: Component view

2.2.1 Mobile client

This is the mobile application. It will be built on top of the mobile platform's specific API, not represented in this diagram, and dependent on the wearable device's API (Google Fit or watchOS) and on the maps service provider's API for Track4Run functionalities. Moreover, it will communicate with the REST server, the Application server mobile client component, through HTTPS protocol, using the REST API the component will provide.

Profile manager This component will handle all functionalities regarding the user, such as registration, login, registering a new wearable device or removing a registered one, changing the preferences, accepting or refusing specific data requests sent from a company, and the activation or deactivation of AutomatedSOS and Track4Run services.

InfoPacket manager This is the component in charge of periodically getting the body parameters from a wearable device and location coordinates, and sending them to the Application server mobile client component.

Emergency manager This component is part of the AutomatedSOS service: it will handle an emergency situation, displaying useful information on the screen.

Running event manager This component is part of the Track4Run service: it will allow the user to create a new event, view the available events near his/her position, and enroll into them. It will also let the user view informations about an event in progress, like the position of the runners on a map.

Notification manager This component will simply display notification to the user when such is received from the server, such as a new specific request for data.

2.2.2 Application server mobile client

This is the REST server that will interact with the mobile client, and will forward every information received from it to the Database server component to store them. It will expose various REST endpoints to handle every action received from the mobile client; for this reason, its composition is very similar to the one found in the mobile client component. However the implementation of the interfaces used will be different: for example, the Emergency handler component, in this case, will interact with the external Ambulance dispatcher system through its API to dispatch and ambulance to the user's location.

2.2.3 Application server Data4Help

This is the REST server aimed to be used from companies. It will let them make requests for data, for a group of users or for a specific user, and subscribe for new informations.

Request manager This component will handle the data requests received from companies: through the help of the Data access manager component, will determine if the company can have access to such user's data, in case of a group request, or will register the request in case it's a specific one, and then eventually will send the data to the company as a response.

Subscription manager This component will handle the subscriptions a company may have, sending the data from the users to the company as soon as it's received.

Data access manager This component will determine if a company can have access to some data, after it has made a request for it. In particular, will count how many users satisfy a group request and authorize or reject it based on this number, and will also regulate the access to users data every request grants.

2.2.4 Database server

This is server is the only component that have access to the DBMS. Every other component will interact with the API this component makes available to read and write on the database. This allows the other components to be independent from the specific implementation of the database controller that will directly interact with the database, providing the ability to change how the system interfaces to the DBMS without altering all the components mentioned above. Moreover this design gives a further layer of security making the system resilient to a possible *CIA* security violation

2.2.5 Web server

This component will be a static server offering information about TrackMe and documentation on our APIs for third party companies. Also, as stated in the RASD, it will offer a form for companies to register on our service and request an API key.

