**Table of Contents**

[**1.** **Introduction** 3](#_Toc531944635)

[**1.1.** **Purpose** 3](#_Toc531944636)

[**1.2.** **Scope** 3](#_Toc531944637)

[**1.3.** **Definitions, Acronyms, Abbreviations** 4](#_Toc531944638)

[**1.3.1.** **Definitions** 4](#_Toc531944639)

[**1.3.2.** **Acronyms** 4](#_Toc531944640)

[**1.3.3.** **Abbreviations** 4](#_Toc531944641)

[**1.4.** **Document Structure** 4](#_Toc531944642)

[**2.** **Architectural design** 6](#_Toc531944643)

[**2.1.** **Overview** 6](#_Toc531944644)

[**2.2.** **Component view** 6](#_Toc531944645)

[**2.3.** **Deployment view** 7](#_Toc531944646)

[**2.4.** **Runtime view** 7](#_Toc531944647)

[**2.5.** **Component interfaces** 7](#_Toc531944648)

[**2.6.** **Selected architectural styles and patterns** 8](#_Toc531944649)

[**3.** **User interface design** 9](#_Toc531944650)

[**4.** **Requirements traceability** 9](#_Toc531944651)

[**5.** **Implementation, integration and test plan** 10](#_Toc531944652)

[**5.1.** **Implementation plan** 10](#_Toc531944653)

[**6.** **Effort Spent** 10](#_Toc531944654)

[**7.** **References** 10](#_Toc531944655)

# **Introduction**

## **Purpose**

The purpose of this document is to give more technical details than the RASD about TrackMe system in order to provide an overall guidance to the architecture of the software product.

While the RASD presented a general view of the system and what functions the system is supposed to execute, this document aims to present the implementation of the system including main components and their interfaces, run-time behaviours, the high-level architectures and the corresponding deployment design. It also presents in more details the implementation and integration plan, as well as the testing plan.

## **Scope**

The project TrackMe, which is a service-based on mobile application and web application, has two different targets of Customers:

* Third-Parties
* Users

First of all, the system must provide the registration and login services.

To log into the system, both the Users and the Third-Parties will use their own credentials, such as username and the related password.

More precisely, the sign up and the sign in processes are carried out via:

* web app by Third Parties
* mobile application by the Users

Furthermore, the system allows Third Parties to require accessing to Users’ data via web app. More precisely, after choosing the type of request (single User data request or data w.r.t a specific group of anonymous Users) and filling out the corresponding fields, Third-Parties can make a data request.

On the other side the mobile app allows Users to see their own pending requests, to accept or reject a new single User data request or to withdraw a previously accepted request.

Group requests are, however, handled directly by the system in order to verify if it is able to properly anonymize the requested data.

The answer to a specific data request must be communicated on the corresponding Third-Party’s web app and for the accepted requests the system will provide a set of APIs in order to allow Third Parties to access data.

Through the mobile app, Users can also keep track of their health status by comparing them with the threshold values and update some fields of personal data (weight, height, etc.).

Finally, Users via mobile app can also subscribe to the AutomatedSOS service. For these Users the system must monitor constantly their health status in order to notify an external Ambulance Service in case of emergency.

## **Definitions, Acronyms, Abbreviations**

## **Definitions**

* **3-tier architecture:** acsbkcb
* Ascancja
* sacnla

## **Acronyms**

* **RASD:** Requirement Analysis and Specification Document
* **DD**: Design Document
* **MVC**: model view controller
* **REST**: REpresantional State Transfer
* **API:** Application Programming Interface
* **GPS:** Global Positioning System
* **Bpm:** Beats Per Minute
* **SSN:** Social Security Number

## **Abbreviations**

* **[Gn]:** n-th goal
* **[Rn]:** n-th functional requirement
* **Tp-a:** Third-Party administrator

