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

MASTER THESIS

TITLE: uPyxis, a real-time geolocated information network

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Antoni Carenys Roca

DIRECTOR: Dolors Royo Vallés

DATE: December, 20th 2012

Títol: uPyxis, a real-time geolocated information network

Autor: Antoni Carenys Roca

Director: Dolors Royo Vallés

Data: 20 de desembre de 2012

Resum

En els últims anys, gràcies al augment de l'ús d'internet per part de persones de tot el món, estan apareixent varis serveis de xarxes socials, els quals estan canviant la forma de comunicar-se i de interactuar amb el món i entre les persones.

A part dels camps d'amistats, aquestes nous serveis estan intentant explotar noves àrees com les professionals, la compartició d'imatges, o els missatges. A més, amb la popularització de l'ús dels smartphones, moltes d'aquestes xarxes comencen a incloure serveis de geolocalització, y també estan naixen noves xarxes basades exclusivament en la geolocalització.

Aquest projecte busca crear un nou servei que actualment no s'ofereix, a partir de l'ús de les capacitats dels smartphones actuals com son la geolocalització, la instantaneïtat graciés a l'accés a internet, i a la capacitat de les persones de ser sensors del seu entorn.

Aquest nou servei, esta centrat en proveir d'un servei útil per a persones curioses que volen conèixer en tot moment tot el que està passant al seu voltant, per a persones que no saben quines activitats realitzar en un moment donat, y per a persones que volen anunciar esdeveniments, activitats o incidents que veuen en un lloc concret.

Per aconseguir això, s'ha dissenyat, desenvolupat y provat un servei de xarxa social basat en l'anunciament y cerca d'activitats y esdeveniments que passen ara mateix al voltant dels usuaris y que tenen una curta durada. Aquest servei esta basat en tres pilars basics, informació d'esdeveniments y activitats, temps real y geolocalització. Aquest servei serà usat des d'una plana web y un Smartphone Android, i farà servir bases de dades y un webservice en el servidor.

En la seva realització s'han estudiat i fet servir varies tecnologies com Java, PHP, MySQL, SQLIte, Javascript o JQuery.

Title: uPyxis, a real-time geolocated information network

Author: Antoni Carenys Roca

Director: Dolors Royo Vallés

Date: December, 20th 2012

Overview

In recent years thanks to the increase of the use of internet by people from around the world, are appearing a lot of social networking services which are changing and evolving the way of communicating and interacting with the world and among people.

Apart from the field of friends, these new social networking services are trying to explode new areas as professionals, sharing pictures or messaging. In addition, with the widespread use of smartphones, many of these networks are beginning to include geolocation services, and also are springing some new networks that are based only on geolocation.

This project seeks to create a new service currently not offered, based on the use of the capabilities of current smartphones such as geolocation, the instant internet access, and the ability of people to be sensors of their environment.

This new service is focused on provided a useful service to curious people who want to know at all times what is happening near them, to people who do not know what activities to do at a given time, and to people that want to announce some events, incidents or activities that they see on some place.

To achieve this, it was designed, developed and tested a social network service based on the announce and location of activities and events happening right now around a user and that have a short duration. This service is based on three main basic pillars, information about events or activities, real time, and geolocation. The service will be used from a website and from an Android smartphone, which also will use databases and a webservice that will be on the server.

In its realization have been studies and used various technologies like Java, PHP, MySQL, SQLite, Javascript and JQuery.

INDEX

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

In recent years, are emerging various types of social networking services, becoming the true stars of the internet, and changing the way we communicate and interact with the world.

Due to that is very difficult to unseat an already well-established network, the new networks are trying to exploit new fields as professionals, sharing of images, messaging, or geolocation.

Networks using the geolocation tend to locate people or places. Networks focusing on user location do not totally succeed, because users do not always like to share its location. Moreover, networks focused on sites location only focus on locating sites that are always there, but not the activities that are happening right now inside the site or near the user.

As a result, the thesis aims to create a service to announce, find or alert about activities and events that are happening right now around the user, and that have a short duration. The service will be used from a website and a smartphone, so will be designed, developed and tested a website and an Android application for android which also will use a database and a webservice that will be on the server.

Next, on this first chapter will be explained the idea of this thesis, the main objectives to accomplish, and the personal motivations to do this thesis.

Once explained the main idea and objectives of this thesis, on the second chapter will be explained and compared different types of social networking sites, the most important technologies available to accomplish the different parts of the thesis, and also will be exposed the reasons to choose each used technology.

The third chapter will be centered on the design of the operation of the uPyxis service. This includes the description of the designed architecture for the system, the description of the requisites of each part of the thesis, and the designed way to accomplish these requisites.

On the fourth chapter will be explained how the designed service, application and website are implemented and deployed. This includes the tools that have been used, the final developed files that compose the different parts of the thesis, and the steps to publish the service to the world.

After the implementation, on the fifth chapter will be showing the results in terms of appearance. Also on this chapter, will be showing the results of the tests of use realized by some users.

On the last chapter are exposed the conclusions about this thesis, explaining the achieved objectives, the environmental impact of this service, and the future work for this thesis.

1.1. Idea

The idea of this thesis is to create uPyxis, a social network service to be consumed from a website or a smartphone, where registered users can post, search and be alerted of activities and events taking place right now near where they are located and that have a short duration.

Such activities may be happening now, or may start in a short period of time, and furthermore, users can interact with the activities, commenting, adding images and making on them check-ins.

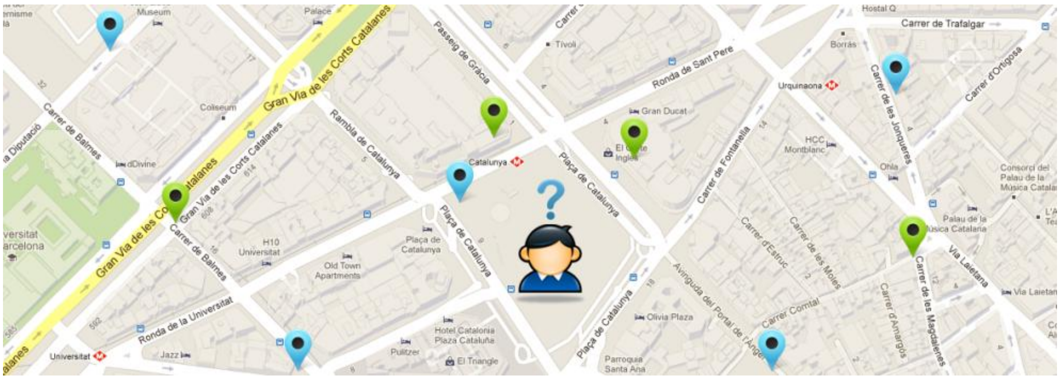

Figure 1 Idea of the thesis

To better understand the thesis idea, below will explain three possible use cases, one for the creation, one for the search and one last for the alert.

- Use Case 1. Juan is in the afternoon in a place where there is signing books a known writer, as he believes that this information will be useful to other people, decides to take his smartphone, and announce that activity in uPyxis. At night he goes out for drinks at a bar in the city center in which are doing a jazz concert, and see that in the next table there is a known football player so he decides to announce the jazz concert and that there is the known football.

- Use Case 2. Toni is with his friends at the Plaça Catalunya in Barcelona, and they are bored without knowing what to do. Toni starts uPyxis, the application geolocate him and among other things it says that at 150 meters there is a well-known writer signing books, that at 200 meters are filming an episode of a TV show, and that in half an hour at 400 meters begins a film in a free outdoor cinema.

  After discussion, they decide to go to the outdoor cinema to see the film. On the way to the cinema they hear sirens, and they see a column of smoke in the distance. They wonder what must occur, and Toni decides search the word "fire" on uPyxis, and as a result he obtains that at 755 meters there is a big fire on a factory, which solves their questions.

At the outdoor cinema, Toni makes check-in in the related activity of uPyxis. As the movie is becoming boring, decides to comment on that activity in uPyxis, and other people who are also in the film are discussing the same.

- Use Case 3. Marc has to carry Jordi from Barcelona to Girona with a car. As sometimes there are car accidents that generate congestion, Jordi decides to start uPyxis and configure an alert to notify them during one hour if there is an accident within 20km around.

  Halfway, Jordi receives a notification from uPyxis saying that the user Lidia has announced that there is an accident at 15km that generates congestion at the highway. As there is an exit near, they decide to dodge de congestion leaving the highway and entering to the next entry.

1.2. Objectives

The objectives for this thesis are to provide the service with a website and an Android application to interact with it.

The website should be able to locate the user and display the nearby events in a list or a map. The user can filter the events and see its detailed information, location and comment them. Each user will have a public profile page that displays their activity.

The smartphone application will interact with a webservice and in addition of the objectives described for the website, should allow the user to create new events, and do check-ins and add photos at every event that has not yet ended.

1.3. Personal motivations

On the realization of this thesis, i have two important personal motivations

The first motivation is that with the realization of this thesis i will learn how to perform a service that consists of an Android application, a website and a database.

The other motivation is that doing this thesis, i am doing a service that i needed a lot of times, and I think that can be useful for many people who do not know what to do at a certain time or are curious and want to know what happens all the time around them.

CHAPTER 2. STATE OF THE ART

This chapter explains the different technologies that have been used in each basic pillar of the project, and why these technologies have been chosen, comparing them with other technologies possible.

As this project will be a social network service, first will be explained and compared different social networking services. Later will be exposed and compared the available mobile technologies to develop a mobile application centring on Android and iOS, and continuing with the available technologies to develop a webpage. Finally, as this project is based on geolocalization, will be explained and compared the technologies in terms of geolocalization.

2.1. Social networking services

This project will be a social networking service, so first we need to know the other similar services, how they are oriented in terms of users and information, and the differences and similarities with this project.

A social networking service is an online service, platform, or site that focuses on facilitating the building of social networks or social relations among people who, for example, share interests, activities, backgrounds, or real-life connections. A social network service consists of a representation of each user (often a profile), his/her social links, and a variety of additional services. Most social network services are web-based and provide means for users to interact over the Internet, such as e-mail and instant messaging. Social networking sites allow users to share ideas, activities, events, and interests within their individual networks. (See [1] for more information)

Actually, there are hundreds or thousands of social networking services, but only a few dozen are extended and known, and most of them are very similar on the type of service that offers or its method of use.

Initially, social networking services, not included the geolocation in their services but slowly the large majority begin to use some geolocation to enrich their services. Furthermore, the popularization of geolocation on smartphones, caused the appearance of social networks based on geolocation. Due to this, social networking services that will be studied below were divided into two groups, the typical, and location-based.

The social networking services studied, are the most representative of their groups, and have been chosen, networks that do not have a similar use between them. It can be analysed a lot of things, but for this project we will focus on three areas, first if users want information about persons or otherwise they want information about things, secondly if the information is about live temporary events or not, and third, the importation that is given to the location.

2.1.1. Typical social networking websites

The main different representatives of this group are Facebook, Twitter, Linkedin and Badoo.

2.1.1.1. Facebook

Facebook was launched in February 2004, and it's the most used social networking service by worldwide monthly active users. On Facebook users add other users as friends and their updates their profiles. Like facebook there are a lot of similar sites like Tuenti, Google+, or Myspace.

This site is oriented basically to obtain information about other people, and only a little to obtain information about events or other similar things. This site is not entirely meant to explain the things that happen in real time, rather it is used to explain things of your life once they have finished, such as uploading pictures of a trip.

Here, the location of information plays a minor role, and is only used a few times to indicate from where the information is sent or where an activity has been carried out.

2.1.1.2. Twitter

Twitter was created in March 2006, and it's a service that enables its users to send and read text-based messages of up to 140 characters, known as "tweets", and has become one of the top 10 most visited websites.

In contrast to Facebook, Twitter is far more oriented to obtain information about things happening like TV shows or news, without losing information of what do the users. Another important point about the information posted on twitter is that most is about things that happen in real time.

The geolocation on this site is not very important because most of the information that is written is not necessary locate it, and the few geolocated messages sent by users, are used only for some use statistics.

2.1.1.3. Linkedin

Linkedin was launched in May 2003, and it's a site for people in professional occupations, using it for professional networking. Like Linkedin there are some similar sites like Xing or Viadeo.

On LinkedIn, the most important is the information of users such as their curriculums, but also have some importance the information about job offers, or discussion groups about labour issues. As usual, the main information in this network is not about real time things, and location is not used at all on it.

2.1.1.4. Badoo

Badoo was launched in November 2006, it is a dating-focuses network, and it is the world's fourth-largest social network with users spread across 180 countries.

The main information is based on information that the user fills in their profile, and such information is information that can be considered in real time.

Moreover, thanks to its application for smartphones, the location plays a very important role, to be able to detect the users who are close to you and with similar profiles to your interests.

2.1.2. Location based social networking websites

The main different representatives of this group are Foursquare, Loopt, Yelp and Eventful.

2.1.2.1. Foursquare

Foursquare was launched in March 2009, and here users makes check-in at venues, using a smartphone by selecting from a list of venues that the application locates nearby to the user. Each check-in awards the user points and sometimes badges

The network turns around the geolocation, so it plays a key role in this network and the most important information is not the user, but is that of the different places that are around the user, with ratings and comments that are made to them.

The information that is obtained in this network is about things that are always there, and not about things that are happening right now and that later will no longer exist. For example, the information we get are restaurants, hotels, universities, etc.., But not about what happens in these places at this precise moment.

2.1.2.2. Loopt

Loopt was founded in 2005 and it produces mobile location-based services that allow users to discover the world around them, and find and enjoy the friends, places and events around them via smartphones. Like Loopt there are a lot of similar sites like Whrrl, Buzzd, and Brightkite.
Loopt, in addition to showing geolocated information sites like Foursquare do, shows the location of your friends, and shows the events that are doing your friends. This information is becoming a little more real-time, but not its main pillar.

2.1.2.3. Yelp

Yelp was founded in October 2004, ant it's a social networking, user review, and local search web site. Like Yelp there are a lot of similar sites like Qype, and other sites similar to a yellow pages.

On Yelp, users punctuate and comment the different venues, so the most important thing here is also the information about places, and not personal information of users. Furthermore, the geolocation is important to geolocate the various venues, but as in the case of Foursquare and Loopt, is information about static things and not about things that happen now.

2.1.2.4. Eventful

Eventful was founded in 2004, and in this network users can search for events worldwide by time, location, performer, and descriptive keyword. Also users can create private or public calendars, including "smart" calendars which automatically update when events matching search criteria are added or existing events are modified.

This network is limited to leisure and cultural information that will happen next, but do not use a lot the geolocation due to that hasn't application for smartphones.

2.1.3. Conclusions

As a conclusion, we can see that none of the networks analysed comply 100% with all areas described above, they stay halfway in all areas, or one fails them.

As seen in the graph on the left of *Figure 2*, typical social networks remain half in all areas, because they focus more on the user information, and how they use the geolocation is not completely relevant. Twitter is the only one of these networks that is closer to fulfilling those areas, but does not have an important use of geolocation, and is not easy to view the geolocated information.

As shown in the graph on the right in Figure 2, location-based networks are much closer to fulfilling all areas because they make great use of the location, and are focused on the event information and not the user personal information.

They fails in the type of information that they shows because is static information that is always there, such as restaurants or shops, and show no real-time information of what is happening now in these places.

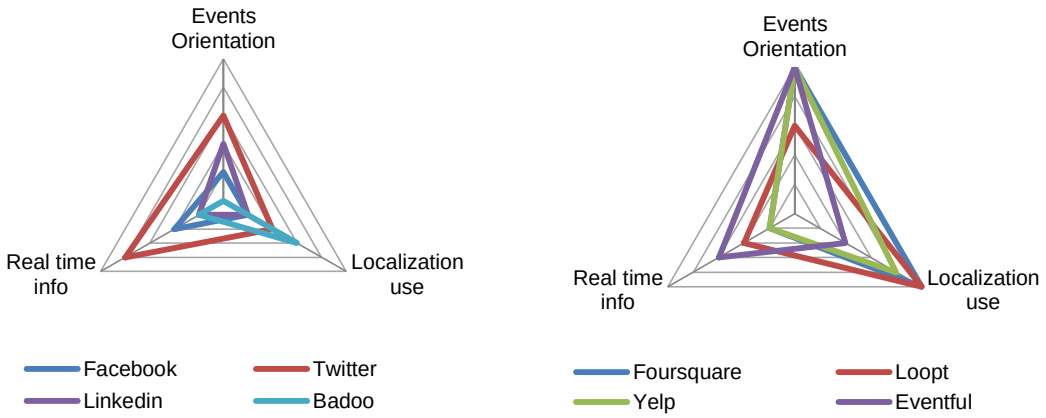

Figure 2 Social networking services, three areas analysis

As we can see on the *Figure 3*, this is the same graphic but with the uPyxis project. uPyxis will comply the three areas at 100%, because the important information of the network are the events that are happening right now, and all of these events are geolocated.

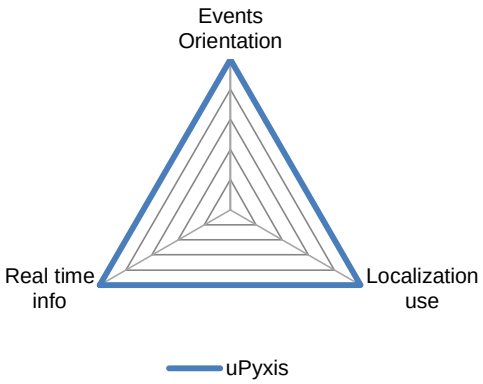

Figure 3 uPyxis, three areas analysis

2.2. Smartphone technologies

In this section we will discuss the options available to develop the mobile application of the project. Given that we are talking about a Smartphone systems application makes no sense to analyse the set of mobile systems, so we will focus only on the Smartphone systems. First we will describe the different operating systems available, and we will justify the choice of android, and later we will analyse android more deeply, allowing taking more decisions and knowing some limitations on the requirements.

2.2.1. Smartphone operating systems

As can be seen in *Figure 4*, according to market researcher IDC [2], currently, Android globally has gone up to be the undisputed king, followed at some distance by iOS, while the other competitors as Blackberry OS, Symbian and Windows Phone are in residual positions.

The same IDC shows that if you look only in the Spanish market, the reign of Android is even greater, as it achieves a share of over 80%.

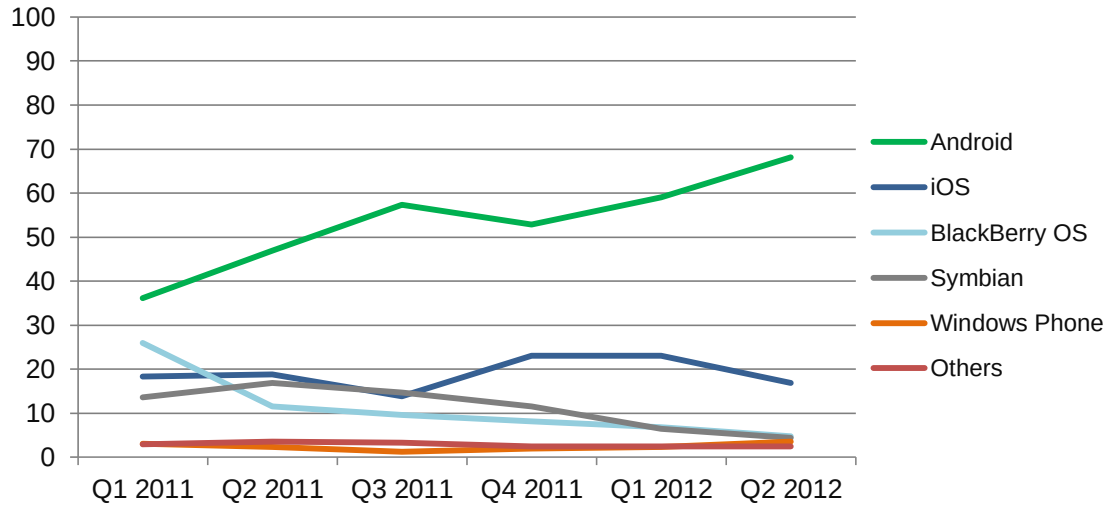

Figure 4 Smartphone operating system shares

Having identified the major operating systems, it was decided to make a first elimination, and leave only as candidates iOS and Android. It has taken this decision because we want to deliver the application to as many users as possible, and seeing the evolution of market shares in the short term these two operating systems will remain by far the referents in sales.

Next, we will explain briefly the iOS and Android operating systems, focusing on their strengths and weaknesses developing an application on them, showing that each platform has its own language and tools.

2.2.1.1. iOS

iOS is the Apple's mobile operating System. Originally released in 2007 for the iPhone and iPod Touch, it has been extended to support other Apple devices such as the iPad and Apple TV. The user interface of iOS is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons

The official language for iOS is Objective-C, and with this language can be created applications for iPhone, iPad and iPod Touch. This language is difficult to understand if you have not used before, as it has a difficult to read syntax, and requires writing long lines.

To develop for iOS, is also required a tool named Xcode. Xcode is the official development environment and it only exists for Apple and Mac, so to create iOS applications we need an Apple computer.

Furthermore, in order to distribute applications in the App Store and to test the developed applications in our own Iphone / Ipad, we must purchase a developer license which costs € 79 annually

2.2.1.2. Android

Android is an open source Linux-based operating system designed primarily for mobile devices such as smartphones and tablet computers, developed by Google in conjunction with the Open Handset Alliance.

Android uses the Java language to program applications and a platform SDK that runs on Windows, Linux and Mac. Java is easier to learn and program that Objective-C due to its simplicity.

If we want a development environment, we can use the ADT plugin for Eclipse that includes a simulator, which is also cross platform, free and open source.

To test applications on your phone, we do not need to buy any license, and if we want to distribute our applications in Play, we only require a single payment of $ 25 for registration.

Android has the problem that due to that it is open source and that most companies can tweak it to suit their devices, they have created many versions and subversions, generating some compatibility problems in some functions of the applications.

2.2.1.3. Conclusions

Because both Android and iOS support HTML5, one option would be to develop a web application in HTML5 which is compatible with both systems. With this technology we would get friendly design and fast implementation, but otherwise we could not access to basic elements such as controlling the phone's camera or control at 100% the use of geolocation, so this option is discarded due to the mentioned reasons.

From my point of view, with nothing more in mind that the market share and the use of the various Market Stores that the users do, if you want to develop a payment application, you would have to choose iOS, because its users buy many more apps than Android, but if you want to develop a free application as

in this case, it is better to choose Android because its market share is far higher.

If to this we add that is more agile develop an application for Android, that i already have the basic knowledge to start and also that i have the basic materials such as an Android phone on which to perform tests, it seems clear that in this case, the best option is to choose Android.

2.2.2. Android

Once chosen Android, is now going to discuss a little more thoroughly some features of this operating system, starting with the aforementioned varieties of versions that are installed in the devices of the market.

