POLITECNICO DI MILANO
Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

TrackMe

Design Document
(DD)

Reference professor:
Elisabetta Di Nitto

Mandatory Project:
Alessia Buccoliero, matricola 920484
Emilio Corvino, matricola 920429
Gianluca Drappo, matricola 920155

Anno Accademico 2018-2019

Version 1.0
10/12/2018

Contents

Chapter 1

Introduction

1.1 Purpose

This document will provide an a description of the three new services the *TrackMe* company is willing to launch with respect to their design. Such description will include an analysis of the components the applications will be made of, and the interaction among them.

A detailed list of the goals, the requirements and the domain assumptions of the applications are provided in the **Appendix A**.

1.2 Scope

Nowadays being able to retrieve users data is very important for many companies operating in several fields (think about insurance, health, fitness...), since it allows them to provide assistance, tailor their services to the user specific needs, and so on. With the rising amount of smart devices capable of gathering data from the wearer, this necessity could easily be satisfied: smart-watches and fitness bands are being refined and they are also becoming more affordable.

On the other hand, there is the need to ask permission to the users to be allowed to exploit their data and they have to be informed about what is being gathered and who is requesting it, especially with the recent the introduction of GDPR (General Data Protection Regulation) in Europe.

TrackMe is willing to satisfy these necessities through ***Data4Help***, a service capable of balancing users' privacy with companies' need of data. This will be done allowing both Individuals and Third-parties to register to the service, so that the latter can perform requests over the former, which can accept or refuse them. Moreover, *TrackMe* wants to launch two more services exploiting ***Data4Help***'s framework:

- ***AutomatedSOS***, a software dedicated to Elderly people that will call an ambulance whenever the health signs of the person (gathered through ***Data4Help***) are below a critical threshold;

- ***Track4Run***, a software for runs organisation, where spectators can follow the participants thanks to the localisation provided by ***Data4Help***.

1.3 Definitions, Acronyms and Abbreviations

In this section the definitions, the acronyms and the abbreviations used throughout the document are explained in detail.

1.3.1 Definitions

- **Application, Software, System**: these terms refer to *Data4Help*, *AutomatedSOS* or *Track4Run*, depending on which of them is being described, in their entirety (design and implementation alike).
- **Critical threshold**: this expression refers to the values that, when trespassed, make *AutomatedSOS* call for an ambulance. (See [D2.2] in the **Appendix A**).
- **Health status:** this expression refers to the status of the Individual inferred from the health signals gathered through smart devices.
- **Maps, Map service:** these terms refer to Google's map service, Google Maps.
- **Multiple, group or aggregated search:** these terms refer to the data requested by Third-parties through *Data4Help* involving a group of Individuals.
- **Smart devices:** the ones taken into consideration are smart-watches and fitness-bands.
- **Subscription:** this term refers to the request of continuously updated data performed by a Third-party. Such data can belong to an Individual or may refer to a group search.
- **User:** this term refers to all possible customers of *TrackMe*, such as Individuals, Third-parties, Elderly people, Organisers, Runners, Visitors (see **section 2.3** of the RASD for further details).
- **Frequency and Granularity:** these terms are related to the subscriptions. The frequency represents the period of time that has to pass between updates, while the granularity represents how the data is aggregated (e.g. average bpm per hour/day/week; number of steps per hour/-day/week). Both of these parameters can be set by the Third-party after they check the box for the subscription.

1.3.2 Acronyms

- **API:** Application Programming Interface.
- **DB:** Database.
- **DMZ:** Demilitarized zone.
- **HTTPS:** HyperText Transfer Protocol over Secure Socket.
- **OS:** Operative System.
- **RASD:** Requirement Analysis and Specification Document.
- **UML:** Unified Modeling Language.

1.3.3 Abbreviations

- **D4H:** *Data4Help*.
- **ASOS:** *AutomatedSOS*.
- **T4R:** *Track4Run*.
- **[D.1.k]:** *Data4Help*'s k-th domain assumption.
- **[D.2.k]:** *AutomatedSOS*' k-th domain assumption.
- **[D.3.k]:** *Track4Run*'s k-th domain assumption.
- **[D.2-3.k]:** *AutomatedSOS*' and *Track4Run*'s k-th domain assumption.
- **[G.1.k]:** *Data4Help*'s k-th goal.
- **[G.2.k]:** *AutomatedSOS*' k-th goal.
- **[G.3.k]:** *Track4Run*'s k-th goal.
- **[R.1.k]:** *Data4Help*'s k-th requirement.
- **[R.2.k]:** *AutomatedSOS*' k-th requirement.
- **[R.3.k]:** *Track4Run*'s k-th requirement.

1.4 Revision History

Version	Comments
1.0	First delivery of DD

Table 1.1: Revision history table

1.5 Reference Documents and Software Tools

1.5.1 Reference Documents

1. Specification document: *Mandatory Project Assignment A.Y. 2018-2019.pdf*
2. Software Engineering 2 course slides
3. Previous mandatory project examples:
   (a) Specification document: *Mandatory Project Assignment A.Y. 2017-2018.pdf*
   (b) *DD to be analyzed.pdf*
4. 1016-2009 - IEEE Standard for Information Technology–Systems Design–Software Design Descriptions

1.5.2 Software Tools

1. *Overleaf*, an online LaTeX editor;
2. *Visio*, a diagramming and vector graphics application by Microsoft;

1.6 Document Structure

The structure of this DD document follows IEEE standard, therefore it is divided into five chapters:

1. Introduction
2. Architectural Design
3. Requirement Traceability
4. Implementation, Integration and Test Plan
5. Effort Spent

The overall aim of this document is to describe the architecture and design of the applications *TrackMe* wants to develop.

In the first chapter a general overview of the purpose of the software-to-be has been provided, together with the description of the context in which it is going to operate. Being an introductory chapter, all the terminology that will be used in the rest of the document has been clearly defined.

In the second chapter the architectural design of the software-to-be is described, highlighting the components it is going to be made of, the way they will be deployed, the interaction among them and the interfaces through which they will communicate. Furthermore, the architectural styles and patterns are listed and justified here.

In the third chapter it is explained how the requirements defined in the RASD map to the design elements defined in the second chapter.

The fourth chapter presents a way in which the implementation of components of the system can be led, along with the integration order of said components. Moreover, a test plan for the application is provided.

The fifth chapter contains the effort spent by each member of the group in order to realise the DD.

The Bibliography is provided after the fifth chapter. All the sources of information exploited are listed there.

The Appendix A in the end of the document contains the goals, the requirements and the domain assumptions as they have been formalised in the RASD.

For a description about the User Interface, refer to the Requirements Analysis and Specification document, **section 3.1.1**.

Chapter 2

Architectural Design

2.1 Overview

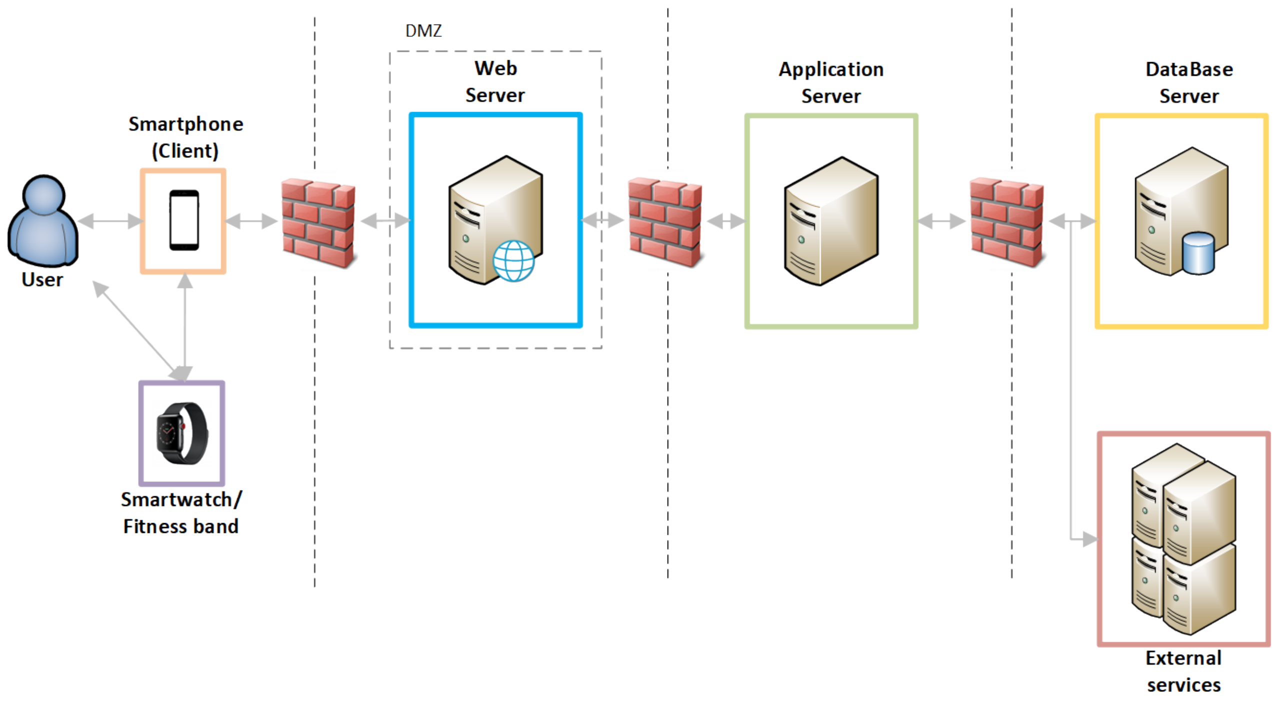


Figure 2.1: Overview of the system high level architecture

Figure 2.1 shows the high level architecture that the three applications will adopt. They will be deployed in a 4-tier client-server architecture, composed by:

- a distributed presentation layer on the Client, that also retrieves information from the smart-watch or fitness band;
- a Web Server for load balancing, failure detection, message filtering and forwarding;
- an Application Server that runs the business logic;
- a Database Server containing the applications' data.

They will also have to communicate with external services (Google Maps and an ambulance service). For further details on the 4-tier client-server architecture, refer to **section 2.7**.

There are three firewalls among the four tiers, so as to guarantee a level of security as high as possible. The one between the Client and the Web Server filters all the requests coming from the former, allowing only a restricted amount of operations.

The Web Server, however, is still considered to be in an untrustworthy zone (DMZ) because it is in close contact with the web and the (possibly malicious) Clients. Therefore, after the request is received and analysed by the Web Server, it is sent to the Application Server through a second firewall, that should guarantee that only well formed request are accepted.

The last firewall is placed between the Application Server and the Database Server and it makes sure of the fact that only certain operations on the data can be performed by specific components (see **section 2.4**). This firewall also filters the messages exchanged with the external services.

All the firewalls work in the opposite direction too, allowing only those operation that the applications are built to offer.

2.2 Component view

In this section there is an analyses of components of the entire system. In particular, each subsection is dedicated to one application and is structured as:

- UML Component Diagram;
- Detailed description of each component.

Descriptions are focused on component characteristics; the interaction between components, instead, is provided in detail in the **section 2.4**.

2.2.1 Data4Help

In the **picture 2.2** there is the UML Component Diagram of Data4Help system. The interaction between components allows D4H to reach its goals and meet its requirements. Each component and its features are better described below.

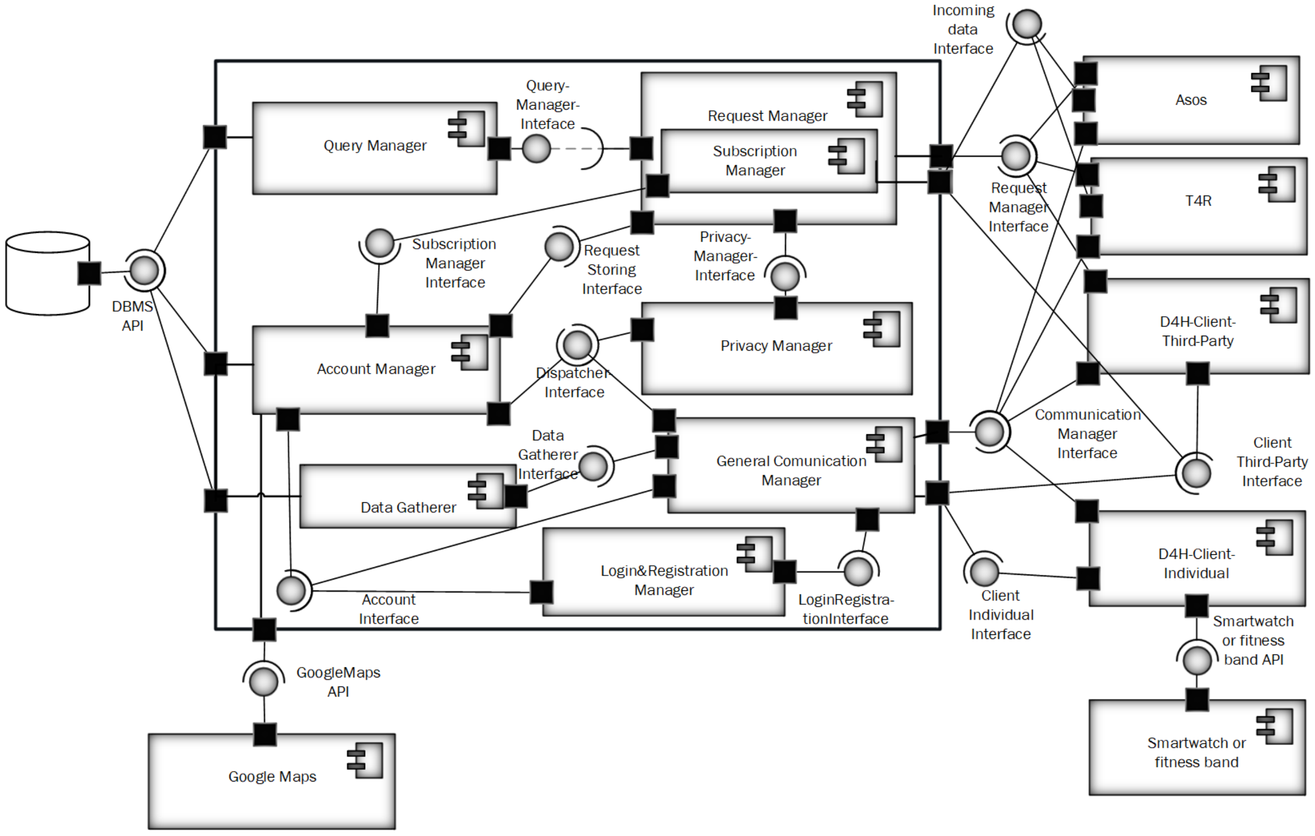

Incoming data Interface
Query Manager
Query-Manager-Inteface
Request Manager
Subscription Manager
Asos
T4R
Request Manager Interface
DBMS API
Subscription Manager Interface
Request Storing Interface
Privacy-Manager-Interface
D4H-Client-Third-Party
Account Manager
Dispatcher Interface
Privacy Manager
Data Gatherer Interface
General Comunication Manager
Communication Manager Interface
Client Third-Party Interface
Data Gatherer
Login&Registration Manager
Account Interface
LoginRegistra-tionInterface
Client Individual Interface
D4H-Client-Individual
Smartwatch or fitness band API
GoogleMaps API
Google Maps
Smartwatch or fitness band

Figure 2.2: UML Component Diagram of Data4Help system

Data4Help components are:

- **Login&registration Manager**: through *Account Manager* component, it permits unregistered users to create a new account (either Individual or Third-party) and to already registered users to access their personal profile.
- **Account Manager**: this component allows D4H to manage user accounts; it has to store all the data related to user accounts in the DBMS and show stored data in case of personal data requests. Moreover it needs to communicate with the *Subscription Manager* to store subscription data.
- **Privacy Manager**: it is the component in charge of managing user privacy settings, in particular it asks Individuals for consent during an Individual request (and eventual subscription).
- **Request Manager**: it receives all the request. In case of request without subscription (either Individual or group request) it, itself, handles the

request (by querying the db with *Query manager*); if it's a request with subscription, instead, this component forwards the request to its sub-component *Subscription Manager.*

- **Subscription Manager**: it is the component in charge of managing requests with subscription; at the beginning a *Supervisor* object is created; it instantiates a *Worker* at each received request. Periodically the supervisor pings workers and, if someone does not answer, through the *Account Manager* it finds out which subscriptions are lost and creates new workers to substitute them.
- **Query Manager**: when a request from *Request Manager* arrives, it performs a QUERY to the DBMS.
- **General Communication Manager**: this component acts as a message "switch" between internal components and external users and viceversa.
- **DBMS**: this is the component for the Database, here D4H stores account data, user gathered data and keeps track of all kinds of requests (single and multiple user, with or without subscription).
- **D4H-Client-Third-party**: this is the Front end for the Third-party client; this component interacts with *General Communication Manager* and *Request Manager.*
- **D4H-Client-Individual**: this is the Front end for the Individual client; this component interacts with the *General Communication Manager.*
- **Smartwatch** or **Fitness band**: this component represents the external device that D4H system exploits in order to gather Individual data.
- **Google Maps**: this is an external components that D4H uses to map user position on a map.

