MASTER THESIS

TITLE: Advanced personal social network API for third-party mobile applications

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Alberto Carlos Toro Sánchez

DIRECTOR: Antoni Oller Arcas

DATE: 07-20-2011

Título: Advanced personal social network API for third-party mobile applications

Autor: Alberto Carlos Toro Sánchez

Director: Antoni Oller Arcas

Fecha: 20-07-2011

Resumen

Hoy en día nuestro mundo está permanentemente conectado, esto forma parte de la vida cotidiana que nos rodea. El creciente aumento de teléfonos inteligentes o *Smartphones*, la necesidad de controlarnos y saber cómo se encuentran los demás, llevan a la realización de este proyecto.

El principal objetivo del proyecto es desarrolla un concepto de aplicación que permita la recolección de datos sobre el usuario para mostrarlos a uno mismo y a los demás.

Inicialmente se han realizado estudios previos de varias tecnologías que pueden ser de utilidad para llevar a cabo el diseño y la implementación de la aplicación, y de esta manera, hacer llegar el servicio de *AAAida* hasta usuarios de *Smartphones*. Se han realizado casos de uso y pruebas para comprobar las funcionalidades de cada tecnología de cara a implementarlas en la aplicación.

Teniendo en cuenta el resultado de las pruebas se ha diseñado una arquitectura para que el usuario pueda consultar sus valores de las medidas como por ejemplo el peso, colesterol, glucosa, etc. o ingresar nuevas. Sobre este diseño se han ido añadiendo nuevos diseños de funcionalidades.

Finalmente se ha implementado el diseño realizado que permite al usuario realizar las funciones marcadas. Se han realizado con éxito test sobre la implementación para comprobar que los objetivos marcados se han conseguido, como son los comentado previamente, localización del dispositivo en un mapa, realidad aumentada para localizar lugares de interés como policía o farmacias y el tracking de rutas o paseos del usuario.

Title: Advanced personal social network API for third-party mobile applications

Author: Alberto Carlos Toro Sánchez

Director: Antoni Oller Arcas

Date: 07-20-2011

Overview

Nowadays, our world is permanently connected; this is part of the daily life that surrounds us. The fast grown of *Smartphones* and the need of control us and how other people are, are the main reasons why this project is done.

The main objective of the project is to develop a conceptual application that can collect user data to show it to other users or that can be checked for oneself.

Initially, previous studies of several technologies have been done, which can be useful to carry out the design and implementation of the application, and thus, bring the *AAAida* platform to *Smartphone* users. Moreover, use case and proves have been completed in order to test functionality of each technology toward implementing the application.

Taking these test results as a reference, the architecture have been designed to allow the user to consult his measures data such as weight, cholesterol, glucose, etc. or enter new ones. Over this design new features have been added.

Finally, the proposed design have been implemented, which allow the user realize the objective functions. The test over the implementation have been performed with success to check that the previously described objectives have been achieved, like device location on a map, augmented reality to locate emergency places like police or pharmacies and tracking of routes of user walks.

To my family, friends and colleagues whose support and encouragement have allowed me to finish this Master Thesis

INDEX

FIGURES

INTRODUCTION

We live in a world that is permanently connected with Internet. This fantastic tool is part of our daily life. Moreover, the grown of the mobility services was unthinkable some years ago.

These ideas move the Internet services market to offer solutions that allow people to stay aware with the things surrounding them. If the things that surround us are changed by people, the concept of social networks appears and it brings new possibilities and services to Internet.

This mobility and the desire, which can be created by the demand itself, of knowing in real time the state of the people that are important, or maybe pure entertainment, are synergies that must be taking advantage of these to create new services based on the social network concept. These new services help to take care and/or monitories people who are important, starting definitely by oneself.

These ideas come together to offer these features in a unique idea which generates a new service, *AAAida* [1]. The social network that allows knowing and sharing the status of the people and things that you concern and worry, ensuring the peace of mind you deserve.

This work grants to user the possibility of accessing to *AAAida* platform when and where it is needed. Furthermore, it would allow the user control or monitor his health state or if necessary, offer help in case of emergency.

This project seeks to achieve this goal ruled by for these ideas, searching viable solutions that allow develop a conceptual application, which helps to reach the mentioned goal.

The *Master Thesis* is organized in the following way: In the first chapter, we will place in the work environment, a brief overview of what *AAAida* is and his owners *Alteraid* [2], the mobility project for *AAAida*, its context and the objectives that must be followed to complete the project successfully.

The second chapter explains a proof of concept, remarking the possible functions of the application and follow up on what it can offer to users.

The third chapter is a brief explanation about the platform architecture and which is the mobile application location in the architecture. Each one of the platform components are explained and show the design followed in this project. Below this, more exhaustive explanation about the components, including the interaction with the platform and between other components, is done.

In the fourth chapter the implementation of the concept is shown. The followed steps for the realization, the tools and technologies used, the necessary

environment for developing the application and a brief explanation of what can be found in the application are explained.

The following chapter, the fifth, shows the explanation of work planning and the approximate cost of the development of this *Mater Thesis.*

Finally, the extracted conclusion of the project performing and implementation are commented. It is followed of future improvements for the implementation, the environmental impact that may result in the project. To conclude this chapter the personal conclusions are shown.

CHAPTER 1. PROJECT DESCRIPTION

The context of this project is the new social platform called *AAAida* which is being developed by *Alteraid*. *Alteraid* is a spin-off of *Universitat Politècnica de Catalunya (UPC)* [3]; born in the *Escola d'Enginyeria de Telecomunicacions i Aeronautic de Castelldefels* [4] which main goal is to take advantage of the growing of social networks in the Internet.

AAAida takes the idea of social network sharing the person or object states between other users. The main objective of this platform is the people who want to take care of themselves, to other people or to objects that are important for their.

The project is centred in the need of expand this concept to the people that uses smartphones, see *Annex II*. , especially the Android ones and make easier the way to use *AAAida*. It is needed to prepare a door called *Application Programming Interface (API),* which could perform the input and output of data from the *AAAida* to devices via web services.

This chapter is focused in introducing the basic concepts to understand the planning and development of this mobile application. First of all, the *AAAida* platform and the mobile application are explained in a brief introduction. Next, the basic concepts needed to understand the basic about social networks, augmented reality and location. The main objectives of this *Master Thesis* complete this chapter.

1.1. AAAida

As can been seen at the beginning of the chapter, *AAAida* is a project created to integrate the social network idea with the concept of care about things or people.

The main part of the project is *AAAida*. This contains the *Web* that allows to users upload or review states, and the database where the data is saved to be used were ever is needed. The other part of *AAAida* is the mobile application, which can be used at anytime and anywhere, to upload new states about him to the platform.

1.1.1. The platform

As it has been explained before, the platform is composed by a *Web*, which is accessible for all the users, and a database.

The *Web* allows users update their status, to be to follow their evolutions by themselves or by other users. In this site, any type of measure can be

uploaded, e.g.: weight, height, blood pressure, temperature, tracking, etc. Moreover, the users can create links, or bonds, of other people or things to be monitored as well, for example, creating a bond to the family car and check a daily status about the fuel consumption. These bonds can be real or not.

The database of the platform is important, too. It stores all the uploaded information via *Web* or external applications like mobile applications. The structure of the data is based on a patient model; this structure is explained in the following chapters.

In order to access to database storing, the *Web* client needs to call the *AAAida API* functions, like other *AAAida* clients.

1.1.2. The mobile application

This *Master Thesis* is focused in the development of the *AAAida* mobile application that must use the *AAAida API*. This application is a tool to make easier the status or measurement updates in the platform. Moreover, previously uploaded data can be reviewed in the application.

The main idea for developing of a mobile application is to approach *AAAida* to new users that in anytime of the day and everywhere they can update the status without the need of a computer to use the *Web*.

An example can be read in the following use case:

"Carmen and Pilar are sisters, who constantly need to know about each other. The two sisters use the AAAida platform to be updated about the status of the other sister. Carmen has planned to spend two week's holiday, so she cannot be aware of having a computer to connect to AAAida and updated her status and data. Her sister recommended her to install a mobile application for the smartphone that allows her to update data and status and also, she will be able to check the evolution of the status in charts. Furthermore, she can upload the walked distance in each tour and record it in the phone to have a memory of where she was and where she walked exactly".

Another use case where the application can take sense is for example, in a digital photo frame, which in addition to displaying photos it has integrated the *AAAida* application. This frame could be a gift from *Carmen* to his elder uncle *Benet*, which shows automatically the family's albums. At one moment, the application will blow an alarm showing that the uncle should take his blood pressure. The alarm blocks the slideshow until he uploads data in simple and intuitive way.

1.2. Basic concepts

There are different concepts that will be explained in this section, which refer to social networks, location based context awareness and augmented reality.

1.2.1. Social networks, Web 2.0 & Semantic network

The *social networks* [5] are a growing global phenomenon that affects to almost all the countries. This kind of networks uses the known theory of the *six degrees of separation* [6]. This theory says that anyone on Earth is connected with another person through a known chain that does not have more than five intermediaries.

The networks are social structures composed by groups of people that are connected by one or more kind of relationship. The group of people is composed by individuals or organizations and in the social networks are known as *"nodes"*.

There are many types of *social networks* but roughly can be distinguished three modalities: those that allow share knowledge, those that help to personal relationship and those that allow carrying out different projects between registered users.

Nowadays, the *social network* idea is correlated with the concepts of *Web 2.0* [7] and *semantic network* [8].

On the one hand, the *Web 2.0* consists of portals that have dynamic pages, generated automatically on user's petitions. Furthermore, this kind of *Web* always has updated content without the necessary interaction of the users. It is possible by keeping a constant connection with the server side and executing part of the *Web* software in the client side; these performance front the *Web 1.0* make the *Web 2.0* more flexible and interactive. The first concept of *Web 2.0* appeared in 2004 introduced by *Dale Dougherty* and *Craig Cline* in a conference [7].

On the other hand, the concept of *semantic network* is linked with the *Web 2.0* like a step to a *Web 3.0* [9]; this kind of Web offers contents in a smart way and more oriented each kind of user. For example, with tagged contents, it can show a filtered page to the user, based on his needs. These needs or preferences are related with tags, which are related with specific contents.

As can be seen before in this chapter, *AAAida* is focused in a personal relationship between the users, more precisely, in a health relationship.

1.2.2. Location based context awareness

The main idea that can be extracted from *Location based context awareness* [10] is to obtain the position of the device knowing first what is surrounding it. With this and adding the information obtained through the *GPS* hardware, the device could be located faster and accurately. The obtained information must be processed in order to give a position, which is called geolocation [11].

Geolocation is the identification of the real-world geographic location and it refers to the practice of calculating the actual location.

The geolocation in mobile devices can go with two location sets: the inverse geolocation and the geographic location.

The geographic location is related with geographic positioning and is given with a set of geographic coordinates, latitude and longitude. This kind of location can be obtained through GPS device hardware, as can be said before.

Instead, the inverse geolocation is related to positioning but can be distinguished from geographic location by a greater emphasis on determining a meaningful location (e.g. a street address) rather than just a set of geographic coordinates. To obtain the inverse geolocation just is needed the geographic location and an external services like *Google Maps* [12] to translate the type of location.

1.2.3. Augmented reality

Augmented Reality has become a trendy technology used in mobile phones and other portable devices.

Fig. 1.1 Augmented reality demonstration, source [13]

There exist many definitions of augmented reality. The more accepted one was given by R. T. Azuma [14] in 1997. It says *Augmented Reality (AR) is a variation of Virtual Environments (VE), or Virtual Reality as it is more commonly called*. The main difference between the *AR* and *Virtual Reality* is that for the last is a computed-generated environment instead of adding virtual objects (3D models) inside a real world representation, see *Fig. 1.1*.

This technology can be applied in many fields, for example:

- Video games
- Advertisements
- Art
- Military industry
- Automobile industry
- Place locator
- Healthcare

1.3. Objectives

Based on the terms explained in last sections, now the objectives of the *Thesis* can be defined.

- *AAAida*:
    - Create an *API* based on web services [15] allowing access to 3rd party implementations.
    - The *API* must be integrated with the social network

- *Mobile application*:
    - Upload new measurements as: weight, blood pressure, glucose and cholesterol.
    - Review previous measurements in a temporal graph
    - Auto location in map
    - Using augmented reality tool, which shows the nearest interesting points such police, pharmacy, hospitals, etc.
    - Tracking:
        - Save tracking and upload to the platform: distance, duration, etc.
        - View previous tracking in a map
        - Make backup of saved tracking and restore it if it is needed

To use the application, the users must introduce the email and password to sign in. Once in the principal sight, from the menu the user may access to upload new data about the chosen measure and review or monitor previously uploaded data in a temporal graph.

Take a walk is important too for the health, therefore the user will be able to record routes activating the tracking option and stopping it at the end of the route. The obtained data will be processed in a map where it will be shown, with

the allowed GPS precision, the entire path and the exact start and arriving places. Furthermore, additional data about the tracking, for example the walked time, will be displayed.

The application will offer the possibility of real time location as an option inside the tracking. It can display the name of the surrounding streets by clicking over there.

Another option is the augmented reality that will show the interesting points like police, hospitals, pharmacies, etc., over the camera caption and offering to the user extra information about the points, for example, telephone, street name, etc.

CHAPTER 2. SPECIFICATIONS

This chapter explains the proof of concept of using *AAAida* with the clients *Web* and mobile and the different functionalities that they can offer using the *AAAida API*. At the end of chapter a comparison between the use cases for these clients is done.

2.1. Functionalities

In order to define the specifications of the project developed, first the functionalities have to be detailed.

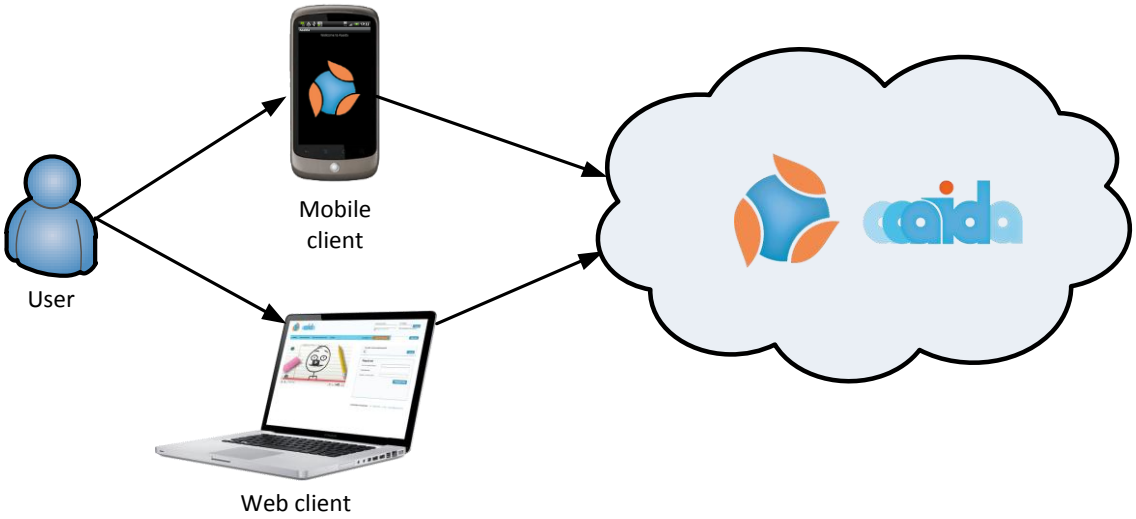


Fig. 2.1 AAAida platform

The different modules that compose the platform are: the *Web client*, the *Mobile client* and *AAAida*. A briefly introduction to the functionalities of each part are explained in next points.