Figure 5 shows the distribution of versions of Android smartphones which acceded to Google Play in early October 2012, you can see that you can find devices with earlier versions of Android (1.5 or 1.6) and also some with the latest versions (4.0 or 4.1), while the vast majority have the 2.3 or 4.0 version

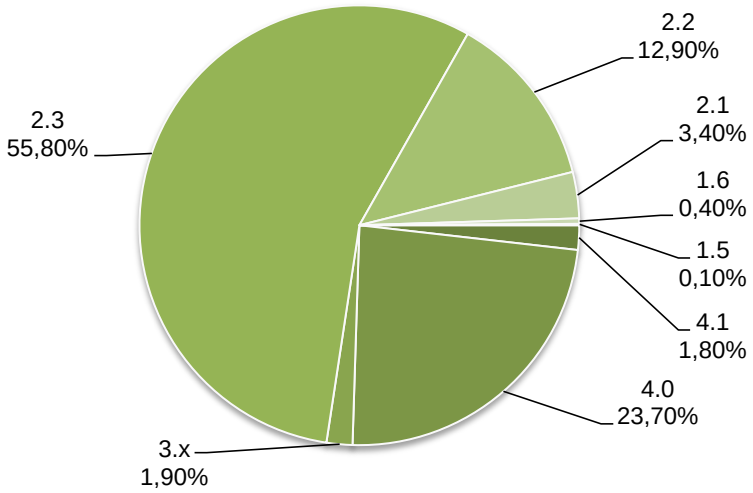

Figure 5 Distribution of versions of Android on active devices

From the previous graph it has been decided that the application will be compatible from version 2.1 onwards, this ensures that it will be compatible with 99.5% of active terminals currently on the market, but due to alterations made by manufacturers, there can always be some minor compatibility issue in some terminal.

Another important factor to consider before designing the application is the size and density of the existing screens, to achieve the highest possible compatibility. Must be understand that depending on the size of the letters on

small screens they may be unreadable, or due to that some icons are not prepared for a certain density, they will look bad.
From the *Figure 6* that shows the current distribution of sizes and densities of the Android terminal screens can be seen that the vast majority of screens are of a normal size or larger, which is equals to between 3 and 7 inches, and usually it have a high density.

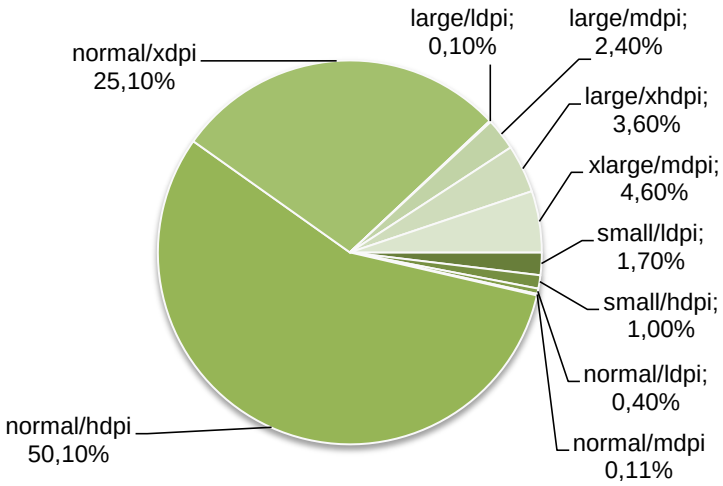

Figure 6 Distribution of size and density of screen on active devices

From these data, it was decided that when developing the application interface is going to put the icons and images in different resolutions, to increase compatibility, and that the amount of information displayed and their sizes, shall be designed towards a screen from 3 inches.

2.3. Web technologies

The web server is one of the basic pillars of the project. The server is responsible of generate and send web pages and other files needed for its correct visualization for example images, javascripts, CSS, etc.

In addition, the webservice will also be located on the web server which will answer the requests of the Android app, and the web server will also contain the database where all information about the events and the users will be stored.

Below we describe the possible technologies to generate and send the website to the users, we will continue with an analysis of the available options for the webservice, and finally we will explain the technology used in the database.

2.3.1. Webpage

Long ago, the code of each web page of a website was served in a single document, but now, to make the code more understandable and to improve the loading speed is divided into multiple documents.

As seen in the schema of the *Figure 7*, on our website we will serve a document with the HTML code of the page, in another document all CSS styles of our website, and in another all the Javascript code of our website. Thus, given that CSS and JavaScript documents are always the same for all the pages and that the modern browsers get used to cache these files, these files will only be loaded the first time, improving the loading speed and decreasing server load.

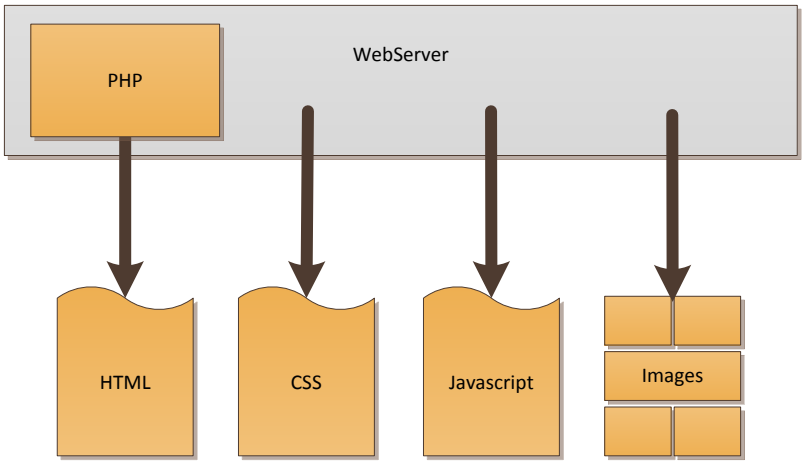

Figure 7 Shema of serving the webpage

Our website will be dynamic because it will be personalized for each user profile and its location, so we will need a web programming technology that allows us to do this. The two main languages in this area are PHP and ASP.net, which will be analyzed briefly below with CSS and Javascript technologies named above.

2.3.1.1. ASP.net vs PHP

ASP is not really a language, is an acronym for Active Server Pages. Supports multiple languages, current ASP programming languages are Visual Basic Script, JScript, and C #, among others. The major drawback of ASP is to be a proprietary system that is used only by Microsoft Internet Information Server (IIS). This limits their availability to WIN32 servers only.

PHP (acronym for PHP: Hypertext Preprocessor) is an open source language very popular especially suited for Web development and can be embedded into HTML. With PHP you are not limited to output HTML. Among PHP skills include: imaging, PDF files and even Flash movies (using libraries) on the fly.

PHP can be used in any of the major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows,

Mac OS X, RISC OS, and probably some more. PHP supports most modern web servers, including Apache, IIS, and many others.

One of the many features that have PHP is its support for a large number of databases. Write a web page with enabled access to a database is incredibly simple using a specific extensions (e.g. for mysql).

Given the benefits named above, that we have some basic knowledge of PHP and that is intended to use the XAMPP pack in which is included PHP, the Apache web server and the MySQL database, it was decided to use PHP to develop web

2.3.1.2. CSS

In addition to the logic of a web, another very important part is the style. To facilitate and minimize work was created CSS. CSS is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a mark-up language, as for example HTML or XHTML.

In the HTML code the page will be only structured, and each item of the HTML code will be associated to a style class defined in the CSS file, with its height, color, transparency, shadows, etc… When the browser displays the page, the browser displays the page as CSS says. Thus, if there are elements with the same styles we will not have to write a new style for each item but will be associated with the same style class.

In the CSS file will be described all styles from all over the web, so that the browser only has to load it the first time and then keep it in cache.

2.3.1.3. Javascript & JQuery

JavaScript is a programming language that is used mainly to create dynamic web pages, but unlike PHP, runs on the client side, that is on the web browser.

Javascript, among other things allows a page to be dynamic, incorporating effects like text that appears and disappears, animations, actions that are activated by pressing buttons, etc. It also serves to download data via Ajax, and to generate parts of the website on the fly.

Technically, JavaScript is an interpreted programming language, so it is not necessary to run compiled programs. In other words, scripts written in JavaScript can be tested and executed directly in browser without any intermediate processes.

As in the case of CSS, will be created a specific and unique file containing all javascript developed functions that will be used on the web.

2.3.2. Webservice

Another important part is the communication of the mobile application with the web site, in order to make the various requests such as login, create events, list events, etc., and get the answers to each of the requests.

For these operations, will be implemented a webservice on the server. This webservice will receive various input parameters, will do the necessary operations to obtain the results, and will respond with the requested data in a structured format.

There are various types of webservices, but we will focus on analysing the two most famous and used, which are SOAP and REST. We will see that in the case of SOAP, the returned data are in XML format, and that in the case of REST can be in XML or JSON format, so we will also discuss between XML and JSON to choose one of them.

2.3.2.1. SOAP

SOAP (Simple Object Access Protocol) is a standard protocol that defines how two objects in different processes can communicate using XML data exchange.

SOAP is a complete XML-based infrastructure, each object can have methods defined by the programmer with the necessary parameters. The main advantage of SOAP is that it provides a mechanism for describing services to users, and to announce its existence.

A SOAP problem is that the data structures are very heavy, and they occupy a lot of space, which can slow its reading and sending.

2.3.2.2. REST

While the term REST was originally referred to a set of principles of architecture, currently is used in the broadest sense to describe any simple web interface that uses XML (or JSON) and HTTP, without the additional abstractions of protocols based on message exchange patterns such as the SOAP web services protocol

REST is much lighter and can be implemented using wide range of tools, which will reduce the bandwidth and will have a shorter learning curve. However, customers have to know what to send and what to expect since the developer is the one that defines the format of input and output data.

REST, always use HTTP as a method of communication, and XML or JSON to exchange data. Each URL is an object on which we can use the methods POST, GET, PUT and DELETE. It uses the language of the web.

Unlike SOAP which establishes the rules of communication both input and output of data, REST rules are of input only. While SOAP is based on well-defined standards of communication, REST depends directly on the HTTP protocol.

Although that REST was designed so that their responses were in XML format, currently as REST is a technology in which the programmer can customize it to their desire, you can also respond in JSON.

JSON, short for JavaScript Object Notation, is a lightweight format for data exchange. JSON is a subset of the literal notation of JavaScript objects that does not require the use of XML. The simplicity of JSON has led to their widespread use, especially as an alternative to XML in AJAX. One of the supposed advantages of JSON over XML as a data interchange format is that it is much easier to write a parser for JSON.

An XML file is much more. It is a set of structured data. As such, it supports queries, has a readily ascertainable structure (DTD, XML Schema), can be easily visualized, can be easily processed, etc.

2.3.2.3. Conclusions

Has been decided to choose REST because it is much lighter and for connections in a mobile terminal it is a very important factor. And our webservice for now will be for our own use, will not be public, so we see a lot easier to handle a REST service.

For answers we have chosen that will be in XML format. Although JSON is lighter and faster to deal with, we believe that moving little data as in our case is not going to see much difference, and also prefer to lose some efficiency but to see more clearly the structure of the data to send. If in the future the webservice is open to the public, surely we will migrate to JSON.

2.3.3. Databases

To save all user information and events, we will use databases in both the web server and on the mobile terminal.

For the android terminal, we are limited to using SQLite because it is the implemented by default. For the web server, there are several types, but has decided to use MySQL because it is open, free, and it is included on the XAMPP package that we will install.

Below we will briefly describe MySQL and SQLite.

2.3.3.1. MySQL

The MySQL database has become the world's most popular open source database because of its high performance, high reliability and ease of use. It is also the database of choice for a new generation of applications built on the XAMPP stack (Apache, MySQL, PHP / Perl / Python.) that we will install. Many of the world's largest and fastest-growing organizations including Facebook and Google rely on MySQL to save time and money powering their high-volume Web sites, business-critical systems and packaged software.

MySQL runs on more than 20 platforms including Linux, Windows, Mac OS, Solaris, IBM AIX, giving you the kind of flexibility that puts you in control.

2.3.3.2. SQLite

SQLite is a database engine currently very popular for offering very interesting features such as small size, no server need, require minimal configuration, be transactional and of course be open source.

Android comes standard with all the necessary tools for creating and managing SQLite databases, and include a complete API to perform easily all necessary tasks.

2.4. Geolocalization

This project turns around geolocation, so we have to correctly analyse the two most important uses that we will use in this context and see that there are different options or technologies in each one. On the one hand we need the use the maps to show the location of the events created by users, in the other hand we need to obtain the user's location to display the nearest events.

2.4.1. Maps

For the project will need to use the maps in 3 different ways, the first is to show a navigable map on the Android application which will show the location of events with icons, the second is the same but on the website, and the third is to generate static maps with the position of an event.

On the internet we can find a large number of map services but for this analysis we have chosen the four most important and representative. They all have very similar features, so we will focus on analysing what really matters to us and the things that can limit us, which usually are its limits of use, and the quality of its APIs.

2.4.1.1. Bing Maps

Bing Maps is a web mapping service provided as a part of Microsoft's Bing suite of search engines. Microsoft has positioned Bing maps as an alternative to Google maps, especially for providing local information. A unique feature of Bing maps is their 'bird's eye' view, which gives aerial views from several perspective angles.

Free accounts for Bing maps have some usage limits on the maps API, so you are limit to 125,000 sessions or 500,000 transactions in a 12 month period.

2.4.1.2. OpenStreetMap

OpenStreetMap is a collaborative project to create a free editable map of the world. The maps are created using data from portable GPS devices, aerial photography, other free sources or simply from local knowledge and this data is available under the Open Database License.

OpenStreetMap data is free for everyone to use, but you can’t do a heavy usage of its servers, because OpenStreetMap has a strict tile usage policy to protect the service which is paid for by donations, and run by volunteers.

OpenStreetMap also uniquely offers another approach: Create your own tile server, rendering the raw map data yourself. This might be a sensible option if you run a very popular website, or if you require only a limited area of the world to be shown.

2.4.1.3. Mapquest

MapQuest was one of the first providers for maps on the web, and today it's encouraging the transition to open maps. MapQuest is the only company that lets you choose between using licensed maps or open maps. Even using licensed map data, it has free accounts with no limits on map views. However, it does limit you to 5K calls per day, and after testing it, we see that it is quite slow generating maps. The open map option, which uses OpenStreetMap has no limits.

MapQuest also has servers for map tiles, which it lets you use with other APIs for free, even for heavy use. Using MapQuest's tile servers with open source APIs such as OpenLayers or Leaflet is a popular option.

2.4.1.4. Google maps

Google maps is the map service of Google, is the most widely used at present, which has a more powerful API, and in which can find more examples and tutorials on the internet, such as how to create heat maps.

Given that it is the most used, Google recently decided to put some limits on the usage of its API. These limits are 25,000 API loads per day and 25,000 loads of static maps. In case of exceeding these limits, you need to pay $ 4 per 1,000 extra loads.

In addition, Google Maps has an API developed for Android, that greatly facilitates the use of maps in an Android application, and this API currently have no usage limit.

2.4.1.5. Conclusions

If we compare the different usage limits, we see that MapQuest and OpenStreetMap are limitless but otherwise their service are slower, and at the moment is not considered the option of implementing OpenStreetMap in our own server. Furthermore, Bing annually has a limit of 125,000 sessions, equivalent to 350 days maps loads, and the Google maps limit is 25,000 loads per day.

From the above, because we have some experience with Google maps and can be found much more documentation, we decided to choose Google maps, to implement both the web and Android application.

Due to that in when the events are listed, a map will be displayed as its avatar in those without main photo, this could quickly exceed the limits of static maps, so it was decided that when an event is created, will be downloaded in the server a static map generated with the option of MapQuest OpenStreetMap, and this downloaded map will be assigned as the avatar of the event that has not main picture.

2.4.2. Geolocalization techniques

Here will be described the three main techniques used in terms of geolocation by Android and by the modern browsers (W3 Geolocation API) to give the location. Such techniques are performed transparently to the user, and nearly equal from the point of view of the developer.

2.4.2.1. Mobile towers

This method is based on the triangulation of mobile phone antennas. Are sent to a location service, the different identifiers of nearby antennas, and this gives us back our position. The accuracy of the identification by the mobile phone antennas depends on the density of the antennas and the available data in the database of mobile identification location.

Accuracy by this method is not very high and may contain errors that may be up to several kilometres.

2.4.2.2. WiFi hostspots

This method is similar to the previous, but in this case is based on the triangulation of WiFi access points. When using this method, we send the MAC addresses and SSID detected around us to a service that has an enormous database with the location of each WiFi access points.

Google has one if these services and their databases are populated by data gathered by the StreetView cars, and also from the same data that is sent from smartphones to request a location.

With this type of location, we improve the accuracy of the previous system, and we can be accurate with an error of between 20 and 30 meters.

2.4.2.3. GPS

Is a triangulation positioning system of the signals received from satellites, which allows determining the position of a device in any place of the Earth with an accuracy of between 1 and 15 meters.

To calculate the distance between the device and the satellites is measured marking the time delay of the signal emitted by the satellites. Once we know the delay and that the signal has travelled at the speed of light, we can determine the distance over which the satellite is located. To make this process work you need to receive signals from at least two other satellites.

2.4.2.4. Conclusions

In Android we can detect the position by choosing two options. The first is through telephone antennas and with wifi networks, and the second is using GPS. Because the first option is much faster, it has been decided that in the application, it first will detect the position with the first option to give the initial results, and immediately it will use the second option to be updating the position for other petitions.

CHAPTER 3. DESIGN

Once analysed the state of the art, and chosen the different technologies to be used, will now proceed to explain the design for the operation of the service, describing the operation and requisites of each of the parts of its architecture.

In the *Figure 8* you can see the basic architecture that has been designed for the service. On the left there are applications that the user will use to interact with the service, such as the browser and the Android app.

In the middle there are the web server, which is the brain of the service, is an Apache server is programmed in PHP, and is composed of, the part that generates the web page, the webservice that communicates with the Android application, and the Core where you will find the main functions. And finally on the right side you can see the part of the database.

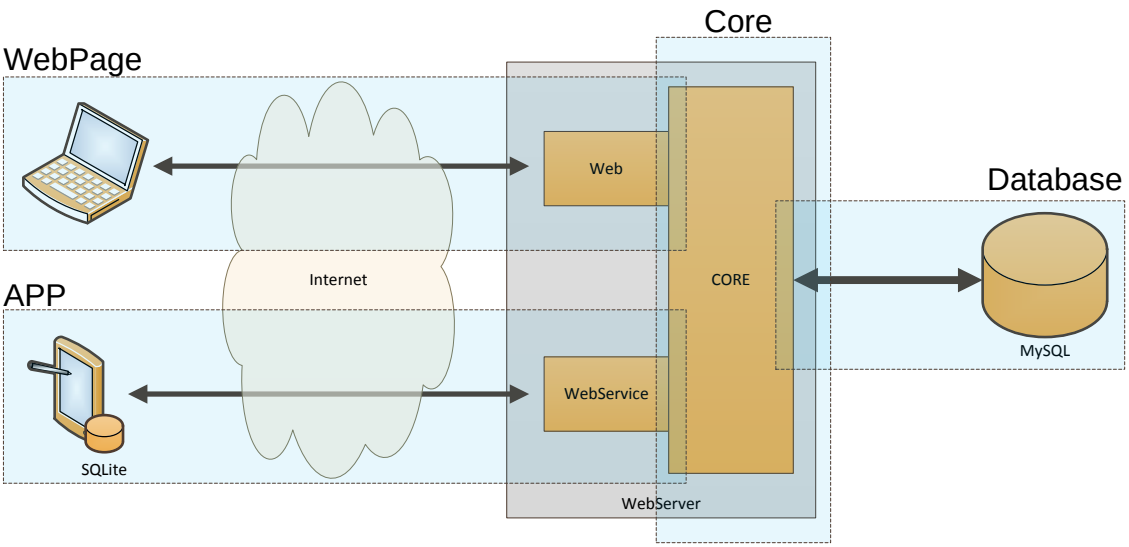

Figure 8 General architecture

To explain its design, architecture explanation will be divided into 4 parts, the part of the website, the Android application, the core, and the data part. These four parts are explained below.

3.1. Android APP

This part includes the Android app and the webservice that is located on the server, because it is the way that has the application to communicate with the server.

In the following *Figure 9,* can be observed a diagram useful to understand paths between screens that users may make since they start the application.

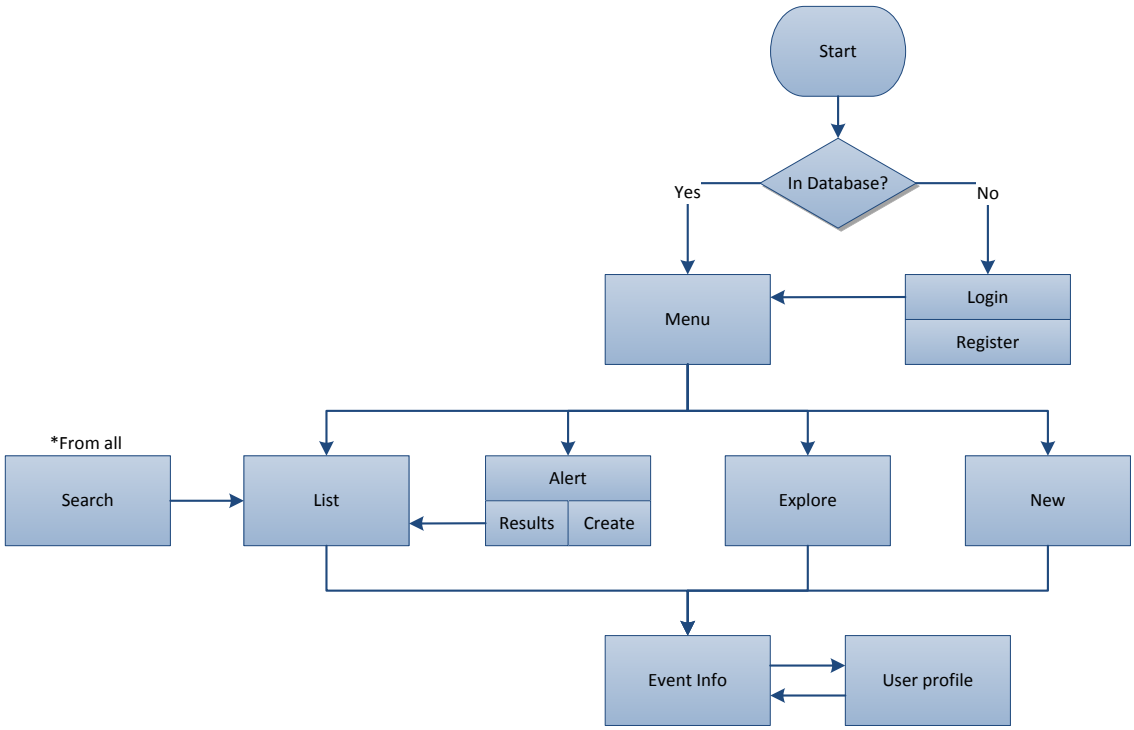

Figure 9 APP general diagram

As seen, when the application starts, if not done previously, users must register or log-in and immediately will appear in screen menu. From this screen, they can access to other screens with event listings, map display, alerts or create new events.

From the screens on which events are displayed, they can navigate to the screen with the complete description of the events, and from this they can access to user profiles. Furthermore, from almost any screen they can access to the search screen.