2.2.2 AutomatedSOS

The **picture 2.3** is a representation of the UML Component Diagram of AutomatedSOS system. Each component has a specific role in the system and it is better described below.

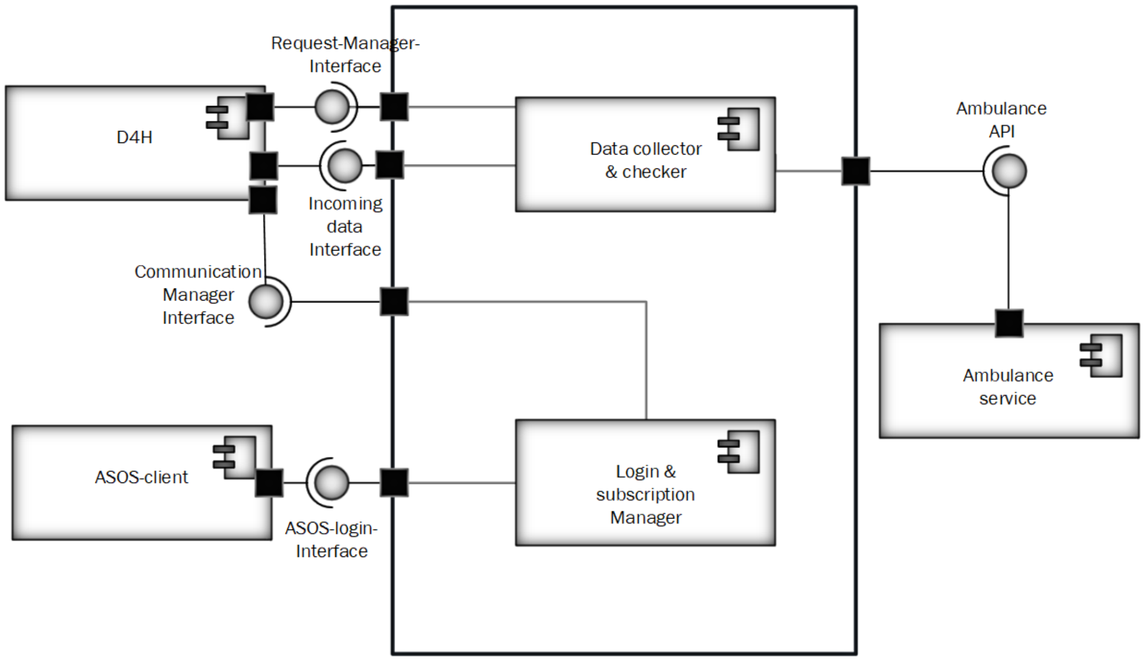


Figure 2.3: UML Component Diagram of AutomatedSOS system

AutomatedSOS components are:

- **Login&Registration Manager**: this component allows to sign in using D4H credential, checked through D4H as external component, and to perform a subscription request.
- **Data collector & Checker**: after a subscription, AutomatedSOS system is "subscribed in D4H" to the user data and this component receives periodically user gathered data, checks their value and, in case of emergency, dispatches an ambulance, called through the *Ambulance service* component.
- **ASOS-Client**: this is the Front end for the ASOS client; it only interacts with the *Login&Registration Manager* in order to login and perform a subscription.
- **Ambulance service**: This is the external component representing the Ambulance service that ASOS exploits in order to offer its SOS service.
- **D4H**: this is the external component representing the entire D4H system; ASOS interacts with it using *Communication Manager Interface* to receive gathered data and *Request Manager Interface* to perform subscriptions.

2.2.3 Track4Run

The UML Component Diagram for Track4Run is shown in the **figure 2.4**; the roles that each component plays is defined below.

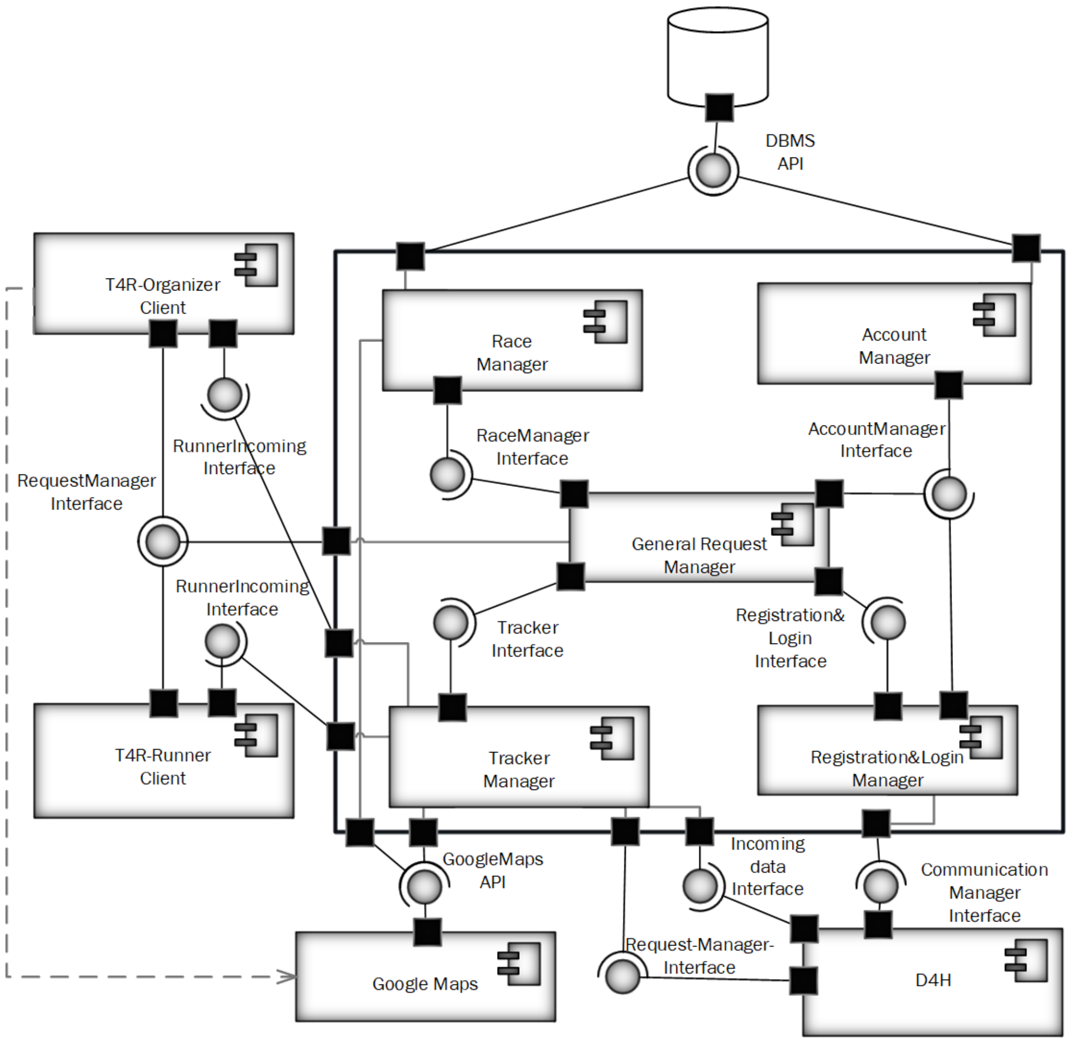


Figure 2.4: UML Component Diagram of Track4Run system

Track4Run components are:

- **Registration&Login Manager**: through the *Account Manager* component, it permits unregistered users to create a new Organiser account and to already registered users to sign in as Organisers or as Runners using their D4H credentials, checked through D4H as external component.

- **Account Manager**: this component allows T4R to manage user accounts; it has to store all the data related to user accounts in the DBMS, all the data related to races and to show stored data in case of personal data requests.
- **General Request Manager**: this component acts as a message "switch" between internal components and external users and viceversa.
- **Race Manager**: this is the component in charge of races' management. It creates new runs when a creation request arrives and stores all the data related to races in the DBMS;
- **Tracker Manager**: this component is dedicated to participant tracking; it periodically receives the runners' positions from Data4Help through *Incoming Data Interface*, sends them to *Google Maps*'s external component and forwards computed maps to users (both registered and visitors) that have made requests.
- **T4R-Organizer Client**: this is the Front end for the T4R organiser client; it interacts with the *Race Manager* component to create a new race and with the *Account Manager* component to check the races associated with the Organiser.
- **T4R-Runner Client**: This is the Front end for the T4R participant client; it can make enrolling requests trough the *Race Manager* or ask for its personal data to the *Account Manager* component.
- **D4H**: this is the external component representing the entire D4H system; T4H needs to interact with this component in order to verify Participants credentials and receive the runners' position during a race.
- **Google Maps**: this is an external component that T4R uses to locate runners' positions on a map.
- **DBMS**: this is the component for the Database, here T4R stores users and races data.

2.3 Deployment view

This section shows the configuration of the runtime processing nodes and the components that live on them. In the **figure 2.5** there is a representation of the Deployment diagram for the entire system. The firewalls and DMZ are omitted in the diagram for readability, but they are treated in the Overview section.

In the package *Client* there are five nodes, one for each client app; since the apps are multi-platform the nodes can either be on iOS or Android and all of them communicate with the *Web Server* node through the TCP/IP protocol. Client nodes can be:

1. **D4HIndividual** (it is the only client that communicates with a *Smartwatch or Fitness band* through a Bluetooth connection)
2. **D4HThird-party**
3. **ASOSUser**
4. **T4RRunner**
5. **T4ROrganizer**

In the *Server* package there are four different physical nodes; three of them are one per application: D4H, ASOS and T4R. A main role is played by the fourth node: the **Web Server**. The *Web Server* communicates, on the front-end, with client applications and, on the back-end, with application servers. It provides the clients with the required information queried from a specific application server and encapsulated in a structured format such as JSON or XML; by masking the presence of back-end servers it provides an effective security measure. The *Web Server* is also in charge of monitoring, as a Watch-dog, all Application Server components and replacing them in case of failures. Moreover, it also acts as a load balancer, distributing the computation among the various nodes. The *Web Server* and the three application nodes communicate among each other through the TCP/IP protocol.

The *Database Server* package contains two physical databases; one is dedicated to the Data4Help system, the other one stores Track4Run data. Only D4H has the access to the D4H database and only T4R has the access to the T4R database. Both of them communicate with their databases through the TCP/IP protocol.

The two remaining nodes represent external services that apps make use of, in order to reach their goals:

- **Ambulance service**: it communicates only with the ASOS nodes, through TCP/IP protocol.
- **Google Maps service**: It is connected with D4H and T4R nodes through TCP/IP protocol.

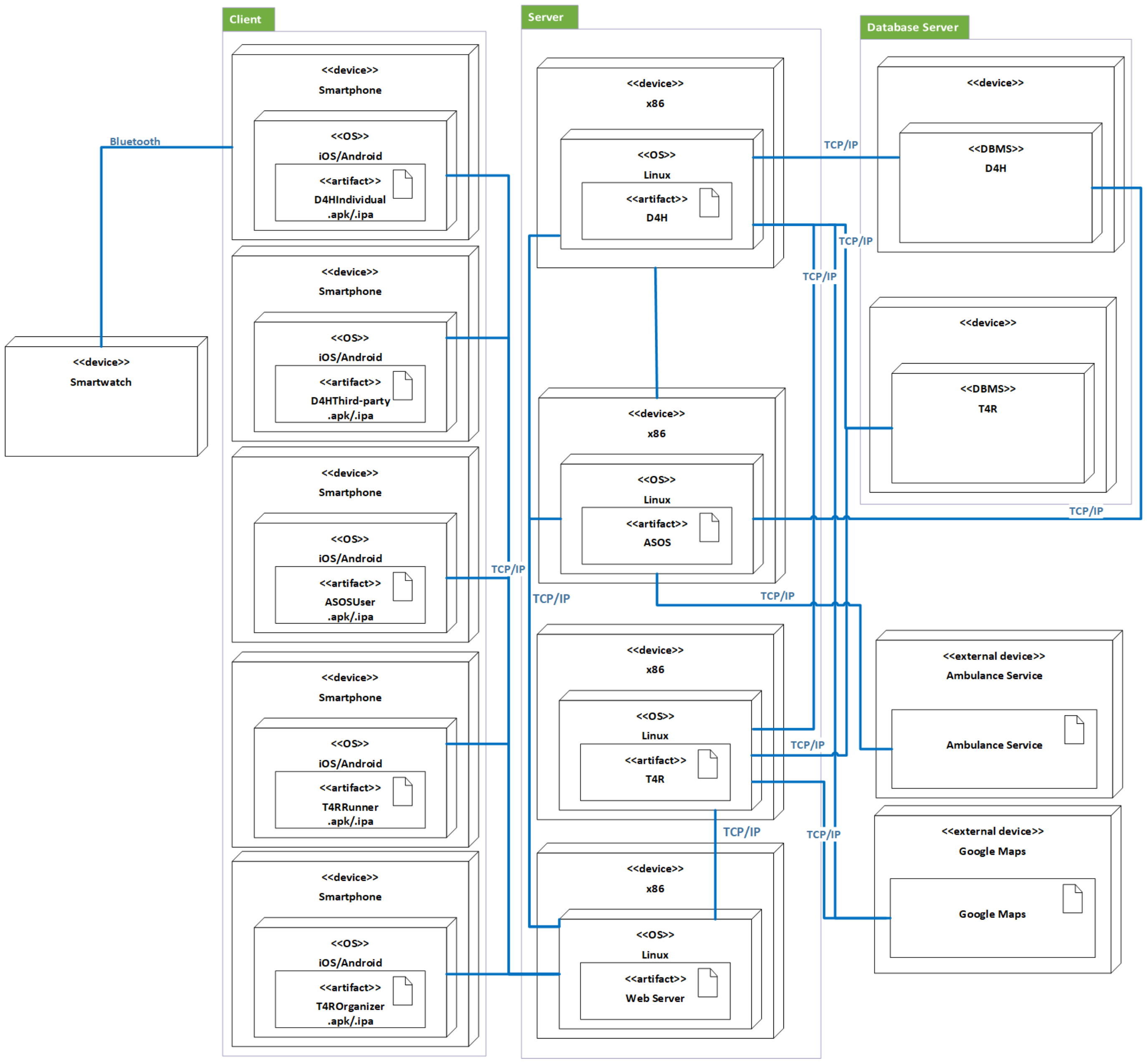


Figure 2.5: Deployment Diagram for the entire system

2.4 Runtime view

In this section will be analysed the runtime view of the system. It shows, more in detail, how the system works in every use case, presenting the interaction between the internal components, discussed above in the section 2.2.