- **Web client:** This is an entry point for the platform and can be used to show the user information and the information of other users.
- **Mobile client:** This is the mobile application that the user can use to interact with de platform. It is an entry point of data for *AAAida*.
- **AAAida:**
    - **AAAida Manager:** This is the part that processes the information obtained from the mobile application, the *Web* or other inputs like third-party applications. All these inputs use the *AAAida API.*

- **Database:** This is the mandatory module for the data storing. It is transparent for the user and cannot be accesses directly, always via the *AAAida Manager*.

2.1.1. Web client

This is the original entry point for *AAAida*. There are not functional requirements; the user needs only a browser to access to the social network.

2.1.1.1. User management

The *Web client* offers the possibility to access to the social network by login with a user name and associated password. Once is inside the service, the user can create and share bonds with other users. These bonds contain the measures with their values.

2.1.1.2. Data management

The user can create new measures and upload values to it. When a new bond, measure or value is created, it has associated a tag, introduced also by user. These tags are used to share bonds or measures with other users and also to organize de information contained in the social network.

2.1.1.3. Application management

With this functionality the user can manage the applications that can access to his information. These applications are created to extend the usability of *AAAida* using the *AAAida API*. Applications such as memory games or mobile applications like the developed in this project.

If the user wants to grant access to the mobile application, he only needs to tick over the application list and then download the application on mobile via *Android Market* [16] or by checking a *QR code* [17] located in *AAAida* official web with the mobile camera.

2.1.2. Mobile client

All users that have a *Smartphone* can install, as is said in *Application management*, the *AAAida* 3rd parties applications. One of these applications is *AAAida*, which is an application that allows to user the access to the social network contents in a friendly way.

The application architecture is *Service Oriented Application (SOA)* architecture. All the mobile client works thanks to the web services offered by the *AAAida*

API, which allow access to the platform via login and upload or download the information about the user and his measures; the same like *Web client.*

Furthermore, there are some extras. The user can do tracking of his routes, search points of interest with the *AR* tool integrated in the application and use the geolocation to locate the device on a map.

The application can only treat a specific list of measures to avoid compromising the proper functioning of the application. This is the list of measures that can be treated by the smartphone application:

- Weight
- Glucose
- Blood pressure
- Cholesterol
- Tracking

2.1.3. AAAida

This is the platform that provides all the services that can be accessed by both Web *client* and *Mobile client*. As is said before, the access is done via web services to his *AAAida API*.

2.2. Use cases

The following figure is a representation about the all functionalities offered by the platform form the user point of view.

Fig. 2.2 The different use case for clients

As can be seen in the Fig. 2.2, both clients can do the principal functionalities like login or upload values to social network. But there are differences, for example, in the *Web client* the values are showed an organized in a wall, while in the mobile are showed in a chart, but only the measures specified in the *Mobile client* section.

The Application management only is accessible through the *Web client*; it has no sense administrate applications through a 3rd party application.

CHAPTER 3. ARCHITECTURE & DESIGN

This chapter is a brief explanation about the architecture which is based this project. It shows the *AAAida API* and mobile application architecture, examples of formal use cases to understand the interaction between components and the design that the mobile application follows.

The next figure shows a scheme of *AAAida* platform design, where are described the different parts of the project, the communication and relationship between them.

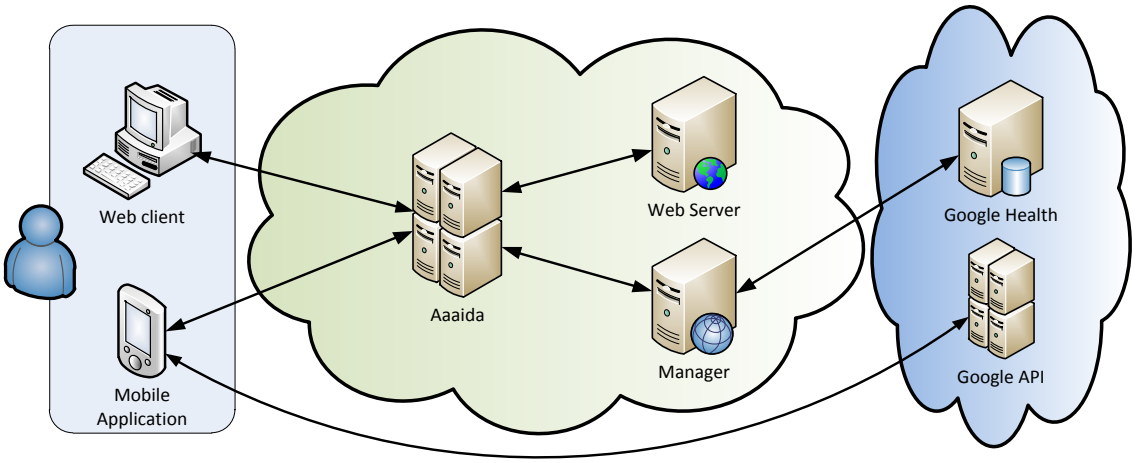


Fig. 3.1 AAAida components platform

As it can be observed in the *Fig. 3.1*, the *AAAida* platform consists in several modules, among which the *Manager* can be specially highlighted.

This is the list of modules:

- **AAAida:** It is the central module that receives all the request of access and data treatment. His function is to redirect each request to its correct destination: the *Manager* and the *Web Server.*

  - **Manager:** The manager module is the responsible of the communication between the external service of *Google Health* [18], and the whole *AAAida* platform. The manager offers an *API* which connects via web service both the *Web* client and third-party applications such as mobile application.

  - **Web Server:** It works as application server which simply generate Web platform and display the data demanded by the user in a web page.

The platform also uses external services:

- **Google Health:** This component is the responsible of storing the measures and data of the user. It has its own *APIs* in order to be used.

- **Google APIs:** These are a group of *APIs* needed for the mobile application to work correctly. The needed *APIs* are:

    - *Maps API:* This *API* is used in the *Route tracking*.
    - *Search API:* This *API* is needed for *POIs* searching.

As can be seen in *Fig. 3.1*, the mobile application uses the external service *Google APIs* [19] in order to obtain suitable information about geographical information from *Maps API*, and what surround the user like *POIs* using *Search API*.

The following sections show the architecture for the platform, the mobile application platform and its design.

3.1. Platform architecture

This section shows the whole architecture of the project and the different components that make up the mobile client.

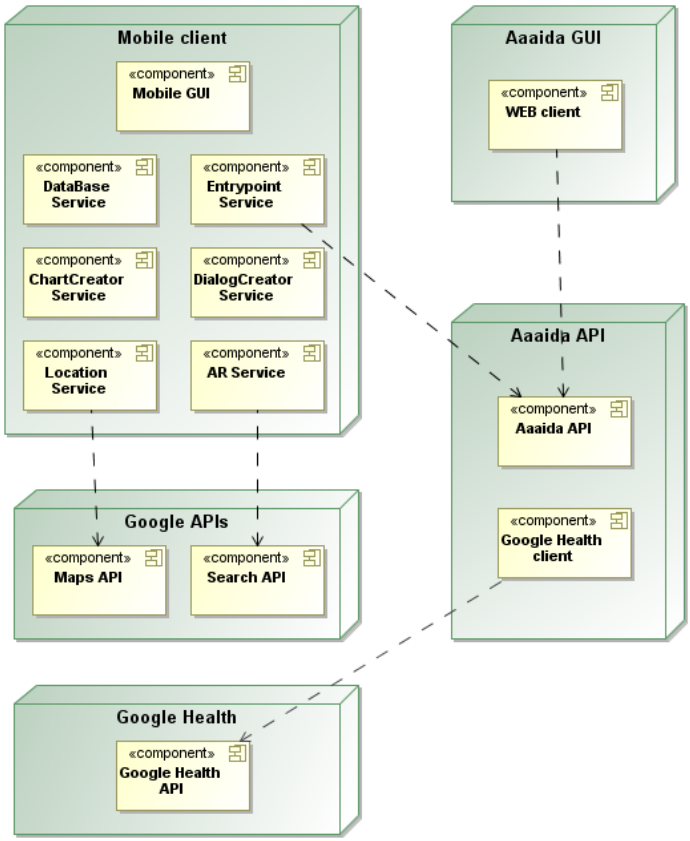


Fig. 3.2 Platform architecture

As can be seen in the *Fig. 3.2*, the architecture is made up by different modules that have a relation each other, the *AAAida Manager*, which is de mandatory module for the whole project interaction. The exception can be seen in the interaction between the *Location* and *AR* services with the *Google APIs*.

Focusing in the mobile client, this module consists in the following components:

- **Mobile GUI:** This component is the needed for the presentation in the mobile application, displays information and interact with the user.

- **Database Service:** The component that keeps the user information like the path routes and tracking details. Also offers the possibility of store login information.

- **Entrypoint Service:** It sends and receives information from the *AAAida API* and therefore the possibility to access user data stored in *Google Health*.

- **ChartCreator Service:** This component calls a tool that creates a chart with the given information.

- **DialogCreator Service:** This is the central dialog creator that generates and displays all the used dialogs in the application.

- **Location Service:** This component interacts with the *Maps API* offered by *Google*, and offers information like real time position, geolocation, etc.

- **AR Service:** This is the last component of the mobile client. This interacts with the *Search API* offered by *Google* in order to obtain *POIs* information and show these in the *Mobile GUI*.

The *Mobile GUI* allows to the user to perform the most important actions of the project. These have already been outlined in *CHAPTER 2*, for example:

- Send and receive information that is useful to perform login, save user information and display previously saved information.

- Locate user in the street in real time, showing in a map the location. Also shows routing details and a map with previous tracks.

- Showing in the caption camera the important *POIs* that surrounds user at each times.

3.1.1. Mobile client architecture

In this section the architecture for the mobile client is showed. This is a zoom from the *Fig. 3.2* focused on the *Mobile GUI* and his interaction with other mobile client components.

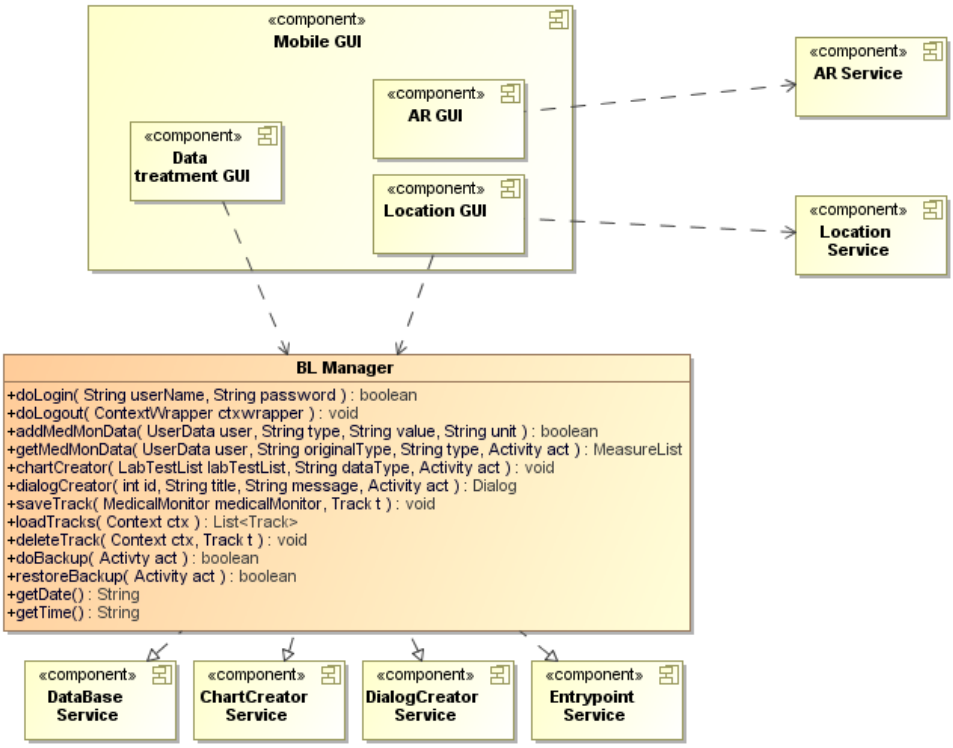


Fig. 3.3 Architecture of Mobile client

As it can be observed in the *Fig. 3.3*, the *Mobile GUI* is fragmented in different components that link the user interface with the *business logic Manager*, the mandatory components for the mobile localization tracking and the augmented reality.

The *business logic Manager* is the component that allows the communication with the *Service* layer components, it is like an internal *API* that simplifies the use for the *GUI* components of the *Database Service*, *Entrypoint Service* and the *ChartCreator Service*.

In the following sections the different components are shown and explained.

3.1.1.1. Data treatment GUI

This component is the mandatory for the general data handling both to show and collect data. All users' requests go through this component.

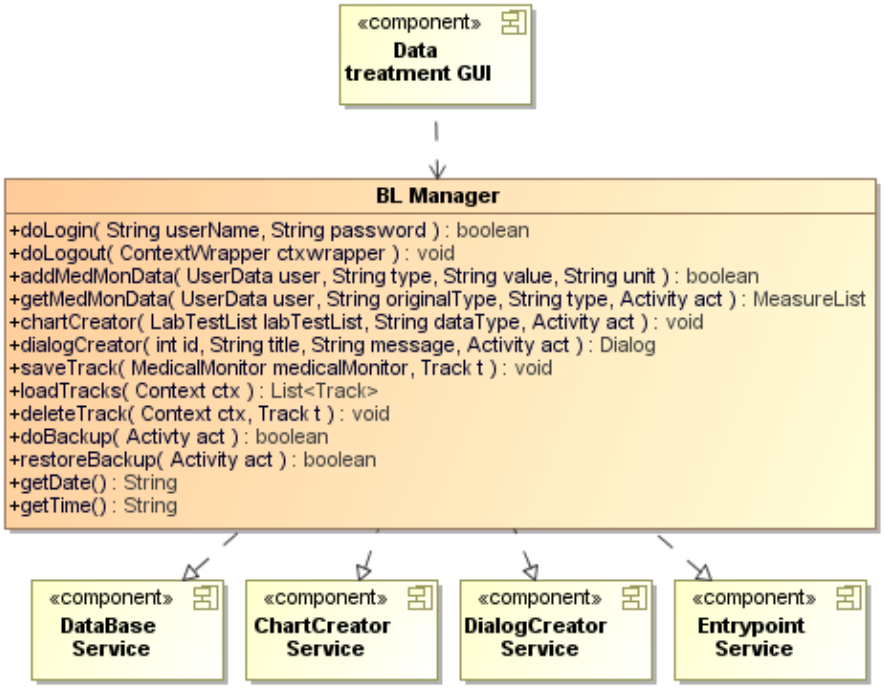


Fig. 3.4 Data treatment component

The *Fig. 3.4* shows the relations between the *Data treatment GUI* and the components for the database and communication with the *AAAida Manager*.