Below we will define all functional and non-functional requirements of the application which help to explain and understand the *Figure 8*, these requirements should be for meeting the objectives marked in this project for the application part, and must allow a good experience for the user.

3.1.1. Functional requirements

Below is listed and explained the different functional requirements of the application. In the *Appendix III* you can find several functional diagrams that expand the information described in this section, knowing the logical steps needed to do the application part.

3.1.1.1. Login, logout, register

The APP will allow the user to login with his username and his password, and once logged in, will be and option to logout. If the user doesn't have an account, he will be able to register a new account, introducing a *username*, a *password*, an *email* and a *name*, also, the APP will show the *terms and conditions*.

3.1.1.2. Menu

Once logged in, the APP will show a menu with the four principal options, and also the logout option. The four principal options are *Events*, *Explore*, *New* and *Alert*. Each menu option will be formed by an image and a text.

3.1.1.3. List events

The APP will request to the server and show a list with the events, ordered by proximity or time of creation. For each event row will show its title, the first 250 characters of the description, an image, the distance to the event location, and the remaining time. When the users touch one of the rows, will be open the event screen.

Also the user will be able to filter the events results by category, or status (Started, Not Started, All).

3.1.1.4. Explore events

The APP also will be able to show the events results in a map. Each result will be shown with a blue or a green icon, depending on the status of the event. When the user touches one of the icons the title of the event will be shown on the top, and when the users touch the same event again, will be open the event screen.

Also the user will be able to filter the events results by category, or status (Started, Not Started, All).

3.1.1.5. Show event info

The event's info screen will have three tabs, one with its info, a second with the event comments, and a third with the event check-ins.

The info tab will show the main image of the event (if exists), the description, the events main details, a map with its location, and a horizontal list with its images. Also, will be a group of buttons to put a comment, check-in or image. If the user is the event creator, he will be able to delete the event; otherwise will be able to report the event.

The comments tab, will show a list with the event comments ordered by descending time of creation. For each row, will be shown the username, the user image, the time of creation and a button to report or delete the comment. When the users touches one of the rows, will be opened the user's profile.

The check-ins tab will show a list with the event check-ins ordered by descending time of creation. For each check-in, will be show the username, and its image. When the user touches one of the check-ins, will be opened the user's profile.

3.1.1.6. Show user profile

The user's profile screen will have a static zone with the user's name, username and image, and another zone with three tabs, one with the user created events, a second with its comments, and a third with its check-ins.

The lists shown inside of each tab, will have the same style that the list events option, and the list of the comments shown on the event info screen. If the user touches one of the lists rows, will be opened the event info screen.

3.1.1.7. Create events

With the APP the user will be able to create new events. First will be shown a map centred in the users location, where the user will select the point where will be located the new event.

Once selected the location, will appear a form to introduce the event details as the title, category, an optional subcategory, description, duration (1h, 2h, 4h, 8h, or 12h) and the minutes left to starts the event (0m, 15m, 30m, 1h or 2h). Also, the users will be able to take a photo and attach it to the event.

Once the event will be sent to the server and is correctly created, will be opened the event info screen, otherwise and error popup will we shown with the error description.

3.1.1.8. Create alerts

The user will be able to create defined alerts and the APP will periodically asking the server if there are some near events that matches with the defined alerts. Will be a screen with a list of the created alerts, and a button for create a new one. For each created alert row will be a button to delete the alert, a button to see the results, and another to edit the alert.

When creating a new alert or editing an existing one, the user will introduce a name for the alert, duration and distance, and the event filters as title, category, subcategory and description. The results screen will have the same style that the List Events screen.

3.1.1.9. Search events

The APP will have a search option formed by a simple search and an advanced search. In the simplest search the user only will be able to introduce a word or words that will be searched either in the description, title or subcategory. In the advanced search, the user can define the title, description, category, subcategory, and distance.

3.1.2. Non-functional requirements

3.1.2.1. Language

Due to that the project is orientated to Spanish and Catalan people, tthe APP will be available in Catalan and Spanish and also in English as an universal language. Depending on the Phone language will be shown in one or other language.

3.1.2.2. Android version

Due to the fragmentation that we have seen in the 2.2 point, to have the biggest public, the APP will works on every Android Smartphone with at least the 2.1 version of the Android OS.

3.1.2.3. Cache

The APP will implement a little cache to save the images due to that the interaction between screens of the APP will generate that some images can be shown several times in a short period. With the cache also the APP will show the images faster, and the server will have fewer loads.

3.1.2.4. Database

The APP will use the android's SQLite database because after the user login, it needs to save its session id to send it in each petition to the server. In addition, in the database will be stored the alerts and its results.

3.1.2.5. Market

Actually, to reach the largest number of users it's essential to put the developed APP on the android's market, so will be needed an account on it, and upload the app there with its requirements.

3.1.3. WebService

The webservice is a key part of the application, because it is the part that allows the communication of the application with the service. As mentioned in the previous chapter, the webservice will be programmed in PHP, is designed as a REST webservice and their answers will be in XML format.

It must have features to resolve all the requests for information or insertions necessaries for the above requirements. For each function of the webservice will be implemented a separate file, so each one will be accessed via a GET request to a different web address. The different parameters for each function will be passed on the same address with the format "&parameter=value".

The functions implemented in the webservice are the explained below. In the *Appendix II* you can find highly detailed information about the functions of the webservice.

- ***login:*** To login. Returns the user ID and the session number generated for that user.

- ***logout:*** To announce that the user log-outs of the service, and that the server needs to delete its related session number.

- ***register:*** Receive new user's personal data to create a new user.

- ***getUser:*** Returns the user's personal information and / or their comments and / or events and / or their checkins and / or images,

- ***getEvent:*** Returns the event information and / or their comments and / or their checkins and / or their images.

- ***getLocations, getLocationsMap, getLocationsAlert, getLocationsSearch, getLocationsSearchAdv:*** Returns a list of events close to a set of coordinatesand that correspond to a filter, ordered by proximity or creation time.

- ***setEvent:*** Used to create a new event based on the different parameters that are attached into the petition.

- ***setComment:*** To add a new comment relating to an event.

- ***setCheckin:*** To add a new chckin relating to an event.

- ***setImage:*** To add a new image relating to an event.

- ***setReport:*** To insert a report relating to an user, an event, a comment or a image.

- ***setDelete:*** To delete an event, a comment or an image.

In all requests except on the login, we must pass two parameters to indicate that the request comes from a user logged-in, these parameters are the user ID and the session.

In addition, to provide security to the service, the webservice should check that in each request are sent all the required fields, and further that each of these fields have an allowed length and all of its sent characters are permitted.

3.2. Web

This part covers the creation and display of the website, which is the other gate of interaction with the service by the end user.

In the next *Figure 10* can be seen a diagram that serves to paths between screens that users may make since they loads de website.

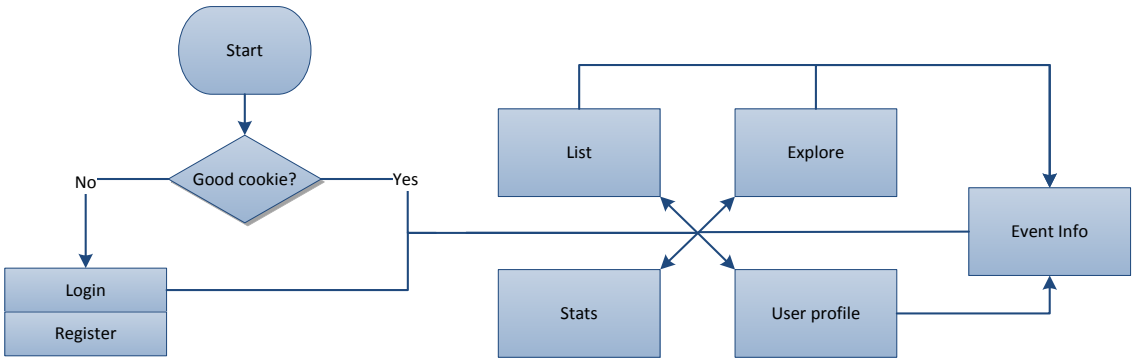

Figure 10 Web general diagram

As seen, when accessing to the website, first is verified by a cookie if the user has previously logged, otherwise log-in and registration screen appears.

Within the site, there will be four main pages that can be accessed from the top menu, which will display a list of events, explore a map with events, view service statistics, or view and edit your profile.

In the pages that are displayed events, the users can go to another page with full information on the events, and also to the profile page of the user that creates the event.

Below we will define all functional and non-functional requirements of the website which help to explain and understand the *Figure 10*, these requirements should meet the objectives marked in this project for to the website part, and should allow a good user experience.

3.2.1. Functional requirements

Below are listed and explained the different functional requirements of the website. In the *Appendix III* you can find several functional diagrams that expand the information described in this section.

3.2.1.1. Login, logout, register

The webpage will allow the user to login with his username and his password, and once logged in, will be and option to logout. If the user doesn't have an account, he will be able to register a new account, introducing a username, a password, an email and a name, also, will be an option to show the terms and conditions. Also, if the user doesn't remember its password will be an option to recover it.

3.2.1.2. Menu

On the top of the web, will be a fixed bar with the logo of uPyxis, a search fields and a menu. This menu will have four principal options, the first one is the home page, the second is the explore map page, the third the statistics page, and the last one is the user's profile page.

3.2.1.3. List events (Home page)

The webpage will request to the server and show a list with the nearest events, ordered by proximity or time of creation. For each event row will show its title, the first 250 characters of the description, an image, the creation time, and the related user.

The user will be able to open the event clicking on the row, or open the related user when clicking on the username. When the users touch one of the rows, will be open the event screen.

Also the user will be able to filter the events results by category, or status (Started, Not Started, Finished, All) with a top and lateral menu.

3.2.1.4. Explore events

The webpage also will be able to show the events results in a map and also on a lateral list. Each result will be shown with a blue or a green icon, depending on the status of the event.

If the user clicks on an event of the lateral menu, will be centred to the event, also when the user put the mouse over an icon, the related row will be highlighted. When the user touches one of the icons of the map will appear a

popup box on the map with the title of the event, the image and a short description.

Also the user will be able to filter the events results by category, or status (Started, Not Started, All) with a top menu.

3.2.1.5. Show statistics

The website should show statistics about the service activity. These statistics are displayed in two ways, the first is through a graph and the second through a heat map.

In the case of the graph, it will show the evolution of the number of events created during the last month grouped by days, and during the last week grouped by hours.

For the heat map will be displayed with different shades of color the zones with more events created, which will know the zones where is most used uPyxis, and by using searches can also know the zones where occur a certain type of events more often.

3.2.1.6. Show and edit users info

You will see the profile of an user, which will contain a background image, an avatar, user statistics, and personal information. Also will appear the user activity on uPyxis, and it can be filtered to show only the events, the comments or the check-ins.

If a user opens his personal profile, will have an option to edit its settings. In this option the user can change his personal information, his password, add a new avatar image, and change the background image of the profile.

3.2.1.7. Show events info

The event's info page will show main info related with the event, its comments, images, checkins, and will allow to the user to insert a comment.

The main info of the event include the main image, the title, description, map, creator, start time, duration, category and subcategory. A menu will allow the user to load with Ajax the comments, images and check-ins of the event, and also include a comment.

With some icons, the user will be able to share the link of the event page and a short description on the main social networks as twitter, facebook and google+. Also, the user will be able to report the event or some of the comments and images that other users insert, or if it is the creator of the event, it will be able to delete it.

3.2.1.8. Search events

On the fixed bar of the top of the web, the user will be able to do event searches introducing some words. If the search is done on the explore map zone, it will done on the map, it's done on the statistics zone, it will be done on the statistics zone, also the search will be done on the home page.

3.2.1.9. Change web settings

The page will have a bottom menu with some setting options. These options will allow to the user to change manually the language of the web, change the location to a fixed town of Spain or set it to autodetection, open an 'about us' page, open the terms and conditions, and open a contact form to send us some comments.

3.2.1.10. Info about to how download the APP

Also, the bottom menu will have an option to open de info page of the Android application. On this page will be shown a little description about the application, some screen captures of the aplication, and a link to the android market page of the application to download it.

Also, this page will have a short form to introduce an email, and a button that will allow to the visitor to send an email with the link of the application download page, to do easy to download it.

3.2.2. Non-functional requirements

3.2.2.1. Language

Due to that the project is orientated to Spanish and Catalan people, the webpage will be available in Catalan and Spanish and also in English as a universal language. Depending on the webbrowser language will be shown in one or other language, also the user will have an option to change manually the language, and this will be saved on a cookie.

3.2.2.2. Domain

In order to have visibility on the Internet, the website must have its own domain, which will be www.upyxis.com, and should be placed in a hosting provider that at least allows the use of mysql and php.

The addresses of the different sections of the website that users see on its address bars, must be clean, ie must not contain the ".php" nor the names of

the parameters. This will be done by inserting internal rules for rewriting urls and redirections in the htaccess file.

3.2.2.3. Security

In the htaccess file mentioned above (available in Annexes) will be add the code necessary to prevent common attacks such as SQL or XSS attacks, avoid listing the directories or their contents and avoid reading the web by harmful robots.

Also, in all the variables that are sent to the server, and cookies will be reviewed on the server to check that do not contain illegal characters or malicious strings.

3.2.2.4. Cookies

The website uses cookies to store some variables. In particular, store a cookie with the session number returned after login, the location manually customized, and the preferred language. Also related to our website, cookies will also be saved for the use of Google Analytics to know the usage statistics of our website.

Due to the use of cookies, a new European directive [4] requires to inform the user that cookies will be used, and also the finality of them. Due to this, its use is explained in the terms and conditions, which users must accept to register.

3.1. Core Design

The part of the Core is done in PHP and can be considered the most important part of the service because consists on all the functions necessary for the service to works correctly inserting and removing information.

Core functions will be called from both the webservice and from the web page part, and will make calls to the database to store and retrieve the needed data.

To explain this functions, we will divide into four areas, those related to the user, to the events, to the search, and those related to the proper use of the service.

3.1.1. User related functions

The first user-related function that is needed is the function that allows inserting a new user, verifying repeated entries such as users with the same nickname, email, etc. Once registered, a function will send to the user an email with a link that must access to validate and to use the account. Another similar function is to update the user data, such as its name, customizing its profile, statistics, etc.

Is also needed a log-in function that validates that the password is inserted correctly, and then generates a code for that user session, and also another function for the log-out that must delete the session code to make effective the logout.

Other necessary functions are functions to recover the user-related information. This would consist of different functions depending on whether you want to recover, i.e. personal data, comments, check-ins or images.

3.1.2. Event related functions

The first basic function related to the events, is the function that allows their creation. If there were no main image, it must also connect to Mapquest and download an image with the map of its location.

Other functions needed in this area, are those that will allow inserting comments, checkins and images related to an event. In the case of images, it will be necessary to generate and store the same image in various sizes.

In addition, will also be necessary functions for the recovery of the event information and its comments, checkins and images. To retrieve this information, it should keep in mind that this event, comments, checkin or image has not been reported or deleted by the user. In this case on the description of the comment or of the event will be shown a text saying that it has been deleted

3.1.3. Search

A very important part of the service, are the functions to search events. The search, will accept the variables that define an event ie category, status, title, description, subcategory, etc.., requires that the results are events near to the user. To find the closest events we uses the events coordinates and the coordinates of the user's location.

To determine the distance between two coordinates we use a SQL select statement containing the formula of the following *Figure 11*

$$6371 * acos(\, cos(\, radians('latitude1')\,) * cos(\, radians('latitude2')\,) * cos(\, radians('longitude2') - radians('longitude1')\,) + sin(\, radians('latitude1')\,) * sin(\, radians('latitude2')\,)\,)$$

Figure 11 SQL Haversine formula

This formula for SQL is based on Haversine formula The Haversine formula is used generally for computing great-circle distances between two pairs of coordinates on a sphere. For detailed information, please see [3].

3.1.4. Karma & Report system

Due to that there may be users with malicious intent that can create false events or offensive content, it is necessary to implement a quality assurance system based on karma and the cooperation of the users.

By default, each user, event, comment, and image will have a karma with value '100 ', and as they create or receive interaction, its karma will increase as shown in *Figure 12*, up to a maximum value of '200 '.

Action	User	Event	Comment	Image	Related user
Insert: event	+1	-	-	-	-
Insert: comment	+1	+1	-	-	-
Insert: checkin	+1	+1	-	-	-
Insert: image	+1	+1	-	-	-
Report: spam	-5	-10	-10	-10	-5
Report: content	-5	-10	-10	-10	-5
Report: fake	-5	-10	-10	-10	-5
Report: troll	-5	-10	-10	-10	-5
Report: other	-5	-10	-10	-10	-5
Report: repeated	-	-10	-10	-10	-1
Report: finished	-	-10	-	-	0

Figure 12 Karma values of each action

If some element receives a report from a user, the karma would be reduced for both the element reported and the creator of the element, reducing it as indicated in the above *Figure 12*.

Once the karma arrives to '0 ', the event, comment or image is deleted and will no longer appear. If is a user who arrives to '0', it cannot create content in the system for a period of 12 hours if the first time, 24 hours if the second, and so on.

To avoid misuse of this system of self-control, the following restrictions are imposed:

- One report for same IP every hour to the same element
- One report for same IP every 5 minutes
- One report for same user every minute

3.2. Database

As mentioned above, it will use two databases, SQLite in android application, and MySQL on the server. Below are the tables needed on each one.

3.2.1. Android APP

Following are the tables designed for SQLite, for much more detailed information of all its fields, please see Annex.

3.2.1.1. Tables and relations

For the SQLite database of Android, will be needed three different tables as shown in the following *Figure 13*.

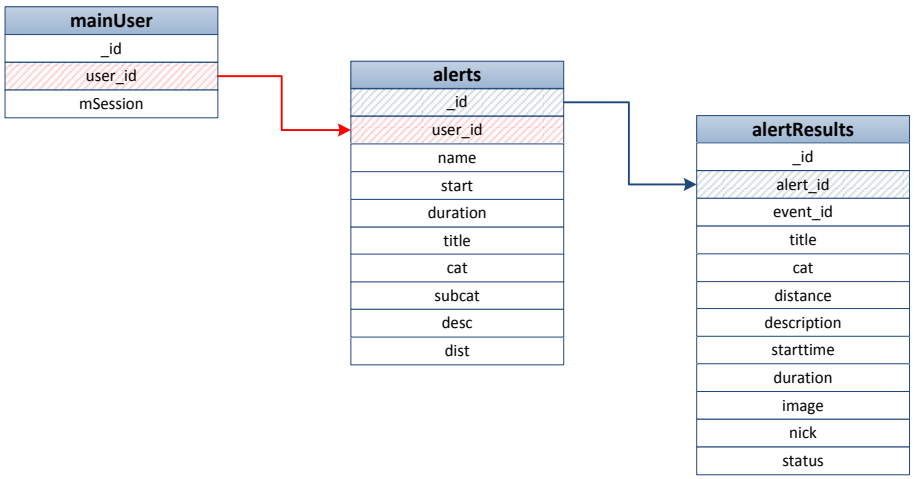

Figure 13 Tables of the Android database

Table 'mainUser' will be used to save the user ID and session number that is received once made the log-in. These data will be sent on every request that is made to the webservice.

Table 'alerts' will be used to store the information about the alerts created by the user. Each alert will be linked to a specific user id, because on a same device can log-in different users in a short time.

Table 'alertResults' will store the event information resulting from searches of each of the alerts. Each event result will be linked to an alert id.

3.2.2. Server

Following are the tables designed for MySQL, for much more detailed information of all its fields, please see *Appendix I*.

3.2.2.1. Tables and relations

Due to that on the server will store all the information, it need a lot of different tables. In total will be needed 9 tables, which can be seen in *Figure 14* with their relationships.

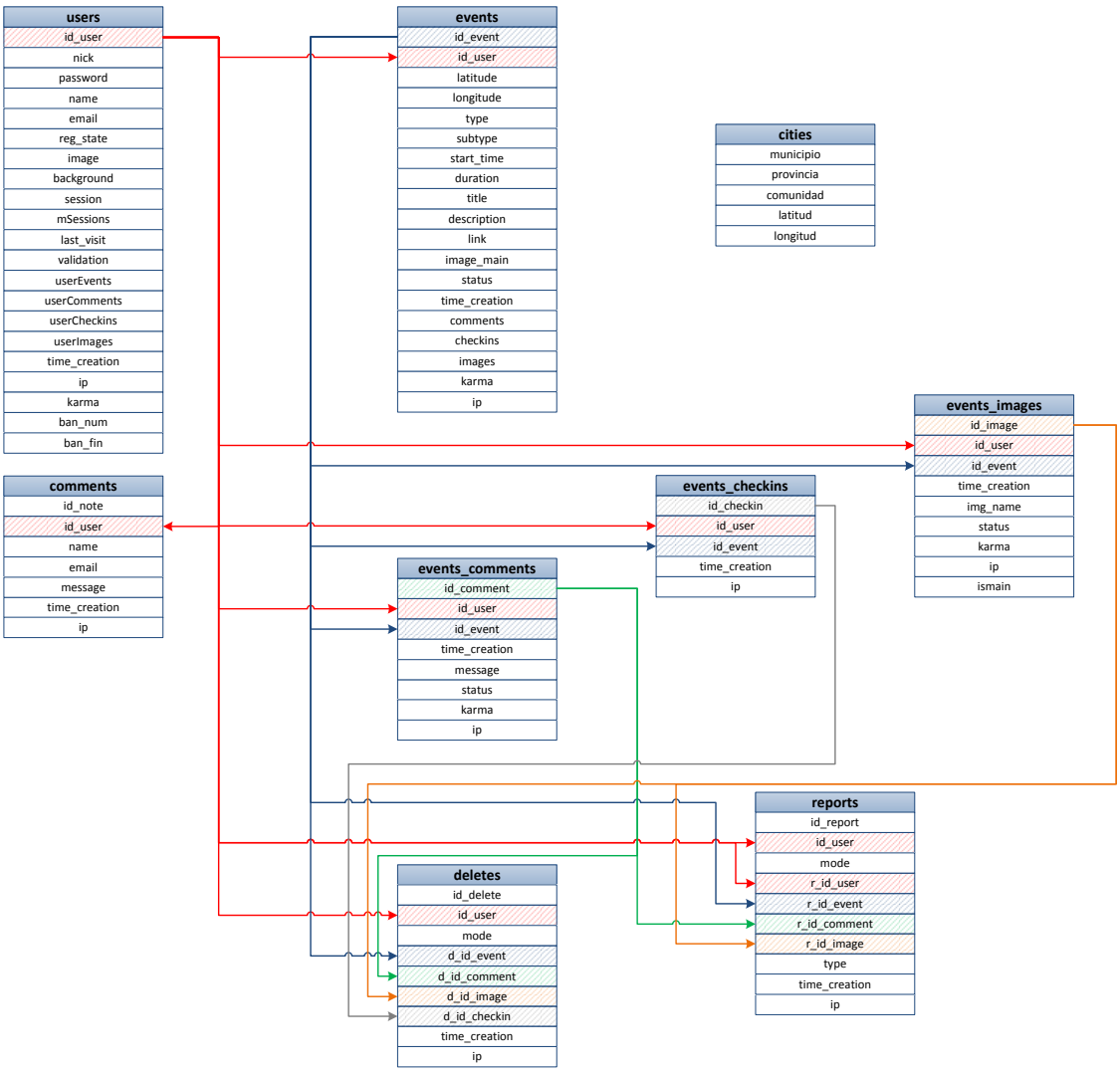

Figure 14 Tables of the server database

The 'users' table contain all information related to a user such as personal data, customizations, statistics, meetings, etc. Each user will have a unique user ID that will identify it.