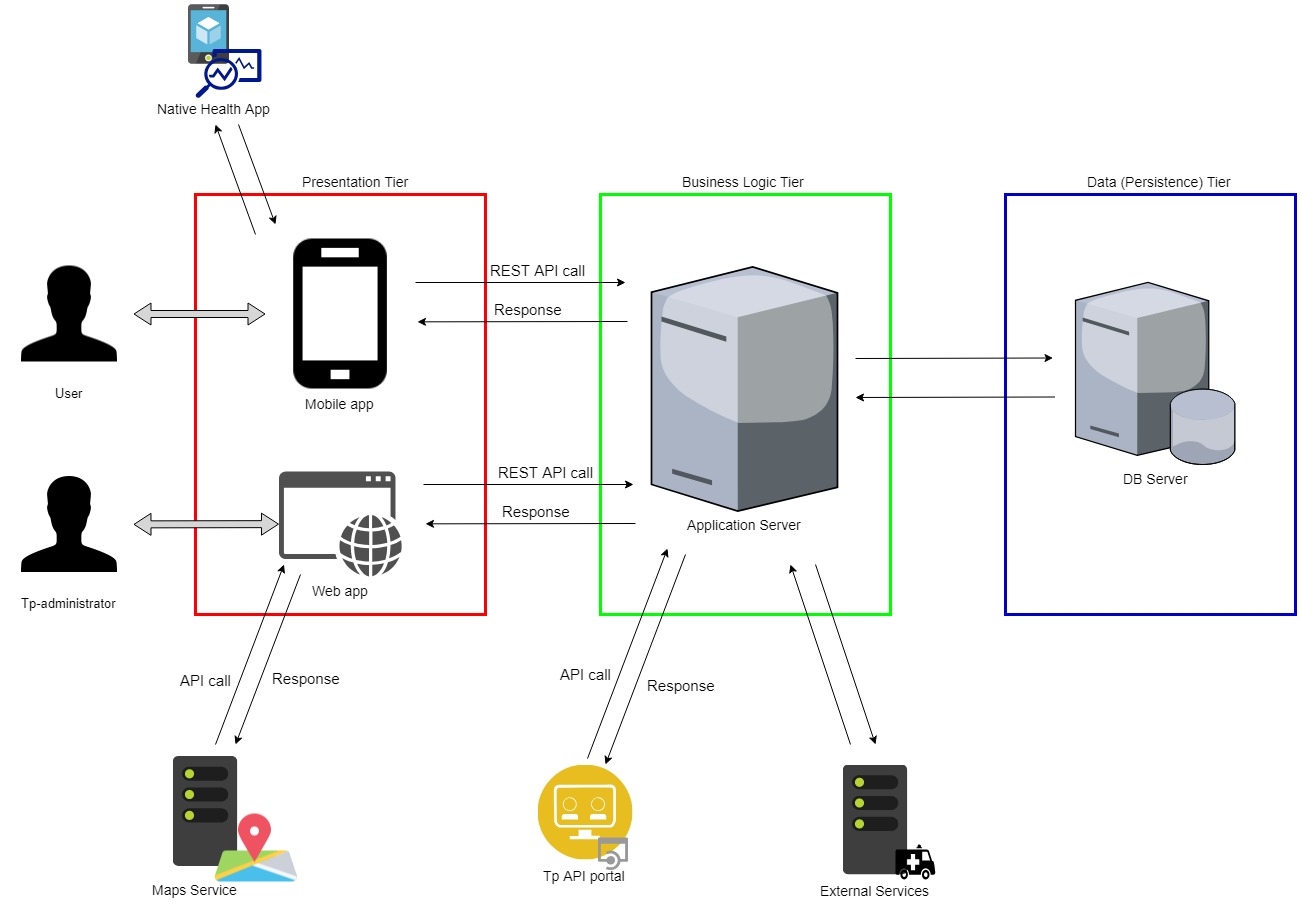
## **Document Structure**

This document is divided into six sections:

* Section 1 gives an introduction of the design document. It contains the purpose and the scope of the document, as well as some abbreviation in order to provide a better understanding of the document to the reader.
* The second section deals with the architectural design of the application.   
  It gives an overview of the architecture and it also contains the most relevant architecture views: component view, class view, deployment view, runtime view and it shows the interaction of the component interfaces. Some of the used architectural designs and designs patterns are also presented here, with an explanation of each one of them and the purpose of their usage.
* Section 3 refers to the User Design Interface previously presented in the RASD document through some mock-ups w.r.t the Users and the Third Parties interfaces.
* The fourth section explains the mapping between the requirements previously defined in the RASD and the design elements that are defined in this document
* The fifth section provides the description of the implementation and testing strategy adopted in the whole project and of the order in which it is planned to integrate such subcomponents.
* Section 6 shows the effort spent by each group member while working on this project.
* Section 7 includes the reference documents