2.4.1 Data4Help

Concerning Data4Help, the main functionalities taken into account are: the registration procedure, the log in procedure, the subscription for a request, the unsubscription and both single user request and multiple users request.

Registration and Login

In the diagram below is presented the workflow of the application during the registration and the log in. This diagram shows the case of registration and the log in of an individual to the system. However, the client type is not specified because the behaviour is the same also in case of Third Party. The main component is the Account Manager, that inserts in the database the data of new users and, communicating with the Login and Registration Manager, checks the credential after a log in request.

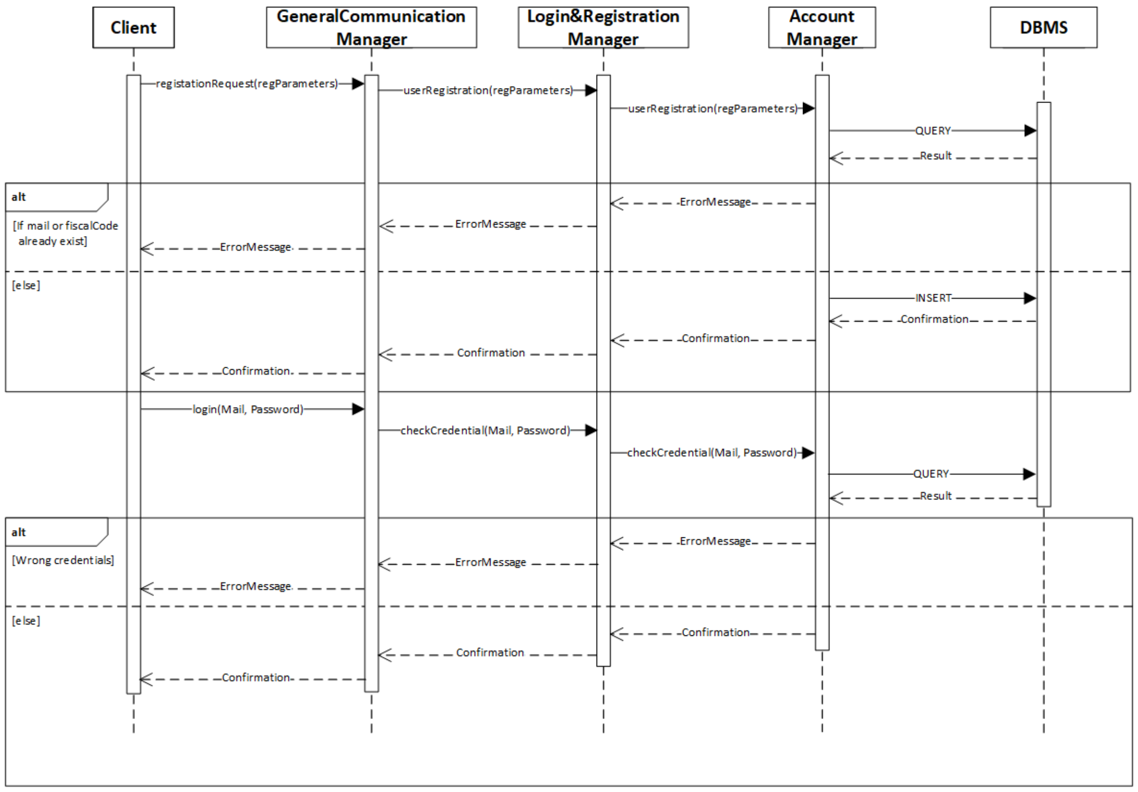


Figure 2.6: Registration and Login sequence diagram

Single User data request

This diagram analyses the single request made by a Third-party to a specific registered Individual. The request is taken by the Request Manager, that verifies the correctness of the demanded fiscal code. Then, in case of positive match, the responsibility passes to the Privacy Manager that, through the General Communication Manager, informs the user and collects his/her consents. After this procedure the data are sent to the requester and the Account Manager saves the request's details in the database, in order to make the data available to the Third-party also at a later time.

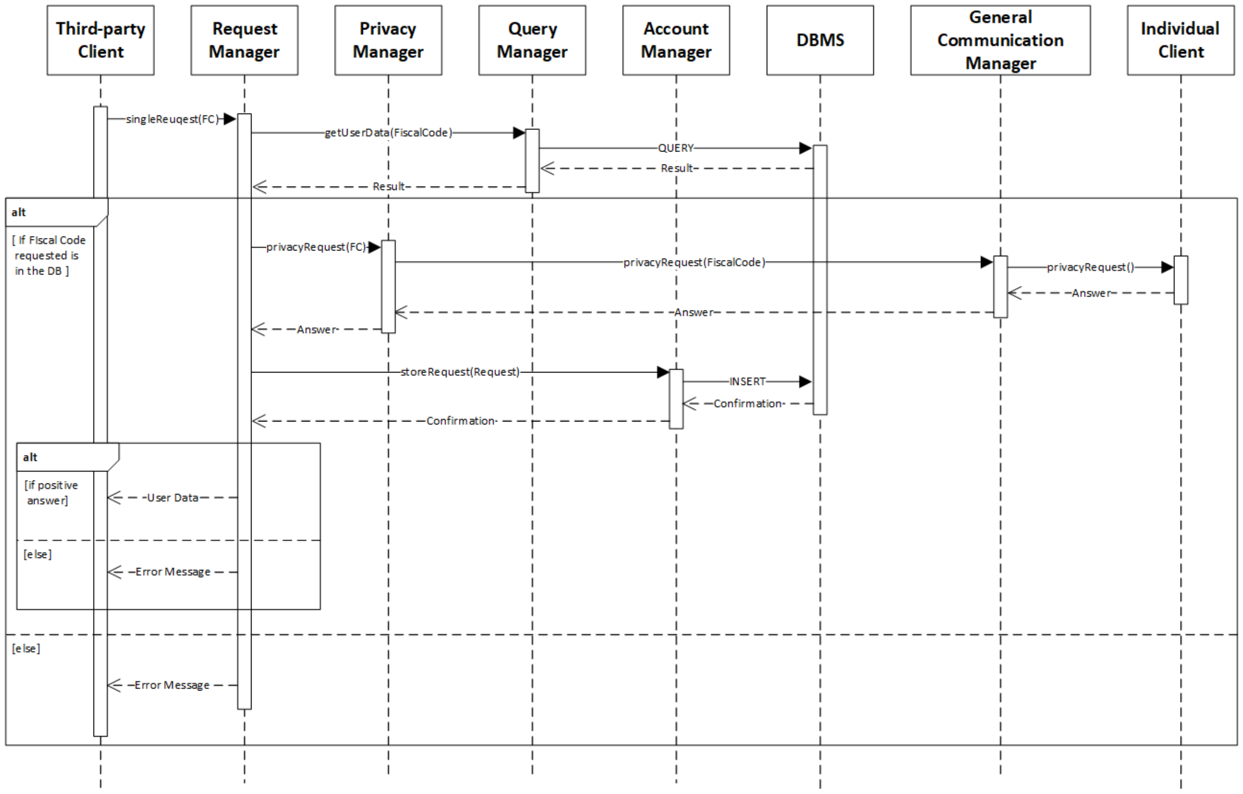


Figure 2.7: Single user request sequence diagram

Aggregated data request

For an aggregated data request the procedure is very similar to the previous one, indeed there are the same actors, except for the Privacy Manager, and the roles taken by them are mostly the same. However, as already discussed in the RASD, the system can not send the data that match with the filters received, if they belong to a little amount of people. Therefore, the Request Manager, before sending them, has to verify if the corresponding number of people is greater than 1000, then, in case of positive answer, it can send them to the Third-party.

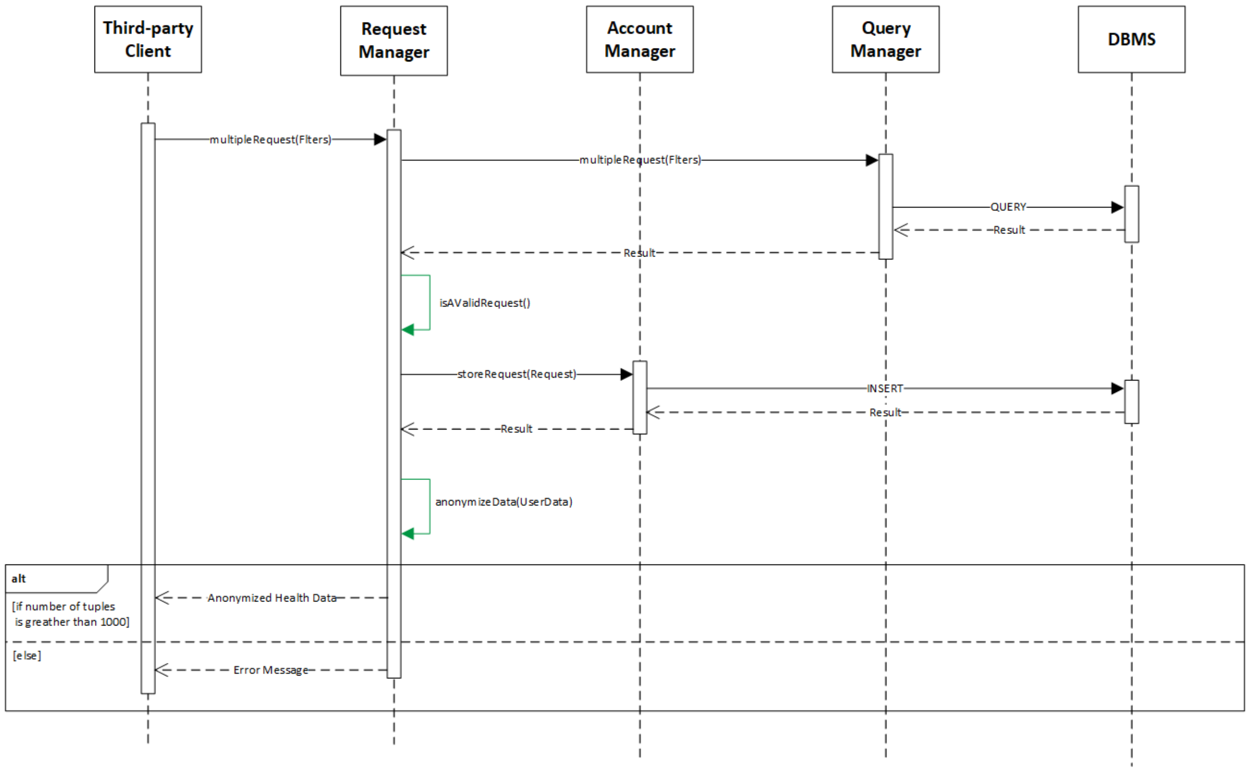
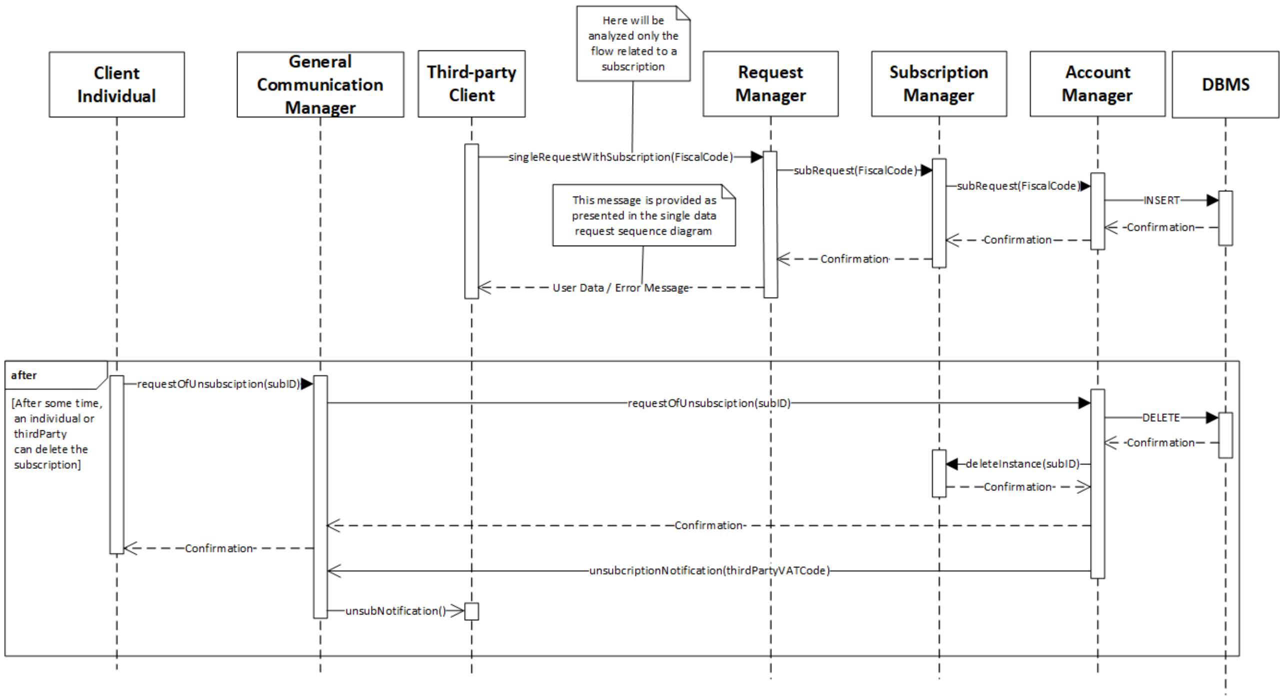


Figure 2.8: Aggregated data request sequence diagram

Single User Data Subscription and Unsubscription

The following diagram shows the behaviour of the system in case of single request with subscription and the consecutive unsubscription to it. In the first part is analysed how the system reacts in case of subscription request. The Request Manager passes the control to a subcomponent, the Subscription Manager, that generates a new instance to manage the subscription in its entire life. Then the flow is the same of a normal single request, the first data collected are sent immediately to the Third-party, with the only difference that, when the Account Manager stores the request details, it includes also the information about the subscription. On the other hand, the part in the bottom describes what happened when, after some time, the user wants to end the subscription. The Account Manager receives the request, it deletes the subscription in the database and informs the Subscription Manager, that in turn deletes the instance related to that subscription. Furthermore, it notifies with a confirmation the user and sends a notification to the Third-party connected to it. The same happens also in case of opposite situation, when a Third-party wants to end a subscription. In that case, the confirmation is sent to it and the notification to the corresponding Individual.