Table 'events' contain all information related to an event such as its details, statistics, etc.. Each event will have a unique event id to identify it, and each event will be linked to the user id of the creator.

The tables 'event_comments', 'event_checkins' and 'event_images' contain information of all comments, checkins and images that users create in the events. Each will have a unique identifier, and will be related to the event id to which it belongs and user id that creates it.

The tables 'deletes' and 'reports' are used to store the information of all reports and deletes that the users do. With this table we can use it to keep track of what users do with these systems. Each will have a different identifier and will be related to the element to which the action is performed and the user who performs it.

Table 'comments' saves the comments that visitors send us over the web. We receive these comments by email, but are stored in this table in case there is a problem with the email system.

Table 'cities' contains all municipalities of Spain, along with the province to which they belong and their geographical coordinates. It is used when the user does not want to use the localization in the web and want to preset its location.

3.2.2.2. Schedulers

Because it is necessary to update the state of events as time goes on, between 'not started', “started” and 'finished', and the user bans expire, is necessary to use schedulers on the database.

These schedulers are used to periodically launch automatic queries in the database to detect events that its status should be updated, and that users should be unbanned.

Due to this, we need to implement three schedulers to be launched every minute to update the states of the events, and another to be launched every hour to remove bans of the users.

CHAPTER 4. IMPLEMENTATION

This chapter explains how the application and the website are deployed. This includes the tools that have been used, the various output files and the steps to publish the service to the world.

4.1. Android APP

4.1.1. Development environment

The Android app has been developed from zero with some programming skills on Java, but not in Android. For this we have used the Eclipse development environment, in which was installed the Android SDK and the ADT plugin.

Eclipse is a multi-language software development environment comprising an integrated development environment and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages.

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux, Mac OS or Windows.

The custom plugin for the Eclipse IDE, called Android Development Tools (ADT) is offered by Android. This plugin is designed to gives a powerful, integrated environment in which to develop Android apps. It extends the capabilities of Eclipse to let quickly set up new Android projects, build an app UI, debug your app, and export signed or unsigned app packages for distribution.

All these tools are installed and used on a laptop with Windows 7 64-bit, Intel Core i5 with 4 cores at 2.27 Ghz and 4Gb of RAM.

To test and debug the application, we have used the Android emulator that comes with the SDK and a Samsung Nexus S with Android 4.4 and a 4-inch screen HDPI.

4.1.2. Layers

The application development has been separated into three layers, which are explained below. The files belonging to each layer can be found listed in the *Appendix IV*.

4.1.2.1. Presentation

The presentation layer contains the code used to draw the user interface, ie the various screens that the user navigates. In Android, the screens are designed using XML files, in these XMLs are inserted labels that represents container of information, and in each label are placed a set of attributes such as margins, colours, width, etc.

We have created an XML file for each screen, and XML subfiles for other elements such as the top bar, the format of the list items, etc.

4.1.2.2. Logic

The logic layer contains the code that processes the data, and in our case we have divided the programmed files in this layer into four categories:

- **Activities.** This kind of files are the main of each screen, are responsible of the loading of the interface, and perform the main logic regarding this screen

- **Adapters.** They are responsible to receive a list of data items, such events or comments and to decide how to display each of the items in a list on the screen.

- **Objects.** These are files with the structure of various objects that are to be used as events, user profiles, map layers, etc..

- **Others.** In this category we have included files with general functions, and two very important files which are the "cache" and "alertService".

  The object cache is used to create a cache of the images that are received. Given that on the navigation that an users make can load the same images several times in a short period, this cache can improve the speed of loading and minimize the Smartphone data consumption.

  The alertService generates a service that is used to perform queries and notifications when the user has created alerts. This service is needed to keep running queries, because due to the way that Android deal with processes, if the user put the main application in background, it would surely be finished in a short period.

4.1.2.3. Data

In the data layer we programmed files that are used to retrieve data.

We can find a file that takes care of all queries to read or write to the SQLite database. This file is also responsible for creating the tables when you run the application for the first time, and update them when necessary in new versions of the application.

The other two files of this layer are those that are responsible for making all the queries to the webservice and check the answers, and the file that is responsible for parsing XML responses, and create arrays of objects with the information.

In the logical layer is also found the webservice hosted on the server. It has been used to implement it a folder "webservice" which have hosted all the php files that represent each of the functions available in the webservice.

4.1.3. Resources

This section explains how we have used the various resources required for the implementation, such as images or strings.

4.1.3.1. Images

Because all smartphones not have the same performance, nor the same screen densities, to improve efficiency we have used the same images at different resolutions as shown in *Figure 15*.

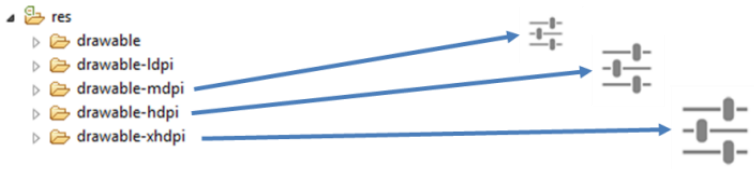

Figure 15 Different density images location

In this way, the Android system automatically displays the image that best suits each Smartphone according to their specifications.

4.1.3.2. Strings

To make the application available in several languages, three files were generated with the same strings but each in a different language, as shown in *Figure 16*.

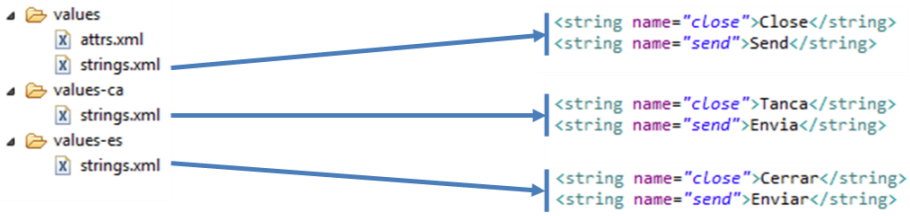

Figure 16 Files with strings in different languages

Thus, when the application starts, the Android system automatically load the file with the strings in the language in which the user has configured its phone.

4.1.3.3. Constants

Because many constants are repeatedly used in several places, and to take more control of them, they have been grouped into a single archive as shown in *Figure 17*.

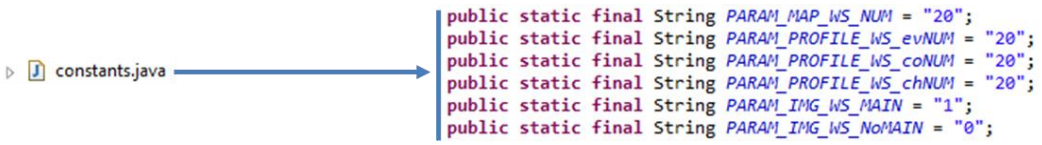

Figure 17 File with constants

4.1.4. Publish

Now they are going to explain the steps taken to publish the application and that other people can install it.

4.1.4.1. Keys and signing APP

To distribute and install the application outside the development environment is necessary to generate a private key using keytool, then compile the application, and finally sign the application with the previously generated key.

4.1.4.2. Google Play

With the previous step already the application can be sent to other people, but for wider distribution, it is necessary that the application will be available on google play.

For this, first of all we need to register as Publisher, this costs $25, and immediately we can use the googlePlay console to customize he application marketing. As users downloading the application will be able to view the statistics.

4.2. Web

Below is explained the different layers in which we divide the development of the website. Detailed information about the developed files can be found in Annex XX.

4.2.1. Development environment

The parts of the website, the core and the webservice, have also been developed from zero, without using content management systems like Joomla or Liferay. Were performed using PHP with already some knowledge of this language. To do this has not been used any powerful development environment, only Notepad++ and google Chrome browser for debugging and testing. For the web server and the PHP interpreter has been installed XAMPP.

Notepad++ is a text editor and source code editor for Windows. It aims to be a lightweight and robust editor for a variety of programming and scripting languages. One advantage of Notepad++ over the built-in Windows text editor, Notepad, is tabbed editing, which allows working with multiple open files.

Google Chrome is a freeware web browser developed by Google that uses the WebKit layout engine. Integrated in Google Chrome, there is a set of tools for web developers, which allows analyzing and debugging the various parts that make up the web page that the users sees both the HTML, and CSS and Javascript.

XAMPP is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

All these tools are installed and used on a laptop with Windows 7 64-bit, Intel Core i5 with 4 cores at 2.27 Ghz and 4Gb of RAM. Later has been deployed the web server in a shared hosting on 1and1.

4.2.2. Layers

The development of the web can be also separate into three layers, which are explained below.

4.2.2.1. Presentation

In the presentation layer of the web, we can found a set of php files that generate the structure of the interface of all sections of the website. The styles and colors of this structure are defined in a css file that has been developed.

The interface that has been developed is based on a fixed top bar, which is the main menu and browser. The body of the web is with a gray background with the main information on the left in white boxes and on the right there are submenus and secondary information. In the bottom of the page are several links with information and options.

4.2.2.2. Logic

The main functions of the site are in a related but separate files from which are responsible for generating the interface. There is a file of functions with the functions of each section of the website. These files are responsible for validating and processing information, create links with verification codes, send emails, etc.

A part of those files there are other logic files very important and used in most requests. We will focus on explained these last files, which are:

- _**IPObten.php** This file contains the functions to return the IP of the user who made the request. Used to store the IP each time a user creates new content such as events, comments, etc.. and so know the origin of such requests. Initially also returned the ISP, but was removed because it took more than 1 second to return the name of the ISP.

- _**readCookie.php**. This file contains the functions to read the user id and session number that is saved in a cookie at login, it also verifies that they match and that the data have not been tampered , and then was loaded the main infomation of the user .

- **upload.php**. This file contains the functions necessary to receive an image, either the user profile or event, and immediately make the necessary copies in different sizes.

4.2.2.3. Data

The functions that perform queries to the MySQL database are mixed with the logic, but there is a separate file called _DBconnect which is called every time that the php build a connection to the MySQL, because it has the access data and generates the necessary link with the database. For this php connection with the MySQL, we use the mysqli PHP extension.

4.2.3. Resources

This section explains how we use the different resources necessary in the website such as images or scripts.

4.2.3.1. Images

The images used on the website, and that are sent to the android application, are divided into three different folders.

- **/img/** In this folder are general images used on the web such as logos, wallpapers, and icons

- **/img/events/** In this folder are the pictures of the events. The images are generated and stored in five different sizes, depending on the purpose and the means by which they are intended, for example for the web, for the application to full screen, etc.. The size is identified by the first character of the file name. Inside this folder is another folder with the the downloaded maps from MapQuest with the the positions of the events.

- **/img/users/** In this folder are the images of user profiles. The images are generated and stored in two different sizes according to the purpose in which it are displayed.

4.2.3.2. Scripts & CSS

The folder / scripts / contains all the necessary JavaScript and CSS files for the web. Programmed files, are one with the javascript, and another with the css styles, all other files that are required, are files like jquery, colorbox, etc.

4.2.3.3. Strings

As in the case of the application, in order to ensure that the website is available in several languages, three files were generated with the same strings but each in a different language.

When loading the web, check first of all if you have saved a cookie with the preferred language, if so it loads the appropriate strings file. If not, will consult the http headers sent in the request and load the file with the specified language.

4.2.3.4. Constants

As in the case of the application, because many times many constants are used in various sites of the code, and to take greater control of them all has been grouped into a single file called _constants.php.

4.2.4. Publish

4.2.4.1. Hosting & Domain

It has hired a 1and1 shared Linux hosting with a price of 5 € per month in the second year, which allows the use of PHP, MySQL and FTP to upload files.

Moreover, this free hosting includes two domains, of which one was used to obtain the domain www.upyxis.com

A problem encountered in this hosting is that some PHP extensions were not available for security, which has caused that the part that downloads internally MapQuest maps were to be programmed differently.

4.3. Database

4.3.1. Development environment

The part of the databases has also been developed from zero, with almost no previous experience in relation to using these databases. For this we have used the MySQL database that comes by default in XAMPP as explained above, and that database has been administered through phpMyAdmin.

PhpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a Web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. PhpMyadmin is also included in de XAMPP package de we installed.

4.3.2. Publish

In 1and1 shared hosting, has also been used MySQL and phpMyAdmin, since it is what came by default, but when publishing the various tables and schedulers developed, there have been some problems.

These problems are due to that the hosting contract is of low capacity, the MySQL database that comes by default does not allow the use of schedulers, so we cannot use this technique to update states and events users.

Because of this, the possible solutions that have been found to solve this problem and to update the states of the various elements are the following:

- Hire a better hosting, which allows use schedulers, or having the ability to install a service to be launched every few minutes to update the states of the elements. This solution would involve an increase in spending.

- Use an old computer on my house, and do that every minute it connects to a PHP file of the 1and1 server which would perform the updates. This solution has the problem that we depend on that the availability of this second computer.

- That each requests from the web or application to request listings of events, automatically throw a piece of code that updates the states. This solution greatly increases the cost of computation if many requests.

Finally, we choose the last option because at the moment is a service for demonstration and with few users. However, if the number of users increases, we will decided to use the first option and hire a more powerful hosting.

CHAPTER 5. RESULTS

After the implementation of the service, on this chapter will be shown the final results in terms of appearance, and the results of tests of functioning and ease of use realized by users.

5.1. Final appearance

Below are some screenshots of the final appearance of the Android app and the web page, please, to see more screen shots go to the *Appendix V.*

5.1.1. APP

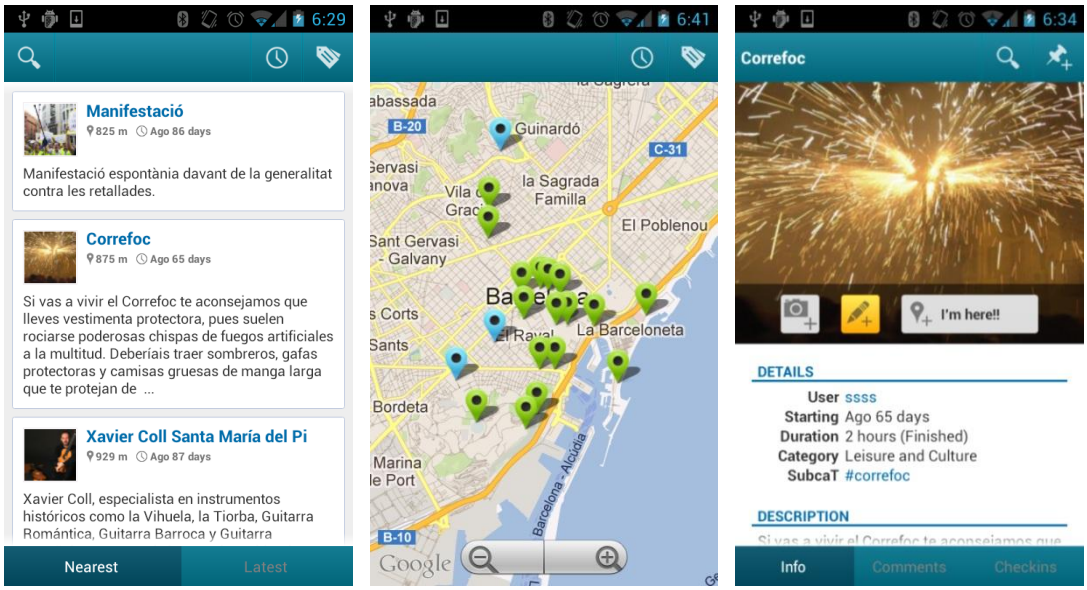

Figure 18 App screenshots

Figure 18 shows three screenshots of the application. All have a top bar with shortcuts, and the body of the application is light colored.

The first two screenshots show two ways to view the events. The first is a list sorted by proximity or time with basic information about the event. The second is the representation of events on a map, with icons of different colours depending on the status of the event.

The third is the screen capture of the event information, which shows its main image, the actions to take, their details, description and map. In addition it have multiple tabs to view its comments and its check-ins.

5.1.2. Web

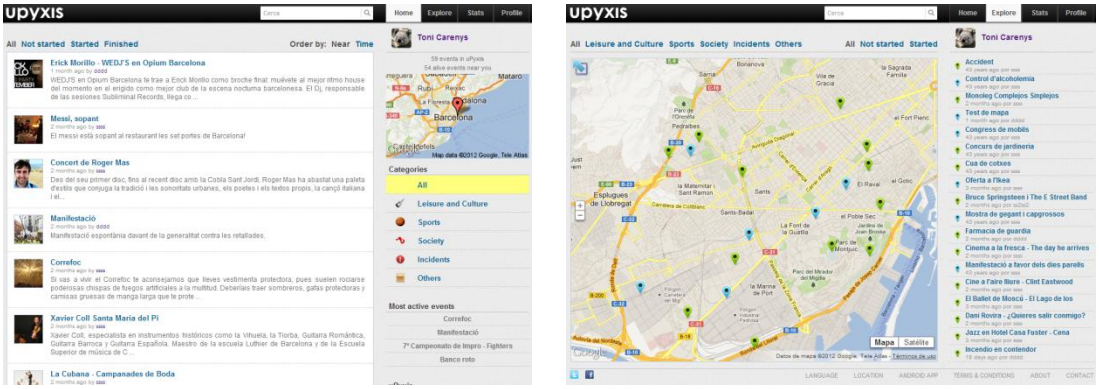

Figure 19 Website screenshots

The *Figures 19 and 20* shows four screenshots of the website. All have a fixed black bar above with the search field and a menu, and the body have a gray background and the main information is under boxes with white background.

The two screenshots in *Figure 19* represent two ways to view the events. The first is a list sorted by proximity or time with basic information about the event. The second is the representation of events on a map, with icons of different colours depending on the status of the event, and the list of the events represented on the right.

The first capture of the *Figure 20* shows the screen with the information of the event, which has a structure similar to its equivalent on the application. The second is the screen with the user profile, in which appears its background image, its avatar, personal data, statistics and a menu to see his last activity

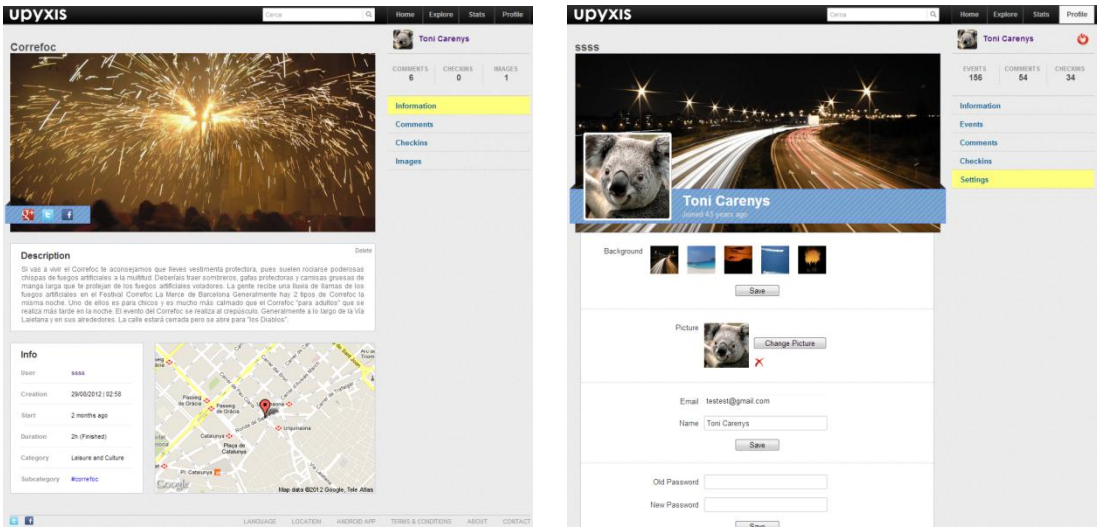

Figure 20 Website screenshots

5.2. Usability tests

Since who will use the website and the application can be very varied users, some more technical than others, it is necessary to know the ease of use of the application and the website.

To perform this check, we asked a group of people who use both the application and the website, at least requiring them to perform certain actions such as register, creating events, etc, and finally they will complete a short survey about the use of the website and the application, from which we can see if it has a good usability or not.

Figure 21 shows the results obtained in the survey related to the application. Detailed data can be found in the *Appendix VI.*

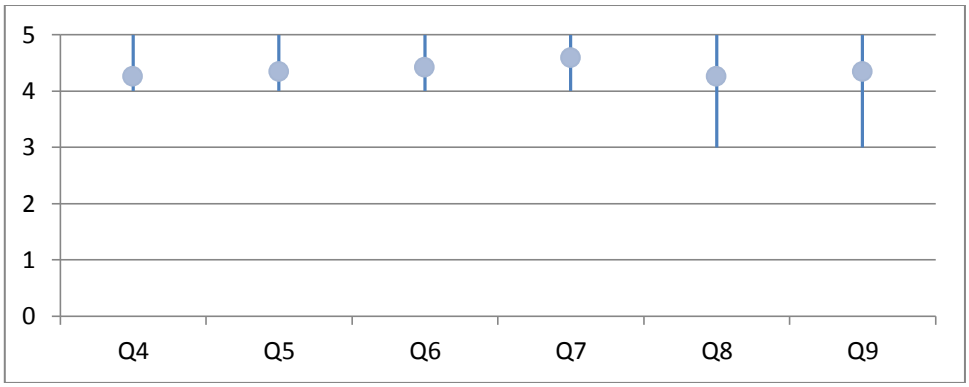

Figure 21 APP survey results

Figure 22 shows the results from the surveys related to the website. Detailed data can be found in *Appendix VI.*

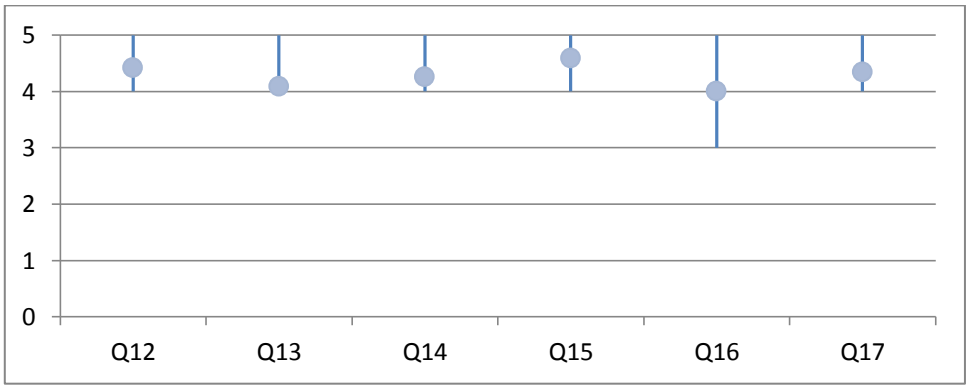

Figure 22 Website survey results

CHAPTER 6. CONCLUSIONS

During this project, we designed, developed and tested a social network service based on the location of activities and events happening right now around the user and that have a short duration. This service is based on three main basic pillars, information, real time, and geolocation.