2.3 Deployment view

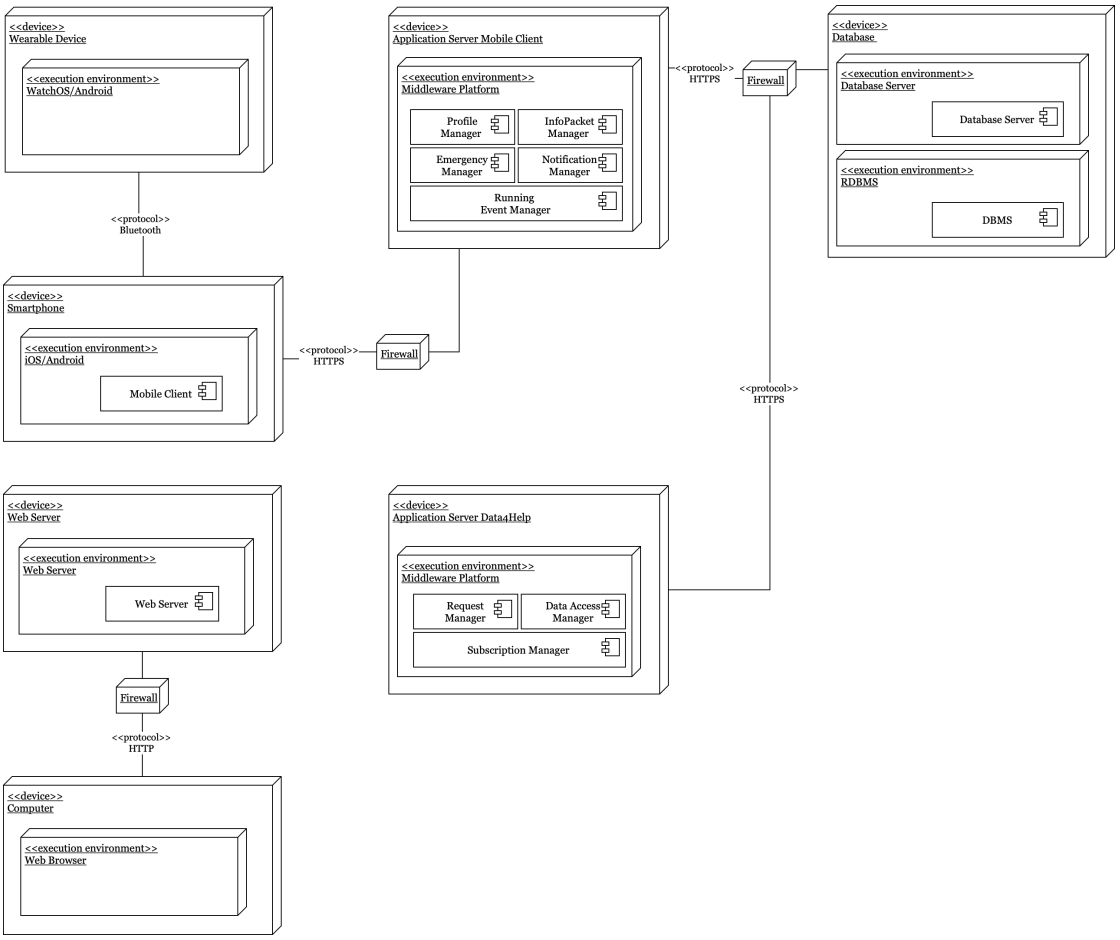

Figure 4: Deployment Diagram

2.4 Runtime view

In this section are shown the sequence diagrams of some features. They are useful to clarify the runtime behaviour of the components involved for each functionality.