Client Individual
General Communication Manager
Third-party Client
Here will be analyzed only the flow related to a subscription
Request Manager
Subscription Manager
Account Manager
DBMS
singleRequestWithSubscription(FiscalCode)
subRequest(FiscalCode)
subRequest(FiscalCode)
INSERT
Confirmation
This message is provided as presented in the single data request sequence diagram
Confirmation
Confirmation
User Data / Error Message
after
[After some time, an individual or thirdParty can delete the subscription]
requestOfUnsubsciption(subID)
requestOfUnsubsciption(subID)
DELETE
Confirmation
deleteInstance(subID)
Confirmation
Confirmation
Confirmation
unsubcriptionNotification(thirdPartyVATCode)
unsubNotification()

Figure 2.9: Subscription and Unsubscription request sequence diagram

Group search unsubscription

In case of unsubscription for a group search, the system manages the request in an easier way compered to the single request unsubscription. Since no user is directly connected, the Account Manager just deletes the information on the database and informs the Subscription Manager, that behaves in the same way respect to the single request unsubscription. At the end the confirmation is sent only to the Third-party that made the request.

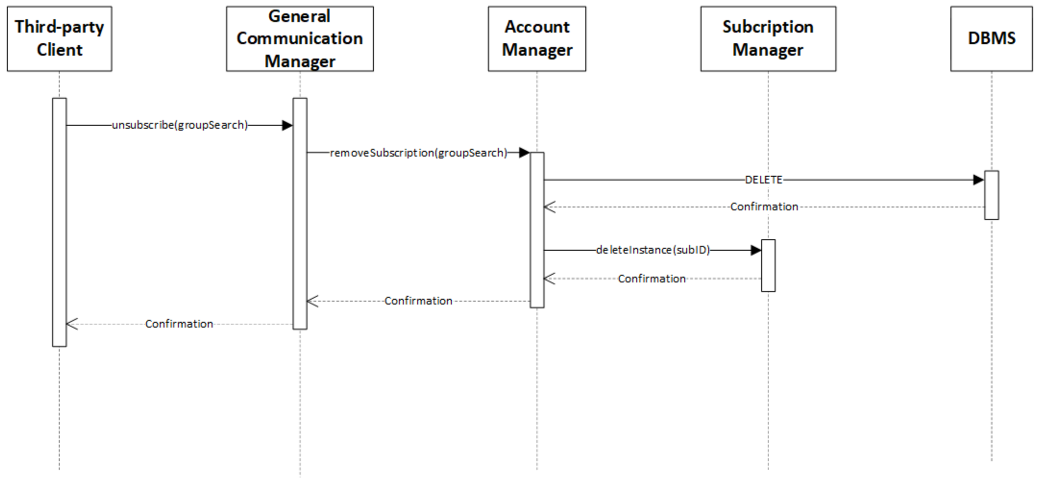


Figure 2.10: Group search unsubscription sequence diagram

Data dispatching during subscription

The next diagram displays how the Subscription Manager carries out its work during the subscription. Until the end of the subscription, it periodically extracts the data from the database thanks to the Query Manager and it sends them to the Third-party subscribed. The only important thing to highlight is that with Subscription Manager in the diagram, it is intended the instance correlated to a specific subscription, because there isn't a component who manages all the subscription but, as earlier discussed, each subscription is managed by a different instance. In the diagram is presented the case of single request subscription, but the same takes place also for a group request subscription.

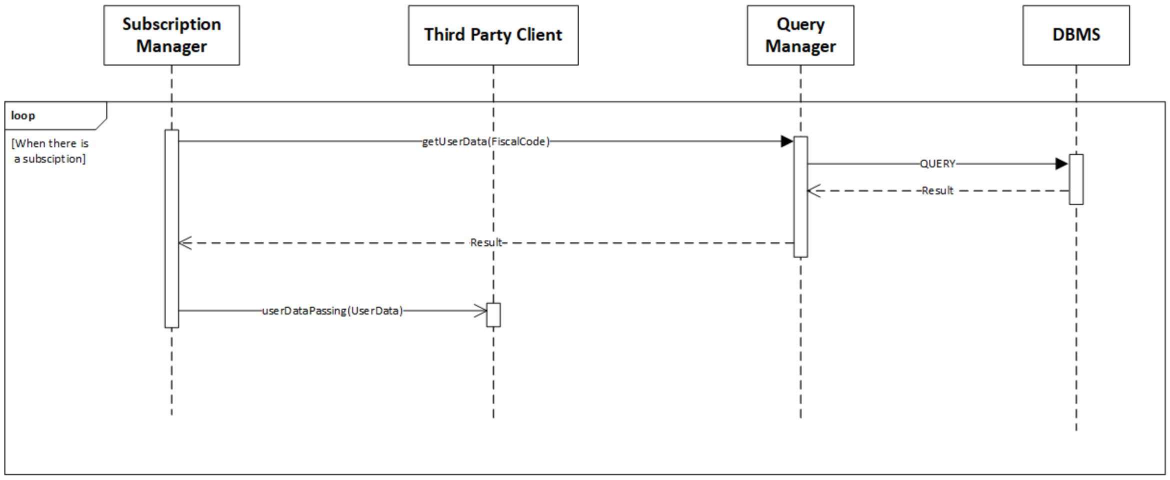


Figure 2.11: Data dispatching during subscription sequence diagram

Data Collection and Visualisation

The diagram below describes how the user data arrives to the system. Essentially the Individual Client collects the data from the smart-watches and sends them to the Data Gatherer that stores them in the database. In addition an Individual can have his/her data displayed by sending a request to the Account Manager that returns the requested data.

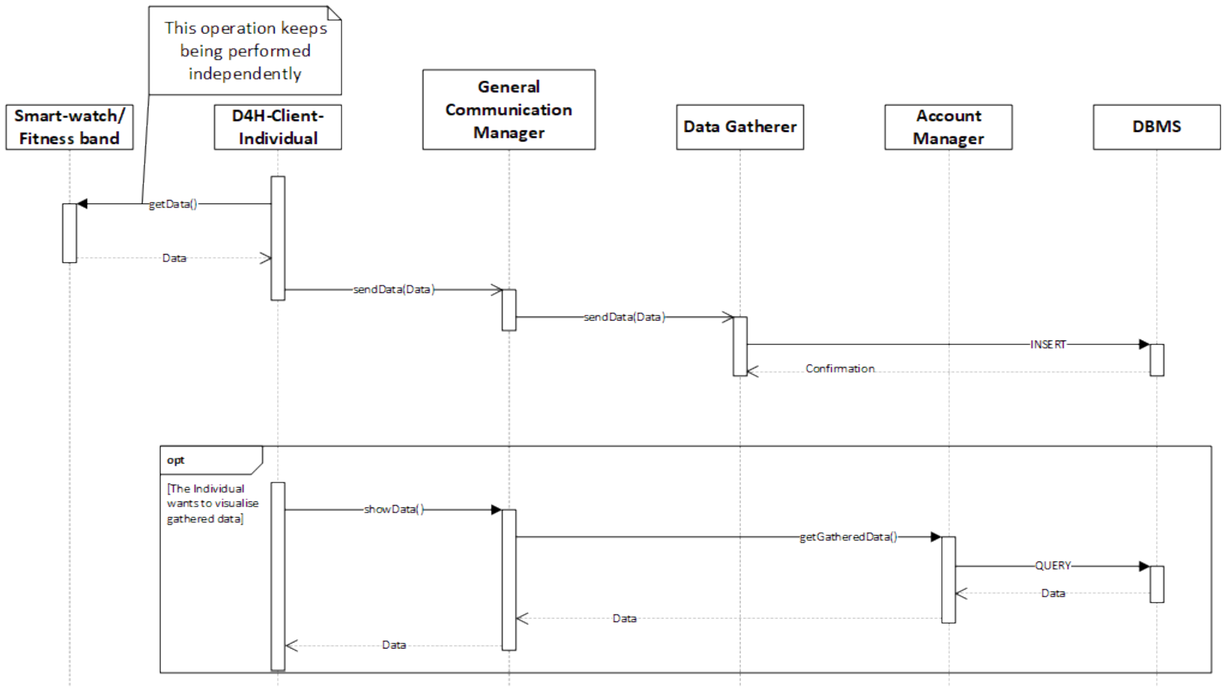


Figure 2.12: Data collection and visualisation sequence diagram

Third Party Request History

Thanks to the system a Third-party can visualise its old requests. These kind of requests are taken by the Account Manager that provides to the Third-party all the history of the occurred requests. Afterwards, if demanded, it provides also the details of each specific request, if the past request was accepted.

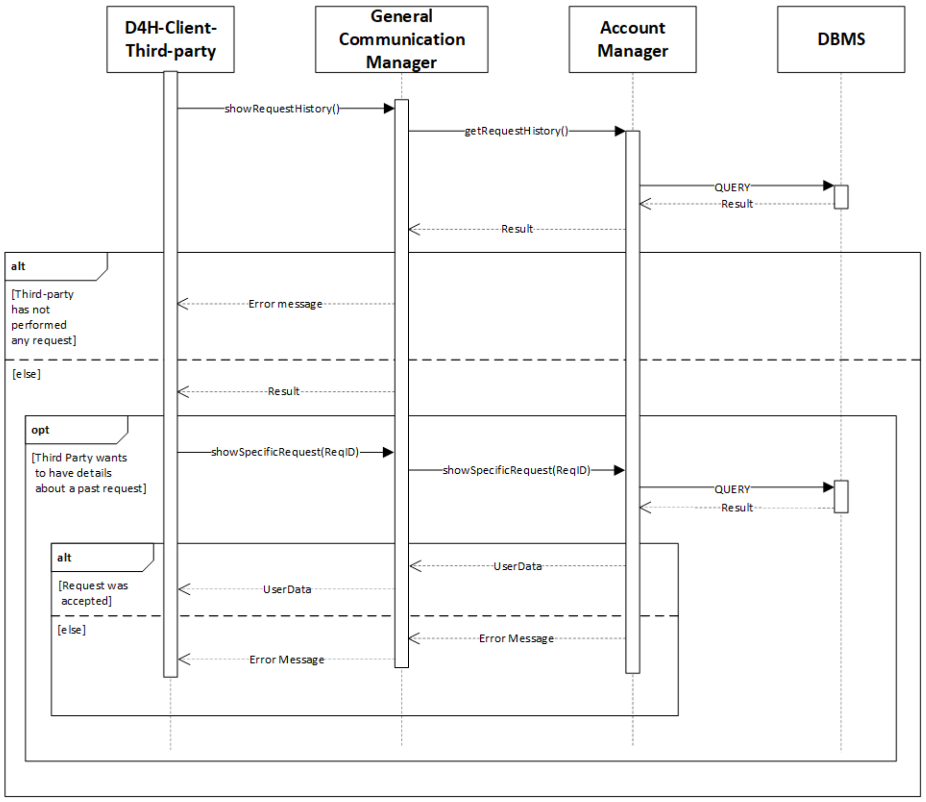


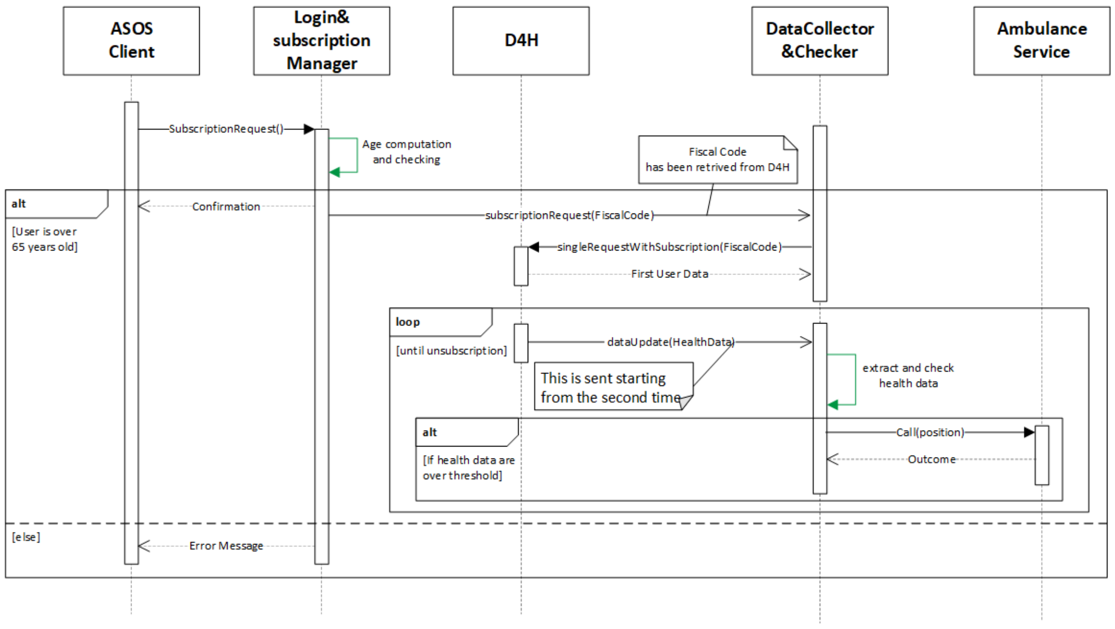
Figure 2.13: Third Party request history sequence diagram

2.4.2 AutomatedSOS

Concerning AutomatedSOS the only functionality highlighted is the ambulance dispatching. AutomatedSOS interacts with Data4Help, collecting the users data needed to monitor the elderly people registered to it and interacting with the ambulance service in case of emergency.

Subscription and Ambulance Dispatching

First of all, an individual, that is already logged in Data4Help, needs to subscribe him/herself to the service. The system checks if his/her age matches with the bound imposed (65 years old), and then, it procedes to the subscription as follow. It makes a single request with subscription to D4H as a normal Third-party, the Individual receives the request with a notification coming from ASOS. After the acceptance, D4H starts to send data periodically to the Data Collector and Checker that analyses them and, in case of health parameters out of the thresholds set, calls the ambulance service, forwarding the position of the interested Elderly person.