Thanks to this project, was provided a useful service to curious people who want to know at all times what is happening near them, to people who do not know what activities to do at a given time, and to people that want to announce some events, incidents or activities that they see on some place.

After describing the idea and to choose the main objectives in the introductory chapter, on the second chapter have conducted an analysis of the current social networks and different technologies available to carry out our service. In that same chapter, we have chosen to do an application for Android, which communicates with a REST webservice that responds in XML format. The server side was chosen to do it in PHP using a MySQL database, and on the website was chosen to use the Google Maps service.

In the third chapter described architecture for the service, and all the necessary requirements of each of the parts of the service have been described, both functional and nonfunctional. In this same chapter was explained the use of the haversine formula to calculate distances between two coordinates, the designed tables of the databases, and also the design of the autocontrol system.

Once designed the service, in the fourth chapter was explained how was implemented each of the parts, explaining the final structure, the created files, and how to solve the problems encountered along their development, such as that end server did not accept the use of schedulers in the database.

Finally, in the fifth chapter have shown the final results, and has asked several users to answer a few questions after the test the service. The result of these questions have been very positive, achieving an average score of 4.32 out of 5, and has also served to see improvements or changes that users want.

6.1. Achieved objectives

When writing this document, can be considered to have achieved all the objectives described in the introduction, and that have also been able to introduce additional features.

Although as is normal during the development have appeared some problems, these have been solved and the service completed successfully, which is consumed from the Android app and website, and all such parts and functions developed contains requirements mentioned.

Moreover, when we was doing the service, has been decided to add new features that were not previously described on the introduction, such as autocontrol system that provides the possibility to users to report content in order to provide a service with quality content, or show of statistical charts and a heat map on the website.

6.2. Environmental impact

Directly, the environmental impact that can generate the service done in this thesis is not very important. Service directly affects on an increase of power consumption, by requiring servers to function, and computers or smartphones to be used. By generating this energy required to work, also influence the generation of greenhouse gases such as CO2.

Indirectly, the use of the service, could affect both positively and negatively to the environment, because the simple fact that people discover and attend certain activities shown in the service could lead to higher energy consumption for parties, or otherwise in environmental improvements if in the case of activities to save the nature.

Furthermore, the widespread use of the service could have a sociocultural impact on society. This impact could change the way people look for things to do and also may change its usual channels to gossip and know about incidents.

6.3. Future enhancements

A future technical improvement would be transform the webservice to public like an API, which from other applications could find events. In this case, would migrate the format in what data is returned, from XML to JSON, and in this way we gain an improvement in efficiency when returning large amounts of data.

Another technical improvement would be to hire a better server, where you could program the necessary schedulers in MySQL database to update the states of events and users automatically.

On the other hand, would have to add and improve service functional characteristics, such as improving the issue of user privacy, and put options to choose what activity can be shown in the profile, and what not. Other features that should be added is the ability to add friends or create user groups that tend to create or attend similar activities.

REFERENCES

[1] Wikipedia - “Social Networking Services”,
http://en.wikipedia.org/wiki/Social_networking

[2] IDC - “IDC Worldwide Mobile Phone Tracker”
http://www.idc.com/getdoc.jsp?containerId=prUS23638712

[3] Wikipedia - “Haversine formula”
http://en.wikipedia.org/wiki/Haversine_formula

[4] EUR-Lex - “Directive 2009/136/EC”
http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:01:en:HTML

BIBLIOGRAPHY

1- Stack Overflow,
http://www.stackoverflow.com/

2- Android developers official site,
http://developer.android.com

3- Google developers official site
https://developers.google.com/

4- W3Schools Online Web Tutorials,
http://www.w3schools.com/

5- SgOliver - “Curso de Programación Android”
http://www.sgoliver.net/

6- Wikipedia,
http://en.wikipedia.org/

7- Flot official site and tutorials
http://www.flotcharts.org/

8- Recursive awesome – “Implementing Pull To Refresh”
http://www.recursiveawesome.com/blog/2011/04/29/implementing-pull-to-refresh-in-your-android-app/

9- Geo developers blog
http://googlegeodevelopers.blogspot.com.es/

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

APPENDIX

TITLE: uPyxis, a real-time geolocated information network

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Antoni Carenys Roca

DIRECTOR: Dolors Royo Vallés

DATE: December 20th 2012

APPENDIX INDEX

LIST OF APPENDIX FIGURES

I DATABASES TABLES

For the project, are needed 2 types of databases, MySQL for the web server and SQLite for the Android's system. Now I will describe the structures of each table used for each type of database

.

I.1 Web Server

For the MySQL database of the web server, there are 9 different tables. Now I will explain the uses and the structure of each one.

I.1.1 Events

As shown in the *Figure I.1,* the event table is used to store all the information about an event such as its id, the id and IP of the creator, its coordinates, its details, number of check-ins, comments and images, its actual karma, etc.

Field	Type	Null	Default	Extra
id_event	int(11)	No	-	-
id_user	int(11)	No	-	-
latitude	float(10,6)	No	0.000000	-
longitude	float(10,6)	No	0.000000	-
type	int(3)	No	1	-
subtype	varchar(100)	-	-	-
start_time	timestamp	-	0	-
duration	int(11)	No	-	Minutes
title	varchar(200)	No	-	-
description	varchar(3000)	No	-	-
link	varchar(300)	No	0	-
image_main	varchar(50)	No	0	Name
status	int(3)	No	0	0: Hidden 1: Not started 2: Started 3: Finished 11: Del by user 12: Del by adm 13: Del by report
time_creation	timestamp	No	Current_ Timestamp	-
comments	int(10)	No	0	-
check-ins	int(10)	No	0	-
images	int(10)	No	0	-
karma	float(6,2)	No	*100.00*	-
IP	varchar(50)	No	-	-

Figure I.1 Structure of the table 'events'

I.1.1 Events_checkins

As shown in the *Figure I.2* **Structure of the table 'events_checkins'**, the event_checkin table is used to store all the information about a check-in such as its id, the id and IP of the creator, the related event, and the time of creation.

Field	Type	Null	Default	Extra
id_checkin	int(11)	No	-	-
id_user	int(11)	No	-	-
id_event	int(11)	No	-	-
time_creation	timestamp	No	Current_Timestamp	-
IP	varchar(50)	No	-	-

Figure I.2 Structure of the table 'events_checkins'

I.1.2 Events_comments

As shown in the *Figure I.3,* the event_comment table is used to store all the information about a comment such as its id, the id and IP of the creator, the related event, the time of creation, its status and its karma.

Field	Type	Null	Default	Extra
id_comment	int(11)	No	-	
id_user	int(11)	No	-	
id_event	int(11)	No	-	
time_creation	timestamp	No	Current_Timestamp	
message	varchar(2000)	No	-	
status	int(3)	No	0	0: Normal 11: Del by user 12: Del by admin 13: Del by report
karma	float(6,2)	No	*100.00*	
IP	varchar(50)	No	-	

Figure I.3 Structure of the table 'events_comments'

I.1.3 Events_images

As shown in the *Figure I.4,* the event_images table is used to store all the information about an image such as its id, the id and IP of the creator, the related event, its time of creation, its status and its karma and also if is the main image of an event or not.

Field	Type	Null	Default	Extra
id_image	int(11)	No	-	-
id_user	int(11)	No	-	-
id_event	int(11)	No	-	-
time_creation	timestamp	No	Current_Timestamp	-
img_name	varchar(50)	No	.	Image name
status	int(3)	No	0	0: Normal 11: Del by user 12: Del by admin 13: Del by report
karma	float(6,2)	No	*100.00*	-
IP	varchar(50)	No	-	-
ismain	tinyint(1)	No	0	0: Main 1: Secondary

Figure I.4 Structure of the table ‘events_images’

I.1.4 Users

As shown in the *Figure I.5,* the users table is used to store all the information about a user such as its id, all of its personal details, the related event, its time of creation, its status the customization details, its statistics and its karma and bans.

Field	Type	Null	Default	Extra
id_user	int(11)	No	-	-
nick	varchar(15)	-	-	-
password	varchar(32)	-	-	Md5
name	varchar(100)	-	-	-
email	varchar(200)	-	-	-
reg_state	int(11)	No	-	1: Not validated 2: Ok 11: Del by user 12: Ban by report 13: Ban by admin
image	tinyint(1)	No	0	0: Without avatar 1: With avatar
background	int(1)	No	1	-
session	varchar(32)	-	-	Md5
mSession	varchar(32)	-	-	Md5
last_visit	timestamp	No	Current_Timestamp	-
validation_code	varchar(32)	-	-	Md5 code for the verification mail
userEvents	int(11)	No	*0*	-
userComments	int(11)	No	*0*	-
userCheckins	int(11)	No	*0*	-
userImages	int(11)	No	*0*	-
time_creation	timestamp	No	0	-
Ip	varchar(50)	No	-	-
karma	float(6,2)	No	100.00	-
ban_num	int(3)	No	0	-
ban_fin	timestamp	No	0	-

Figure I.5 Structure of the table ‘users

I.1.5 Reports

As shown in the *Figure I.6,* the reports table is used to store all the information about a report that is done by a user. It stores its id, the user id, the id of the report destination, the type of report, etc.

This table is used to track the usage of the report system and to prevent the misuse of the same.

Field	Type	Null	Default	Extra
id_report	int(11)	No	-	-
id_user	int(11)	No	-	-
mode	int(3)	No	-	-
r_id_user	int(11)	-	-	-
r_id_event	int(11)	-	-	-
r_id_comment	int(11)	-	-	-
r_id_image	int(11)	-	-	-
type	int(3)	-	-	-
time_creation	timestamp	No	Current_Timestamp	-
ip	varchar(50)	No	.	-

Figure I.6 Structure of the table ‘reports’

I.1.6 Deletes

As shown in the *Figure I.7,* the reports table is used to store all the information about a delete that is done by a user. It stores its id, the user id, the id of the deleted item, etc.

This table is also used to track the usage of the deletes and to prevent the misuse of the same.

Field	Type	Null	Default	Extra
id_delete	int(11)	No	-	-
id_user	int(11)	No	-	-
mode	int(3)	No	-	-
d_id_event	int(11)	-	-	-
d_id_comment	int(11)	-	-	-
d_id_image	int(11)	-	-	-
d_id_checkin	int(11)	-	-	-
time_creation	timestamp	No	Current_Timestamp	-
ip	varchar(50)	No	*None*	-

Figure I.7 Structure of the table ‘deletes’

I.1.7 Comments

As shown in the *Figure I.8,* the comments table is used to store all the information about the comments that the visitors send to me about the webpage. It stores the id of each comment, the id of the user (if logged), and the different details of the same.

The comments are sent via email to me, but in case of failure of the email service, this table works as a backup for the comments.

Field	Type	Null	Default	Extra
id_note	int(11)	No	-	-
id_user	varchar(50)	No	-	-
name	varchar(50)	No	-	-
email	varchar(100)	No	-	-
message	text	No	-	-
time_creation	timestamp	No	Current_Timestamp	-
Ip	varchar(50)	No	-	-

Figure I.8 Structure of the table 'comments'

I.1.8 Cities

As shown in the *Figure I.9,* the cities table is used to store all the cities of Spain and its coordinates. It is used for the manual location selection if the users don't want to use the auto location system.

Field	Type	Null	Default	Extra
municipio	varchar(255)	No	-	-
provincia	varchar(255)	No	-	-
comunidad	varchar(255)	No	-	-
latitud	double	No	-	-
longitud	double	No	-	-

Figure I.9 Structure of the table 'cities'

I.2 Android

For the SQLite database of the Android system, there are 3 different tables. Now I will explain the uses and the structure of each one.

I.2.1 MainUser

As shown in the *Figure I.10,* the mainUser table is used to store the user id and its related session that is retrieved by the server after the login.

This table is used to remind the user when the application starts, and also the session is sent in every request that is made to the server.

Field	Type	Null	Default	Extra
_id	int	No	-	Primary
user_id	text	No	-	Primary
mSession	text	No	-	Unique

Figure I.10 Structure of the table ‘mainUser’

I.2.2 Alerts

As shown in the *Figure I.11,* the alerts table is used to store all the information about the alerts that the users define. It stores its id, all the details of the alert and the id of the user that creates the alert. This last field is important due to that if in a same smartphone logins more than one user, each user only can be see its defined alerts.

Field	Type	Null	Default	Extra
_id	int	No	-	Primary Autoinc.
user_id	text	No	-	-
name	text	No	-	-
start	text	No	-	-
duration	Int	No	-	-
title	text	-	-	-
cat	int	-	-	-
subcat	text	-	-	-
desc	text	-	-	-
dist	int	No	-	-

Figure I.11 Structure of the table ‘alerts’

I.2.3 alertResults

As shown in the *Figure I.12,* the alertsResults table is used to store all the information about the events sent by the server as a result of a petition of an alert. It stores an id for each result, its related alert id and all the details of the event.

Field	Type	Null	Default	Extra
_id	int	No	-	Primary Autoinc.
alert_id	int	No	-	-
event_id	text	No	-	-
title	text	No	-	-
cat	text	No	-	-
distance	text	-	-	-
description	text	-	-	-
starttime	text	-	-	-
duration	text			
image	text			
nick	text	-	-	-
Status	text	No	-	-

Figure I.12 Structure of the table ‘alertResults’

II WEBSERVICE

The following shows information regarding the use of the functions of the webservice. For each function is described its purpose and is showed a table with the input parameters and the required format for each of them, and the response format.

II.1 Session

To ensure that the requests come from users logged into the service, and to identify the user that makes the request, in all functions except the login and the register, there are two mandatory parameters, which are defined in the following *Figure II.1*.

Field	Longitude	Format	Optional	Type
mSession	32	0-9 a-f	no	GET
uid	1 - 5	0-9	no	GET

Figure II.1 Required parameters to validate the session

II.2 Functions

Listed below are all functions of the webservice, with its input parameters and output formats. In case of failure of the webservice or absence of any input parameters is returned an error message with the word "KO".

II.2.1 login.php

The login function is used to log into the service and obtain the session ID, input parameters are shown in *Figure II.2*.

Field	Longitude	Format	Optional	Type
nick	4 - 15	a-Z 0-9 _	no	GET
pss	6 - 20		no	GET

Figure II.2 Details of the parameters of login.php

This function returns the generated session number and the ID of the user in an XML in the following format:

```
<user>
    <info>
        <session>    </session>
        <id>    </id>
    </info>
</user>
```

These data will need to be sent in all future requests to validate the requests.

II.2.2 logout.php

The logout function is used to log out of the service and clear of the database server the session id associated with that user. No parameters are sent more than needed to identify the session mentioned above.

If everything goes well, this function returns an "OK".

II.2.3 register.php

The register function is used to register a new user on the service, the required input parameters are those of *Figure II.3*

Field	Longitude	Format	Optional	Type
nick	4 - 15	a-Z 0-9 _	no	GET
pss	6 - 20		no	GET
name	2 - 30		no	GET
email	5 - 200	email	no	GET

Figure II.3 Details of the parameters of register.php

If everything goes well, the new user is inserted and is sent an email with the account validation link. This function returns an "OK".

II.2.4 setEvent.php

SetEvent function is used to insert a new event in the service, the required input parameters are those shown in *Figure II.4.*

Field	Longitude	Format	Optional	Type
latitude	1 - 30	0-9 .	no	GET
longitude	1 - 30	0-9 .	no	GET
type	1	0-9	no	GET
starttime (minutes)	1 - 3	0-9	no	GET
duration (minutes)	1 - 3	0-9	no	GET
title	4 - 50		no	GET
description	20 - 1000		no	POST
subtype	1 - 31	0-9 a-Z _ #	yes	GET

Figure II.4 Details of the parameters of setEvent.php

If everything goes well, this function returns the id of the new event in an XML with the following format:

```
<events>
    <event>
        <ev_id>    </ev_id>
    </event>
</events>
```

II.2.5 setComment

SetComment function serves to add a comment concerning an event, the required input parameters are those shown in *Figure II.5.*

Field	Longitude	Format	Optional	Type
event_id	1 - 5	0-9	no	GET
message	1 - 500		no	POST

Figure II.5 Details of the parameters of setComment.php

If everything goes well, this function returns an "OK".

II.2.6 setCheckin

SetCheckin function is used to insert a new check-in from a user in an event that has not yet finished, the input parameters required are those shown in *Figure II.8*

Field	Longitude	Format	Optional	Type
event_id	1 - 5	0-9	no	GET

Figure II.6 Details of the parameters of setCheckin.php

If everything goes well, this function returns an "OK".

II.2.7 setImage

SetImage function is used to upload and insert a new image, either primary or not, in a new event, the input parameters required are those shown in *Figure II.7.*

Field	Longitude	Format	Optional	Type
event_id	1 - 5	0-9	no	GET

Figure II.7 Details of the parameters of setImage.php

If everything goes well and are generated the images in different sizes, this function returns an "OK".

II.2.8 getUser

GetUser function is used to request information about a user. Depending on the parameters that are sent, it can return only personal information, only its comments, only its events, only its check-ins, or everything. The input parameters required are those shown in *Figure II.8*

Field	Longitude	Format	Optional	Type
user_id	1 - 5	0-9	no	GET
info	1	0-1	yes	GET
ev	1	0-1		GET
ev_m	1	0-1	yes	GET
ev_l	1 - 5	0-9		GET
co	1	0-1		GET
co_m	1	0-1	yes	GET
co_l	1 - 5	0-9		GET
ch	1	0-1		GET
ch_m	1	0-1	yes	GET
ch_l	1 - 5	0-9		GET

Figure II.8 Details of the parameters of getUser.php

If is sent a "1" in the "info", returns the personal information.

If is sent a "1" in the "ev", returns the information about the different events created by that user. Depending on the parameters ev_l and ev_m, are sent recent or old events.

If is sent a "1" in the "co", returns the information about the various comments created by that user. Depending on the parameters co_l and co_m, are sent recent or old comments.

If is sent a "1" in the "ch", is returned different information about check-ins made by that user. Depending on the parameters sent ch_l and ch_m, are sent recent or old check-ins.

All this information is sent in an XML with the following format:

```
<user>
    <info>
        <id>  </id>
        <nick>  </nick>
        <image>  </image>
        <name>  </name>
    </info>
    <events>
        <event>
            <id>  </id>
            <type>  </type>
            <date>  </date>
            <title>  </title>
            <message>  </message>
            <image>  </image>
        </event>
    </events>
    <comments>
        <comment>
            <id>  </id>
            <ev_id>  </ev_id>
            <type>  </type>
            <date>  </date>
            <title>  </title>
            <message  </message>
            <image>  </image>
        </comment>
    </comments>
    <checkins>
        <checkin>
            <id>  </id>
            <ev_id>  </ev_id>
            <title>  </title>
            <image>  </image>
            <type>  </type>
        </checkin>
    </checkins>
</user>
```

II.2.9 getEvent

GetEvent function is used to request information from an event. Depending on the parameters that are sent, it can return only information of the event, only its checkins, only its images, or everything. The input parameters required are those shown in *Figure II.9.*

Field	Longitude	Format	Optional	Type
event_id	1 - 5	0-9	no	GET
info	1	0-1	yes	GET
co	1	0-1		GET
co_m	1	0-1	yes	GET
co_l	1 - 5	0-9		GET
ch	1	0-1		GET
ch_m	1	0-1	yes	GET
ch_l	1 - 5	0-9		GET
img	1	0-1		GET
img_m	1	0-1	yes	GET
img_l	1 - 5	0-9		GET

Figure II.9 Details of the parameters of getEvent.php

If is sent a "1" in the "info", it returns the personal information of the event.

If is sent a "1" in the "co", returns the information about the various comments created on this event. Depending on the parameters co_l and co_m, are sent recent or old comments.

If is sent a "1" in the "ch", returns the different information about checkins made to this event. Depending on the parameters sent ch_l and ch_m, are sent recent or old checkins.

If is sent a "1" in the "img", returns the different information about images of this event. Depending on the parameters sent img_l and img_m, are sent recent or old images.