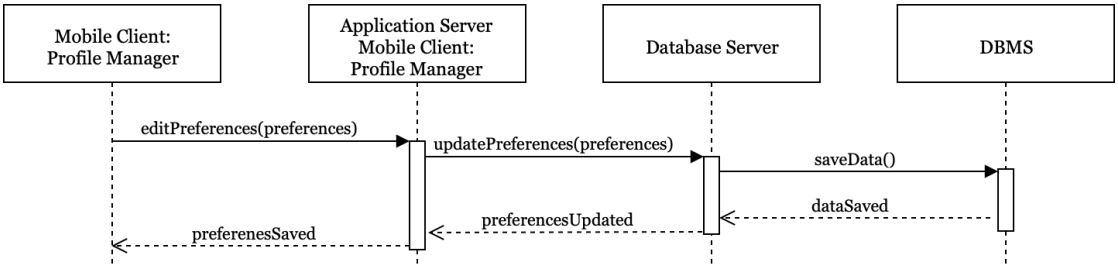

Figure 5: Sequence Diagram Edit Preferences

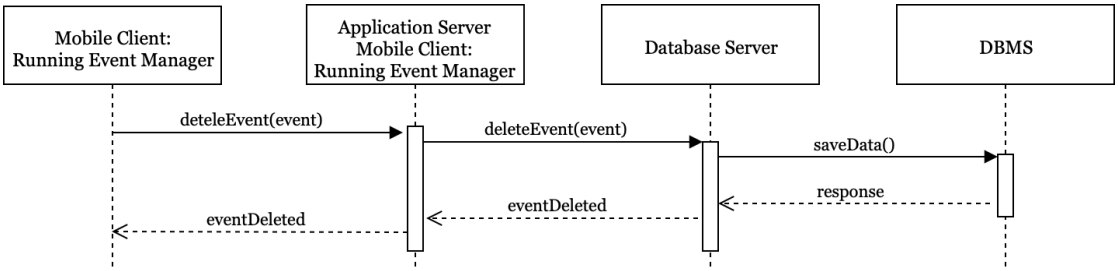

Figure 6: Sequence Diagram Delete Running Event

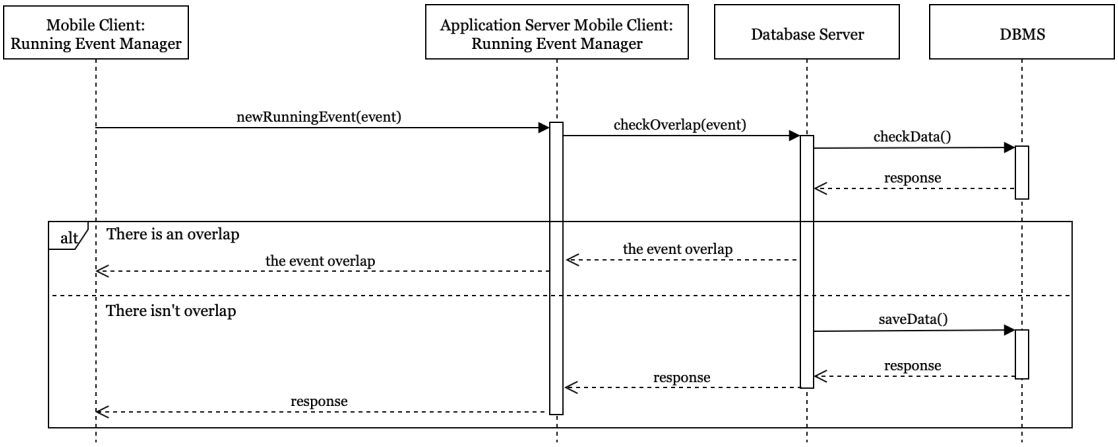

Figure 7: Sequence Diagram Create Running Event

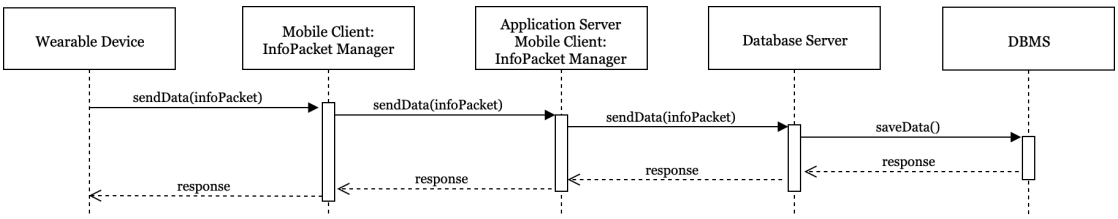

Figure 8: Sequence Diagram Data4Help Service

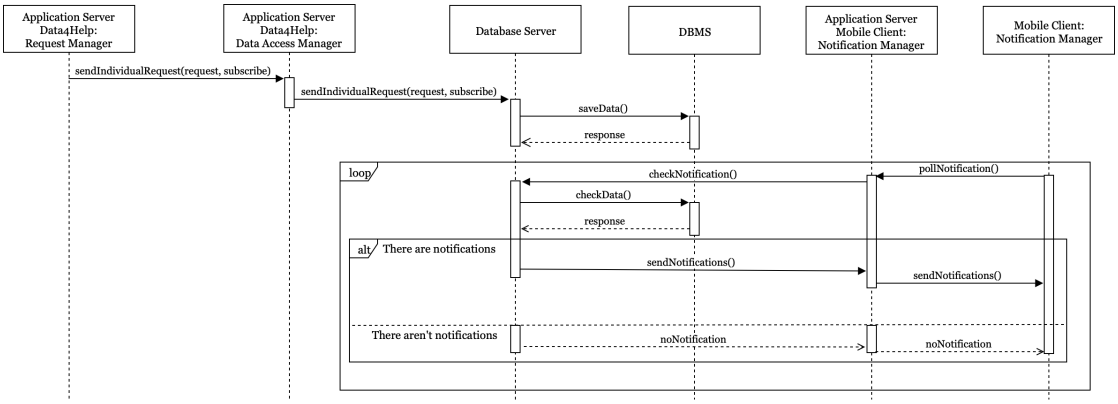

Figure 9: Sequence Diagram Data4Help Notification

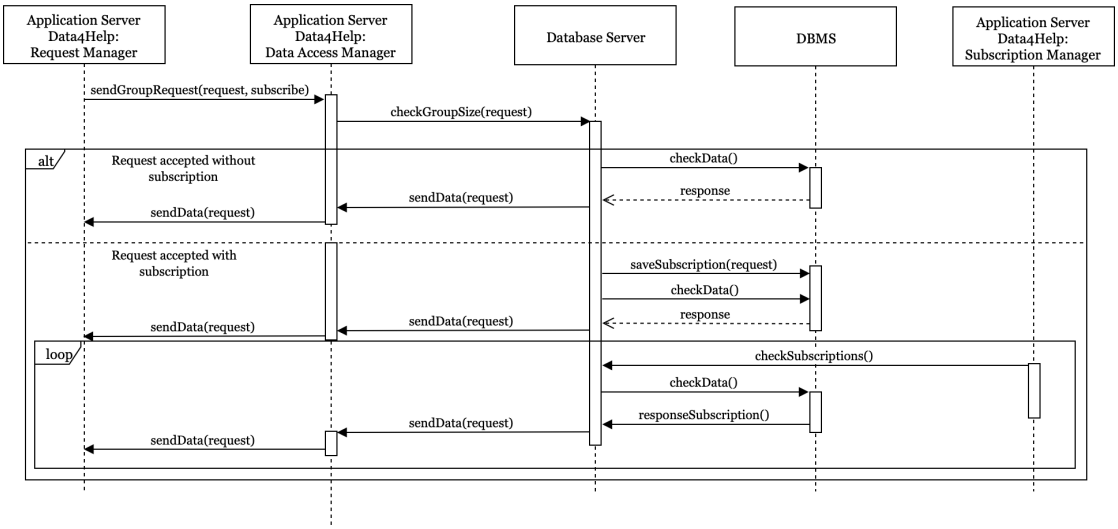

Figure 10: Sequence Diagram Data4Help Group Request Service