ASOS Client
Login& subscription Manager
D4H
DataCollector &Checker
Ambulance Service
SubscriptionRequest()
Age computation and checking
Fiscal Code has been retrived from D4H
alt
[User is over 65 years old]
Confirmation
subscriptionRequest(FiscalCode)
singleRequestWithSubscription(FiscalCode)
First User Data
loop
[until unsubscription]
dataUpdate(HealthData)
This is sent starting from the second time
extract and check health data
alt
[If health data are over threshold]
Call(position)
Outcome
[else]
Error Message

Figure 2.14: Ambulance Dispatching sequence diagram

2.4.3 Track4Run

Concerning T4R the main functionalities touched are: The run organisation, the participants enrolment, the runners tracking and the ranking history requests. Some of the others features have been already explained in the previous sections, for example the Organiser registration and log in is very similar to the log in and registration for a Third-party, with the only difference that the components are the ones of T4R.

Run Organisation

In the following diagram is explained how the system works to allow the Organiser to plan a run. After the log in, the Organiser can make a new race request, handled by the Race Manager. It checks if there are other races at the same location at the same time, and after this verification, it contacts Google Maps. The latter is opened on the Organiser's device, in order to allow him/her to create the path desired, by adding manually on the Google Maps application each stop. Afterwards, when the path is defined the Organiser, through the T4R application, sends a notification to the Race Manager that stores the race details inside the database.

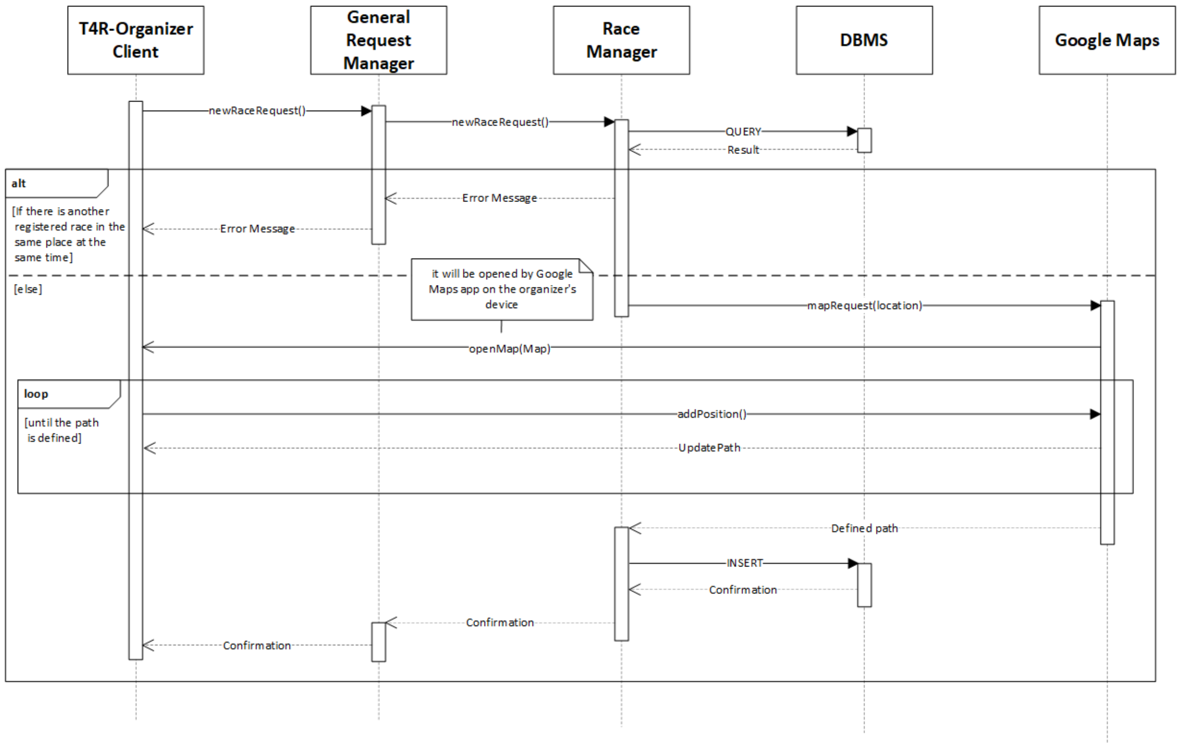


Figure 2.15: Run organization sequence diagram

Participant Enrolment

After the race registration, any Data4Help Individual user can enrol to it. The Race Manager receives the request from the General Request Manager, and checks if the Runner is enrolled to an other race at the same time. After this control, it verifies if the run already reached the maximum number of participants and, if it doesn't exceed, it adds the enrolling Participant. In order to allow the Visitor to track the runner in a specific race, as will be explained in the next diagram, the General Request Manager, during the check-in of the runners, before the beginning of the race, sends a notification to the Tracker Manager that requests a single user subscription to Data4Help. The runner receives a request of subscription from T4R in his/her D4H application and, by accepting it, he/she confirms the participation to the race and the Tracker Manager starts to receive the data related to that Participant.

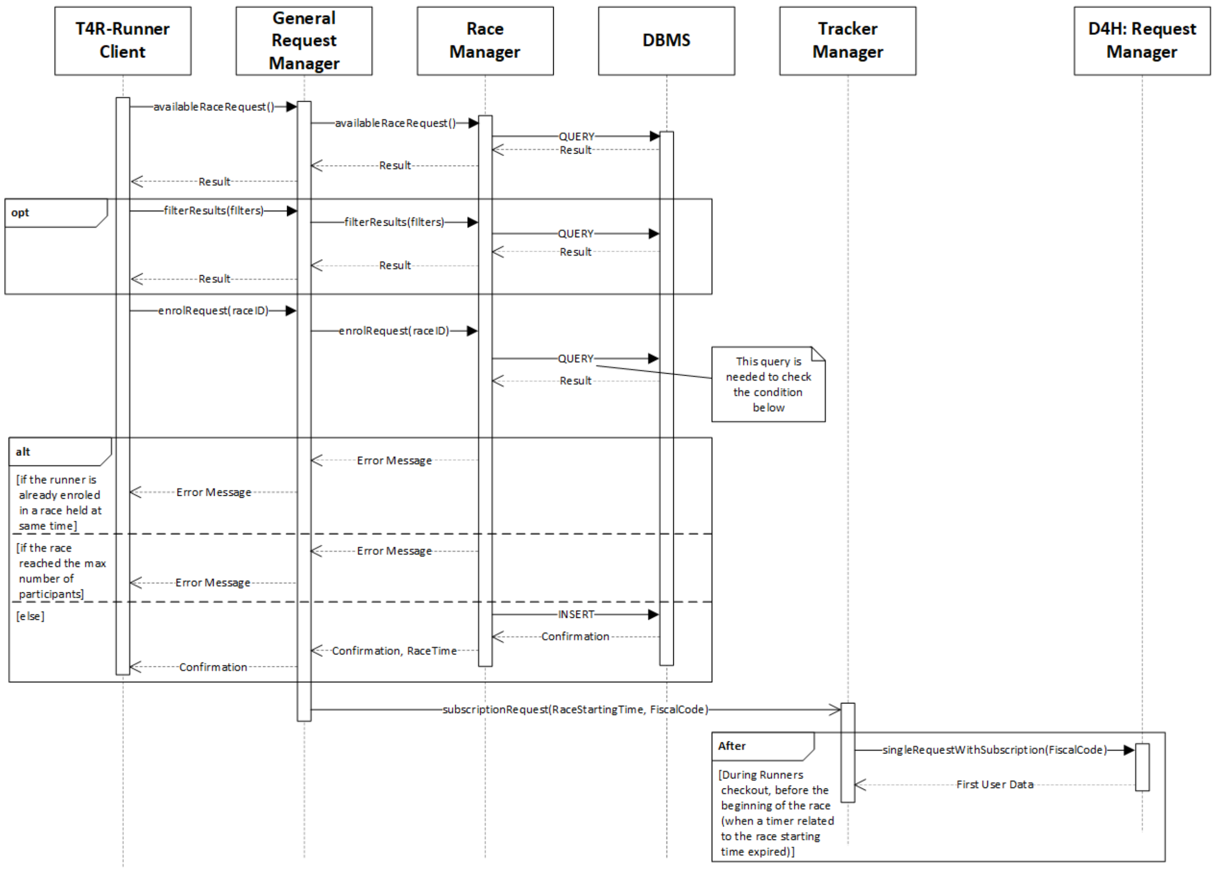


Figure 2.16: Participant Enrolment sequence diagram

Participant Tracking

As described above, the spectator of a run, thanks to T4R, can track the participant during all the race. They just have to download any application, Runner or Organiser edition, and without logging in they can enter in the tracking section. Therefore, a Visitor can request the tracking for a specific race, this request passes through the General Request Manager and arrives to the Race Manager, that extracts all the Participants fiscal codes from the database, and sends them to the Tracker Manager. It filters the data received from D4H and, by sending the position to the Map service, it gets a map with the position of all the runners. This map is forwarded to the client periodically, in order to give the possibility to the Visitor to track all the participant at any time.

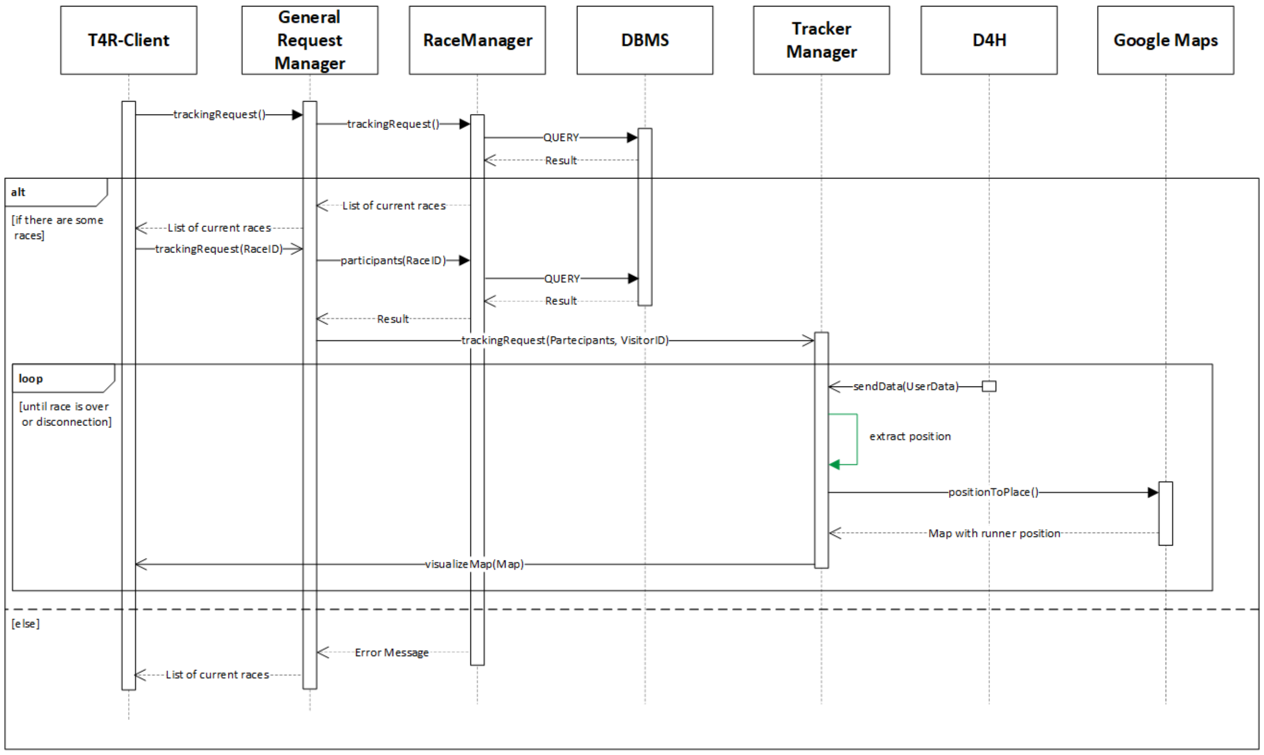


Figure 2.17: Participant tracking sequence diagram

Ranking History

The last diagram shows the possibility of any client, both Organiser and Runner, to get the ranking of the past runs. The request is processed by the Account Manager that extracts the information from the database and returns them to the client.

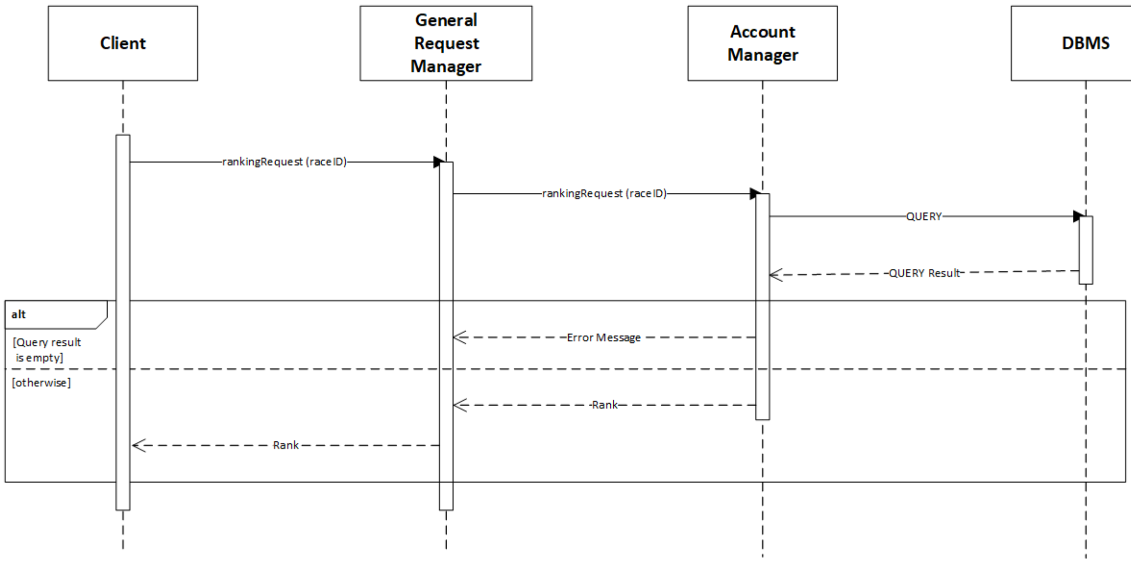


Figure 2.18: Ranking history sequence diagram

2.5 Component interfaces

This section will discuss in detail the various interfaces of the components presented in the **section 2.2**. In particular for each of them, it will display all the signatures of the methods with a particular focus on inputs and return parameters, in a Java-like style.

To better understand the code, the boolean return parameters are used when the method needs to send a confirmation or an error notification to the caller. While, the other types of parameters are referred to classes that will be developed during the implementation phase.

Moreover, these methods are the same presented in the sequence diagrams.

2.5.1 Data4Help

Concerning Data4Help the interfaces provided by the components are the following.