All this information is sent in an XML with the following format:

```
<event>
    <info>
        <ev_id>  </ev_id>
        <us_id>  </us_id>
        <nick>  </nick>
        <starttime>  </starttime>
        <duration>  </duration>
        <status>  </status>
        <type>  </type>
        <subtype>  </subtype>
        <title>  </title>
        <description>  </description>
        <link>  </link>
        <image>  </image>
        <latitude>  </latitude>
        <longitude>  </longitude>
    </info>
    <us_checkin>
        <isCheckin>  </isCheckin>
    </us_checkin>
    <comments>
        <comment>
            <id>  </id>
            <us_id>  </us_id>
            <nick>  </nick>
            <image>  </image>
            <date>  </date>
            <message>  </message>
        </comment>
    </comments>
    <checkins>
        <checkin>
            <id>  </id>
            <us_id>  </us_id>
            <nick>  </nick>
            <image>  </image>
        </checkin>
    </checkins>
    <images>
        <image>
            <id>  </id>
            <nick>  </nick>
            <name>  </name>
        </image>
    </images>
</event>
```

II.2.10 getLocations

GetLocations function is used to request a list of events close to a set of coordinates and that meet a set of filters. The input parameters required, are those shown in *Figure II.10.*

Field	Longitude	Format	Optional	Type
latitude	1 - 30	0-9 .	no	GET
longitude	1 - 30	0-9 .	no	GET
type	1	0-9	no	GET
status	1	0-9	no	GET
mode	1	0-9	no	GET
dis	1 - 3	0-9	no	GET
num	1 - 2	0-9	no	GET
ev_m	1	0-1	no	GET
ev_l	1 - 10	0-9 .	no	GET

Figure II.10 Details of the parameters of getLocations.php

This function returns a list with information about the various events that meet the requirements. This information is sent in an XML with the following format:

```
<locations>
    <location>
        <id>  </id>
        <title>  </title>
        <type>  </type>
        <distance>  </distance>
        <desc>  </desc>
        <starttime>  </starttime>
        <duration>  </duration>
        <image>  </image>
        <nick>  </nick>
        <status>  </status>
    </location>
</locations>
```

II.2.11 getLocationsAlert

GetLocationsAlert function is used to request a list of events close to a set of coordinates and that meet a set of filters. The input parameters required, are those shown in *Figure II.11.*

Field	Longitude	Format	Optional	Type
latitude	1 - 30	0-9 .	no	GET
longitude	1 - 30	0-9 .	no	GET
title	0-4 - 50		yes	GET
type	1	0-9	no	GET
subtype	0-1 - 31	0-9 a-Z _ #	yes	GET
status	1	0-9	yes	GET
desc	0-4 - 50		yes	GET
dis	1 - 3	0-9	no	GET

Figure II.11 Details of the parameters of getLocationsAlert.php

This function returns a list with information about the various events that meet the requirements. This information is sent in an XML with the following format:

```
<alertLocations>
    <location>
        <id>  </id>
        <title>  </title>
        <type>  </type>
        <distance>  </distance>
        <desc>  </desc>
        <starttime>  </starttime>
        <duration>  </duration>
        <image>  </image>
        <nick>  </nick>
        <status>  </status>
    </location>
</alertLocations>
```

II.2.12 getLocationsMap

GetLocationsMap function is used to request a list of events close to a set of coordinates and that meet a set of filters. The information returned will be used to display these events on a map. The input parameters required, are those shown in *Figure II.12.*

Field	Longitude	Format	Optional	Type
latitude	1 - 30	0-9 .	no	GET
longitude	1 - 30	0-9 .	no	GET
type	1	0-9	no	GET
status	1	0-9	no	GET
dis	1 - 3	0-9	no	GET
num	1 - 2	0-9	no	GET

Figure II.12 Details of the parameters of getLocationsMap.php

This function returns a list with information about the various events that meet the requirements. This information is sent in an XML with the following format:

```
<locations>
    <location>
        <id>  </id>
        <title>  </title>
        <type>  </type>
        <distance>  </distance>
        <status>  </status>
        <latitude>  </latitude>
        <longitude>  </longitude>
    </location>
</locations>
```

II.2.13 getLocationsSearch

GetLocationsSearch function is used to request a list of events close to a set of coordinates and that meets a set of filters and satisfy a search. The input parameters required are those shown in *Figure II.13.*

Field	Longitude	Format	Optional	Type
latitude	1 - 30	0-9 .	no	GET
longitude	1 - 30	0-9 .	no	GET
searchWord	3 - 50		no	GET
dis	1 - 3	0-9	no	GET
num	1 - 2	0-9	no	GET
ev_m	1	0-1	no	GET
ev_l	1 - 10	0-9 .	no	GET

Figure II.13 Details of the parameters of getLocationsSearch.php

This function returns a list with information about the various events that meet the requirements. This information is sent in an XML with the following format:

```
<locations>
    <location>
        <id>  </id>
        <title>  </title>
        <type>  </type>
        <distance>  </distance>
        <desc>  </desc>
        <starttime>  </starttime>
        <duration>  </duration>
        <image>  </image>
        <nick>  </nick>
        <status>  </status>
    </location>
</locations>
```

II.2.14 getLocationsSearchAdv

GetLocationsSearch function is used to request a list of events close to a set of coordinates and that meet a set of filters and a more complex search. The input parameters required are those shown in *Figure II.14.*

Field	Longitude	Format	Optional	Type
latitude	1 - 30	0-9 .	no	GET
longitude	1 - 30	0-9 .	no	GET
title	4 - 50		yes	GET
type	1	0-9	no	GET
subtype	1 - 31	0-9 a-Z _ #	yes	GET
status	1	0-9	yes	GET
desc	4 - 50		yes	GET
dis	1 - 3	0-9	no	GET
num	1 - 2	0-9	no	GET
ev_m	1	0-1	no	GET
ev_l	1 - 10	0-9 .	no	GET

Figure II.14 Details of the parameters of getLocationsSearchAdv.php

This function returns a list with information about the various events that meet the requirements. This information is sent in an XML with the following format:

```
<locations>
    <location>
        <id>  </id>
        <title>  </title>
        <type>  </type>
        <distance>  </distance>
        <desc>  </desc>
        <starttime>  </starttime>
        <duration>  </duration>
        <image>  </image>
        <nick>  </nick>
        <status>  </status>
    </location>
</locations>
```