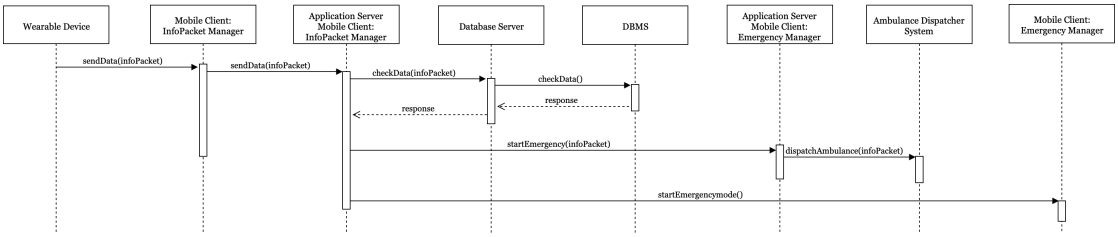

Figure 11: Sequence Diagram AutomatedSOS Service Dispatching Ambulance

2.5 Component interfaces

In this chapter are listed all interfaces that permit to the various components to communicate.

In the Application server components, the APIs exposed are composed of various interfaces that handle the appropriate functions.

Not represented here is the DB Server API, exposed by the Database server component, which is composed of all the functions necessary for the other components to interact with the DBMS.