This interface allows the Request Manager to perform a look up on the database.

```
public interface QueryManagerInterface{
        public UserData getUserData(String
            fiscalCode);
        public List<UserData>
            multipleRequest(List<Filters> filters);
}
```

This interface allows the Account Manager to communicate with the Subscription Manager in case of unsubscription, to notify it that has to delete the instance related to that subscription.

```
public interface SubsciptionManagerInterface{
        public boolean deleteInstance(int subID);
}
```

This interface allows the Request Manager to communicate with the Account Manager, that stores the request in the database.

```
public interface RequestStoringInterface{
        public boolean storeRequest(Request request);
}
```

This interface allows the Request Manager to interact with the Privacy Manager in case of single user request.

```
public interface PrivacyManagerInterface{
        public boolean privacyRequest(String
           fiscalCode);
}
```

This interface allows to the inside components to communicate with the outside through the General Communication Manager.

```
public interface DispatcherInterface{
        public boolean privacyRequest(String
           fiscalCode);
        public void unsubscriptionNotification(String
           thirdPartyVATCode);
}
```

This interface allows the General Communication Manager to forward the user data, collected from the client, to the Data Gatherer that stores them in the database.

```
public interface DataGathererInterface{
        public void sendData(UserData data);
}
```

This interface allows the General Communication Manager to forward the log in and registration request coming from both client types.

```
public interface LoginRegistratioInterface{
        public boolean
           userRegistration(RegistrationParameters
           regParameters);
        public boolean checkCredential(String mail,
           String password);
}
```

This interface allows the components interested in managing accounts to communicate with the Account Manager, that can look up for data or modify the database as requested.

```
public interface AccountInterface{
        public boolean
           userRegistration(RegistrationParameters
           regParameters);
        public boolean checkCredential(String mail,
           String password);
        public boolean subscriptionRequest(String
           fiscalCode);
        public boolean requestOfUnsubscription(int
           subID);
        public boolean removeSubscription(int
           groupSearchID);
        public UserData getGatheredData(String
           fiscalCode);
        public RequestHistory
           getRequestHistory(String
           thirdPartyVATCode);
        public UserData showSpecificRequest(int
           reqID);
}
```

This interface allows the Third Party Client to send requests to the system.

```
public interface RequestManagerInterface{
        public UserData singleRequest(String
           fiscalCode);
        public List<AnonymizedHealtData>
           multipleRequest(List<Filters> filters);
        public UserData
           singleRequestWithSubscription(String
           fiscalCode);
        public List<AnonymizedHealtData>
           multipleRequestWithSubscription(List<Filters>
           filters);
}
```

This interface allows Clients to send several messages (except for request messages) to the system.

```
public interface CommunicationManagerInterface{
        public boolean
           registrationRequest(RegistrationParameters
           regParameters);
        public boolean login(String mail, String
           password);
        public boolean requestOfUnsubscription(int
            subID);
        public boolean unsubcribe(int groupSearchID);
        public void sendData(UserData data);
        public UserData showData();
        public RequestHistory showRequestHistory();
        public UserData showSpecificRequest(int
            reqID);
}
```

This interface allows the Privacy Manager to send the notification that asks for the Individual (corresponding to the fiscal code requested by the Third-party) permission to use his/her data.

```
public interface ClientIndividualInterface{
        public boolean privacyRequest();
}
```

This interface allows the Third-party to receive unsubscription notifications and user data, in case of subscription, from the Subscription Manager.

```
public interface ClientThirdPartyInterface{
        public void unsubscriptionNotification();
        public void userDataPassing(UserData
            userData);
}
```

2.5.2 AutomatedSOS

Concerning AutomatedSOS the interfaces provided by the components are the following.

This interface permits Data4Help to send data related to a subscribed user, to the Data Collector And Checker that verifies the health parameters and in case of emergency, calls the ambulance.

```
public interface IncomingDataInterface{
        public void subscriptionRequest(String
            fiscalCode);
        public void dataUpdate(UserData healthData);
}
```

This interface allows a user to log into the system and subscribe to it, by communicating with the Login and Subscription Manager.

```
public interface ASOSLoginInterface{
        public boolean login(String mail, String
            password);
        public boolean subscriptionRequest();
}
```

2.5.3 Track4Run

Concerning Track4Run the interfaces provided by the components are the following.

This interface allows the General Request Manager to forward requests concerning the race, that come from clients, to the Race Manager.

```
public interface RaceManagerInterface{
        public boolean newRaceRequest();
        public List<Race> filterResults(List<Filter>
           filters);
        public List<Race> availableRaceRequest();
        public int enrolRequest(int raceID);
        public List<Race> trackingRequest();
        public List<String> participants(int raceID);
}
```

This interface permits to communicate with the Account Manager in case of ranking request or log in.

```
public interface AccountManagerInterface{
        public Rank rankingRequest(int raceID);
        public boolean
           userRegistration(RegistrationParameters
           regParameters);
        public boolean checkCredential(String mail,
           String password);
}
```

This interface permits clients to communicate with the internal components of the system through the General Request Manager.

```
public interface RequestManagerInterface{
        public boolean
           registrationRequest(RegistrationParameters
           regParameters);
        public boolean login(String mail, String
           password);
        public boolean newRaceRequest();
        public List<Race> filterResults(List<Filter>
           filters);
        public List<Race> availableRaceRequest();
        public int enrolRequest(int raceID);
        public List<Race> trackingRequest();
        public void trackingRequest(int raceID);
        public Rank rankingRequest(int raceID);
}
```

This interface allows the visitor, logged in to any client, to receive the map with runners' positions from the Map Service.

```
public interface RunnerIncomingInterface{
        public void visualizeMap(Map map);
}
```

This interface permits the General Request Manager to forward the tracking request and the subscription request to the Tracker Manager. Note that the raceStartingTime parameter is used in order to send an Individual request through D4H shortly before the race, so as to perform a subscription to the Participants' D4H accounts.

```
public interface TrackerInterface{
        public void subscriptionRequest(int
           raceStartingTime, String fiscalCode);
        public void trackingRequest(List<String>
           participantsFiscalCodes, int visitorID);
}
```

This interface allows the General Communication Manager to communicate with the Registration And Login Manager in case of log in and registration requested.

```
public interface RegistrationAndLoginInterface{
        public boolean
           userRegistration(RegistrationParameters
           regParameters);
        public boolean checkCredential(String mail,
           String password);
}
```

This interface lets the Tracker Manager receive the data of the runners from Data4Help.

```
public interface IncomingDataInterface{
        public void sendData(UserData userData);
}
```

2.6 Database View

This section contains definitions of data and information structures that have to be implemented in the database.

2.6.1 Data4Help

In **picture 2.19** there is the ER diagram for the D4H database. There are four main entities:

- **User**: used to store D4H user information, it is inherited by Individual and Third-party entities.

- **Health data**: used to store data gathered through the smart-watch, it is associated to exactly one individual and it is characterised by time stamp (date and time), position (latitude and longitude) and health data (BPM, Daily steps).
- **Subscription**: used to store requests with Subscription; it's associated to the Third-party that made the request, to the Individual that the Third-party is asking for and to the particular health data record.
- **Request**: used to store a request for health data and the Third-Party who made the request; can be either Single (requesting a single health data record) or Multiple (requesting multiple health data records).

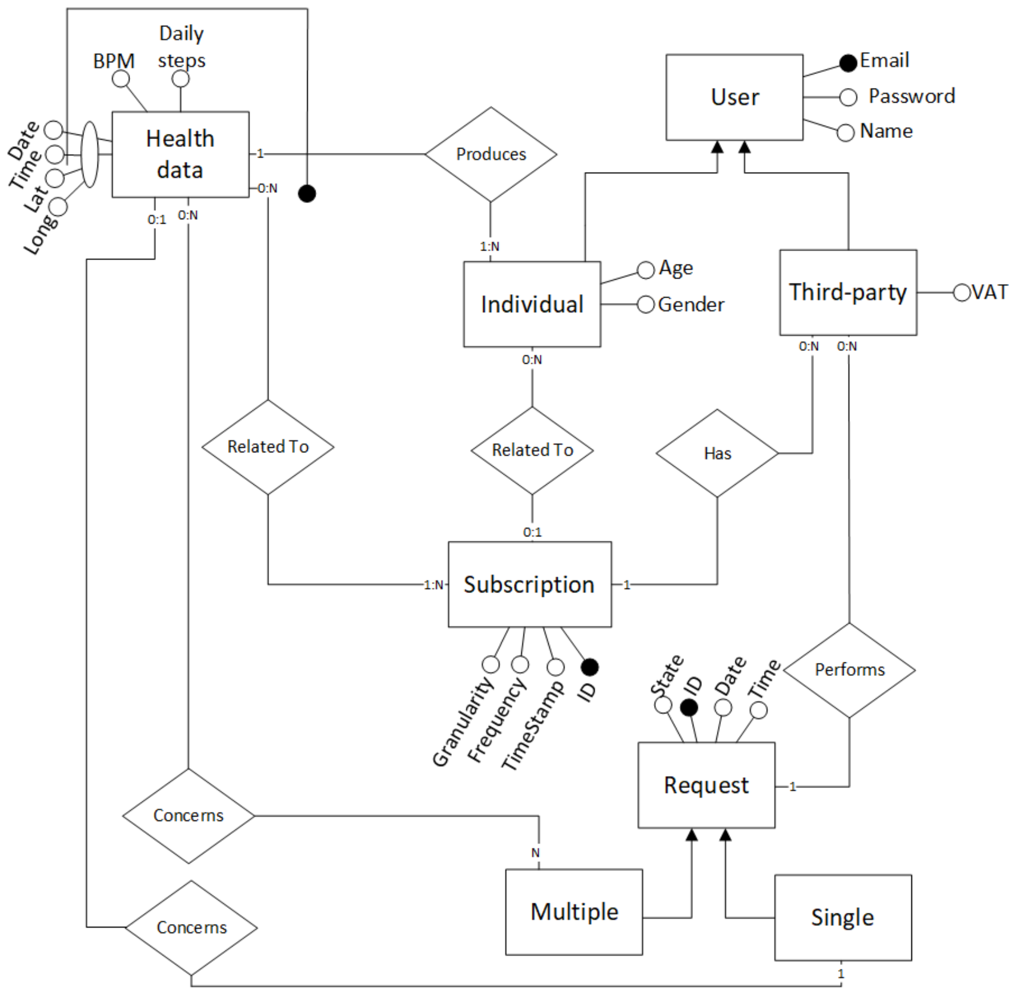


Figure 2.19: D4H database ER diagram

In the picture below (**picture 2.20**) there is the the conversion from the ER diagram described before to a Logic Schema.

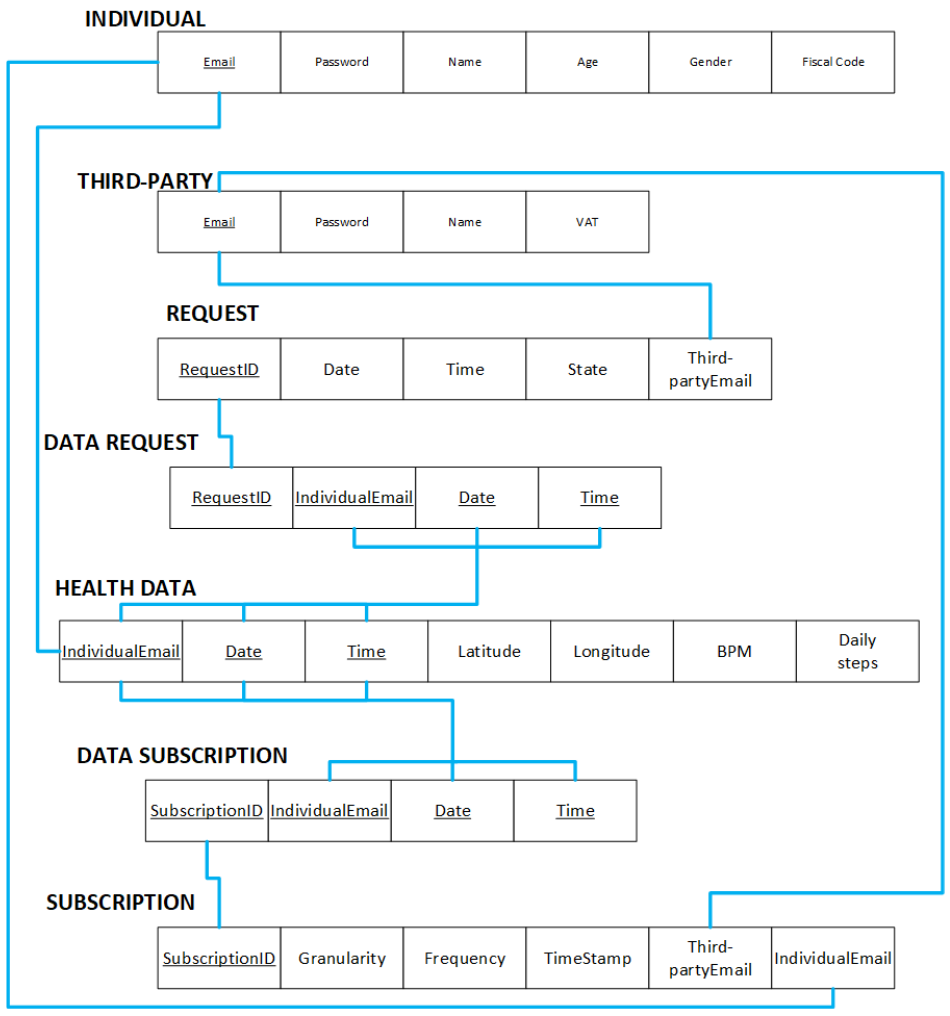


Figure 2.20: D4H Logic Schema

2.6.2 Track4Run

Figure 2.21 represents the ER diagram of the T4R system. Track4Run needs to store data related to Organiser and Participant accounts and information about races.

In this case, he main entities are:

- **Organiser**: it models Organiser account data such as its name, email, password, etc. An organiser can organise zero, one or more races.

- **Participant**: this entity is used to store Participant account information. It is associated to *Race* through the "enrols" relation.
- **Race**: this is the entity containing race details, such as the ID, the location, date and time; there is a relation between *Race* and *Organiser*.
- **Ranking**: it is used to store participants' arrival rankings in past races. Any Participant can be involved in one or more rankings if he enrolled in one or more races in the past.
- **Path**: Every race tuple is associated to a path; it is represented by a starting point, an ending point, a total length and n middle positions between the starting and the ending locations. A single path can be associated to more then one race.