The Data treatment offers different options to users:

- Login
- Logout
- Send measures data
- Receive data for a specific measure
- Create charts with the received data
- Get present time and date
- Do backup and restore this for saved tracks

The *Database Service* component offers the possibility of accessing to Database store and recover login information and previous trackings.

The other component that is used through the *BL Manager* is the *Entrypoint Service,* which is responsible for all connections with the *AAAida Manager.* The next section explains the use and call backs that are done to the *AAAida API*.

3.1.1.2. AR GUI

This component offers to users the augmented reality tool. This is integrated on the application and can be used by calling the *AR Service*.

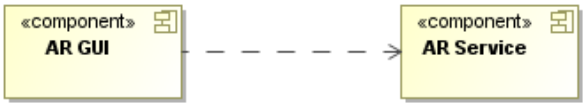


Fig. 3.5 AR component

The *AR GUI* only represents the places that are processed by the *AR Service*.

First of all, the *AR Service* needs the current device localization. This localization is obtained through the internal hardware, thanks to the *GPS*; by means of internal calls in the *AR Service* and through the mobile *OS*, the exact coordinated are stored.

Next, with the stored coordinates, the *AR Service* will search specific *POIs* through the *Search API*, processing the received data and showing the places over the *AR GUI.*

3.1.1.3. Location GUI

This is the last component, mandatory for represent and inform user localization changes in real time.

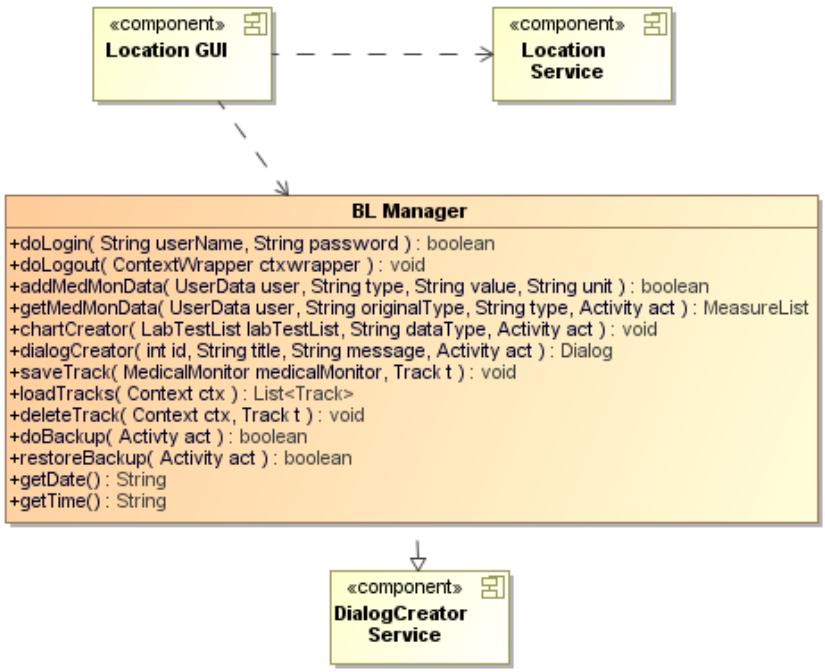


Fig. 3.6 Location component

In the *Fig. 3.6* can be seen the relation between the location front-end, the *Location Service* and the *business logic Manager*.

The most information comes from the *Location Service*, which is necessary to bring to user the location and tracking information. This information can be *e.g.* the present position of the device that could be retrieved in geographic coordinates or give the inverse geolocation such the street name.

To obtain the geographic coordinates is needed the internal mobile hardware such as the *GPS*, to reach the better accuracy in less time the *3G* connection can be used too; this process is automatic and does not requires user intervention. However, the inverse geolocation needs external services like the *Maps API*, and this need the *3G* connection.

3.1.2. AAAida API

This is one of the most important platform components, is the enter point and processor data localized in the manager. Through it the *Google Health* stored data can be accessed or upload new values. As was explained in other chapters, the communications are made through web services.

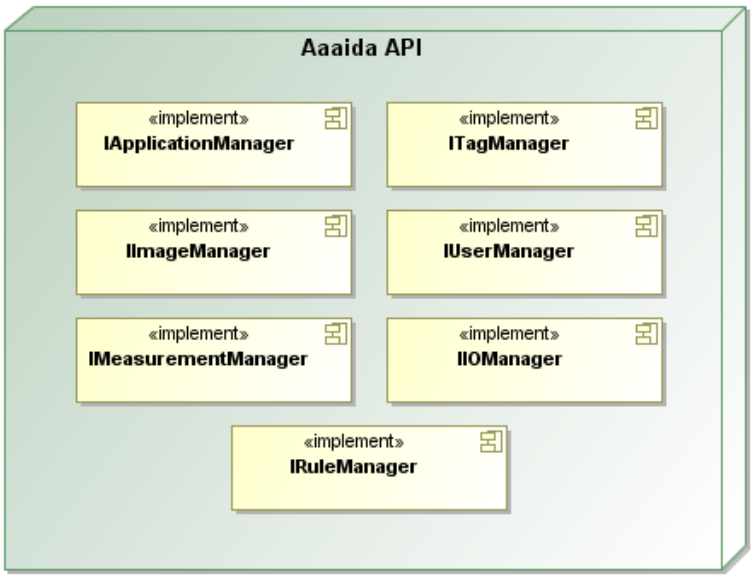


Fig. 3.7 Components of AAAida API

The *AAAida API* offers a variety of methods which is like a gateway between the *Google Health* client and the different *AAAida* clients, such as the mobile application, the web client and third parties applications.

It is needed to say that the mobile application client only needs to use the implementations of *IMeasurementManager* and *IUserManager*. These two *API* implementations offer:

- User creation
- Bond creation
- Bond sharing
- User data request using a user name and password, needed for login
- Request of a specific measure
- Request all the profile measures
- Value upload for a measure
- Creation of a new measure

The following flow diagram shows the use case about the upload of values and the request of a measure values to be shown in a chart. This figure can be viewed bigger in the *Annex III*.

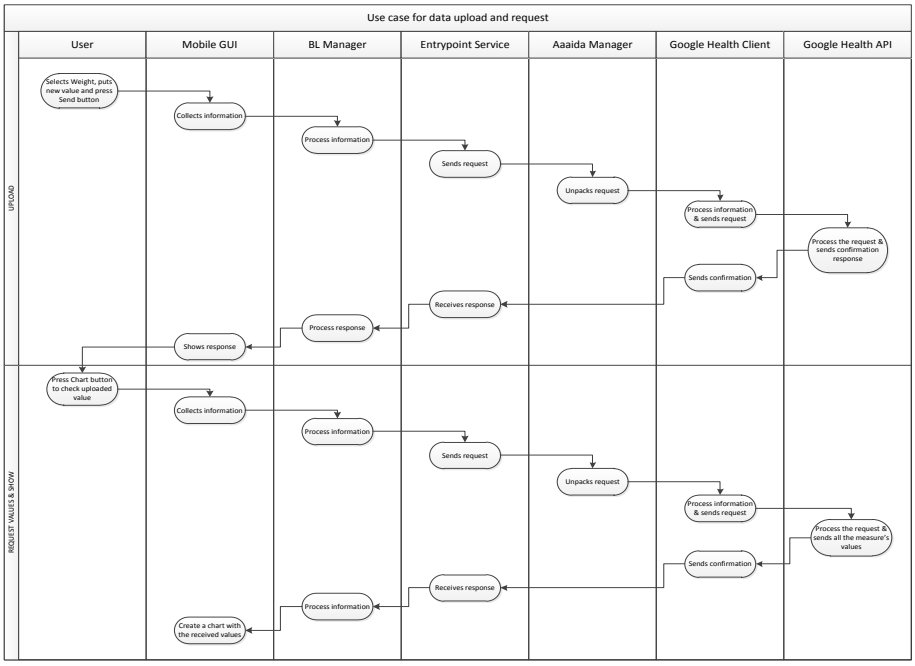


Fig. 3.8 Flow diagram for upload/download

As can be seen in the *Fig. 3.8* both values upload and request information must go through different processes or jumps. In each of them completes different tasks that are in the line with the component that performs, such as the *Entrypoint* component, which is responsible of send requests and receive responses.

Furthermore, in all the processes the user interaction is low. That is because the use cases are simple enough to do, so they require a minimum knowledge in a user-friendly frontend.

In this example, the user only need to choose the wished measure, put the new value and wait for a response from the server; as well in the chart request for a measure. Behind these requests there are several movements between the

different components of the platform, highlighting that the *Google Health* is the alien service in this project. The vital elements are the *API* interface of *AAAida* and the web services that performs the sending parameters between the user's device and the server.

All the user requests go through the *Mobile GUI*, which will be collected and will be sent via web services. The responses are displayed by *Mobile GUI*, too.

3.2. Use cases

The following sections will explain the use cases for the mobile application that show the interaction between the components described above.

The uses cases are the system access, data treatment, route tracking and augmented reality.

3.2.1. System access

Before the user can use the clients, he needs to use the system access functionality.

The platform access needs a verified email account to perform a correct login. Accepting the *Google Health* terms of service can do this verification. Once the user has a verified email, it can be used in the login progress. *Google Health* is used like an external database system.

The *AAAida Manager* is the responsible of connect the client with the database *Google Health*. The manager processes the clients request and adapts this in order to send it to *Google Health* via his own API.

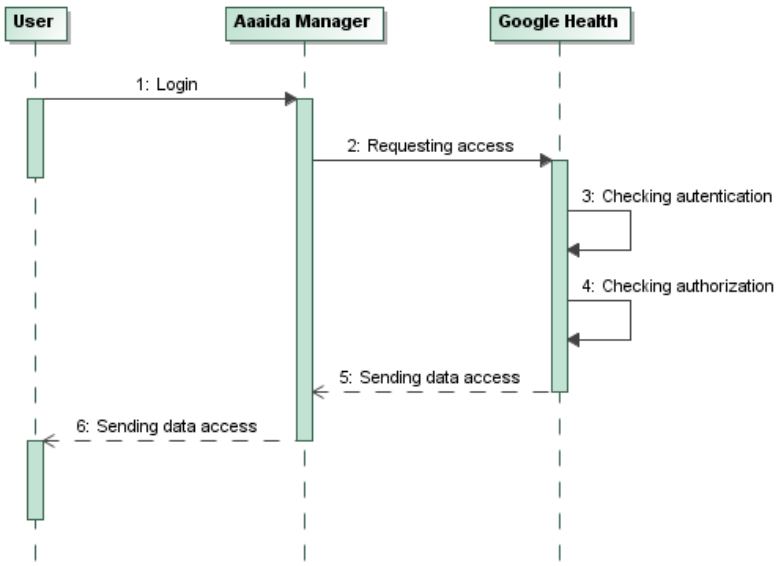


Fig. 3.9 System access flow messages

In the *Fig. 3.9*, when a user wants to use platform, through mobile client or through the *Web client*, he needs to send his authentication information. This authentication information is the couple values formed by his email and the password associated to this email.

Once this information is sent to *AAAida Manager,* the manager resends this with the *Google Health* client component to the *Google*'s tool. In that step, *Google* process the request and check if user can be authenticated and authorized with the given information.

With the granted access, *Google Health* sends the data needed to access to the database service and the manager resend it to the client.

3.2.2. Data treatment

The data treatment is the functionality of the platform needed to upload and download data from the system store.

This process is a black box for the user, no need to know how it works to use properly the platform. The entire data upload, as much like download, is processed in the *AAAida manager*, which is the mandated module.

3.2.2.1. Data upload

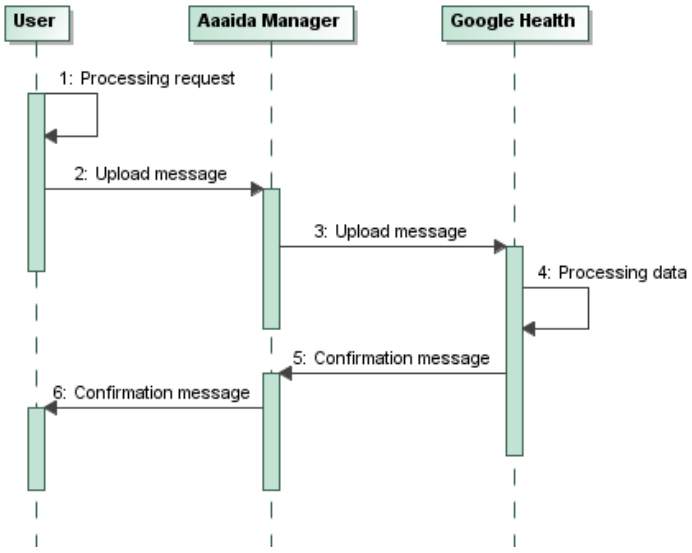


Fig. 3.10 Data upload flow messages

The process of upload data to store is represented in *Fig. 3.10*. The client collects the data given by the user, process it and send it to *AAAida Manager*. The message need the data access obtained in the *System access* process to grant access to the external store, the *Google Health* service.

With the data in the manager, it is repacked and sent it through the *Google Health* client. Once the data arrives to the *Google*'s tool, it is processed and stored.

To end the process, *Google Health* sends a confirmation message of the successful or unsuccessful of the storing process.

3.2.2.2. Data download

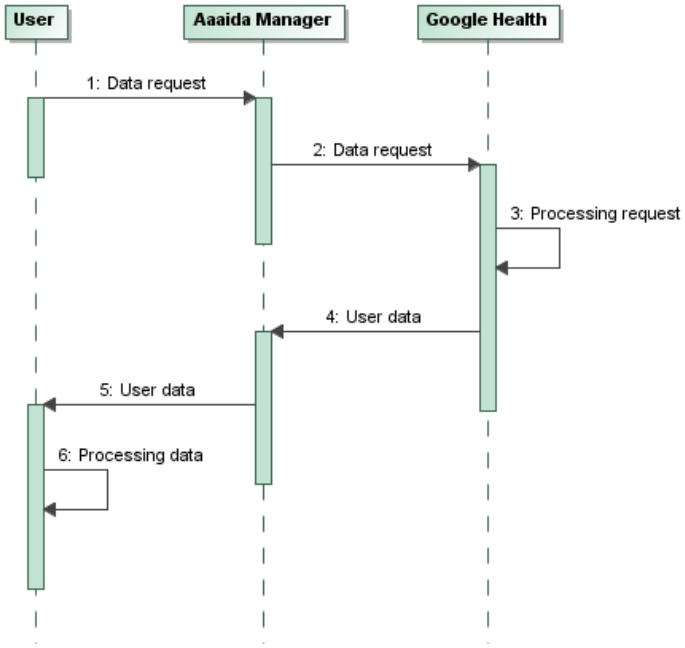


Fig. 3.11 Data download flow messages

The *Fig. 3.11* shows the process of data download from *Google Health* to the client that starts the process. The client makes the data request and it is resent from manager to *Google'*s tool.

