POLITECNICO
MILANO 1863

Software Engineering 2
Design Document

Work team:
Paolo Romeo, Andrea Scotti, Francesco Staccone

AY 2018-2019

Table of contents

1. Introduction

1.1. *Purpose*

This document is thought to be an overview of the TrackMe application, in which is explained how to satisfy the several project requirements stated in the RASD. This document is principally intended for the developers and the testers, with the purpose of providing a functional description of the main architectural components, their interfaces and their interactions, along with the design patterns.

1.2. *Scope*

—Presentation—
TrackMe is structured in a multi-tier architecture. More specifically, the Business logic layer has the task of taking charge of the incoming requests/data, computing checks, and interacting with external third-party services through the use of interfaces. This layer is connected with the Data layer, in which are stored all the Users data (credentials, health data). The Presentation layer is build through the fat Client paradigm in which the client needs to perform computation to temporarily store health data in the local memory in case of missing internet connection between the client and the server.

1.3. *Definitions, acronyms, abbreviations*

1.3.1. Definitions

- Client: A client is a piece of computer hardware or software that accesses a service made available by a server;
- Firewall: A network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules;
- Server: A computer program or a device that provides functionality for other programs or devices, called "clients".

1.3.2. Acronyms

- API: Application Program Interface;
- DBMS: Database Management System;
- DD: Design Document
- GUI: Graphical User Interface;
- HTTP: Hypet Text Transfer Protocol;
- MVC: Model View Controller pattern;
- OS: Operating System;
- RASD: Requirements Analysis and Specifications Document;
- REST: REpresentational State Transfer;

1.3.3. Abbreviations

- Gn: n-goal in the RASD;
- Rn: n-functional requirement in the RASD;

1.4. *Revision history*

1.5. *Reference documents*

- RASD document;
- Mandatory project assignment;

1.6. *Document structure*

The following document is organised in this way:

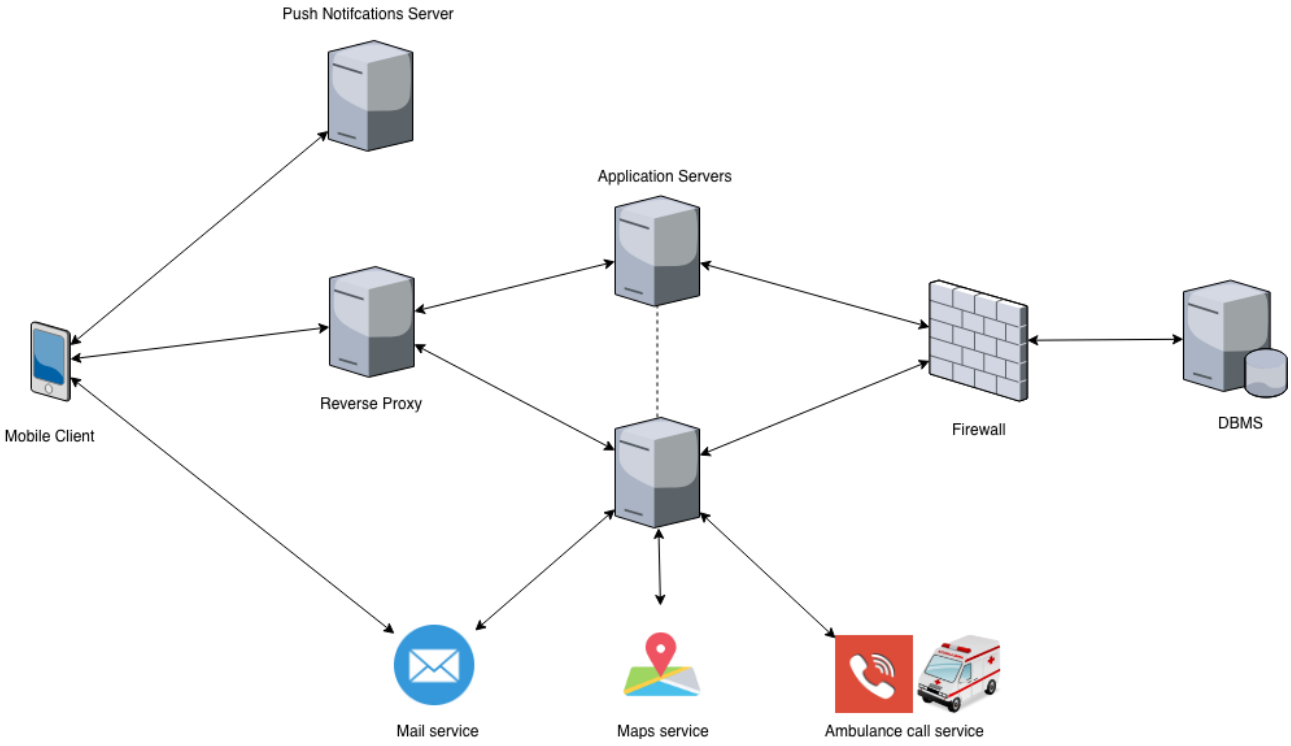
- Architectural Design: this section shows the main components of the system and the connections among them. It will also focus on design choices, styles and patterns.
- User Interface Design: this section includes an improvement of the user interface given in the RASD document. It will be described through the use of UX modeling.