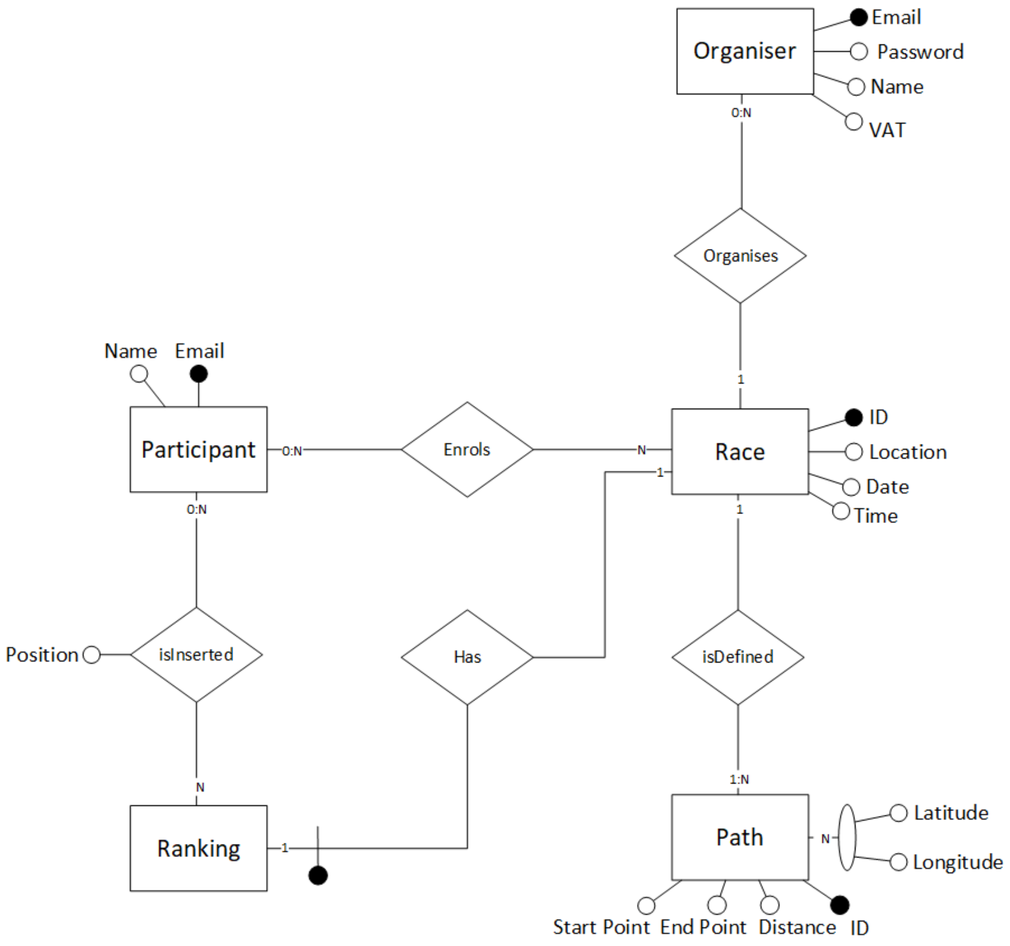


Figure 2.21: T4R database ER diagram

As already done for D4H, also for T4R we show the ER diagram conversion to Logic Schema (**figure 2.22**).

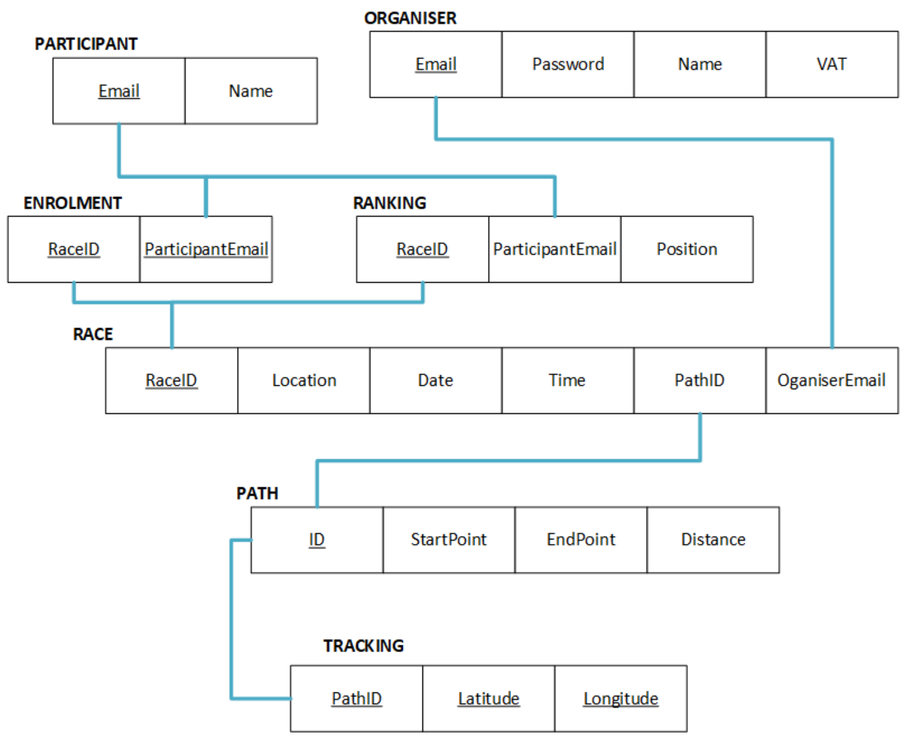


Figure 2.22: T4R Logic Schema

2.7 Selected architectural styles and patterns

In this section, it is discussed about the architectural decisions made for the entire system and about the patterns used, particularly related to the back-end of the system.

2.7.1 Architectural styles

Data4Help and Track4Run systems use the best known and most used architectural style for distributed applications: *Client-Server* with four tiers (Presentation tier, Supervisor tier, Application tier and Data tier, in our case).

Four-tier architecture allows any one of the four tiers to be upgraded or replaced independently.

- **Presentation Tier**: The user interface is implemented on a smart-phone and uses a standard graphical user interface with different modules running on the application server.

- **Supervisor Tier**: It is implemented on the Web Server, that uses the HTTPS protocol to receive and forward messages to the next tier, acts as a balancer for computational loads and replaces failed components.
- **Application Tier**: The system logic is on separate machines, one per application. Each of them is, however, replicated on several machines.
- **Data Tier**: The relational database management system on the database server contains the data storage logic.

Even ASOS system is structured as a Client-Server Application but with only three tiers: presentation tier, delivery tier and application tier (with the same characteristics and functionalities presented before). It does not contain a Data tier, because the system exploits the D4H application to obtain user data and does not need to store any additional information.

2.7.2 Selected patterns

1. **Transaction Based Delivery**: All operations involved in the reception of a message are performed under one transactional context guaranteeing ACID behaviour. This pattern can be applied to *Request Manager* and *General Communication Manager* components in D4H and to the *General Request Manager* in T4R, in charge of receiving and forwarding client messages.
2. **Stateless Components**: Application components are implemented such that they do not have an internal state. It's the case of Application Tier components, whose information are stored in and retrieved from the external database; in this way in case of failure, any information can't be lost.
3. **Dedicated Components**: Dedicated application components are provided exclusively for each client using the application. This pattern is applied to the *Subscription Manager* component, that has a dedicated instance for each request with subscription (both single and multiple requests).
4. **Watch-dog**: Applications cope with failures automatically by monitoring and replacing application component instances if the provider-assured availability is insufficient. The Web Server fulfils this role in the architecture of the three applications: the various components on the application tier are replicated across multiple machines, so if there is a failure in one of them, the Web Server can redirect the traffic on another machine seamlessly, since said components are stateless. Moreover, it is replicated itself so that the applications are guaranteed to work even if a machine fails (this allows to have multiple instances of the watch-dog too). Note that also the Subscription Manager is developed in a watch-dog fashion, being it capable of detecting the failures of the different instances of the sub-component and of replacing them.

5. **Elastic components**: A componentized application uses multiple compute nodes provided by an elastic infrastructure. Also in this case, the Web Server is responsible for the load balancing of the computation across the multiple machines of the application tier. This balancing could quickly change during a race, for example, where the continuously updated position of the Participants could impact on T4R server performance.

Chapter 3

Requirements Traceability

In this chapter, there are traceability matrix, one per application, to explain how requirements defined in the RASD map to the component defined before (in the Appendix A there are detailed lists of Requirements, Goals and Domain Assumptions)

AutomatedSOS

GOAL	REQUIREMENT	COMPONENT	USE CASE ID
[G2.1]	[R2.2]	Login&Registration Manager	Subscription Request
[G2.2]	[R2.1]	Data collector & Checker	Ambulance dispatching

Track4Run

GOAL	REQUIREMENT	COMPONENT	USE CASE ID
[G3.1]	[R3.6]	Registration&Login Manager	Organiser registration and log in
[G3.1]	[R3.7]	Race Manager	Run organisation
[G3.1]	[R3.9]	Race Manager	Run organisation
[G3.2]	[R3.1]	Race Manager	Enrolment to a run
[G3.2]	[R3.2]	Race Manager	Enrolment to a run
[G3.2]	[R3.8]	Race Manager	Enrolment to a run
[G3.3]	[R3.3]	Tracker Manager	Participant tracking
[G3.3]	[R3.4]	Tracker Manager	Participant tracking
[G3.1] [G3.2]	[R3.5]	Account Manager	Rank history

Data4Help

GOAL	REQUIREMENT	COMPONENT	USE CASE ID
[G1.1]	[R1.1]	Registration&Login Manager	Individual registration/log
[G1.1]	[R1.2]	Registration&Login Manager	Third-party registration and log in
[G1.1.1]	[R1.4]	Request Manager	Single user data request and subscription
[G1.1.3]	[R1.6]	Subscription Manager	Single user data request and subscription
[G1.2]	[R1.4.1]	Privacy Manager	Single user data request and subscription
[G1.2]	[R1.4.2]	Privacy Manager	Single user data request and subscription
[G1.1.3]	[R1.7]	Subscription Manager and Account Manager	Single User data unsubscription
[G1.1.3]	[R1.7.1]	Subscription Manager and Account Manager	Single User data unsubscription
[G1.1.3]	[R1.6]	Request Manager	Aggregated user data request and subscription
[G1.1.2]	[R1.5]	Query Manager	Aggregated user data request and subscription
[G1.1.2]	[R1.5.1]	Request Manager	Aggregated user data request and subscription
[G1.1.2]	[R1.5.2]	Request Manager	Aggregated user data request and subscription
[G1.1.2]	[R1.5.3]	Request Manager	Aggregated user data request and subscription
[G1.1.3]	[R1.7]	Subscription Manager and Account Manager	Aggregated user data unsubscription
[G1.1.3]	[R1.7.2]	Subscription Manager and Account Manager	Aggregated user data unsubscription
[G1.1.3]	[R1.6.1]	Subscription Manager	Collected data during Subscription
[G1.2]	[R1.3]	Data Gatherer	Data Collection and Visualisation
[G1.2]	[R1.9]	Account Manager	Data Collection and Visualisation
[G1.1]	[R1.8]	Account Manager	Third-parties request history

Chapter 4

Implementation, Integration and Test Plan

In order to find any possible problem during the implementation as soon as possible, the integration and testing of the applications will be done incrementally during the implementation itself.

A thread approach will be adopted both for the development of D4H and T4R, since there are a lot of dependencies among their internal components. This technique will allow the developers to focus on the offered features of the applications, testing each of them instead of the components. Moreover, in this way, the integration will be brought on together with the implementation for each single feature, linking the various components as the development proceeds, without the need of building stubs and drivers for the sake of testing. The thread approach also facilitates user acceptance and understanding of the development process, since the functionalities of the applications can be presented as they are completed. As for ASOS, its two components and external interfaces can be developed and tested in parallel, proceeding with the integration afterwards, being it a very simple system.

4.1 Implementation

The implementation environments for the three applications are the same:

- Java EE for the application server, because of its high adaptability and compatibility with other environments;
- C# for the Android and iOS applications, using Xamarin framework, that allows to develop cross-platform mobile applications and to deploy them on their respective OSs;
- MySQL for the DBMS, because it is the most widely used relational DBMS.

The first application that has to be implemented is *Data4Help*, since both *AutomatedSOS* and *Track4Run* exploit its external interfaces (see **section 2.5**). The order in which the features are implemented is the same as the one in which they are integrated and tested and it is shown in the next section.

4.2 Integration and Test Plan

As already stated before, the development and testing of D4H must be completed before starting to implement ASOS and T4R, so the development team effort should be focused on that application first. Afterwards, ASOS and T4R can be implemented and integrated in parallel.
Note that only the server-side is described here, but also the presentation and graphical interface in the Client should be brought on feature by feature, so as to have them fully completed one by one.
On the other hand, the database is the first element that should be completed, since every other component (directly or indirectly) depends from it.

The details concerning the functioning of the software components will not be further discussed here, since they have already been presented in **section 2.2**.

4.2.1 Data4Help

There are five features that can be identified in D4H, as shown in the **figure 4.1**. Each of them is identified with a colour and mapped to the components needed to offer it.

The development will be carried on with a thread for each feature, in order of importance.

1. *Data gathering and visualisation*

The first one to focus on is the data gathering from Individuals and their visualisation (light blue rectangle in the **figure 4.1**). Completing this would allow to have real sampled data to be used for the next functionalities. In order to do so, the part of the General Communication Manager in charge of delivering the acquisition data messages must be developed, along with the Data Gatherer, as a whole, for their processing and storing. As for the visualisation of these data, even the part of the Account Manager that retrieves them from the database has to be completed. Also in this case, the General Communication Manager acts like a switch, forwarding the message.

2. *Third-party single request*

Having now the capability of acquiring Individuals' data, the second feature to focus on is the single data requests, and possible subscriptions, performed by Third-parties (red rectangle in the **figure 4.1**), because it involves several component and it is more likely to face difficulties in implementing it. The interested components are:

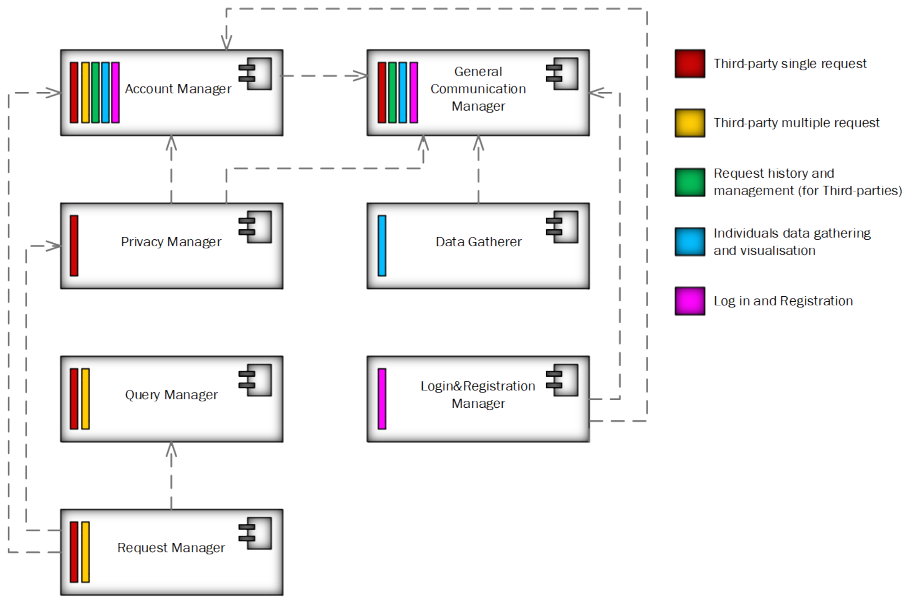

Account Manager
General Communication Manager
Privacy Manager
Data Gatherer
Query Manager
Login&Registration Manager
Request Manager
Third-party single request
Third-party multiple request
Request history and management (for Third-parties)
Individuals data gathering and visualisation
Log in and Registration

Figure 4.1: Dependencies among D4H components

- Request Manager for the request processing;
- Query Manager for data retrieval from the database
- Privacy Manager to ask for the Individual's approval;
- Account Manager to handle subscriptions;
- General Communication Manager to route the messages.