Once the request arrives to the external service, it is processed and *Google Health* generates a new message for response that contains the requested information.

When the client obtains the requested information, it is processed and shown through the web client or mobile application.

3.2.3. Route tracking

This functionality is only available in the mobile platform. Once the user is logged in the system, using the internal locations tools of the mobile platform, the user can saved the routes. The application receives information from *GPS,*

the geographical coordinates at each time. This data is processed to obtain the geographic location to be saved and be sent to the external service that processes the *GPS* data, which is *Google API*. At the end of the route, the user can upload brief information about the route and review more details about this.

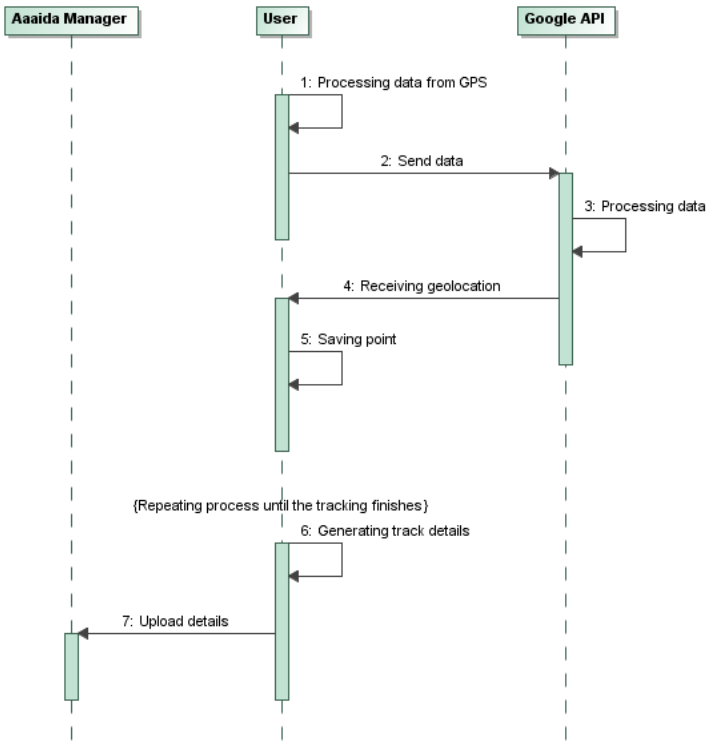


Fig. 3.12 Route tracking flow messages

The *Fig. 3.12* shows the route tracking process. The user process the obtained *GPS* data and sends it to *Google API* where is processed. The *Google API* create a response with the processed information and send it to the client where is saved. This process is repeated until the user stops the tracking tool.

Once the tracking process is over, the mobile client generates information about the route and uploads it to *AAAida Manager.* The upload process is explained in *Data upload* subsection.

3.2.4. Augmented reality

Like the route tracking, is only available in the mobile. With this functionality the user gets his localization and receives important *Points of Interest* (*POI*) that surround him. These *POI*s are requested to *Google API*. The information is displayed in the screen superposed in the camera caption.

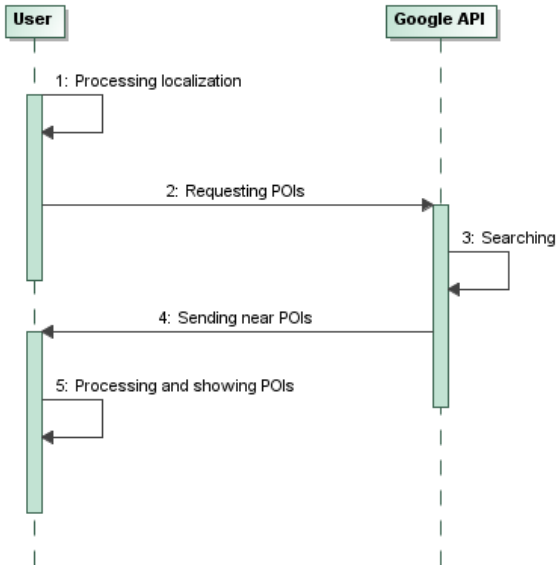


Fig. 3.13 Augmented reality flow messages

As can be seen in the *Fig. 3.13*, this process is similar to the *Route tracking* process, but the location information is obtained once. With the geographic position, client sends it with the name of the places that maybe surround user.

The *Google API* do search with the given information and in the response message, it sends a list of these *Points of Interest* that are near to the user's device.

Finally, the client receives the response with the list of points. This list is processed and displayed to user.

3.3. Mobile client design

This section explains the followed design to develop the different components that are related to the mobile client.

The following subsections and figures are a zoom from a previous figure located in *Mobile client architecture* section.

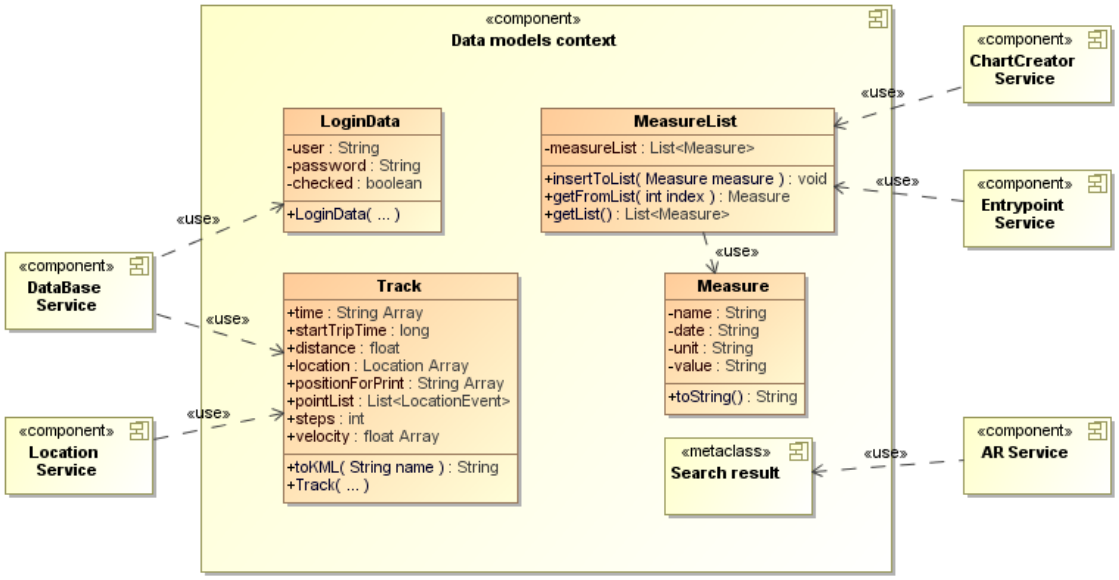


Fig. 3.14 Design for the mobile client application

The *Fig. 3.14* shows the design of the data model context. These models are needed to interact with the different application components between them. The *GUI* layer interacts with the data obtained from the *Service* layer, this layer use the models showed in the figure.

The followed design for the mobile application is in layers. With this type of design, similar to the web model *MVC* [20], the application can be developed and improved more easily.

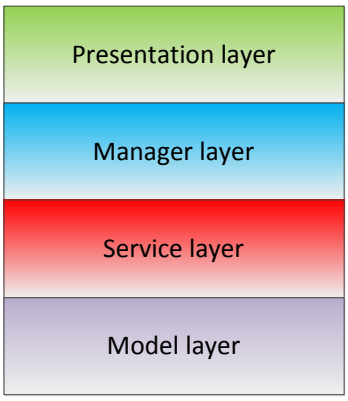


Fig. 3.15 Application architecture layer

The layers that can be seen in the *Fig. 3.15* are related to:

- *Presentation layer* → Mobile GUI
- *Manager layer* → Business logic Manager
- *Service layer* → The group of services
- *Model layer* → Data model context

3.3.1. Service layer components

In this section the different components design from *Service* layer are explained. As is said before, this layer interacts with the external services like *AAAida API* and *Google APIs*.

3.3.1.1. Database Service

The *Database Service* is the responsible of store and recovery of local data like login information and the route tracks. This service allows to application keep information that is needed in future sessions.

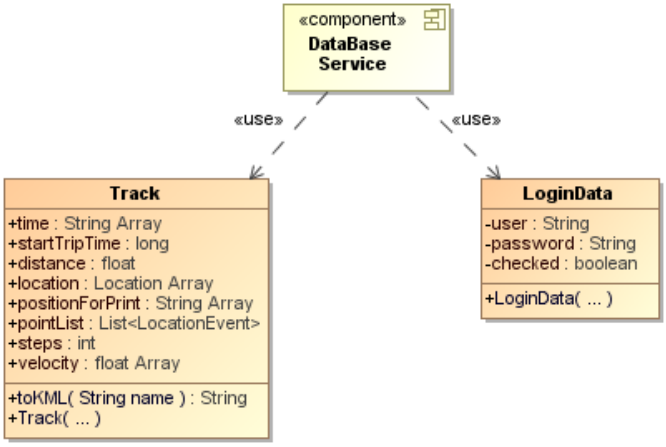


Fig. 3.16 Database service component

The *Fig. 3.16* shows the models needed for *Database Service* correct work.

On one hand, the *LoginData* model, as his name indicates, keeps the information related to the login process. This information is needed when the application is reopened and does auto login.

The *LoginData* model contains the following attributes:

- **user:** The user name.
- **password:** The password for the user.
- **checked:** Boolean that indicates if the remember login box was checked or not to do auto login in other sessions.

On the other hand, the *Track* model has the information related to one route tracking that is saved on the model at the end of a route. When is required the application can restore a *List* of *Track*, one model each route.

The *Track* model contains the following attributes:

- **time:** This is a *String* array that keeps the start time, the trip time and the finish time
- **startTripTime:** This is the same start time in the *time* attribute but here is kept as a *long* type.
- **distance:** The distance between the start point and the end point in *km*
- **location:** This attribute is an array that keeps two *Location* objects, the first and the current location given by the *GPS*. It is used to estimate the distance between locations.
- **positionForPrint:** A *String* array used for print in the screen the start position and the current position. The positions are saved in geographical coordinates.
- **pointList:** This is the *List* that keeps every point gave by the *GPS* hardware. The list is used at the end of route to print each one on a map to show the route.
- **steps:** The number of steps done on a route, if the route was by walk.
- **velocity:** A *String* array that keeps the velocity in *km/h* and, if the route was by walk, the *steps/minute*.

The model is prepared to give all the information in *KML* [21] data type used in *Google Earth* [22] for example.

3.3.1.2. Entrypoint Service

This component is needed to connect the application with the *AAAida API*. It is used for incoming and outgoing data transfer.

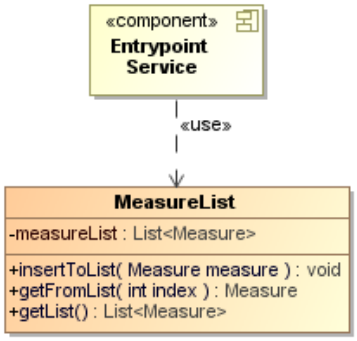


Fig. 3.17 Entrypoint service component

All the transferred information with the *AAAida API* is the measures of the user; it is showed in the *Fig. 3.17*. The *MeasureList* model is used to transport the data received by the service to higher services that need it.

The *MeasureList* model contains:

- **measureList:** This is a *List* of another model that can be seen in the next subsection *ChartCreator Service*

3.3.1.3. ChartCreator Service

The *ChartCreator Service* is used to generate charts in order to show the data requested by the user.

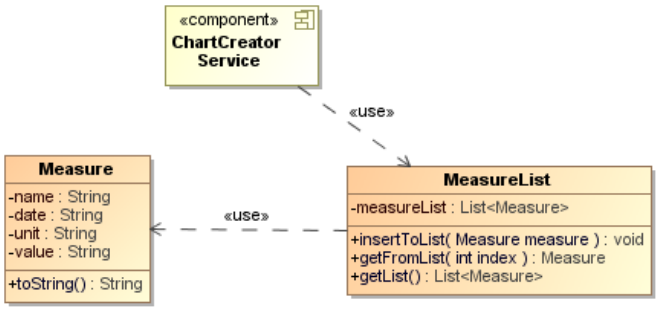


Fig. 3.18 ChartCreator service component

The *Fig. 3.18* shows the model used by the *ChartCreator Service*. These models are the *MeasureList*, explained in *Entrypoint Service* subsection, and the *Measure* model. This model is inside the *MeasureList* in a *List*.

The *Measure* model contains:

- **name:** The name of the kept measure.
- **date:** The saved date for the measure.
- **unit:** The measures need a unit to be represented; it is kept on this attribute.
- **value:** The value for the measure.

3.3.1.4. Location Service

All what is related to location is done by this service. This service prepared and enables the *GPS* hardware to obtain the current device position, used for device location on a map and tracking.

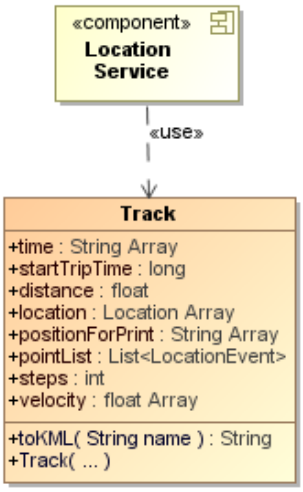


Fig. 3.19 Location service component

In the *Fig. 3.19* can be seen that the *Location Service* needs the *Track* model. This model is explained in the *Database Service* subsection.

3.3.1.5. AR Service

The last component is used to provide augmented reality in the application. As is said in previous sections, the *AR Services* shows points over the screen that maybe can help user.

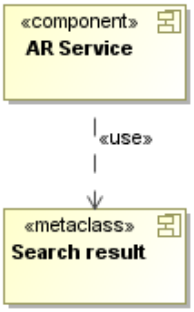


Fig. 3.20 AR service component

The *AR Service* do not have a model as well, it can provide the points with the information obtained from the *Google Search API.*

The information received from the *Search API* offers information such as the street address, phone number, geographic location, etc.

CHAPTER 4. IMPLEMENTATION

In this chapter the different aspects related with the implementation are explained. The implementation takes into account the performed design in *CHAPTER 3*.

First of all, the work environment needs to be defined with the technology and the used tools.

The followed steps and the done implementation are explained in the following sections.

4.1. Work environment

During the project develop several machines had been used. A computer for develop provided by *UPC*, a server where *AAAida* platform was located and a mobile device for testing and improving the application.

Moreover, two operating system had been used, a *GNU/Linux Debian* for server and *Windows 7* in the develop machine. It must be said that the development of the application could have used any type of operating system because the development environment that is multiplatform.