III DIAGRAMS

III.1 APP

Start & login & register

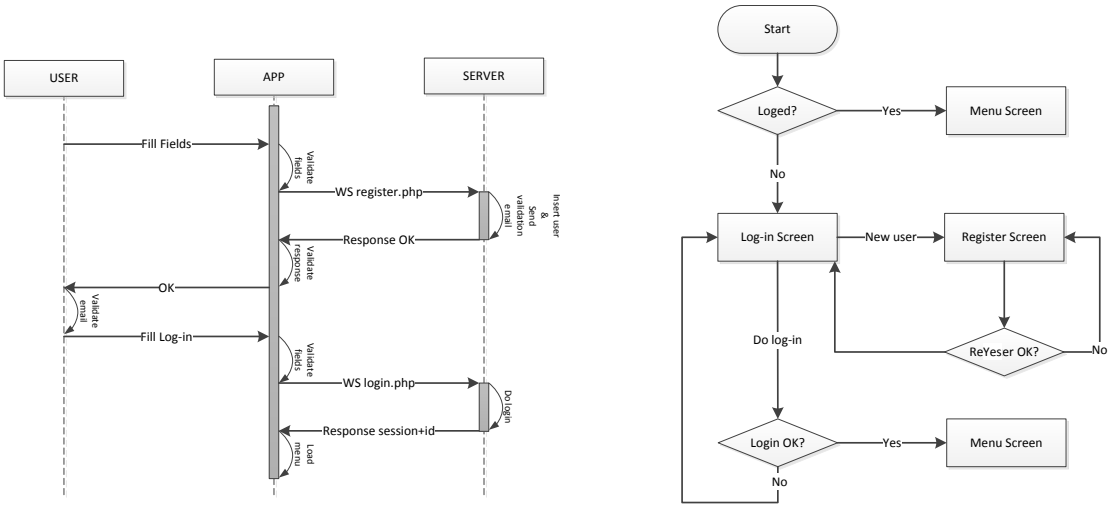

Figure III.1 Start & Login & Register diagrams

Show list

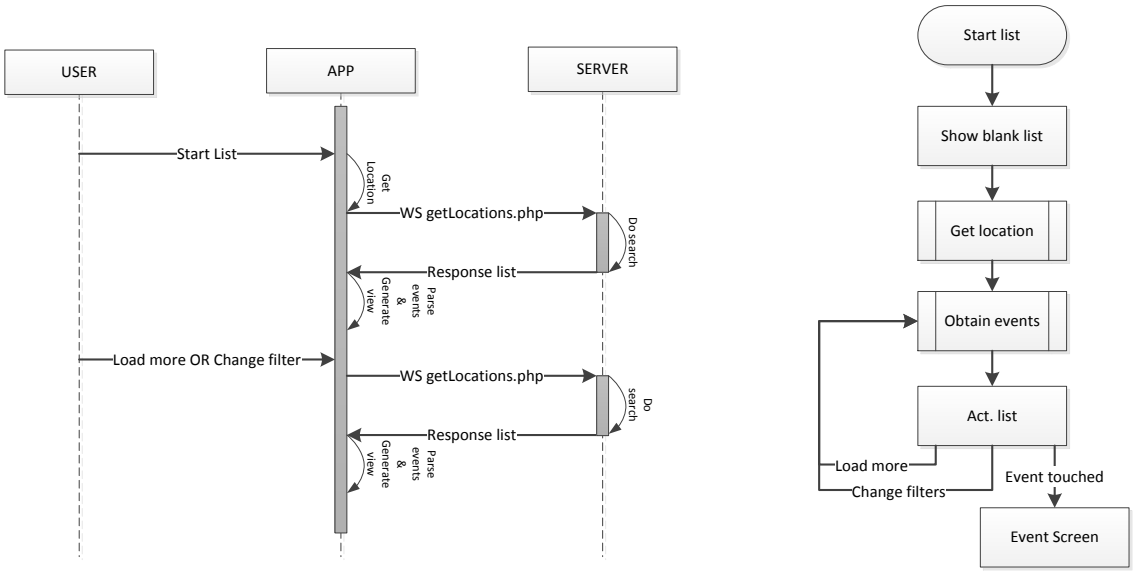

Figure III.2 Show list diagrams

Show map

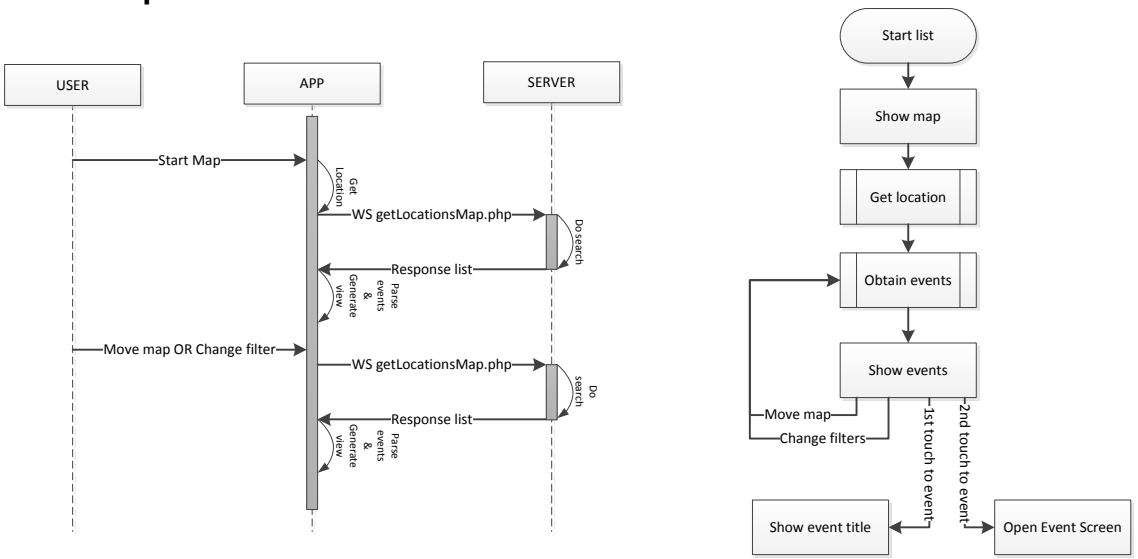

Figure III.3 Show map diagrams

Show event / Interact with event

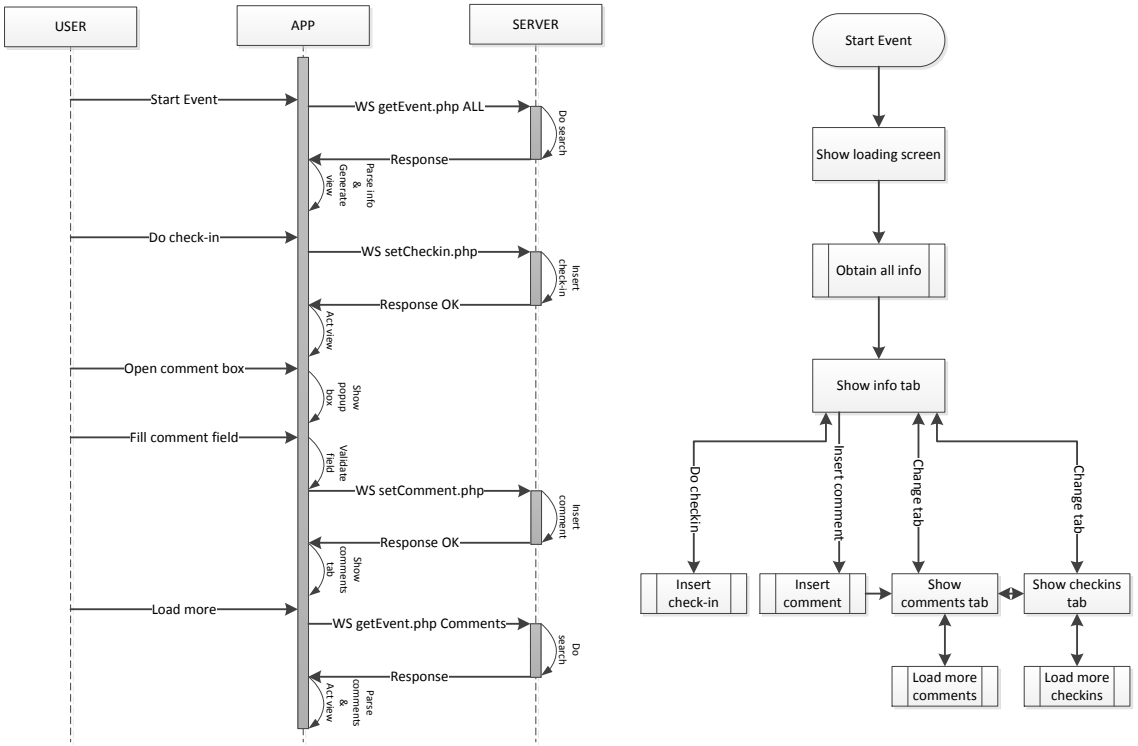

Figure III.4 Show and intercat with event diagrams

Show user

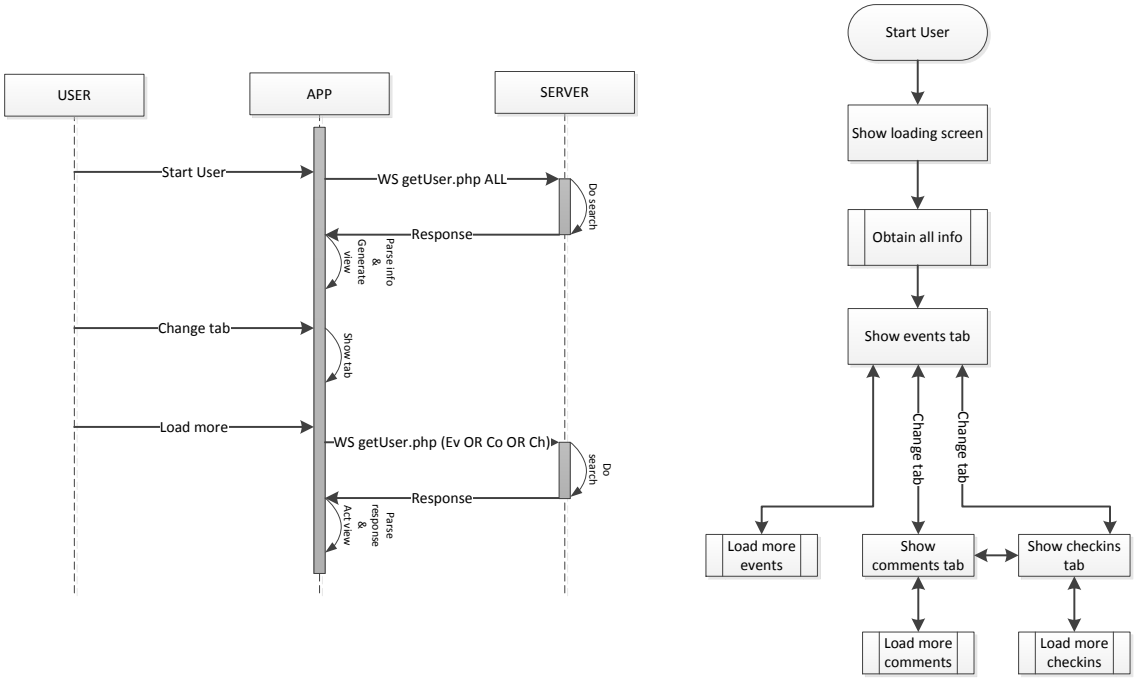

Figure III.5 Show user diagrams

Create Event

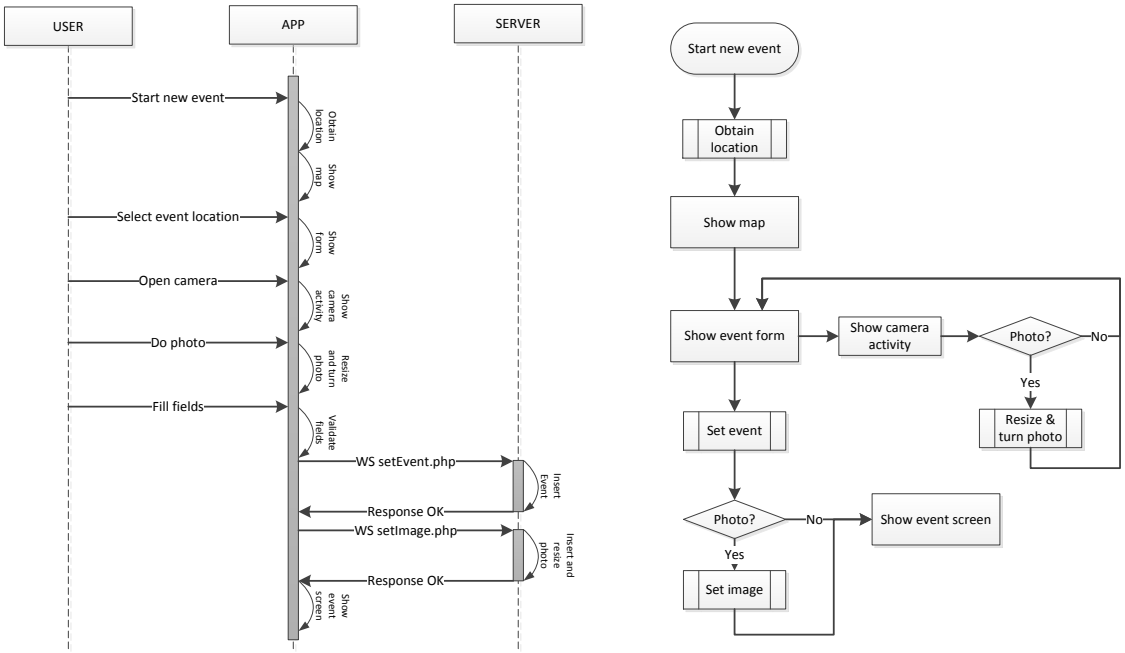

Figure III.6 Create event diagrams

Create alert

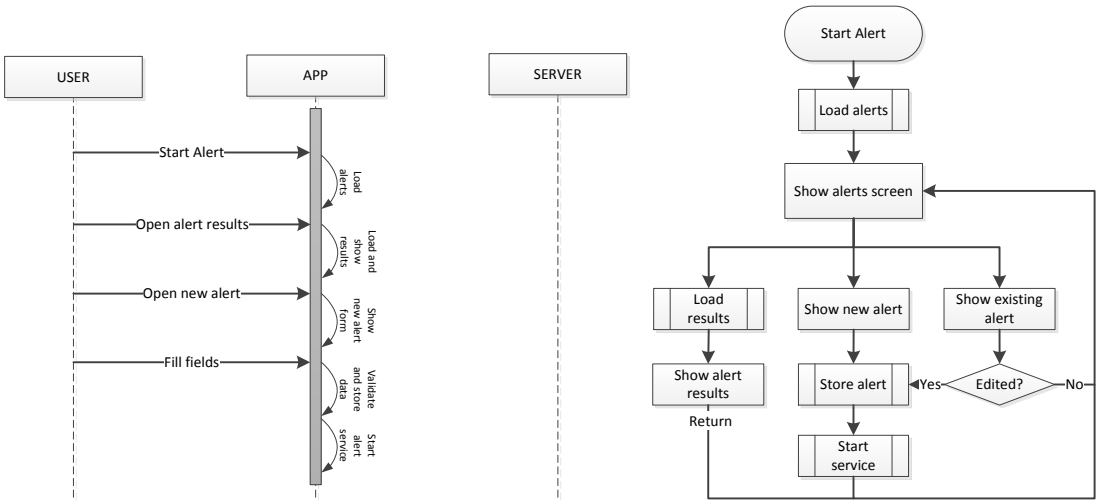

Figure III.7 Show alert diagrams

Run alert service

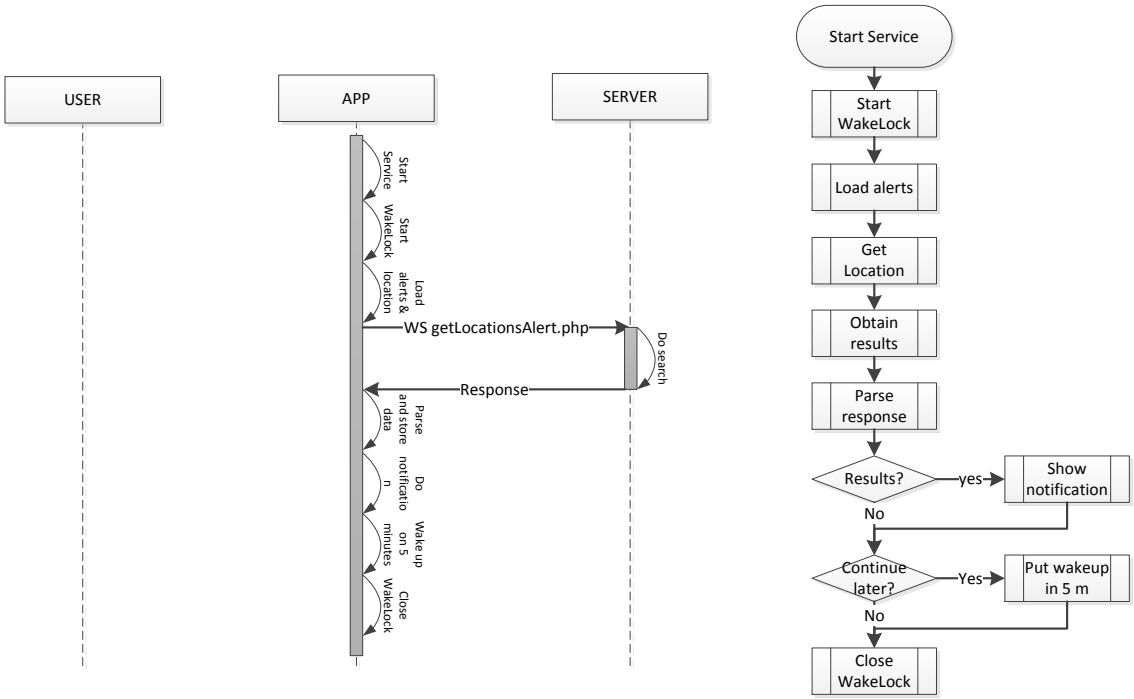

Figure III.8 Alert service diagrams

III.2 WEB

Login / Register

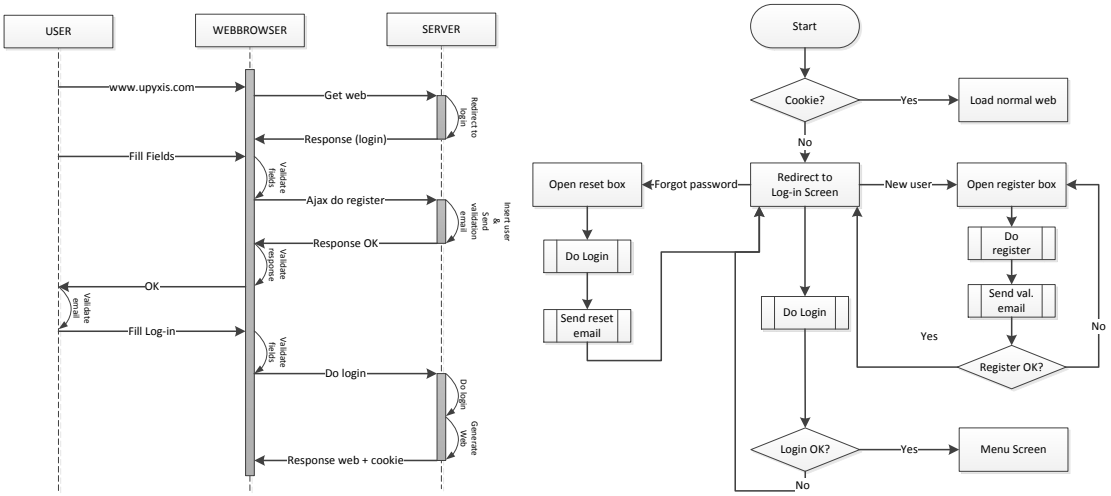

Figure III.9 Login & register diagrams

List events

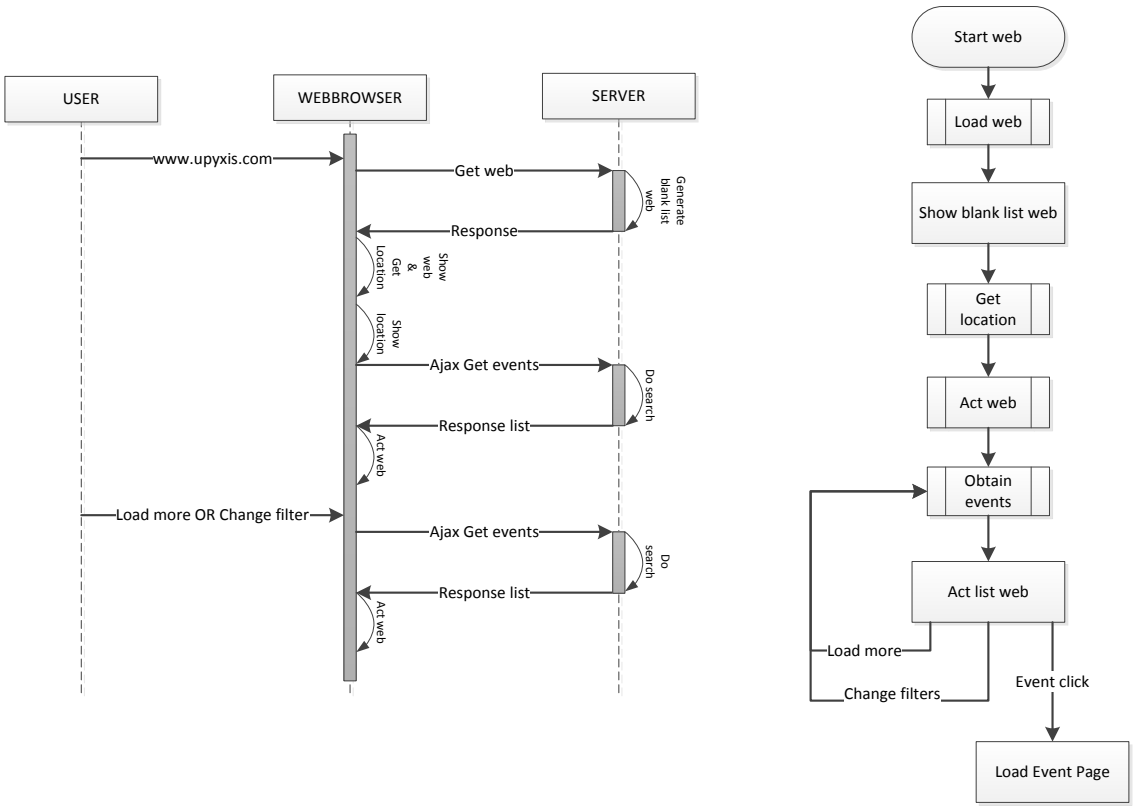

Figure III.10 List events diagrams

Explore

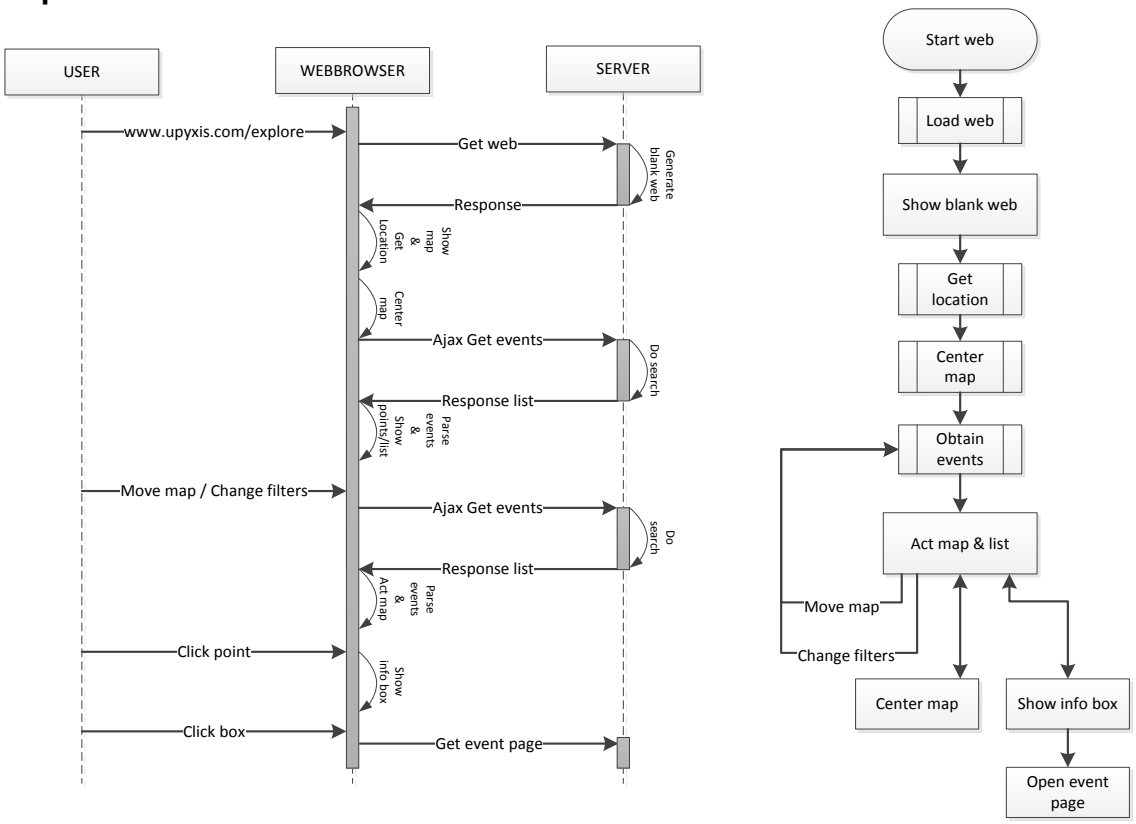

Figure III.11 Explore diagrams

Show statistics

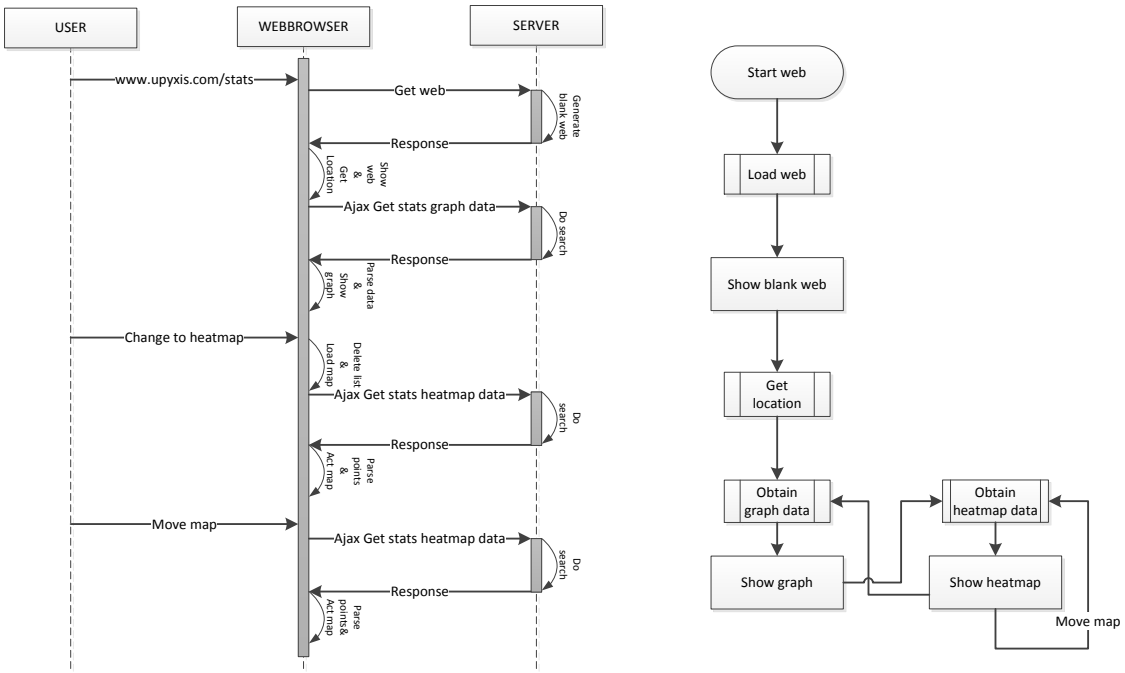

Figure III.12 Show statistics diagrams

Show event / Event interaction

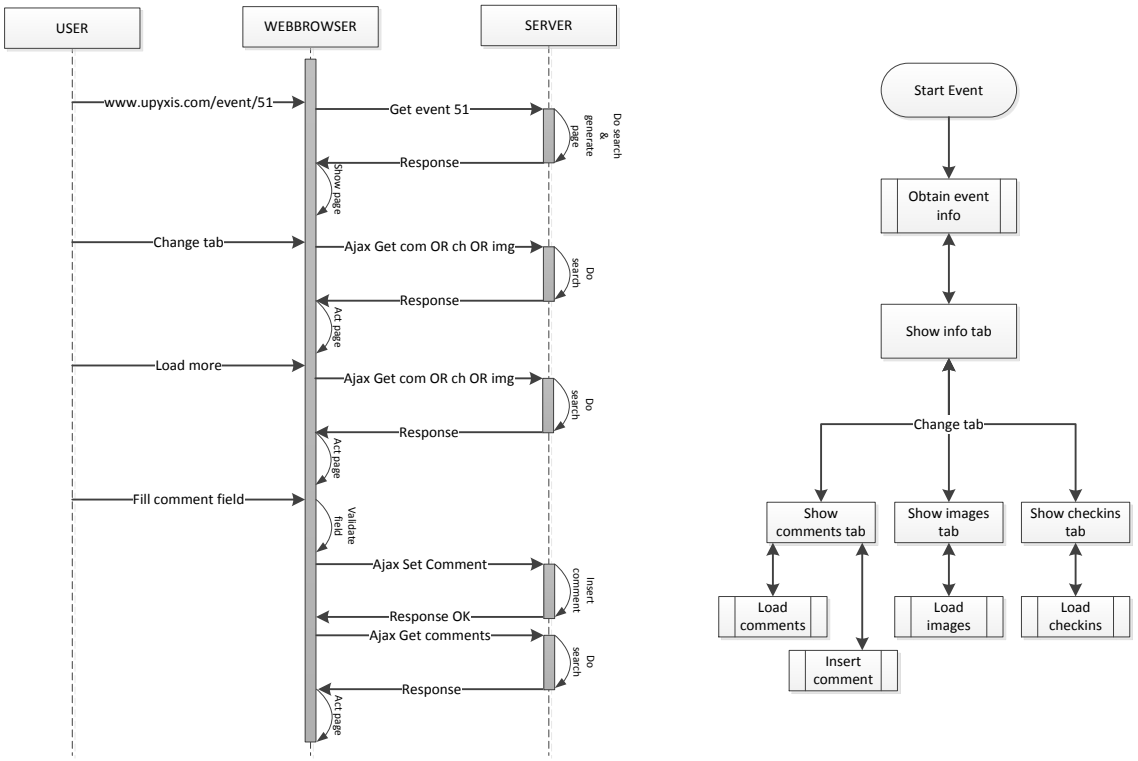

Figure III.13 Show and interact with event diagrams

Show user / User interaction

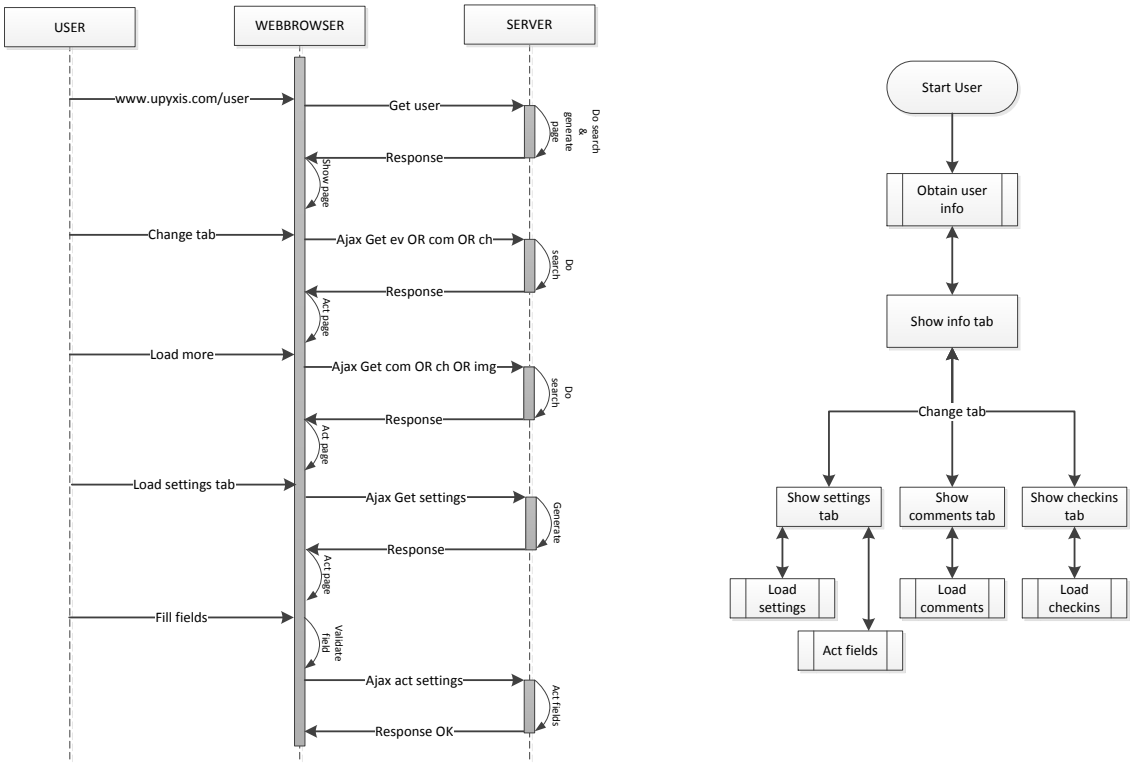

Figure III.14 Show and edit user diagrams

IV IMPLEMENTATION

The following will list the different files developed for this project. You can them grouped in tables by their location and type of file. For each file there is a little description of the use or finality of them

IV.1 APP

First there is listed the files regarding to the Android application. We will separate them in four groups, Presentation, Logic, Data and resources.

As you can see on *Figure IV.1*, an important file is the AndroidManifest. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code

****	
AndroidManifest.xml	Manifest file

Figure IV.1 Manifest file location

IV.1.1 Presentation

The following tables show the developed files that describes the structure of the user interface or its components.

\res\menu:	
menu_llista.xml	Structure of the menu used on the list screen
menu_map.xml	Structure of the menu used on the map screen
menu_newevent.xml	Structure of the menu used on the new event screen

Figure IV.2 Files of the menu folder

\res\layout-land:	
open_cam.xml	Layout of the screen that shows the camera
principal.xml	Horizontal layout of the main menu screen

Figure IV.3 Files of the layout-land folder

\res\layout:	
action_bar.xml	Layout of the main action bar
action_bar_alert.xml	Layout of the action bar of the alert's main screen
action_bar_alert_new.xml	Layout of the action bar of the new alert screen
action_bar_alert_results.xml	Layout of the action bar of the alert results screen
action_bar_event.xml	Layout of the action bar of the event description screen
action_bar_list.xml	Layout of the action bar of the list screen
action_bar_map.xml	Layout of the action bar of the map screen
action_bar_profile.xml	Layout of the action bar of the user profile screen
action_bar_search.xml	Layout of the action bar of the search screen
alert.xml	Layout of the alert main screen
alert2.xml	Layout of the new alert screen
alert_results.xml	Layout of the alert results screen
event.xml	Layout of the event description screen
llista.xml	Layout of the list screen
main.xml	Layout of the login screen
map.xml	Layout of the map screen
newevent.xml	Layout of the new event map screen
newevent2.xml	Layout of the new event form screen
newevent_textview.xml	Layout of the grid elements for time selection
newevent_textview_cat.xml	Layout of the grid elements for category selection
open_image.xml	Layout for the expanded image screen
popup.xml	Layout of the popup to insert a comment
principal.xml	Layout of the main menu screen
profile.xml	Layout of the user profile screen
pull_to_refresh_header.xml	Layout of the headers of the listviews
register.xml	Layout of the register screen
register_terms.xml	Layout of the terms and condition screen
row_alert.xml	Layout of the rows of the alert main screen
row_alert_results.xml	Layout of the rows of the alert results screen
row_alert_spinner.xml	Layout of the rows of the spinner of the new alert screen
row_event_checkin.xml	Layout of the rows of the checkins of the event screen
row_event_comment.xml	Layout of the rows of the comments of the event screen
row_event_image.xml	Layout of the rows of the images of the event screen
row_event_info.xml	Layout of the rows of the event screen
row_llista_event.xml	Layout of the rows of the events of the list screen
row_llista_loading.xml	Layout of the row with a loading icon
row_profile_checkin.xml	Layout of the rows of the checkins of the profile screen
row_profile_comment.xml	Layout of the rows of the comments of the profile screen
row_profile_event.xml	Layout of the rows of the events of the profile screen
row_submenu.xml	Layout of the rows of the popup menu
search.xml	Layout of search screen
search_advance.xml	Layout of advanced search screen
search_advance_form.xml	Layout of form of the advanced search screen

Figure IV.4 Files of the layout folder

IV.1.2 Logic

The logic layer contains the code that processes the data, and in our case we have divided the programmed files in this layer into four folders.

The files of the *Figure IV.5* shows the files that are responsible of the loading of the interface, and perform the main logic regarding this screen

\src\com\upyxis\main:	
event.java	Event activity
llista.java	List activity
main.java	Login activity
map.java	Map activity
newAlert.java	Alert main activity
newAlert2.java	New alert activity
newAlert_results.java	Alert results activity
newEvent.java	New event activity
openImage.java	Expanded image activity
principal.java	Main menu activity
profile.java	User profile activity
register.java	Register activity
register_terms.java	Terms and conditions activity
search.java	Search activity
searchAdv.java	Advanced search activity

Figure IV.5 Files of the main folder

The files of the *Figure IV.6* are the responsible of receive a list of data items, such events or comments and to decide how to display each of the items in a list on the screen.

\src\com\upyxis\adapters:	
event_checkinAdapter.java	Adapter to show the checkins on the event screen
event_commentAdapter.java	Adapter to show the comments on the event screen
event_imageAdapter.java	Adapter to show the images on the event screen
llista_alertAdapter.java	Adapter to show the alerts on the alert screen
llista_alertResultsAdapter.java	Adapter to show the alert results on the alert screen
llista_infoAdapter.java	Adapter to show the events on the list screen
profile_checkinAdapter.java	Adapter to show the checkins on the user profile screen
profile_commentAdapter.java	Adapter to show the comments on the user profile screen
profile_eventAdapter.java	Adapter to show the events on the user profile screen
viewHolder.java	Structure of data

Figure IV.6 Files of the adapters folder

The files of the *Figure IV.7* are files with the structure of various objects that are to be used as events, user profiles, map layers, etc.

\src\com\upyxis\objects:	
alert.java	Structure of the alert object
checkin.java	Structure of the checkin object
eventAndComment.java	Structure of the event or comment object
image.java	Structure of the image object
info.java	Structure of the event info object
mapOverlay.java	Structure of the map overlay object
newEvent_mapOverlay.java	Structure of the event object of a mapoverlay

Figure IV.7 Files of the objects folder

The files of the *Figure IV.8* are files with general functions.

\src\com\upyxis\others:	
PullToRefreshListView.java	Functions of an advanced listview
alertService.java	Service that manages the alerts requests
cache.java	Cache system and its functions
constants.java	File with all the needed constants
funcions.java	File with other general functions

Figure IV.8 Logic files of the others folder

IV.1.3 Data

In the data layer we programmed files that are used to retrieve data.

\src\com\upyxis\others:	
database.java	File that manages the database and that interact with it
webservice.java	File that sends the requests to the webservice
xmlParser.java	File that parses the responses of the webservice and generates list of objects

Figure IV.9 Data files of the others folder

IV.1.4 Resources

The *Figure IV.10* shows the files that contains the strings in each available language in the application. We can see that there are three folders, one for each language.

\res\values-ca:	
strings.xml	Catalan strings
\res\values-es:	
strings.xml	Spanish strings
\res\values:	
attrs.xml	Attributes
strings.xml	English strings

Figure IV.10 Files of the values folder

The next figures show the folders that contains the images in different resolutions for different densities. Also there are some files that describe styles of elements.