- Requirements Traceability: this section shows how the requirements in the RASD are mapped to the design components presented in the DD;
- Implementation, Integration and Test plan: this section shows the order in which the implementation and the integration of the subcomponents will occur and how the integration will be tested.

2. Architectural design

2.1. *Overview: high-level components and their interaction*

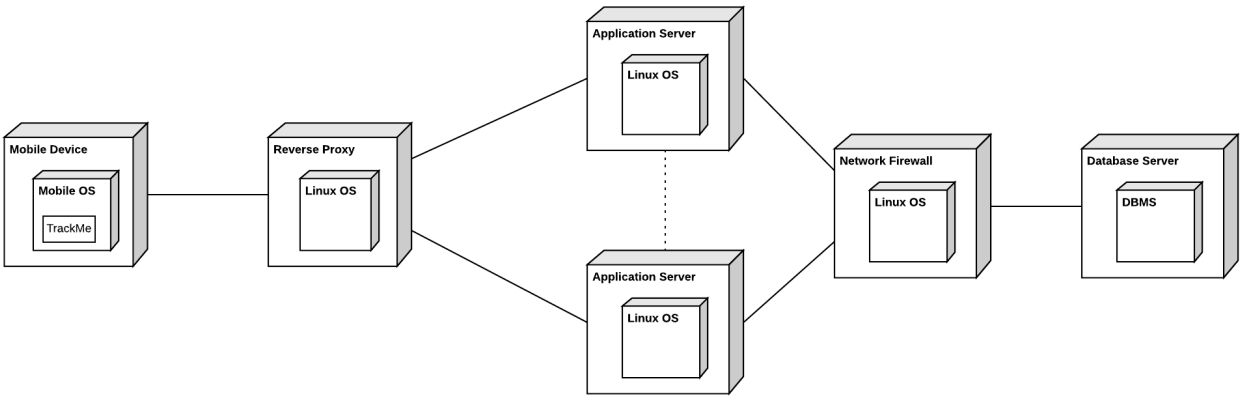
The figure below represents an high level overview of the system. Further details on the system components and their interaction will be further explained in the next sections.



2.2. *Component view*

2.3. *Deployment view*

The figure below shows the deployment diagram of the whole system. Its main goal is to describe the distribution of components capturing the topology of the system's hardware.



As previously stated in the section 1, the system is structured in a multi-tier architecture. The specific role of each node is clarified here:

Clients
The first tier is composed by the mobile clients machines. The client will be able to access TrackMe functionalities through the dedicated native application.

Reverse Proxy
We chose to deploy a reverse proxy on a Linux machine with Nginx server installed on it, in order to safely increase parallelism of requests and scalability of our application. This architecture leads to several advantages:

- Load balancing, distributing requests among different servers
- Optimising content by compressing it in order to speed up loading times
- Event-driven architecture, increasing parallelism by not locking the CPU
- Very conservative memory-wise

Application Servers
This is the middleware level of the architecture: all the business logic of the system is contained in these servers.

Network Firewall
The access to the Database is mediated by a network firewall in order to avoid unauthorised access to the data and the credentials of the user.

Database Server
This is the last layer of the architecture: all the data are stored in a Database Server accessed through a relational DBMS.

2.4. *Runtime view*

2.4.1. **Sign up Runtime View**

2.4.2. **Login Runtime View**

2.4.3. **Join a run Runtime View**

2.4.4. **Organise a run Runtime View**

2.4.5. **Individual request Runtime View**