In the server *AAAida* platform an *Apache Tomcat* [23] is being used, which offers the possibility of deploying a web platform and web service execution that are needed for the correctly work of platform. The server works over the *Java SUN* [24] virtual machine, version 1.6.

Inside the *AAAida* platform, the *Web* application is developed in *Apache Wicket* [25] version 1.4. The *Web* services have been developed with *Jersey Rest* [26] version 1.4,

As develop environment the *Eclipse 3.5* [27] has been used join with the *Android* [28] develop platform, version 11 for tools and *SDK* [29] and version 10 for the *Android API* (*Android 2.3.3*).

Furthermore, for *Master Thesis* wrote the *Microsoft Office 2010* platform and Microsoft Office 2011 Mac version was used. For the *UML* [30] and *diagrams* have been used *MagicDraw 16.7* [31] and *Microsoft Visio 2010* respectively.

Finally, a server prepared for version control has been used. The sever implements *SVN* [32] technology that offers the project develop storing with a version control. In addition, this type of service ensures the work using the uploaded versions as backups, also the possibility of downloading the source code in any machine in any place.

4.2. Tools & technologies

This section shows the tools and technologies that have been used during the development of this *Master Thesis*.

- **Java JRE 1.6:** The *Java* virtual machine, needed to execute the develop environment such as *Eclipse* and *Android SDK*.
- **Jersey WS REST 1.4:** Environment needed to develop server side using *REST* technology.
- **Android SDK 11 & API 10:** The *SDK* version 11 offers tools to develop in *Android* over the *Eclipse* platform for example. The tools are the *Android* virtual machine for local testing, the *DDMS* [33] environment to control device (real and virtual) and a graphical layout developer.
- **DB4O 8.0.184:** An open source object database. It is multiplatform, so can be used in *Java* virtual machines that can be used on *Android* platform too.
- **AChartEngine 0.7:** That *AChartEngine* [34] tool offers an easy way to construct multitouch charts developer over the *Android* platform.
- **Magnitude r2:** *Magnitude* [35] is an open source augmented reality tool that is plugin oriented, is said, can easily integrate objects over the tool that can be showed over the device screen. Based on *ARKit* [36].
- **JSON:** This *JSON* [37] technology is used that facilitates data exchange between client/server and vice versa. Transforms *Java* objects in comprehensible text plain.

4.3. Develop technologies election

This section shows the election of mobile platform developing and the election of a database for the Android application.

4.3.1. Mobile platform

Nowadays there are different mobile platforms in the mobile industry. All are internet oriented and make easy for users the use of these platforms.

There is no platform that is more correct to use, it depends on the approach that you want to give the application or if you prefer to reach more people or have more profits.

The following are the platforms that excel over other platforms:

- **iOS:** Developed by *Apple*, included only on his own mobile devices such as *iPhone, iPod Touch* and *iPad*.

- **Android:** Developed initially by *Android Inc.* and improved later by *Google* when the mentioned company was taken over.

- **Windows Phone:** The youngest platform which *Microsoft* wants to make up lost ground in the industry for its previous platform.

More detailed explication about these platforms can be found in *Annex IV.*

The *Android* election comes with the easy way for develop over his *SDK*, thanks to that uses a high level programming, over *Java*. Moreover, no develop license is required to test the developed code over a device with *Android*, is needed only the *SDK* and correct drivers for device. *Windows Phone* could have been a good alternative, but when the election was made, this platform did not have any device on market.

It is needed to say that the application could be ported to other mentioned platforms without serious problems, since it has a separation of layers similar to which is used in the *Web* world, the *MVC* as can be seen in the *Fig. 3.15*. It would need only transcribe the models and controllers (*Services*) to proper code and *SDK* and implement it over a new *GUI*.

4.3.2. Local database

For the store propose in the application, is needed a flexible and agile database. There are two kinds of data that is needed to be stored in order to be recovered in future sessions, the *route tracking data* and the *login information data.*

4.3.2.1. Route tracking store

Android have inside an implementation of *SQLite* [38], which offers data storing and *SQL* searches. This is not the best solution when a huge amount of data needs to be stored, like in the route tracking tool.

An object database is needed to store the object like they are created, in a faster way and more efficient heading to a low *CPU* consumption. For this *Master Thesis* two object databases has been analysed because they can be used in *Android* developments, they are *Perst* [39] and *DB4O* [40].

These are non-relative databases that store objects to be recovered later indicating the *Class* object that wants to be recovered. As can be seen in the reference [41] the best solution is *Perst* taking into account its prove results, but at the end the database elected was *DB4O* because it allows to store object without the needed to use *Java Annotations* [42] like in *Perst*. This is an

advantage of *DB4O* in front of *Perst* due to it can store objects without source-code or object integrated in *Android*, like the *Location* object.

The advantage of object database against the *SQLite* is that the *SQL* database needs to serialize the objects to a byte array before they can be stored and recover the byte array and do the reverse process to obtain the stored object. With *DB4O* the object is stored as is, without any kind of conversion.

4.3.2.2. Login data store

As route tracking data, the login data information also needs to be stored, for this reason is used an internal tool of *Android* platform called *SharedPreferences*. This tool is used to keep values, such *String* or *Boolean*, and share between the different *Activities*. Moreover, it is used to store information in a *key/value* mode. To restore the stored information only is needed to indicate the *key* associated to each value.

4.3.3. Augmented reality tool

There are several tools to obtain a satisfactory augmented reality. The studied tools were *Layar* [43] and *Magnitude*, both supported for *Android.*

Layar is a great *AR* tool that offer good develop possibilities but is not open source unlike *Magnitude*, and could not be integrated in the *AAAida* mobile application.

The election here is simple, *Magnitude* can offer more flexibility in developing terms and is more easily adaptable for the *AAAida* mobile application objectives.

4.4. Platform implementation

In this section the implementation of the entire platform form a technologic point of view are shown in order to understand how it works.

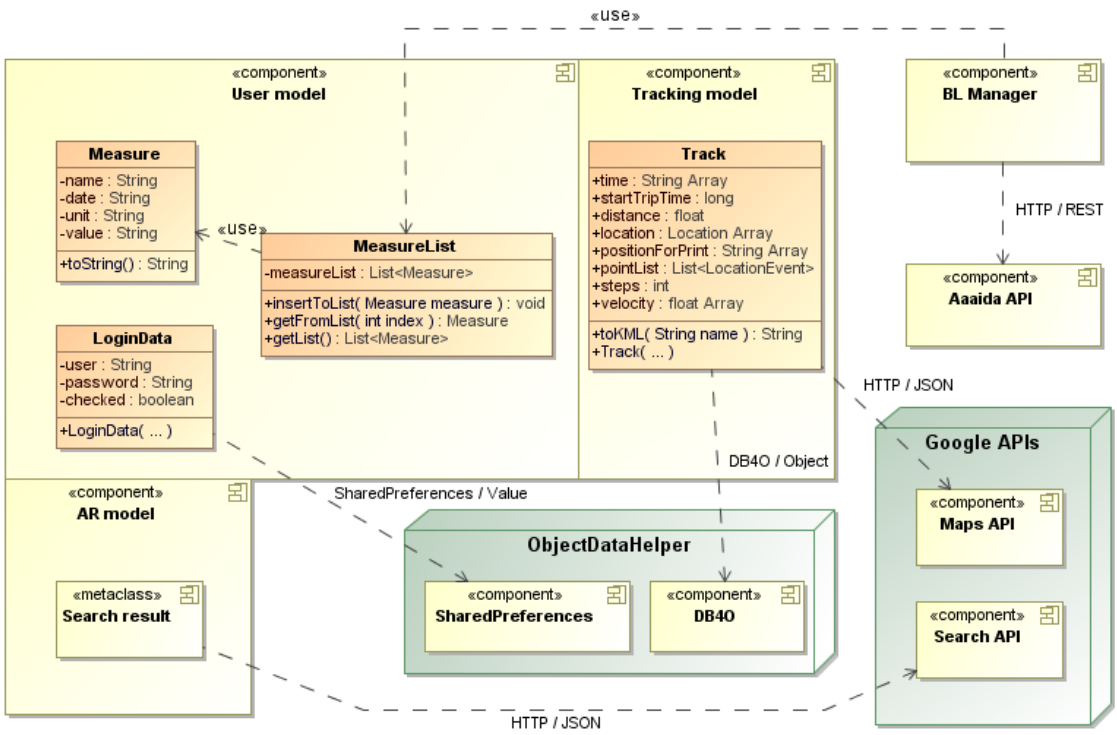


Fig. 4.1 Deploy diagram of the AAAida platform

This *Fig. 4.1* shows a deploy diagram with the current implementation state of the platform.

The different model has been separated in order to show how they are organized and which component uses them. Moreover is showed how the connection between components is done and the required technology in each connection.

On the one hand, inside the *User model* group, *Measure* is used internally by the *MeasureList* model and this is used by *BL Manager* to transport the measure data to *AAAida API* through a *HTTP* connection and using the web service technology *REST*. Furthermore, the obtained information about the login is collected with user agreement and stored with the help of *SharedPreferences* tool, located in *ObjectDataHelper*. Both are internal connection so is not required external links, only need store the values as is explained in *Login data store*.

On the other hand, in *Tracking Model*, the route information is contained in the *Track* model. This object is stored as such in the database *DB4O*, this tool is located in *ObjectDataHelper*. *DB4O* laces of process the information contained in the model and store it. In the recovery process, which is done once the application starts, the database is asked about *Track* model and the obtained result is a *List* of *Track*, is said, all the previous stored routes. The route list can be seen in *Fig. 4.21*. In order to complete the location elements in *Track* is needed a connection, done on *OS* level, with the *Google Maps API* as can be seen in the *Fig. 4.1*.

Finally, for the augmented reality tool there is not any model, there is metadata that is received in *JSON* mode through a *HTTP* connection. This connection is opened when the tool requests the selected *POIs* information to *Google Search API*, see the *Fig. 4.30*.

4.5. Mobile client implementation

The following section will show the result of the mobile client implementation. The explanations of the results are combined with screen captures of the mobile application, which would help to understand the following functionalities.

4.5.1. User data treatment

This section explains the use case of upload and chart measures. Also show the result of upload over the browser with the web client.

4.5.1.1. Login

Before we can upload, download or do other things with the application, the login process is needed.

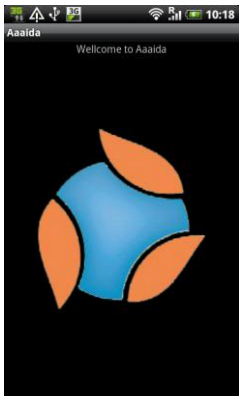


Fig. 4.2 Splash

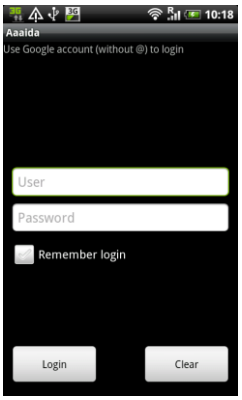


Fig. 4.3 Login view

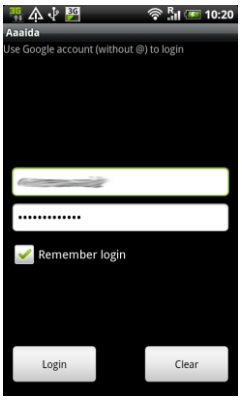


Fig. 4.4 Ready for login

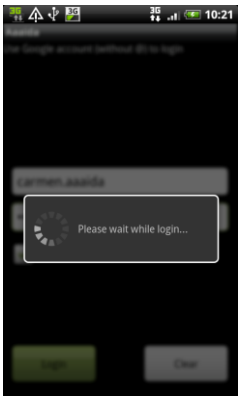


Fig. 4.5 Waiting response

The *Fig. 4.2* is the welcome splash of *AAAida* application, to reach the next screen just a touch over the splash is needed. In the next screen, the login information is requested, once completed and checked *Remember login* if is necessary. That process would block the application until the login is successful. In case of error the *Fig. 4.4* will be the next screen.

4.5.1.2. Upload & chart

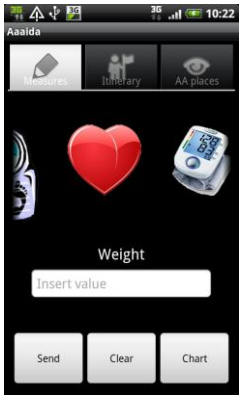


Fig. 4.6 Initial view

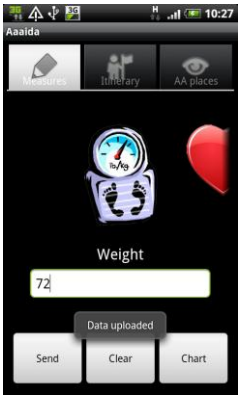


Fig. 4.7 Insert new value

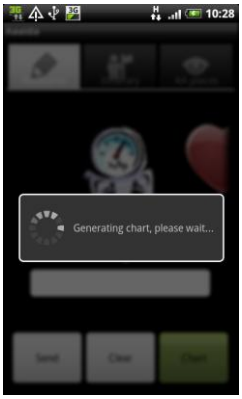


Fig. 4.8 Requesting chart

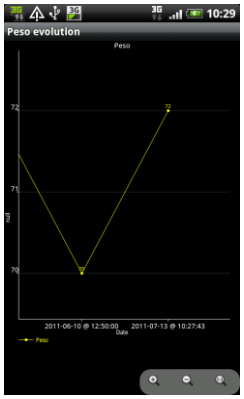


Fig. 4.9 Chart of the measure

Once the login process is completed the next screen in the *Fig. 4.6*, where measures values can be uploaded as can be seen in the *Fig. 4.7* or check previous uploads by pressing the *Chart* button, the application would show the *Fig. 4.8* indicating that is collection the information from the *AAAida* platform and finally the chart is made, see *Fig. 4.9*.

The uploaded information can be checked too on the *Web* client as shows the *Fig. 4.10*.

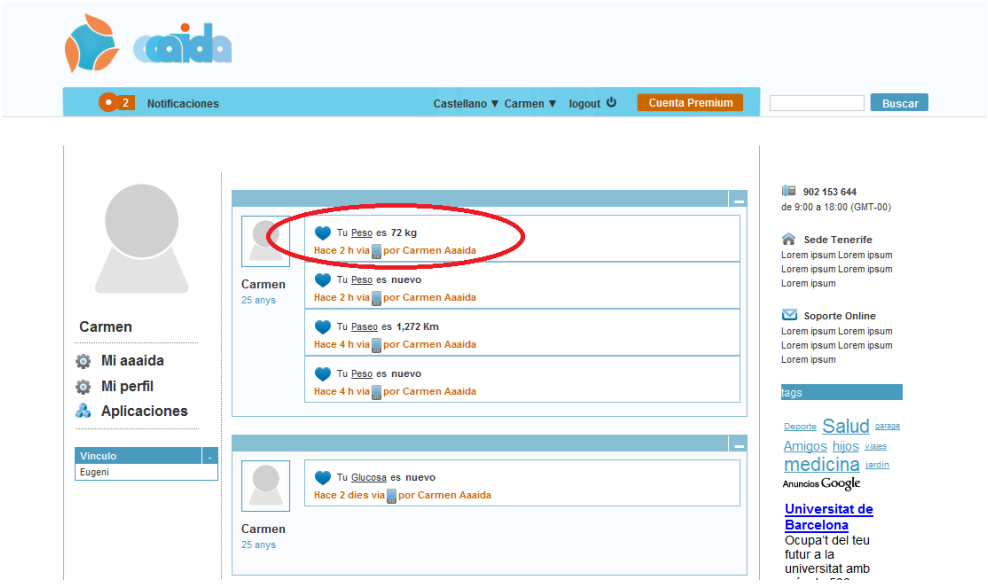


Fig. 4.10 Result of new value showed in the Web client

4.5.2. Tracking & location

This section explains the process followed to create new routes, see the result on the application map and, if is needed, get the device location.