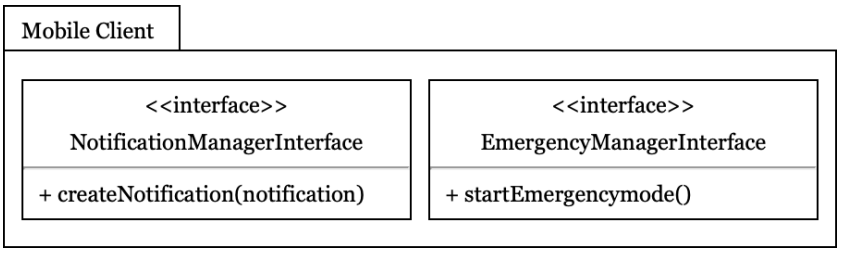

Figure 12: Component Interfaces Mobile Client

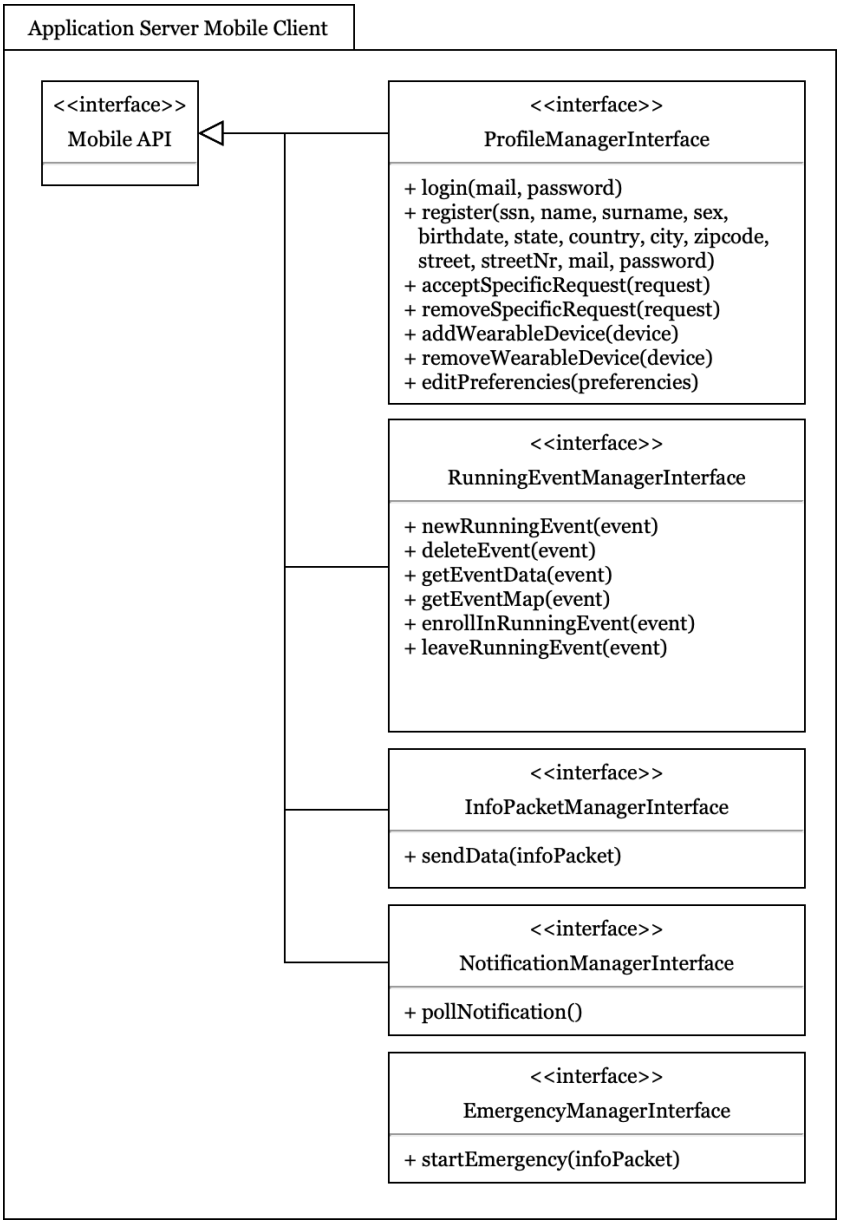

Figure 13: Component Interfaces Application Server Mobile Client

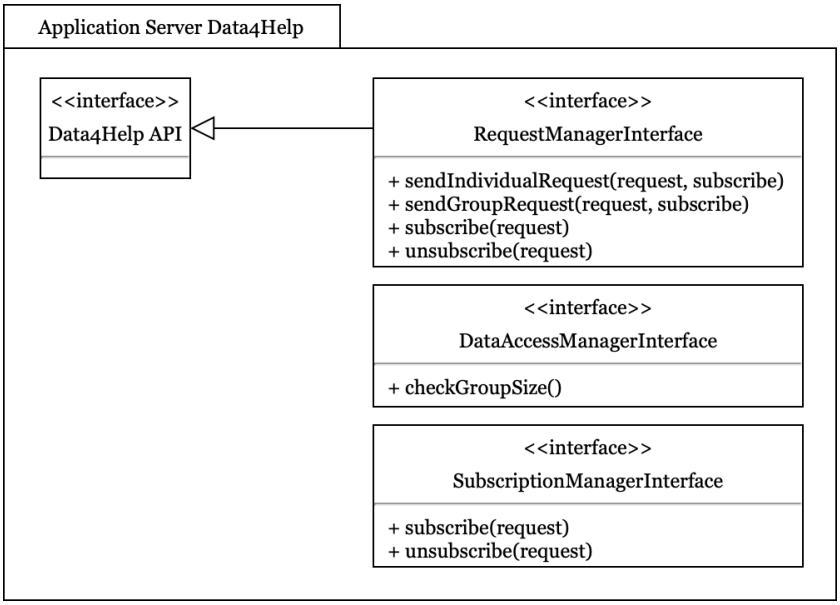

Figure 14: Component Interfaces Application Server Data4Help

2.6 Selected architectural styles and patterns

A client-server model will be followed the mobile application, while the part of the system that is oriented towards third party companies will be composed only by a server exposing them the useful APIs. All servers will be RESTful:

- The endpoints, i.e. the APIs they expose, will have meaningful URIs, pertient to the function they serve.
- The communication will rely on HTTP status codes and, when needed, error messages in the HTTP body for informing a successfull or unsuccessful operation.
- The interaction with them will follow the CRUD paradign to perform any operation.

This design allows for an event driven and resource centric system that is especially performant in managing a large number of customer interactions simultaneously (eg. frequently collected user's parameters).

2.7 Other design decisions

Registration and login functionality for end users will make use of HTTP basic authentication. The passwords shall be stored as hashes on the database following security best practices.

All HTTP traffic shall be encapsulated through SSL to preserve confidentiality.

Every registered company will have a unique authorized by TrackMe APIkey to make REST requests.

3 User interface design

Application UI design was presented in the RASD document (*RASD TrackMe, Section 3.1.1*). Below the UX model for the mobile application of TrackMe is presented.