The sections of these components that deal with this type of request have to be fully integrated and tested at this point. Note that the Privacy Manager is only used for this functionality, so it must be fully implemented and tested in this phase.

3. *Third-party multiple request*

At this point the feature that allows Third-parties to perform group searches, with eventual subscriptions, can be implemented (yellow rectangle in the **figure 4.1**). Its functioning is similar to the single request, but it does not involve the Individual, since the legality of the request is checked by the system. Since there is no communication with the Individual involved, only the Request Manager, the Query Manager and the Account Manager have to be integrated and tested further.

4. *Request history and management*

Provided that the different kind of request can be performed, their chronology and management (green rectangle in the **figure 4.1**) should be developed, integrated and tested. This feature only concerns the Account Manager, and the General Communication Manager for the communication with the Third-party Client.

5. *Log in and Registration*

After having integrated all the core functionalities of D4H, the log in and registration system (purple rectangle in the **figure 4.1**) can be implemented, so that every user can be unequivocally recognised by the system. This last feature has its own dedicated component, aided by the General Communication Manager for the communication with the Client and by the Account Manager for the one with the database.

4.2.2 AutomatedSOS

ASOS structure is quite simple, being made only by two components, as shown in **figure 4.2**. Its development will only require two threads.

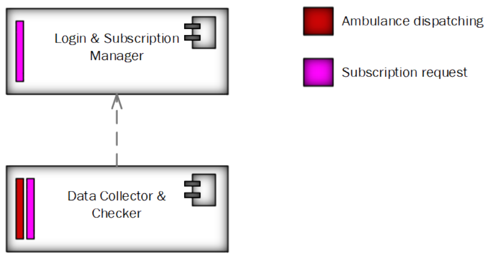


Figure 4.2: Dependencies among ASOS components

1. *Ambulance dispatching*

It is of capital importance for this feature (red rectangle in the **figure** 4.2) to be functioning correctly, hence it has to be thoroughly tested (both function-wise, checking if the ambulance is called *if and only if* the health data received from D4H is over the critical threshold, and performance-wise, verifying that the ambulance is called within 5 seconds after receiving the critical measurement). The only component that deals with it is the Data Collector & Checker, which analyses incoming data and eventually communicates with the ambulance service. Until the next feature is not integrated, this one can be tested using health data that cover all possible situations (warning the ambulance service that it will receive some false alarms).

2. *Subscription request*

This feature (purple rectangle in the **figure 4.2**) allows an Individual to subscribe to ASOS and it works just like a D4H subscription. It involves both the Login & Subscription Manager, that receives the log in request from the user and checks through D4H Communicator Manager Interface if he/she is registered to D4H as an Individual, and the Data Collector & Checker, that performs the actual subscription request through D4H Request Manager Interface. After having integrated this feature, the ASOS system can be tested as a whole using real data.

4.2.3 Track4Run

There are five features that can be identified in T4R, highlighted in **figure 4.3** below.

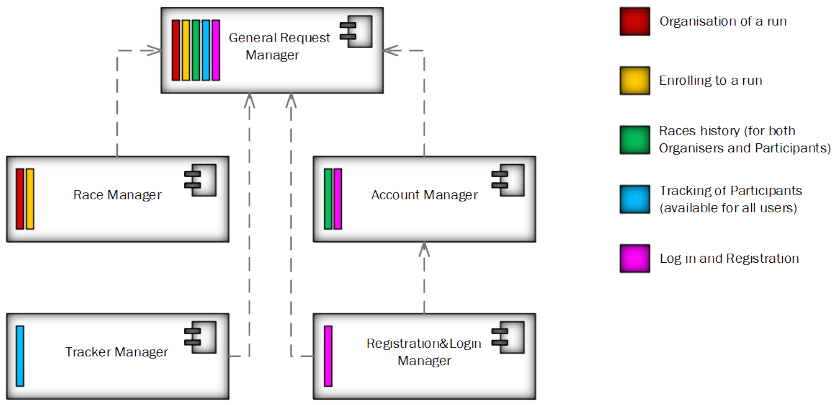

General Request Manager
Race Manager
Account Manager
Tracker Manager
Registration&Login Manager
Organisation of a run
Enrolling to a run
Races history (for both Organisers and Participants)
Tracking of Participants (available for all users)
Log in and Registration

Figure 4.3: Dependencies among T4R components

The various functionalities will be implemented, integrated and tested in order of importance, with a thread for each feature.

1. *Organisation of a run*

The first feature to implement and test is the possibility for Organisers to organise runs (red rectangle in **figure 4.3**), because the next ones can exploit it. This feature is encapsulated in the Race Manager component, aided by the General Request Manager to communicate with the Organiser Client.

2. *Enrolling to a run*

Having races available allows for Participants to enrol to them (yellow rectangle in **figure 4.3**), therefore this should be the next focus, so as to integrate the publish-enrol system and to test it as a whole. In this way, the development of the Race Manager will be completed and fully tested, along with another section of the General Request Manager.

3. *Tracking of Participants*

After the race management has been completed, the tracking system (light blue rectangle in **figure 4.3**) should be implemented and integrated, in order to test its functioning during a fully organised race. This feature is the only one that concerns the Tracker Manager, so this component will be fully integrated and tested by the time this thread reaches termination. Moreover, the part of the General Request Manager that forwards the tracking request consequent to an enrolment will be integrated.

4. *Races history*

At this point all the core functionalities of T4R have been integrated and tested, so the race history (green rectangle in **figure 4.3**), which is a support feature, should be developed and integrated next. The component responsible for this feature is the Account Manager, that communicates directly with T4R's database. It also exploits the General Request Manager in order to communicate with the Clients.

5. *Log in and Registration*

Finally, the log in and registration system (purple rectangle in **figure 4.3**) should be implemented, integrated and tested, so that Organisers can have their own account and Participants' previous registration to D4H can be checked. This last feature has its own dedicated component, aided by the General Request Manager for the communication with the Clients and by the Account Manager for the one with the database.

Chapter 5

Effort Spent

5.1 Alessia Buccoliero

DATE	Number of hours	TOPIC
20/11/2018	2	General discussion
26/11/2018	2	Component definition
26/11/2018	2	Component view first draft
27/11/2018	2	Chapter 1: Introduction
27/11/2018	4	Component diagram definition
28/11/2018	1,5	Component diagram definition
29/11/2018	3	RASD version 1.1
30/11/2018	1,5	Deployment view
30/11/2018	3,5	Architectural style and Patterns; ER diagrams
1/12/2018	1	RASD version 1.1
1/12/2018	4	Overview and Deployment diagram
2/12/2018	1	Component Diagram
2/12/2018	4	Component Interface definition and Implementation, Integration and test plan discussion.
3/12/2018	3,5	Component Diagram
4/12/2018	6,5	Component Diagram Revision
5/12/2018	5	Chapter 2: Component view
6/12/2018	1,5	Chapter 4: Requirements Traceability
7/12/2018	3	Chapter 2: Deployment view
7/12/2018	5	Chapter 2: Database view
8/12/2018	4	Chapter 2: Architectural style and patters
8/12/2018	1	ASOS sequence diagram
10/12/2018	2	Final revision

Table 5.1: Alessia Buccoliero's working hours

5.2 Emilio Corvino

DATE	Number of hours	TOPIC
20/11/2018	2	General discussion
26/11/2018	2	Component definition
26/11/2018	2	Component view first draft
27/11/2018	2	Chapter 1: Introduction
27/11/2018	4	Component diagram definition
28/11/2018	1,5	Component diagram definition
29/11/2018	2	RASD version 1.1
30/11/2018	1,5	Deployment view
30/11/2018	3,5	Architectural style and Patterns; ER diagrams
1/12/2018	2	RASD version 1.1
1/12/2018	4	D4H and T4R ER diagrams
2/12/2018	1	Sequence diagrams
2/12/2018	4	Component Interface definition and Implementation, Integration and test plan discussion.
3/12/2018	3,5	Sequence diagrams
4/12/2018	6,5	Component Diagram Revision
5/12/2018	5,5	Chapter 5: initial writing
6/12/2018	3	Chapter 5: dependency diagrams
7/12/2018	7	Chapter 5: writing
8/12/2018	2	Chapter 2: Overview and Selected Architectural Styles and Patterns writing
8/12/2018	1	ASOS sequence diagram
10/12/2018	2	Final revision

Table 5.2: Emilio Corvino's working hours

5.3 Gianluca Drappo

DATE	Number of hours	TOPIC
20/11/2018	2	General discussion
26/11/2018	2	Component definition
26/11/2018	2	Component view first draft
27/11/2018	2	Chapter 1: Introduction
27/11/2018	4	Component diagram definition
28/11/2018	1,5	Component diagram definition
29/11/2018	2	RASD version 1.1
30/11/2018	1,5	Deployment view
30/11/2018	3,5	Architectural style and Patterns; ER diagrams
1/12/2018	6	RASD version 1.1 and ER design
2/12/2018	5	Component Interface definition and Implementation, Integration and test plan discussion.
3/12/2018	3,5	Sequence Diagram Design
4/12/2018	6,5	Component Diagram Revision
5/12/2018	3,5	Sequence Diagram Design
6/12/2018	4	Sequence Diagram Design and Component Diagram review
8/12/2018	8	Sequence Diagram Design and intro section 2.5
9/12/2018	7	Section 2.5 and Component Interface code
10/12/2018	2	Final revision

Table 5.3: Gianluca Drappo's working hours

Bibliography

[1] Activity diagram tutorial. https://www.lucidchart.com/pages/uml-activity-diagram.

[2] Client-server architecture, three-tier. https://www.techopedia.com/definition/24649/three-tier-architecture.

[3] Cloud computing patterns. www.cloudcomputingpatterns.org.

[4] Cross-platform mobile development - xamarin. https://www.html.it/guide/guida-xamarin/.

[5] Defining stateful vs stateless web services. https://nordicapis.com/defining-stateful-vs-stateless-web-services/.

[6] Deployment diagrams. https://www.uml-diagrams.org/deployment-diagrams.html#node.

[7] Latex tutorial. https://www.overleaf.com/learn.

Appendix A

Goals, Requirements and Domain Assumptions

A.1 Goals

The goals defined for the applications are reported below. For more information, refer to the RASD.

A.1.1 Data4Help

[G1.1] Third-parties have to be able to monitor the location and health status of Individuals.

- [G1.1.1] Third-parties have to be able to request data of an Individual through his/her fiscal code.
- [G1.1.2] Third-parties have to be able to request data of groups of Individuals.
- [G1.1.3] Third-parties have to be able to subscribe to user data, specifying the frequency of updates and the desired granularity of data.

[G1.2] Individuals have to be able to decide whether to share their not anonymised data (gathered through smart devices) or not and to see what they are sharing to whom.

A.1.2 AutomatedSOS

[G2.1] Elderly people's health status has to be constantly monitored.

[G2.2] An ambulance has to arrive to Elderly people's location when their vital signs go below a certain threshold.

A.1.3 Track4Run

[G3.1] Organisers have to be able to define the details of runs.

[G3.2] Runners have to be able to enrol to runs.

[G3.3] Visitors have to be able to track Participants' position on a map.

A.2 Requirements

A.2.1 Data4Help

[R1.1] The system allows Individuals to register, requiring name, surname, fiscal code, password and email in order to create a new Individual account.

[R1.2] The system allows Third-parties to register, requiring the company name, VAT number, email and password in order to create a new Third-party account.

[R1.3] The application retrieves from smart-watches the following data, associating them to the user:

- beats per minute (bpm);
- number of steps;
- location;
- time-stamp of the measurement.

[R1.4] The system allows Third-parties to request data of Individuals and the latter to accept or refuse them.

  [R1.4.1] If the Individual accepts, the system sends to the Third-party all the data gathered from the specified user.

  [R1.4.2] If the Individual refuses or he/she is not registered, the system sends to the Third-party an error message.

[R1.5] Upon Third-parties request, the system can perform parametric searches based on geographical areas, age, genre, time of the day. The result of the query is then stored and evaluated for approval.

  [R1.5.1] If the request is approved, the saved data is provided to the Third-party.

  [R1.5.2] If the request is not approved, an error message is sent to the Third-party.

  [R1.5.3] In order to anonimize data, the application only sends information related to the health status of individuals (e.g. bpm, step, genre, age).

[R1.6] The application allows Third-parties to subscribe to certain data.

- [R1.6.1] The application, if the Third-party subscribed to some data (belonging to Individuals or group searches), sends updates to the Third-party aggregating the data with the specified granularity and frequency.

[R1.7] The application allows Individuals withdrawals of consent for accessing their data and Third-party unsubscriptions.

- [R1.7.1] If the Individual withdraws consent for the access of his data, a message is sent to the subscribed Third-party and data is no longer provided.
- [R1.7.2] If the Third-party no longer wishes to collect data from an Individual, a message is sent to him/her and his/her data is no longer sent.

[R1.8] The system allows Third-parties to access to their request history and results.

[R1.9] The system allows Individuals to check gathered data.

A.2.2 AutomatedSOS

[R2.1] The system has to analyse user data and sends the location of the user to the ambulance service in case of emergency, when health values goes beyond the threshold.

[R2.2] The system gives the possibility to subscribe to the service upon request, only after checking the Individual age from his/her fiscal code.

A.2.3 Track4Run

[R3.1] The system allows Participants to log into the application through their Data4Help account and, after they are logged in, to enrol the the runs available. They should also be able to apply filters to ease the research (e.g.: place, time, date range).

[R3.2] The system allows the enrolling of Participants only if they are not enrolled in a race at the same time.

[R3.3] The system allows Visitors to access the application as host-users, without registration, by giving them only the possibility to track the Participants.

[R3.4] At the end of each run, the systems shows the rank to all Users watching it.

[R3.5] The system stores the ranks of finished races so that both Participants and Organisers can access to the race history.

[R3.6] The system allows the registration of Organisers without requiring also Data4Help registration.

[R3.7] In order to organise a run, the application requires location, data, time and the maximum number of participants.

[R3.8] The system needs to update subscriptions to the runs, decreasing the number of allowed participants. It must not allow further subscriptions if there are no leftover places.

[R3.9] The system does not allow that two or more runs are organised at the same place and time.

A.3 Domain Assumptions

Data4Help

[D1.1] Acquired data is precise enough.

[D1.2] Each Individual can be identified unambiguously through his/her fiscal code.

[D1.3] One anonymized, data cannot be associated to specific Individuals.

[D1.4] Every interaction gets correctly encoded.

[D1.5] Third-parties are assumed to be companies that want to gather data, therefore they have a VAT code.

[D1.6] Third-parties know the fiscal code of the Individual they are looking for.

AutomatedSOS

[D2.1] An ambulance service capable of handling requests exists. It is also capable of receiving the personal data of the Elderly person and his location.

[D2.2] The thresholds beyond which a person is considered to be in need of help are under 40 bpm and above 130 bpm.

[D2.3] The service is offered in Italy (118 is the emergency number for the ambulance service).

Track4Run

[D3.1] Maps and tracking services are accurate enough.

[D3.2] The place where the run is held exists.