# **Architectural design**

## **Overview and High-Level Architecture**

The TrackMe system is based on a widely used 3-tier Architecture:

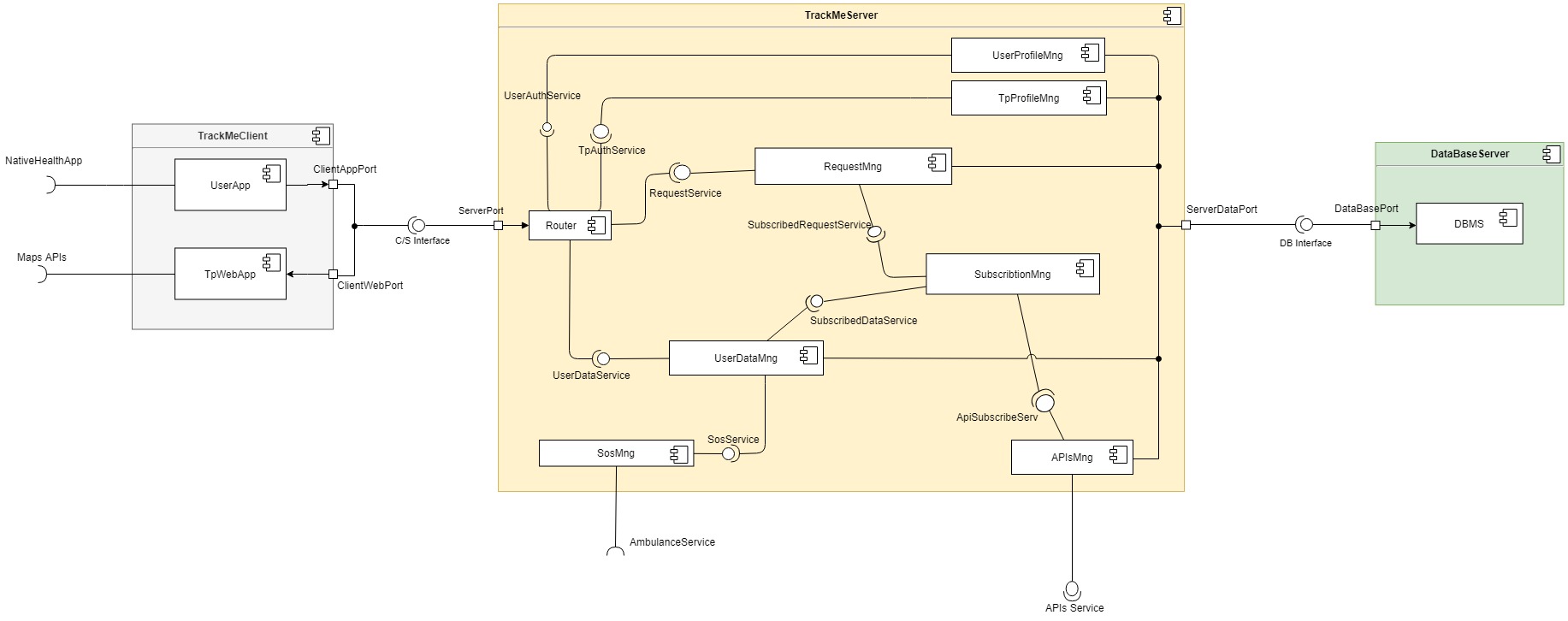
Where:

* **Presentation tier**: hgfg
* **Business Logic tier**: vhv
* **Data (Persistence) tier**: ygug

## **Component view**

The previously introduced components are more precisely described and examined by using a component diagram as follows.  
While describing the parts, the notation will be the interface they are providing in order to be more general.