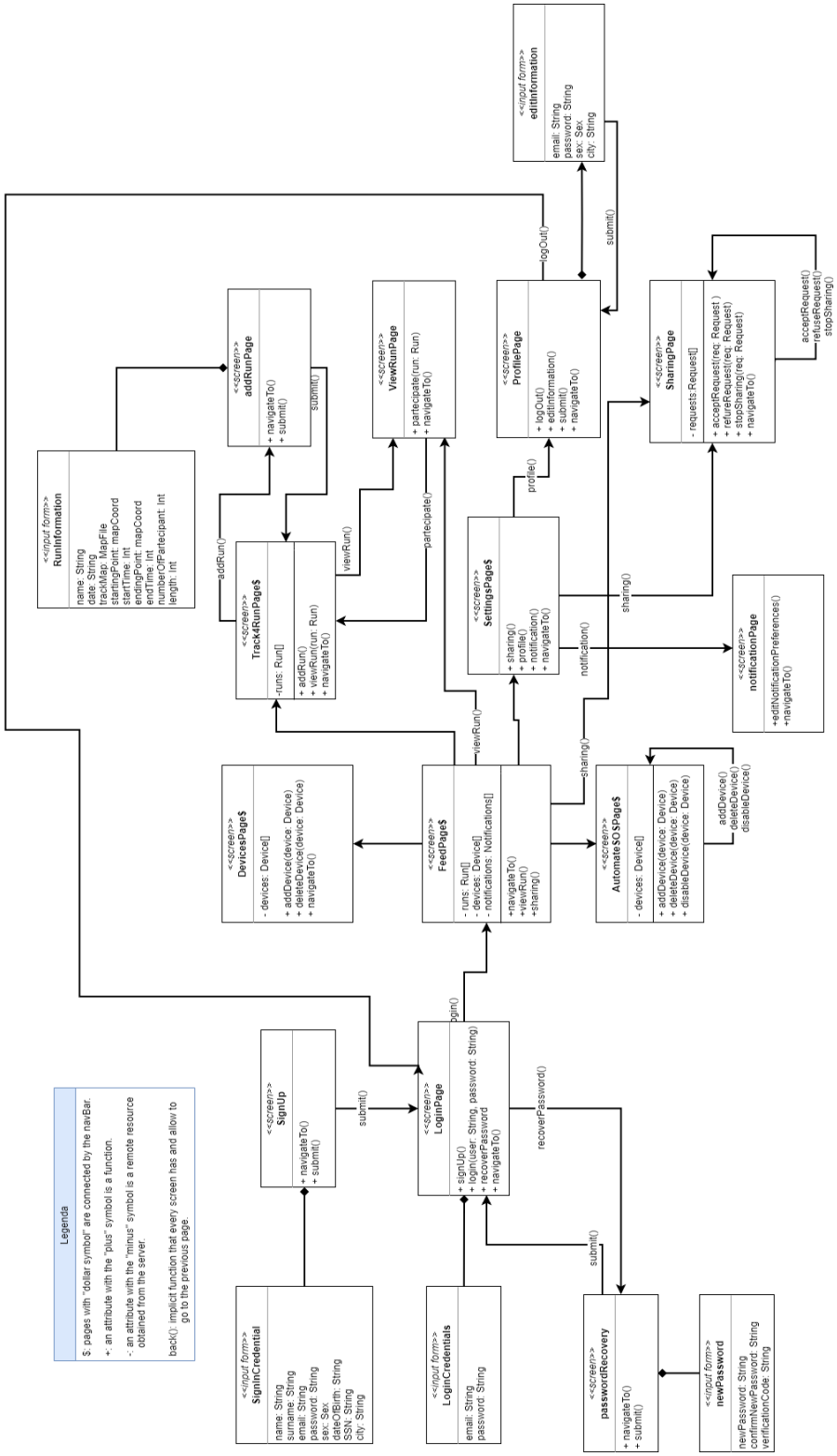

Figure 15: mobile application UX diagram

4 Requirements traceability

The following requirements were described in the RASD:

Data4Help

- **R1** The system must forward any request from a company to single customer's data to the corresponding user.
- **R2** The system must reject any request for data regarding a group of customers when it cannot guarantee the anonimity of its components.
- **R3** The system must periodically collect and store customer's data.
- **R4** The system must periodically update the accessible data with the newly collected one
- **R5** The system must notify the user of the request and permit him to accept or refuse it.
- **R6** The system must update the request status with the answer provided by the user.
- **R7** The system must be able to identify which data can be accessed for each third party company.