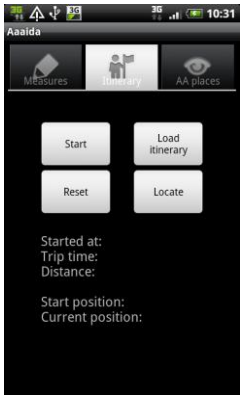


Fig. 4.11 Initial view for Itinerary

By pressing the *Itinerary* tab the reached screen is the *Fig. 4.11*, on this screen the tools of tracking and location are enabled.

4.5.2.1. Tracking

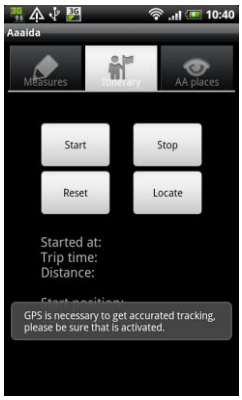


Fig. 4.12 Tracking started

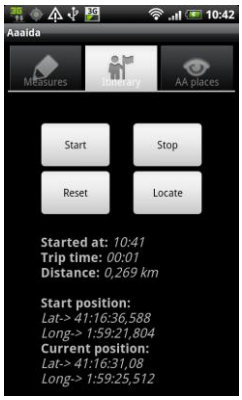


Fig. 4.13 Information of tracking

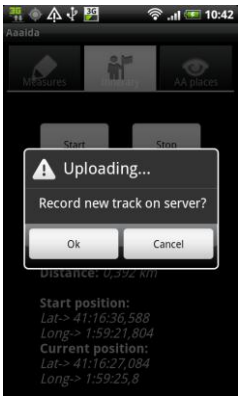


Fig. 4.14 Tracking stopped & upload confirmation

Once the *Start* button is pressed, the *Load Itinerary* button is replaced by *Stop*, see the difference between *Fig. 4.11* and *Fig. 4.12*. At this point the tracking tool is activated and starts to collect the necessary data from *GPS* and *Maps API*. Some information, such time and position, can be seen in the screen, as shows Fig. 4.13.

By pressing the *Stop* button, the application request confirmation to upload data to the *AAAida* platform, see *Fig. 4.14*. If is a positive confirmation, the uploaded value can be seen in the web client, see *Fig. 4.15*.

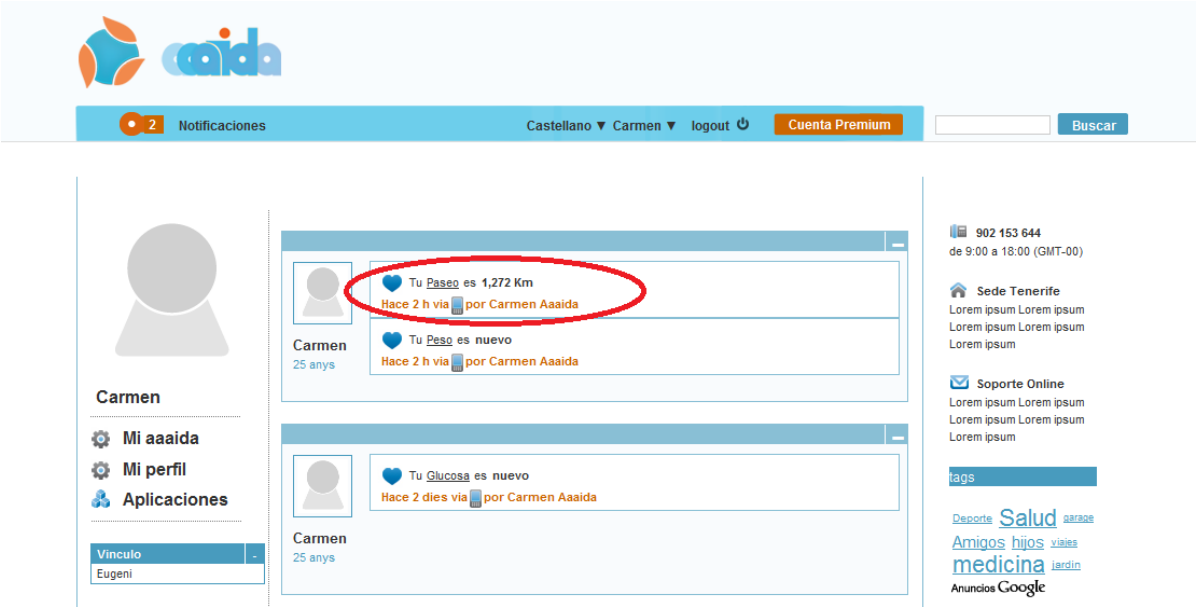


Fig. 4.15 Result of the tracking information upload

When the tracking is finished, the *Load itinerary* button is recovered.

4.5.2.2. Tracking map

When the *Load Itinerary* button is pressed, the application asks for load the last saved route on the application map.

Fig. 4.16 Load map

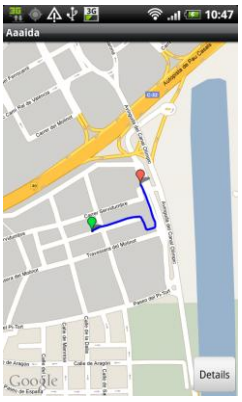


Fig. 4.17 Showing route

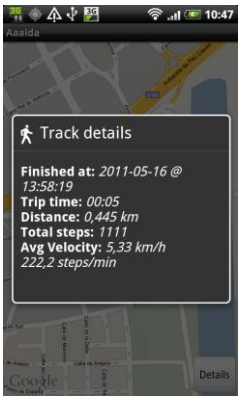


Fig. 4.18 Showing route details

Fig. 4.19 Showing in satellite view

As can be seen in the *Fig. 4.17*, the last route is loaded on the map. Pressing the *Details* button the additional information such as time, velocity, etc. appears (*Fig. 4.18*). The last screen represented by *Fig. 4.19*, shows the same map but

this time with the *Satellite View* layer activate, it would help to recognize faster the location.

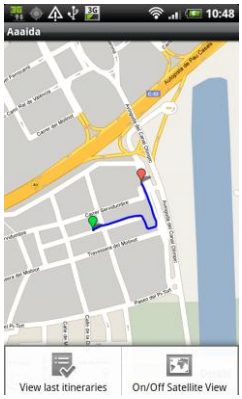


Fig. 4.20 Map menu

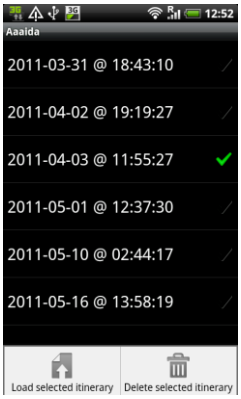


Fig. 4.21 Route list

Fig. 4.22 New route loaded

Fig. 4.23 Street View on start point

When the mobile menu button is pressed, a menu appears over the screen, see *Fig. 4.20*. This menu allows activate the seen *Satellite View* or load more routes as is showed on *Fig. 4.21*. This last view shows the route list, by pressing one route a new appears, to load the selected route or to delete it from the database.

Once the route is loaded, it can be seen in the same map. Other functionality of the *Itinerary Map* viewer is that both start and stop points can been loaded with the *StreetView* tool of Google, it could help to remember the place where were saved these points.

The *Fig. 4.22* and *Fig. 4.23* are referred to a real test of this functionality. The test was done in the *Cursa del Corte Inglés* [44]. The *Fig. 4.23* shows in the *Street View* mode, the start point, that is also the arrive point for this route, the *Plaça Catalunya (Barcelona)*. To see the route with zoom, check *Annex V*.

4.5.2.3. Location

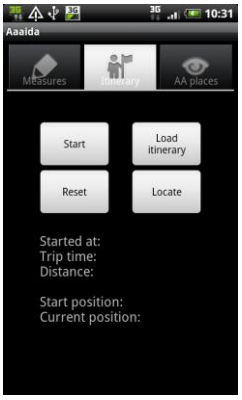


Fig. 4.24 Itinerary view

Fig. 4.25 Location activated

This tool can help to locate the device on a map, and in this way, help the user to know what streets surround him. To access to this tool is needed press the *Locate* button show on *Fig. 4.24*.

Through the *AR off* check button, in *Fig. 4.25*, the augmented reality tool can be activated, see *AR POI search* section.

4.5.2.4. *Backup tracking*

This tool can help to recover lost routes or to replicate the routes in another mobile.

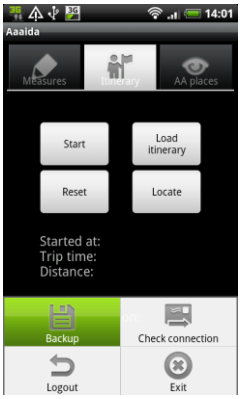


Fig. 4.26 Menu, backup selected

Fig. 4.27 Menu of backup

The menu selected in the *Fig. 4.26* shows how to reach the selection options for backup, in *Fig. 4.27*, that allows to make a backup of the data stored in the database into the *SD* or to restore previous backups from the *SD*.

4.5.3. AR POI search

This tool helps the user to find places such as pharmacy, police, etc. in case of emergency for example.

Fig. 4.28 AA places splash

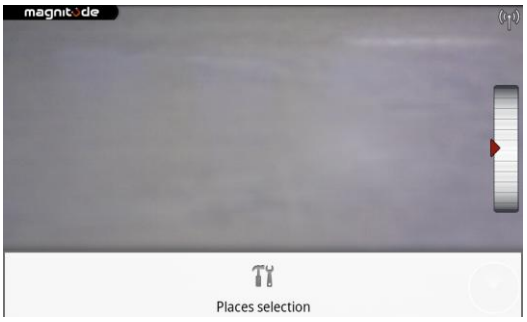


Fig. 4.29 Menu on Magnitude

By pressing *AA Places* tab the application shows a splash, *Fig. 4.28*, indicating that is needed to put mobile horizontally in order to activate the *AR* tool. Once is activated the application starts to find all the *POIs* by default, but it can be configured in the menu, showed in *Fig. 4.29*.

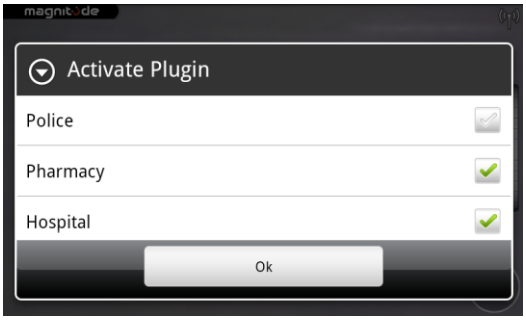


Fig. 4.30 Places selection

Once the wanted places are selected, as is showed in *Fig. 4.30*, the application restarts to request information to *Google Search API* as can be seen in the *Fig. 4.31*. Finally, the tool draws the *POIs* over the camera caption (*Fig. 4.32*)

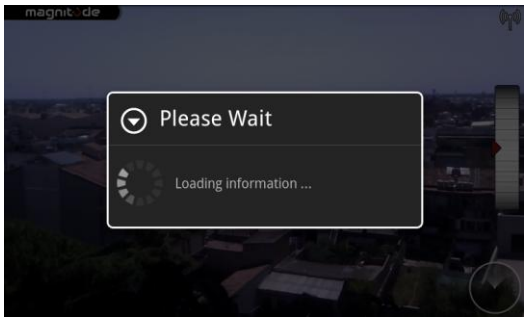


Fig. 4.31 Loading places

Fig. 4.32 Places already loaded & showed

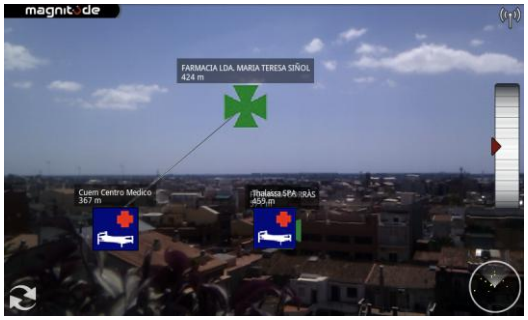


Fig. 4.33 Selecting a place

Fig. 4.34 Information about place

These *POIs* can be moved through the entire screen, it allows seeing the *POIs* which are hide rear another. By pressing once over *POI* the tool launches the *browser* with the information about the place obtained from *Google.*

CHAPTER 5. WORK PLANNING

This chapter shows the different performed tasks of the project with the goal of achieving the objectives that were set at the beginning and the dedicated time for each one. The following lines show a division in groups of the different performed tasks:

- **Previous study:** This task is the research of the State of Art that includes the searching of information, study of the different technologies that can be used and develop of small prototypes or applications to test different options and prove knowledge.
- **Design:** This group includes all the tasks referring to the design components, *UML* schemas, architecture definition and the mobile application.
- **Implementation:** These tasks are related with the software development having as a reference the previously designed schemas.
- **Testing:** Group of tasks devoted to commit tests in the software implementation and final tests.
- **Memory report:** Realization of documents and, then, this *Master Thesis.*

5.1. Dedication time

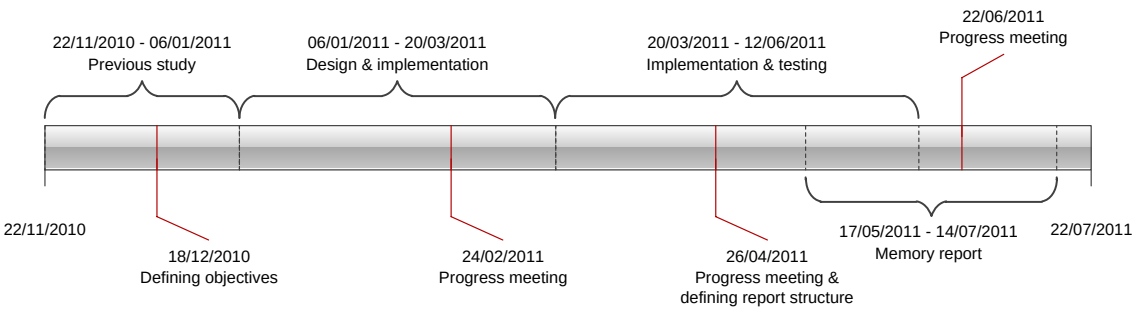


Fig. 5.1 Time line

As it can be observed in *Fig. 5.1*, the tasks have been developed sequentially. During the study of the State of the Art, several tests were committed with the different technologies that were used. Moreover, the project objectives were defined during this study phase of the project once it was clear which technologies can be used and fit with the project objectives. Some progress meetings were held during the project realization in order to determine the way of the project and give more ideas.