The main focus of the diagram is the application server and, except for the database server and its DBMS, the other external services are not depicted here. Indeed, the diagram deals only with the services that they provide, while their structure will not be analysed here, since they are observed as black-boxes.



As we can see the application Server consist of the following components:

* **UserProfileMng**: the component responsible for the authentication request by the user (both registration and log in requests).  
  For each authentication request by a user, after receiving credentials and checking with the Database, it interacts (through the Router) with the mobile app to provide a token associate to that user. This token will be subsequently used by the mobile app to interact with the other components of the application server.
* **TpProfileMng**: the component responsible for the authentication request by a Third Party (both registration and log in requests).  
  It works exactly the same as the UserProfileMng to provide a token to the web app.
* **RequestMng**: the component responsible for the
* **SubscriptionMng**:
* **UserDataMng**:
* **SosMng**:
* **APIsMng**:

## **Deployment view**

ascasc

## **Runtime view**

saccsa

## **Component interfaces**

ascaacs

## **Selected architectural styles and patterns**

## **Overall architecture**

The most suitable architecture for TrackMe would be three tier architecture.

A three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

Three-tier architecture is a software design pattern and a well-established software architecture that enable the distribution of application functionality across three independent systems, typically:

* Client components running on local workstations (tier one)
* Processes running on remote servers (tier two)
* A discrete collection of databases, resource managers, and mainframe applications (tier three)

More precisely:

* **Presentation Tier (first tier):** Responsibility for presentation and user interaction resides with the first-tier components. These client components enable the user to interact with the second-tier processes in a secure and intuitive manner through a set of REST API calls.

In the TrackMe system the presentation tier is a graphical user interface accessible either through a mobile application by individual Users and through a web app by Third-Parties.  
Furthermore, the mobile application communicates with a native health app in order to collect constantly health status values, while the web app use the APIs provided by a Maps Platform.

* **Business Logic Tier (second tier):**  it manages the business logic of the application, and permits access to the third-tier services. The application logic layer is where most of the processing work occurs. Multiple client components can access the second-tier processes simultaneously, so this application logic layer must manage its own transactions.

Without an application logic layer, client components access the database directly. The database is required to manage its own connections, typically locking out a record that is being accessed.

Separating the second and third tiers reduces the load on the third-tier services, supports more effective connection management, and can improve overall network performance.

In TrackMe system this tier communicates also with external services (Ambulance Service) for guarantying the AutomatedSOS service and with the Third-Parties’ APIs portals for sending the required data.

* **Data Persistence Tier (third tier):** The third-tier services are protected from direct access by the client components residing within a secure network. Interaction must occur through the second-tier processes.

There are many benefits to using a 3-layer architecture including speed of development, scalability, performance, and availability.  As mentioned, modularizing different tiers of an application gives development teams the ability to develop and enhance a product with greater speed than developing a singular code base because a specific layer can be upgraded with minimal impact on the other layers.

Scalability is another great advantage of a 3-layer architecture. By separating out the different layers you can scale each independently depending on the need at any given time. This allows to load balance each layer independently, improving overall performance with minimal resources. Additionally, the independence created from modularizing the different tiers provides many deployment options.

As already mentioned, data corruption through client applications can be eliminated since the data passed in the middle tier for database updates ensures its validity.

Moreover, the implementation of several layers makes the data more secure. As clients do not interact with the database directly, it provides less risk and confliction with unauthorized data.

Finally, the actual structure of the database often remains hidden from requesters enabling any change of the database to be transparent. Thus, a process in the middle tier which exchanges data with other applications can sustain its current interface while a modification of the underlying database structure

## **Design patterns**

* **Proxy Pattern**

It is a structural design pattern that provides a substitute or placeholder for another object. The router component functions as a proxy which controls access to the original objects, allowing to perform something either before or after the request gets through to them, represented in the TrackMe system by the other components.

Benefits:

* + It controls the service objects without clients knowing about it
  + It can manage the lifecycle of the service objects
  + It increases the performance of the application, by avoiding duplication of objects which might be huge size and memory intensive
  + It enhances scalability, since it is possible to introduce new proxies without changing the services or clients
* **Event-driven architecture style**

This architecture consists of event producers that generate a stream of events, and event consumers that listen for the events.