AutomatedSOS

- **R8** The system must continuously check the data read from customers subscribed to AutomatedSOS.
- **R9** In case the data indicate an emergency for a customer, the system must dispatch the closest ambulance to his location.
- **R10** After an ambulance is dispatched, the system must notify the customer of its arrival.

Track4Run

- **R11** The system must allow users to define the route, the date and the time of the event.
- **R12** Two events can't overlap in the same place, date and time.
- **R13** The system must show open events to the users.
- **R14** The system must allow users to sort and filter events by date and location.
- **R15** The system must collect and provide in real time the position of participants on a map.
- **R16** The system must allow the creator of a running event to cancel the event before it started.

Following there is a table where each requirement is mapped to the components that fulfill it.

Requirement	**Component(s)**
R1	Request manager Database server Profile manager
R2	Request manager Data access manager Database manager
R3	InfoPacket manager Database server
R4	Database server
R5	Profile manager Notification manager Database server
R6	Profile manager Database server
R7	Data access manager Database server
R8	InfoPacket manager
R9	InfoPacket manager Emergency manager
R10	Emergency manager Notification manager
R11	Running event manager
R12	Running event manager Database server
R13	Running event manager Database server
R14	Running event manager Database server
R15	Running event manager InfoPacket manager Database server
R16	Running event manager Database server

5 Implementation, integration and test plan

5.1 Implementation Plan

TrackMe system is composed of five macro components (*fig. 2*) that can be developed independently and simultanously after interfaces and communications standards are properly chosen.

5.2 Integration Plan

Although macro components will be developed indipendently, it is strongly suggested a top-down approach oriented towards functionalities implementation.

5.2.1 Mobile Client, Applications and Database Servers

The first thing to be developed will be the "core functionality of the system" and Data4Help (*Fig. 17*). Then it will be possible to test the backbone of the system (servers communication, models and APIs). Following the same approach, the development will focus consequently on AutomatedSOS and then on Track4Run.

5.2.2 Web Server

Being a totally independent part of the system, the Web Server will be developed by a different team or after the development of the other components.
The APIs documentation to be hosted on the Web Server will be produced by the Data4Help Application server developers.

5.3 Test Plan

Following the top-down approach, each module should be tested as soon as it is completed and integrated progressively.
Single modules development will implement Continous Integration practices to speed up development. It is advised to implement a *Gated Commit* pattern to avoid "code breaking" commits to the master codebase, that ultimatly will delay TrackMe development

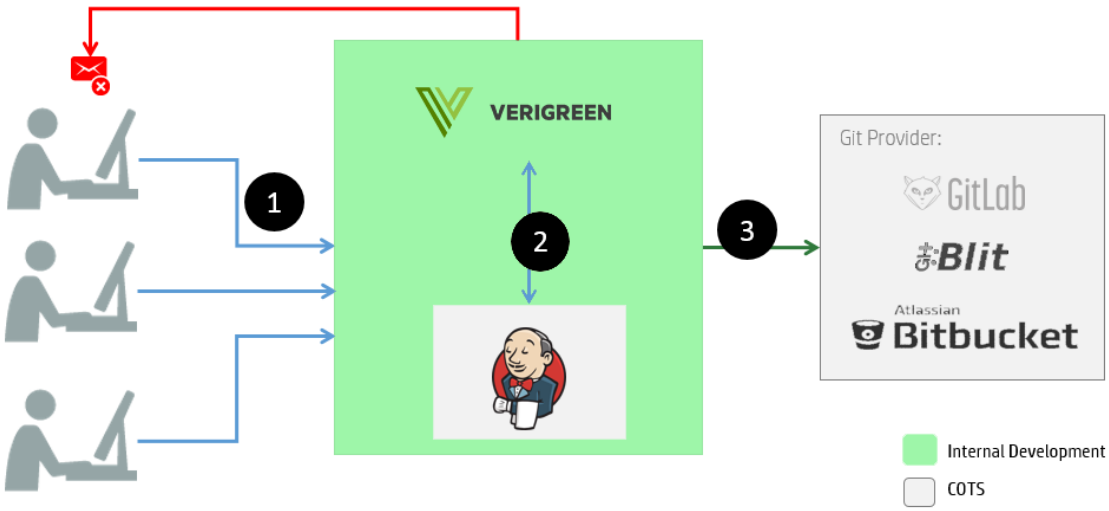

Figure 16: A gated commit pattern using a *jenkins* server with *Verigreen*

5.3.1 System Testing

The application will be firstly beta tested in the urban area of Milan, thus it will be possible to gather statistics about application usage and user base to better plan a final release.
The system must be able to withstand constant data upload from our users, it's advised a load test.
We expect companies to make a large volume of request to Data4Help server, so it should be subject to load and stress tests.