It can be seen that during the components design, the implementation of other components was also developed. These components were tested one by one. Finally the tests were committed between the components, and at the end, global integrated test over the mobile application was executed. For more detailed information see *Annex VI.*

5.2. Committed tasks

Table.5.1 Tasks developed & time

Task	Description	Time
Previous study	Seek of information, state of art and setting Project objectives	40 h
Previous study	Search of technologies and tests	50 h
Previous study	Set up the develop environment and installation of needed tools	10 h
Previous study	Starting tests over the *Android Emulator*	8 h
Previous study	Develop of *Android* applications to test *GUI* components and behaviour	20 h
Design	mock-up's of application *GUI*	8 h
Design	Design of components for internal *Activities* communication	14 h
Implementation	Develop of the first application *Activities*	8 h
Design & implementation	Design and develop of tools for connection with the *AAAida* platform	20 h
Implementation	Develop of an *XML* parser to process received information from *AAAida API*	4 h
Implementation	Develop of models and Login *GUI*	10 h
Design & implementation	Design and develop models for measures treatment	20 h
Implementation	Develop of Measure *GUI*	10 h
Design	Design model and components for location tool	15 h
Implementation	Develop of components for *AchartEngine* implementation (Chart tool)	8 h
Design & implementation	Design and develop route tracking components	30 h
Implementation	Implementation of location and routing	15 h
Implementation	Location *GUI* develop	8 h
Implementation	Route tracking *GUI* develop	8 h
Design & implementation	Design and develop *AR* components	10 h
Design & implementation	Design and develop component for sensors, sensors emulator	8 h
Implementation	Integration of *Magnitude* on *AR GUI*	10 h
Design & implementation	Design and develop of storing components (*DB4O* implementation)	20 h
Test	Testing location on real hardware	5 h
Implementation & test	Testing route tracking and improve of location tools	25 h
Test	Testing route tracking in real situation and over *3G* connection	10 h

Implementation	Implementation of backup tool for routes	8 h
Implementation	Remember login implementation	4 h
Test	*AR* tests	15 h
Implementation	Speed improvements on routeing tracking	8 h
Test	Global application testing	15 h
Implementation & test	General application behaviour improvements, adding icons and global test	10 h
Memory report	Realization of this Thesis	158 h

Total → 612 hours

The following chart shows the breakdown of hours by tasks.

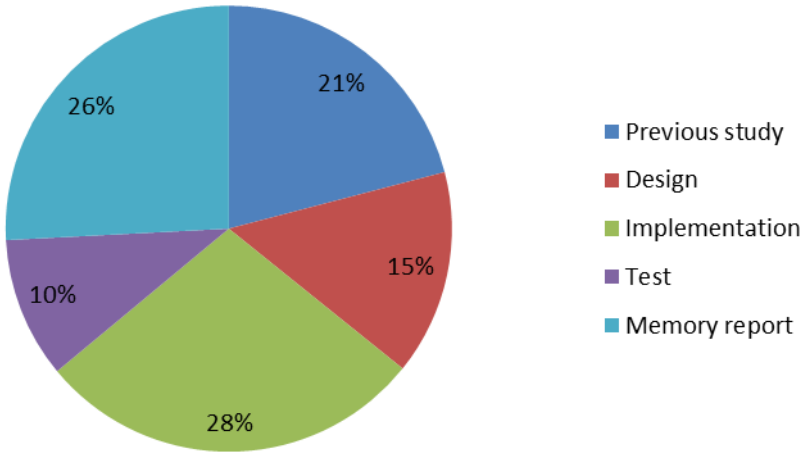


Fig. 5.2 Breakdown of project hours

The total amount of hours committed to this project is 612; as it can be seen, these are over the minimum set by *EETAC* that are 450 hours. This recommendation has been overtaken by the use of different technologies, which need a period of study, testing and the implementation in the project if were needed.

The implementation process of the project is the task that more hours consumed from the total amount of hours, because developing and testing the different use cases that join in a single application.

Included in the implementation phase, the design, implementation and test of the route tracking are the most expensive tasks in hours, without taking into account the previous study tasks. This task needed a mouldable design and code because of the constant changes that suffered and the tests made with every change. These changes were made to improve the relationship between precision and battery consumption. The more accurate route were, the more use of *GPS* and *CPU* time, which cause a substantial increase of battery

consumption [45]. The test always had to be performed outdoors, so it was necessary to upload the application changes on mobile and go out of the work environment.

5.3. Cost estimation

To estimate the cost, the following factor must be taking into account:

- **Dedication time:** As can be seen in the previous section Dedication time, to complete this project were needed 612 hours, approximately 4 hours/day in working days.

  Worked: 612 hours

- **Used tools and material:** This factor could be omitted because all the material used for this project were assumed by *UPC,* and these material is used for other projects too, and were bought before the start of these project, except the mobile phone used for tests. Other costs like electricity were assumed too by *UPC*.

  Inside this material that cannot be cost estimated are the workstation computer, *Windows 7 & Microsoft Office 2010* licenses, and server station. *MagicDraw* were used with demo-license.

  In the develop environment, Eclipse have the *Eclipse Public License* [46] and *Android* have *Apache 2.0* [47] and *GNU GPL 2* [48], both free use. *Magnitude* and *AChartEngine* tools are open source. *DB4O* have a *GPL* license.

  Material: 450€ for mobile phone

- **Personal:** Taking into account the cost per hour established by *UPC* for engineers in practices.

  Personal: One person, 8€/h

With these entire factors, a specific cost cannot be calculated, but approximately.

Total project cost: 5346€

CHAPTER 6. CONCLUSIONS

This chapter contains the conclusions obtained since the start of this *Master Thesis,* a brief set of future improvements for the application and the environmental impact that could have this project.

6.1. Objectives achieved

The main goal of this *Thesis* was to report, design and develop a mobile application that could access to *AAAida* through an *API* which could allow users to upload and check content from this platform. Furthermore, this tool would help user to locate him over a map in case of getting lost and mark over a camera caption the nearest emergency places. Finally, the users are allowed to record his routes by tracking his walks and give important information like time, velocity and steps.

This development has been accomplished successfully by achieving all the established objectives in the *CHAPTER 1*.

There are several improvements that may be interesting for future implementations and could give usefully to this application and could improve the *AAAida* platform. It can be read in the following section, *Future improvements*.

6.2. Future improvements

Like in other projects, at the end of this *Master Thesis* some improvements appeared. There was not more time to implement it on the project, thus these improvements are explained in this section.

- Show consumed calories at the end of tracking, such as time, velocity, etc. and upload data to *AAAida* as a new measure.

- Accessibility for old people in the measure values upload. Make easier the way to choose the value and upload it.

- Show velocity by sections
    - Different velocity, different colour for drew path.

- Allow mobile to create new measures like in the Web client
    - For this improvement, the *AAAida API* is prepared

- Allow *AR* tool to:
    - Create and show personal *POIs*

        - For example *"My home"* and save a photo of the place for *POI* information
    - Search places introduced by user

- Functionality like *FourSquare* application for the *Location* tool
    - Share location in *AAAida*

- Integration in digital frames for dependant people

6.3. Environmental study

The environmental impact is very important in a design project nowadays. The environmental viability is taken into account for the start up or rejection of new projects.

This kind of projects is difficult to study and extract the impact over the environment. There is not much profit for the environment if the servers are on 24 hours of the day online or recharge continuously the mobile due the intense use during the day.

But not all are drawbacks, in the use case applied to health medicine, in a doctor/patient relationship; the doctor would be updated in real time about his patient state. This patient lives outside the city, in other town. With the application the patient only needs to update his state and allow to doctor see the changes. With this interaction between doctor and his patient, the patient do not need to take public or private transport in order to go for a visit, is can CO_2. If this concept is applied with all the doctors and their patients, it would end in a decrease of travelling in relation to medical problems. Therefore, less travels, fewer queues and the wait rooms in the emergency centres less crowed, that is more space for people who need more cares.

6.4. Personal conclusions

The realization of this project has been very interesting and enriching, being a good conclusion for my *Master* studies. This topic has been a very successful election and the conclusion of this has been satisfactory.

These kinds of projects are beneficial for the society. It could change or improve the welfare of people and used to take care of old people, with lack of mobility, with memory problems, etc. These kinds of projects are the best contribution of the R&D for the whole society.

The use of Android as the develop platform has been very useful to learn developing applications over mobiles based on this operating system, which can be useful in my professional career in short and medium-term. The study of new technologies helps a lot to increase the obtained knowledge during the post-graduate studies.

The obtained formation of this Thesis is not focused only in the knowledge that I have obtained about new technologies, but also the availability of design application from the scratch and to learn about project management. The redaction of this kind of reports helps to improve my skills in writing this kind of documents, which I consider also important for the development of my professional career.

In addition, that training should also include the learning about how to persuade to find the correct information and how to learn about the errors committed during the development of this *Master Thesis*. In these kinds of situations it is important to work in a team of people, including both colleagues and Master Thesis Director, who are important to give support and help whenever is possible by answering questions, giving ideas, tackling problems from another point of view. Thus, with calm and knowing that everything has a solution, it becomes to get the satisfaction of achieving the objectives. These moments are the ones which can teach new values and appreciate the assistance obtained.

The strength to overcome oneself, and the improving the personal and professional skills were the main motivations to finally conclude this *Master Thesis.*

BIBLIOGRAPHY

[1] AAAida, Facebook page, URL: <http://www.facebook.com/myAAAida>

[2] Alteraid, Official web, URL: <http://www.alteraid.com/>

[3] UPC, Universitat Politècnica de Catalunya, URL: <http://www.upc.edu/>

[4] EETAC, Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels, URL: <http://eetac.upc.edu>

[5] Social Networks, URL: <http://en.wikipedia.org/wiki/Social_network>

[6] Six degress of separation,
URL: <http://en.wikipedia.org/wiki/Six_degrees_of_separation>

[7] Web 2.0, URL: <http://en.wikipedia.org/wiki/Web_2.0>

[8] Semantic network, URL: <http://en.wikipedia.org/wiki/Semantic_web>

[9] Web 3.0, URL: <http://en.wikipedia.org/wiki/Web_3.0>

[10] Location based context awareness. Last visit: 07-15-2011
URL: <http://www.isprs.org/proceedings/XXXVIII/part2/Papers/79_Paper.pdf>

[11] Geolocation, geographic location,
URL: <http://en.wikipedia.org/wiki/Geolocation >

[12] Google Maps, URL: <http://maps.google.es/>

[13] Mini Augmented Reality Ads Hit Newstands, Technabob.
Last visit: 07-15-2011,
URL: <http://technabob.com/blog/2008/12/17/mini-augmented-reality-ads-hit-newstands/>

[14] Ronald T. Azuma. A survey of augmented reality. Presence, 6, 1997

[15] Web services, URL: <http://en.wikipedia.org/wiki/Web_service>

[16] Android Market, URL: <https://market.android.com/>

[17] QR code, URL: <http://en.wikipedia.org/wiki/QR_code>

[18] Google Health, URL: <http://www.google.com/health/>

[19] Google API, URL: < http://code.google.com/intl/en-EN/more/>

[20] MVC, Model View Controller,
URL: < http://en.wikipedia.org/wiki/Model-view-controller>

[21] KML, Keyhole Markup Language, URL: <http://en.wikipedia.org/wiki/KML>

[22] Google Earth, URL: <http://www.google.es/intl/en/earth/index.html>

[23] Apache Tomcat, Services server, URL: <http://tomcat.apache.org/>

[24] JAVA SUN, Java Virtual Machine, URL: <http://www.java.com>

[25] Apache Wicket, Java web application framework, URL: <http://wicket.apache.org/>

[26] Jersey REST, Web Service technology, URL: <http://jersey.java.net/>

[27] Eclipse, Java program environment, URL: <http://www.eclipse.org/>

[28] Android, Mobile platform, URL: <http://www.android.com/>

[29] SDK, Software Development Kit, URL: <http://en.wikipedia.org/wiki/Software_development_kit>

[30] UML, Unified Modeling Language, URL: <http://en.wikipedia.org/wiki/Unified_Modeling_Language>

[31] MagicDraw, Modeling tool, URL: <https://www.magicdraw.com/>

[32] SVN, Control version tool, URL: <http://en.wikipedia.org/wiki/Subversion>

[33] DDMS, Dalvik Debug Monitor Server, URL: <http://developer.android.com/guide/developing/debugging/ddms.html>

[34] AChartEngine, Chart tool, URL: <http://www.achartengine.org/>

[35] Magnitude, AR tool, URL: <http://code.google.com/p/magnitudehq/>

[36] ARKit, AR development kit, URL: <http://open.arqera.com/node/17>

[37] JSON, JavaScript Object Notation, URL: <http://en.wikipedia.org/wiki/JSON>

[38] SQLite, SQL database implementation, URL: <http://www.sqlite.org/>

[39] Perst, Embedded database, URL: <http://www.mcobject.com/perst>

[40] DB4O, Object oriented database, URL: <http://www.db4o.com/>

[41] Perst comparison with other databases. Last visit: 07-15-2011 URL: <http://www.garret.ru/perstbench.html>

[42] Java Annotations, URL: <http://en.wikipedia.org/wiki/Java_annotation>

[43] Layar, AR tool, URL: <http://www.layar.com/>

[44] Cursa del Corte Inglés, URL: <http://www.cursaelcorteingles.cat/>

[45] Jeffrey Sharkey. Coding for life, battery life. Last visit: 07-15-2011 URL: <http://www.google.com/intl/es-ES/events/io/2009/sessions/CodingLifeBatteryLife.html>

[46] Eclipse License, URL: <http://en.wikipedia.org/wiki/Eclipse_Public_License>

[47] Apache License, URL: <http://en.wikipedia.org/wiki/Apache_License>

[48] GNU License, URL: <http://www.gnu.org/copyleft/gpl.html >

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEX

TITLE: Advanced personal social network API for third-party mobile applications

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Alberto Carlos Toro Sánchez

DIRECTOR: Antoni Oller Arcas

DATE: 07-20-2011

Annex I. Acronyms

3G – Third generation
API – Application Programming Interface
AR – Augmented Reality
BL – Business Logic
CPU – Central Processing Unit
DB4O – DataBase For Objects
DDMS – Davilk Debug Monitor Server
EETAC – Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
GNU – GNU is Not Unix
GPL – GNU Public License
GPS – Global Positioning System
GUI – Graphical User Interface
HTTP – HyperText Transfer Protocol
JSON – JavaScript Object Notation
MVC – Model View Controller
POI – Point Of Interest
QR – Quick Response barcode
SDK – Software Development Kit
OS – Operating System
SOA – Service Oriented Architecture
SQL – Structured Query Language
SVN – Subversion
UML – Unified Modeling Language
UPC – Universitat Politècnica de Catalunya
VE – Virtual Environment
XML – EXtensible Markup Language

Annex II. Smartphones

The Smartphone is the name given to the mobile phone group that offers more advanced computing ability and connectivity than other mobiles.