Events are delivered in real time, so consumers can respond immediately to events as they occur.

In our scope this architecture is used to handle the sending in real time of new produced data to the subscribed Third-Parties.  
TrackMe system uses pub/sub model for this architecture. When an event is published, it sends the event to each subscriber. After an event is received, it cannot be replayed, and new subscribers do not see the event.

More precisely, in TrackMe:

* The event is the update of the health status of a specific User
* The event producer is the mobile application
* The event consumers are the Third-Parties’ APIs portals

Benefits:

* Producers and consumers are decoupled.
* No point-to point-integrations. It's easy to add new consumers to the system.
* Consumers can respond to events immediately as they arrive.
* Highly scalable and distributed.
* Subsystems have independent views of the event stream.

## **Other Design Decisions**

The web application needs an integration with a map service in order to allow Third-Parties to select the geographical area in group requests. It will be integrated using the APIs provided by the Maps platform.

The mobile needs an integration with a native health app installed on the smartphone in order to allows Users’ health status collection.

Others design pattern widely recommended:

* **Model View Controller (MVC)**

For the implementation of the mobile application and the web application is recommended to adopt this commonly architectural pattern which is used for developing user interfaces that divides an application into three interconnected parts. The mainly benefits are:

* + **Faster development process:** MVC supports rapid and parallel development.
  + **Ability to provide multiple views:** In the MVC Model, you can create multiple views for a model. Code duplication is very limited in MVC because it separates data and business logic from the display.
  + **Modification does not affect the entire model:** Modification does not affect the entire model because model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture.

# **User interface design**

The mock-ups for this application were presented in the RASD Document in section “3.1.1. User Interfaces”.

At this point, there are no new functionalities that can be presented with new mock-ups for the application.

# **Requirements traceability**

The design of this application aims to meet all the goals and requirements that have been previously specified in RASD document.   
Below are listed the design components to which TrackMe requirements and goals are mapped:

* **[G1]**: Allow individuals to become registered users of Data4Help (requirements **[R1]**, **[R2]**, **[R3]**):
  + UserProfileMng
* **[G2]**: Allow users to sign up for AutomatedSOS service (requirements **[R4]**, **[R5]**):
  + UserProfileMng
* **[G3]**: Provide the registration to third parties who want access to users’ data (requirements **[R6]**):
  + TpProfileMng
* **[G4.1.]**: Give third parties access to data of a specific user (requirements **[R7]**, **[R8]**):
  + RequestMng
  + APIsMng
* **[G4.2]**: Give third parties access to anonymized data of group of users (requirements **[R9]**, **[R10]**):
  + RequestMng
  + APIsMng
* **[G5]**: Allow users to accept or refuse the requests from third-parties to access their own data and their location (requirements **[R11]**):
  + RequestMng
* **[G6]**: Allow third parties to subscribe to new data and to receive them as soon they are produced (requirements **[R12]**, **[R13]**):
  + UserDataMng
  + SubscriptionMng
  + RequestMng
  + APIsMng
* **[G7]**: Allow customers to insert or update their own personal data and information about their body measurements (e.g. weight, height, etc.) (requirements **[R14]**):
  + UserProfileMng
* **[G8]**: If some parameters of health status are below certain threshold, send an ambulance to the user location, with a reaction time less than 5 seconds (requirements **[R15]**, **[R16]**):
  + UserDataMng
  + SosMng
* **[G9]**: Allow users to keep track of their health status at any time (requirements **[R17]**):
  + UserDataMng
* **[G10]**: Allow users to withdraw the authorisation to third parties to access their data (requirements **[R18]**, **[R19]**):
  + RequestMng

# **Implementation, integration and test plan**

## **Implementation plan**

saicnsacnacs

# **Effort Spent**

Antonio: 6+ 6+

Enrico: 6 + ?+

# **References**

* Specification document *“A.Y. 2018-2019 Software Engineering 2 Mandatory Project: goal, schedule, and rules”*
* RASD document
* Old Design Documents: “*DD to be analysed*” and more previous documents.
* Slides provided on Beep channel
* Architecture styles: “<https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/>”