5.4 API testing

It's critical to test the correct behaviour of TrackMe API. The best approach will be to use a dedicated software like *POSTMAN* to test API as soon as they are developed.

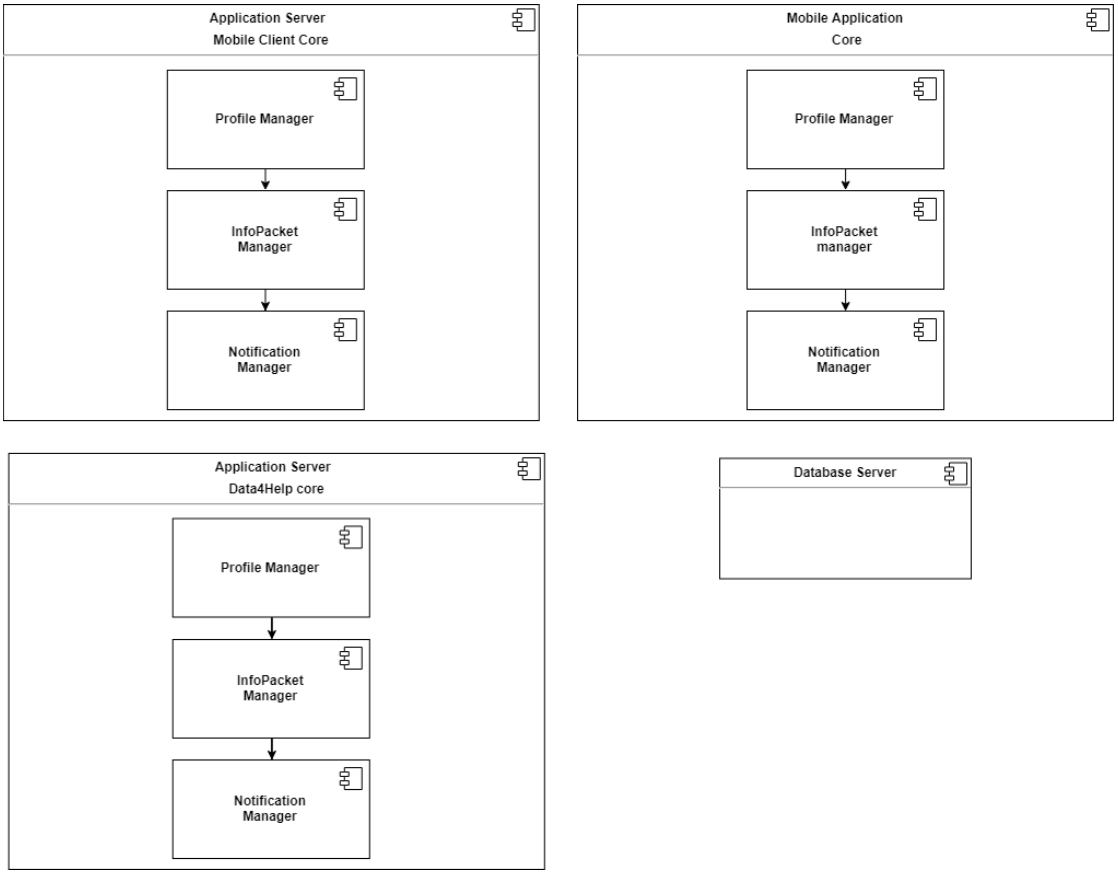

Figure 17: core functionality development

6 Effort spent

Andrei Constantin Scutariu

Date	Hours	Description
25/11/18	0.5h	Initial Structure
28/11/18	0.5h	Introduction
29/11/18	2h	Component view
30/11/18	1h	Component view
01/12/18	1h	Requirements traceability
04/12/18	2h	Reviewing
05/12/18	2h	Architectural styles and patterns
06/12/18	4h	Component Interfaces
07/12/18	1h	Component view
07/12/18	1h	Reviewing
10/12/18	3h	Reviewing
total	18h	

Carlo Pulvirenti

Date	Hours	Description
28/11/18	1h	Architectural Design
29/11/18	3h	Runtime View
30/11/18	2h	Runtime View
02/12/18	4h	Runtime View
04/12/18	2h	Reviewing
05/12/18	3h	Deployment View
06/12/18	4h	Component Interfaces
10/12/18	3h	Reviewing
total	22h	

Sergio Piermario Placanica

Date	Hours	Description
29/11/18	2h	Architechtural Design
30/11/18	1h	Architectural Design
04/12/18	2h	Reviewing
04/12/18	3h	Implementation and Testing
08/12/18	3h	UX diagram
10/12/18	4h	Reviewing
total	14h	