Smartphones and featured phones may be thought of as handheld computers integrated with a mobile telephone and allow to user to run and multitask applications that are native to the underlying hardware, also combine the functions of a camera phone and a personal digital assistant (PDA). This kind of phones run complete operating system software and provides a platform for application developers.

Growth in demand for advance mobile devices boasting powerful processors, abundant memory, large screens and open operating system has outpaced the rest of the mobile phone market for several years.

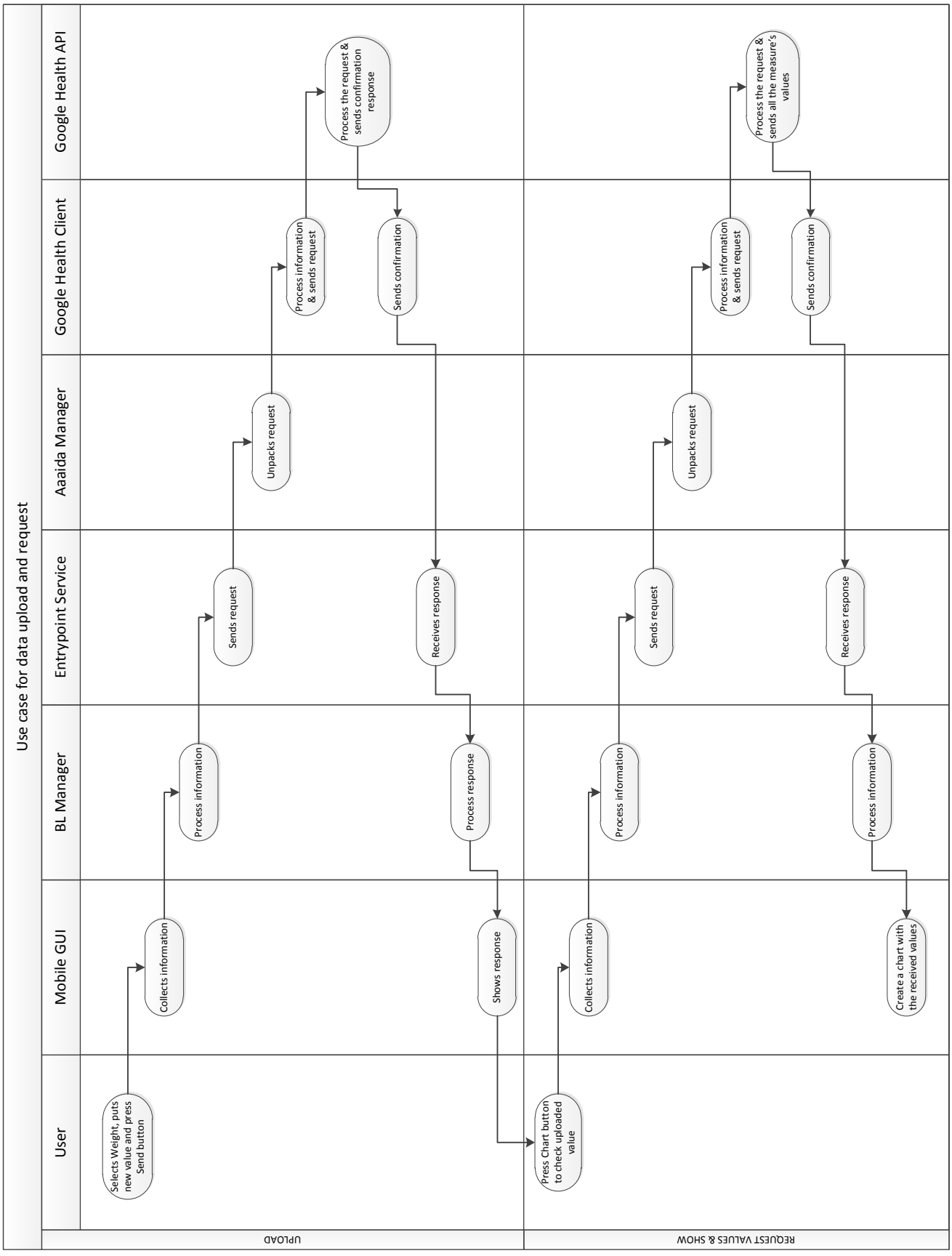
The following table shows the increase or decrease for each platform, comparing the years 2009 and 2010:

Table II.1 Table with OS & quota

Operating system	Quota 2009	Quota 2010
Android	3,5%	1500%
Symbian OS	44,6%	30,6%
iOS	17,1%	16,7%
Blackberry OS	20,7%	14,6%
Windows Phone	7,9%	2,9%
Other	6,5%	2,7%

Annex III. Flow diagram

This is the flow diagram of the use case of upload and request of information.



Annex IV. Mobile platforms

In this section a description of the suitable mobile platforms for the mobile application are described. They are suitable because the simplicity of programing and the developing and testing environment.

I. Android

Android is a software stack for mobile devices that includes an operating system (based on Linux), middleware and key applications. The *Android* SDK provides the tools and APIs necessary to begin developing applications on the *Android* platform using the Java programming language and running such process over *Dalvik VM*, the Java virtual machine integrated over the OS that is write in C++.

Android was developed initially by *Android Inc.* a company that was taken over by *Google* in 2005.

Every *Android* application runs in its own process, with its own instance of the *Dalvik* virtual machine. *Dalvik* has been written so that a device can run multiple VMs efficiently.

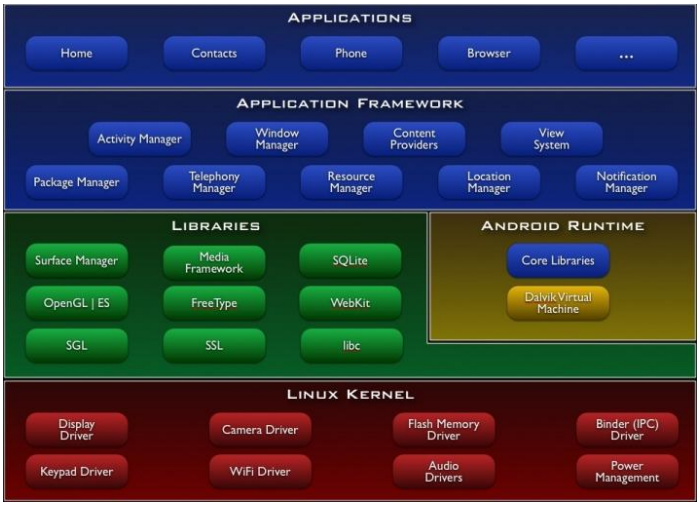


Fig. IV.1 Android platform

As can be seen in the *Fig.* IV.1, the *Android* platform is developed in layers. Each one can use the offered services of the lower layers. With and full access to the same framework APIs used by the core applications, the developers can interact and use to the device hardware such network access, location providers, local storage, etc.

The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

II. iOS

The *iOS*, also known as iPhone OS prior to June 2010, is the *Apple* mobile operating system for the iPhone, iPad, iPod Touch and Apple TV 2 devices. *iOS* is derived from Mac OS X, with which shares the Darwin foundation, and is therefore a Unix-like operating system by nature.

The *iOS* SDK contains the tools and interfaces needed to develop, install, run, and test native applications. Native applications are built using the *iOS* system frameworks and Objective-C language and run directly on *iOS*.

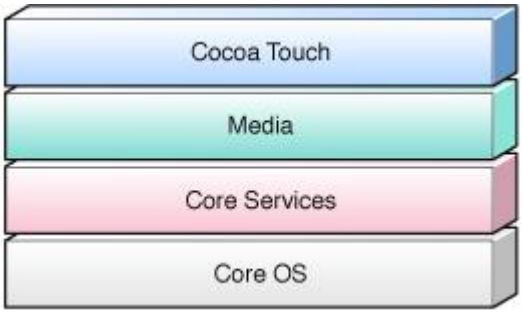


Fig. IV.2 iOS Platform

In the *Fig.* IV.2 the platform is also developed in layers. At the lower layers of the system are the fundamental services and technologies on which all applications rely; higher-level layers contain more sophisticated services and technologies.

iOS offers to the developers the possibility to access directly to lower-level layers if needed to use aspects of those frameworks that are not exposed by the higher layers.

III. Windows Phone

Windows Phone is a mobile operating system developed by *Microsoft* that is the successor of *Windows Mobile* platform. Designed to be used in smartphones, it is primarily aimed at the consumer market rather than enterprise market.

It is based on Windows CE operating system kernel and have, like other operating systems, a group of applications that uses the *Microsoft Windows* APIs. As with Android, this operating system is designed to work with multiple hardware platforms and chipsets, so applications developers need to be able to "compile" on runtime. Microsoft opted for using C# as its main development language and compiling it all over its CLR.

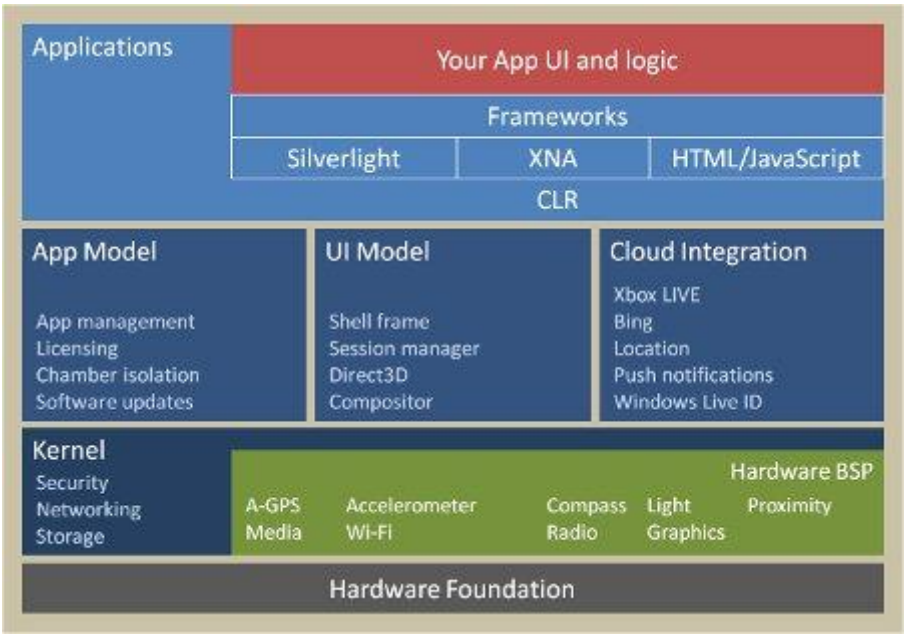
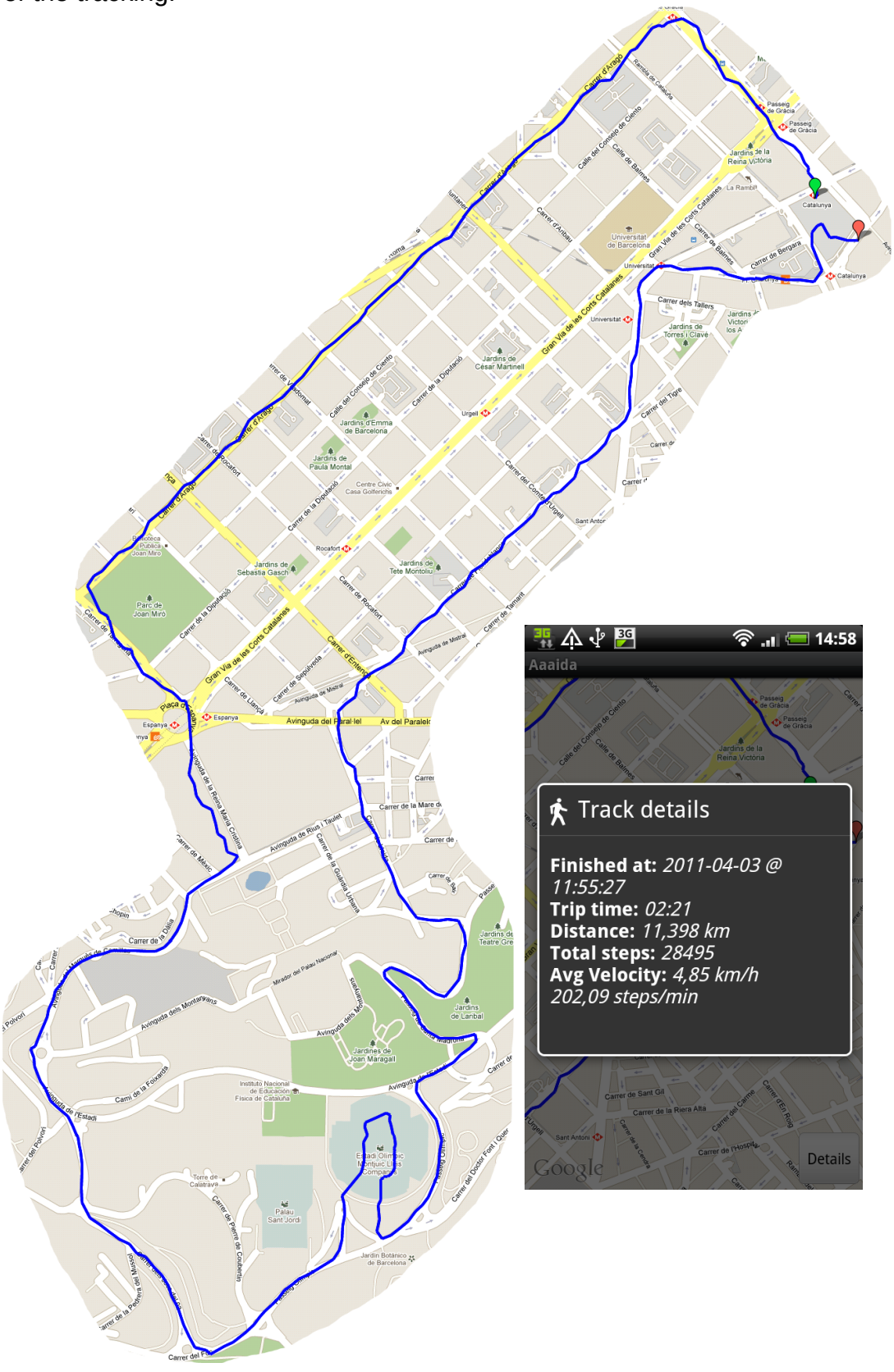


Fig. IV.3 Windows Phone platform

Like *Android* and *iOS, Windows Phone* is also developed to work in layers. That uses the abstract layers to simplify the access to the hardware from the application layer and framework.

Annex V. Example route

This is the zoom of the route saved in the *Cursa del Corte Inglés* and the details of the tracking.



Annex VI. Gantt diagram

This Annex shows the *Gant* diagram of the time line for this *Master Thesis.*