\res\drawable-ldpi:	
ic_launcher.png	Icon
ic_menu_globe.png	Icon
ic_menu_mylocation.png	Icon
ic_menu_smalltiles.png	Icon
ic_pulltorefresh_arrow.png	Icon

Figure IV.11 Files of the drawable-ldpi folder

\res\drawable-mdpi:	
action_camera.png	Icon
action_remove.png	Icon
action_search.png	Icon
action_search2.png	Icon
action_settings.png	Icon
collections_labels.png	Icon
device_access_time.png	Icon
device_access_time2.png	Icon
ic_launcher.png	Icon
ic_menu_globe.png	Icon
ic_menu_mylocation.png	Icon
ic_menu_smalltiles.png	Icon
ic_pulltorefresh_arrow.png	Icon
location_place.png	Icon
noti_icon.png	Icon

Figure IV.12 Files of the drawable-mdpi folder

\res\drawable-hdpi:	
action_camera.png	Icon
action_remove.png	Icon
action_search.png	Icon
action_search2.png	Icon
action_settings.png	Icon
collections_labels.png	Icon
device_access_time.png	Icon
device_access_time2.png	Icon
ic_launcher.png	Icon
ic_menu_globe.png	Icon
ic_menu_mylocation.png	Icon
ic_menu_smalltiles.png	Icon
ic_pulltorefresh_arrow.png	Icon
location_place.png	Icon
noti_icon.png	Icon
pressed_application.png	Icon

Figure IV.13 Files of the drawable-hdpi folder

\res\drawable-xhdpi:	
action_camera.png	Icon
action_remove.png	Icon
action_search.png	Icon
action_search2.png	Icon
action_settings.png	Icon
collections_labels.png	Icon
device_access_time.png	Icon
device_access_time2.png	Icon
ic_launcher.png	Icon
location_place.png	Icon
noti_icon.png	Icon

Figure IV.14 Files of the drawable-xhdpi folder

\res\drawable:	
action_bar.xml	Action bar style
action_bar_item.xml	Action bar item style
action_bar_item_normal.xml	Action bar item style
action_bar_item_pressed.xml	Action bar item style
action_bar_sep.xml	Action bar separator style
action_bar_sep2.xml	Action bar separator style
barra_event.xml	Style of the separator bar of the event screen
button.xml	General button style
button_checkin.png	Icon
button_checkin2.png	Icon
button_comment.png	Icon

button_comment2.png	Icon
button_noborder.xml	No border button style
button_normal.xml	Button style
button_normal_disabled.xml	Button style
button_normal_green.xml	Button style
button_normal_red.xml	Button style
button_photo.png	Icon
button_photo2.png	Icon
corners_event_comments.xml	Style of the corners of the comments rows
corners_image.xml	Style of the corners of the images rows
icon.png	Icon
llista_alert_active.xml	Style of the rows of the alert screen
llista_alert_active_normal.xml	Style of the rows of the alert screen
llista_alert_nactive.xml	Style of the rows of the alert screen
llista_alert_nactive_normal.xml	Style of the rows of the alert screen
llista_alert_pressed.xml	Style of the rows of the alert screen
llista_comment.xml	Style of the rows of the list screen
llista_comment_normal.xml	Style of the rows of the list screen
llista_comment_pressed.xml	Style of the rows of the list screen
llista_event.xml	Style of the rows of the list screen
llista_event_normal.xml	Style of the rows of the list screen
llista_event_pressed.xml	Style of the rows of the list screen
logo2.png	Icon
logo3.png	Icon
map_pin_blue.png	Icon
map_pin_blue2.png	Icon
map_pin_green.png	Icon
menu_alert2.png	Icon
menu_item.xml	Style of an item of the main menu
menu_item_normal.xml	Style of an item of the main menu
menu_item_pressed.xml	Style of an item of the main menu
menu_list2.png	Icon
menu_map2.png	Icon
menu_new2.png	Icon
opacity_event_images.xml	Opacity of an image
pull_to_refresh_header_background.xml	Background of the list header
remove.png	Icon
seekbar_background.9.png	Icon
seekbar_progress.xml	Style of a seekbar
seekbar_progress_bg.xml	Style of a seekbar
set_image.png	Icon
stripe_bg.jpg	Icon
tabwidget.xml	Style of the tabwidget
tabwidget_normal.xml	Style of the tabwidget
tabwidget_pressed.xml	Style of the tabwidget
transparent.png	Icon

Figure IV.15 Files of the drawable folder

IV.2 SERVER

Next, there are the files developed in the server. We show these files divided on three groups, Main, Webservice and Resources. An important file is the htacces.

\

.htaccess

```
RewriteEngine On
Options +FollowSymLinks
RewriteRule ^event/([0-9]+)$ /event.php?e=$1 [L]
RewriteRule ^user$ /user.php [L]
RewriteRule ^user/(.*)$ /user.php?u=$1 [L]
RewriteRule ^explore$ /explore.php [L]
RewriteRule ^explore/$ /explore.php [L]
RewriteRule ^stats$ /stats.php [L]
RewriteRule ^stats/$ /stats.php [L]
RewriteRule ^login$ /login.php [L]
RewriteRule ^login/$ /login.php [L]
RewriteRule ^login$ /login.php [L]
RewriteRule ^search/subcat/(.*)$ /index.php?search=\%23$1 [L]
RewriteRule ^search/(.*)$ /index.php?search=$1 [L]
# Evitar escaneos y manipulaciones malintencionadas de la URL (Ataques de inyección)
RewriteCond %{HTTP_USER_AGENT} ^$ [OR]
RewriteCond %{HTTP_USER_AGENT} ^(-|\.|') [OR]
RewriteCond %{HTTP_USER_AGENT} ^(.*)(<|>|%3C|%3E)(.*) [NC,OR]
RewriteCond %{HTTP_USER_AGENT} ^(java|curl|wget)(.*) [NC,OR]
RewriteCond %{HTTP_USER_AGENT} ^(.*)(libwww-
perl|libwwwperl|snoopy|curl|wget|winhttp|python|nikto|scan|clshttp|archiver|loader|em
ail|harvest|fetch|extract|grab|miner|suck|reaper|leach)(.*) [NC,OR]
RewriteCond %{REQUEST_URI} ^(/,|/;|/<|/>|/'|/`|/%2C|/%3C|/%3E|/%27|/////) [NC,OR]
RewriteCond %{HTTP_REFERER}
^(.*)(%00|%08|%09|%0A|%0B|%0C|%0D|%0E|%0F|%2C|<|>|'|%3C|%3E|%26%23|%27|%60
)(.*) [NC,OR]
RewriteCond %{QUERY_STRING}
^(.*)(%00|%08|%09|%0A|%0B|%0C|%0D|%0E|%0F|%2C|%3C|%3E|%27|%26%23|%60)(.*)
[NC,OR]
RewriteCond %{QUERY_STRING} ^(.*)('|-|<|>|,|/|\\|\.a|\.c|\.t|\.d|\.p|\.i|\.e|\.j)(.*)
[NC,OR]
RewriteCond %{HTTP_COOKIE} ^(.*)(<|>|'|%3C|%3E|%27)(.*) [NC]
RewriteRule ^(.*)$ error.php [NC]
# Evitar que se liste el contenido de los directorios
Options All -Indexes
IndexIgnore *
#Denegar el acceso a robots dañinos, browsers offline, etc
RewriteBase /
RewriteCond %{HTTP_USER_AGENT} ^Anarchie [OR]
RewriteCond %{HTTP_USER_AGENT} ^ASPSeek [OR]
RewriteCond %{HTTP_USER_AGENT} ^attach [OR]
RewriteCond %{HTTP_USER_AGENT} ^autoemailspider [OR]
RewriteCond %{HTTP_USER_AGENT} ^Xaldon\ WebSpider [OR]
RewriteCond %{HTTP_USER_AGENT} ^Xenu [OR]
RewriteCond %{HTTP_USER_AGENT} ^Zeus.*Webster [OR]
RewriteCond %{HTTP_USER_AGENT} ^Zeus
RewriteRule ^.*$ http://www.otraweb.com [R,L]
# Protegerse contra los ataques DOS limitando el tamaño de subida de archivos
LimitRequestBody 10240000
```

Figure IV.16 Content of the .htacces file

IV.2.1 Main

In the next *Figure IV.17* there are the files developed for the webpage.

\	
_DBconnect.php	Used to connect to the database
_IPobten.php	Used to obtain the IP a request
_constants.php	Contains the constants
_functions.php	Main functions
_loadScripts.php	Insert the html with the required scripts of each page
_readCookie.php	Functions that validates the session of the requests
_strings.php	English strings
_strings_CA.php	Catalan strings
_strings_ES.php	Spanish strings
event.php	Generates the layout of the event screen
event_f.php	Functions used for the event screen
explore.php	Generates the layout of the explore screen
explore_f.php	Functions used for the explore screen
index.php	Generates the layout of the list screen
index_f.php	Functions used for the list screen
login.php	Generates the layout of the login screen
login_f.php	Functions used for the login screen
newEvent.php	To insert a new event thought the webpage
stats.php	Generates the layout of the statistics screen
stats_f.php	Functions used for the statistics screen
upload.php	To upload an image thought the webpage
user.php	Generates the layout of the user profile screen
user_f.php	Functions used for the user profile screen
web_about.php	Generates the layout of the about popup screen
web_android.php	Generates the layout of the app popup screen
web_contact.php	Generates the layout of the contact popup screen
web_delete.php	Generates the layout of the delete popup screen
web_delete_s.php	Functions used for the delete screen
web_footer.php	Generates the layout of the footer
web_language.php	Generates the layout of the language popup screen
web_location.php	Generates the layout of the location popup screen
web_menu.php	Generates the layout of the fixed top bar
web_miniUser.php	Generates the layout of the mini profile screen
web_report.php	Generates the layout of the report popup screen
web_report_s.php	Functions used for the report screen
web_terms.php	Generates the layout of the terms and conditions screen

Figure IV.17 Files of the main folder

IV.2.2 Webservice

The next *Figure IV.18* shows the files developed for the webservice

\webservice:	
_constants.php	Constants used on the webservice
_webFunctions.php	Main functions
getEvent.php	To get the information of an event
getLocations.php	To get a list of events
getLocationsAlert.php	To get a list of events
getLocationsMap.php	To get a list of events
getLocationsSearch.php	To get a list of events
getLocationsSearchAdv.php	To get a list of events
getUser.php	To get information of an user
login.php	To log-in
logout.php	To log-out
readSession.php	Functions that validates the session of the requests
register.php	To register a new user
setCheckin.php	To insert a checkin
setComment.php	To insert a comment
setDelete.php	To delete an element
setEvent.php	To insert an event
setImage.php	To insert an image
setReport.php	To send a report to an element

Figure IV.18 Files of the webservice folder

IV.2.3 Resources

The next *Figure IV.19* shows the files required for the web. There are two files developed by us, the PyxisStyle.css and the javascript.js. These two files contain the styles of the webpage and the javascript functions.

\scripts:	
PyxisStyle.css	Developed styles
colorbox.css	Colorbox styles
javascript.js	Developed javascript functions
jquery-pack.js	Jquery functions
jquery.colorbox-min.js	Jquery colorbox functions
jquery.easing.1.3.js	Jquery functions
jquery.flot.js	Jquery flot functions
jquery.imgareaselect.min.js	Jquery functions
jquery.js	Jquery functions
mapInfoBox.js	Functions for the infobox of the map
markerclusterer.js	Functions for the clusterer for the map
markerclusterer_compiled.js	Functions for the clusterer for the map

Figure IV.19 Files of the resources folder

In the *Figure IV.20* there are the images used in the web. Also there are the folders that contain the images of the events and the users.

\img:	
CloseButton.png	Icon
androidAPP1.png	Capture of the android application
androidAPP2.png	Capture of the android application
androidAPP3.png	Capture of the android application
androidLogo.png	Icon
bg.jpg	Background
bg.png	Background
delete.png	Icon
event1.png	Icon
event2.png	Icon
event3.png	Icon
event4.png	Icon
event5.png	Icon
facebook.png	Icon
favicon.ico	Icon
gplus.png	Icon
gradient.png	Icon
i_send.png	Icon
loading.gif	Icon
logoQ.png	Icon
logo_pyxis_allargat2.png	Icon
logout.png	Icon
maximize.png	Icon
minimize.png	Icon
panoramic1.jpg	User background #1
panoramic2.jpg	User background #2
panoramic3.jpg	User background #3
panoramic4.jpg	User background #4
panoramic5.jpg	User background #5
pattern1.png	Pattern
pos_blue.png	Icon
pos_green.png	Icon
pos_red.png	Icon
searchButton.png	Icon
twitter.png	Icon
uPyxis Logo.png	Icon
\img\events:	Folder to store the images of the events
\img\events\maps:	Folder to store the downloaded maps
\img\users:	Folder to store the avatars of the users
dummy.png	Default user avatar
t_dummy.png	Default user avatar

Figure IV.20 Files of the img folder

V SCREENSHOTS

Below are shown and explained different screenshots of the app and the web.

V.1 APP

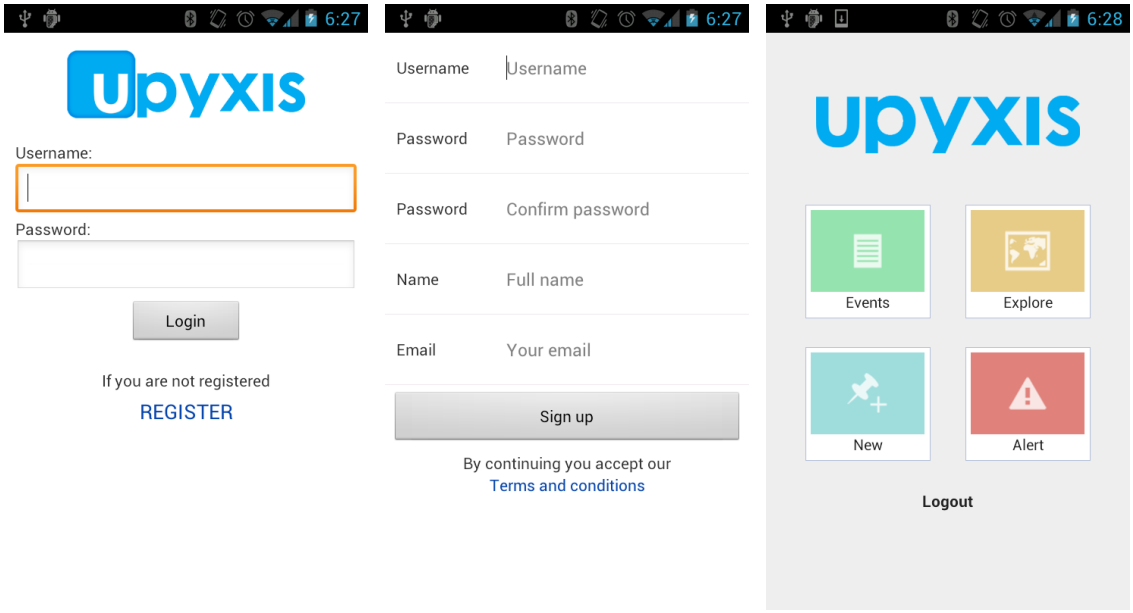

Figure V.1 Screenshots of the login, register and menu screens

The top figure corresponds to the log-in screen, the registration screen, and the screen that appears once logged-in, the main menu.

The figure below shows the screens that appear in the option to create a new event. First is a map to select its location, and the others are to fill in a form with its details.

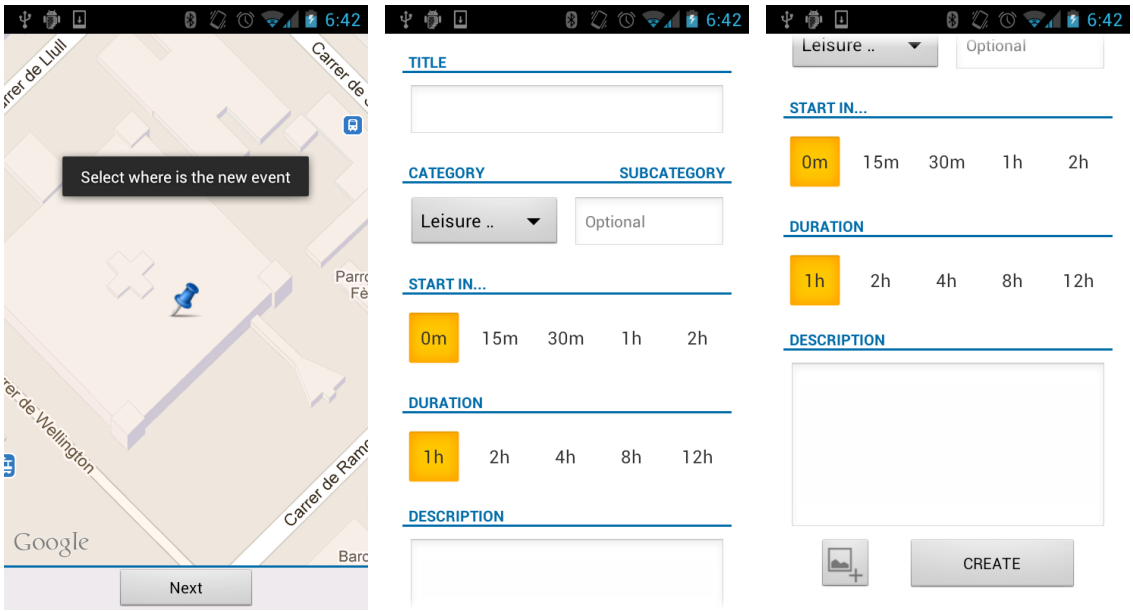

Figure V.2 Screenshots of the new event screen

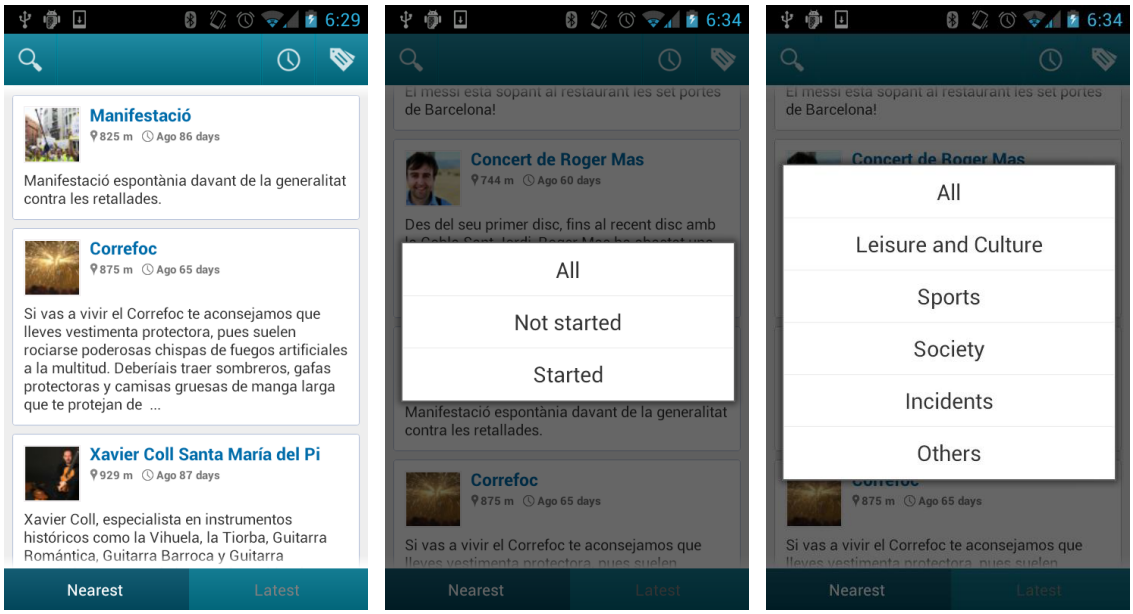

Figure V.3 Screenshots of the list screen

The figure above shows the screen that lists nearby events sorted by proximity or creation date. It also shows the different ways to filter these events, whether by state or category.

The figure below shows the other way to see the nearest events, which is using a map. The map icons are displayed in different colours depending on the status of the events shown. It also shows the ways to filter these events, whether by state or category.

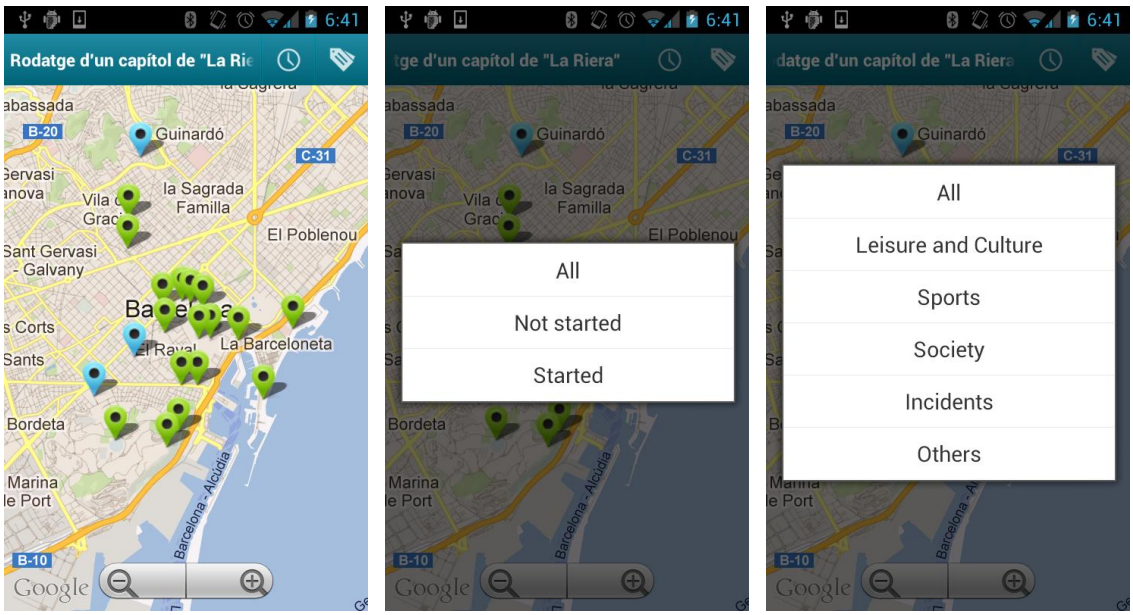

Figure V.4 Screenshots of the map screen

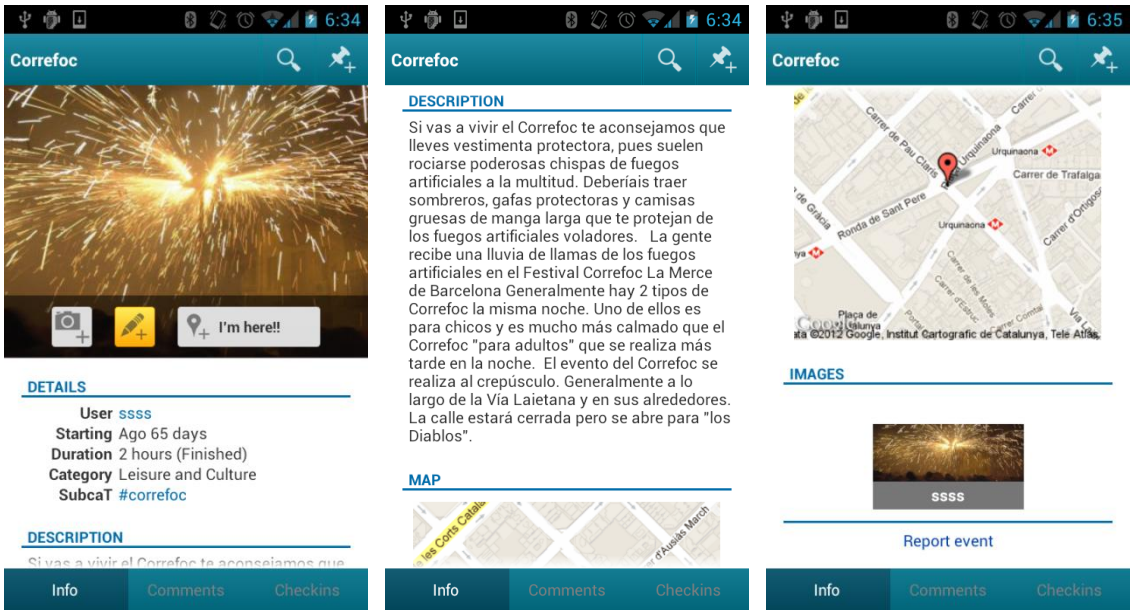

Figure V.5 Screenshots of the event screen

The figure above shows the screen with the complete information about an event. It shows its main picture, the interaction buttons, its details, its description, its location map and the pictures.

The figure below shows the box that is used to add new comments, the tab with the list of comments of this event and the grid with the check-ins of that event.

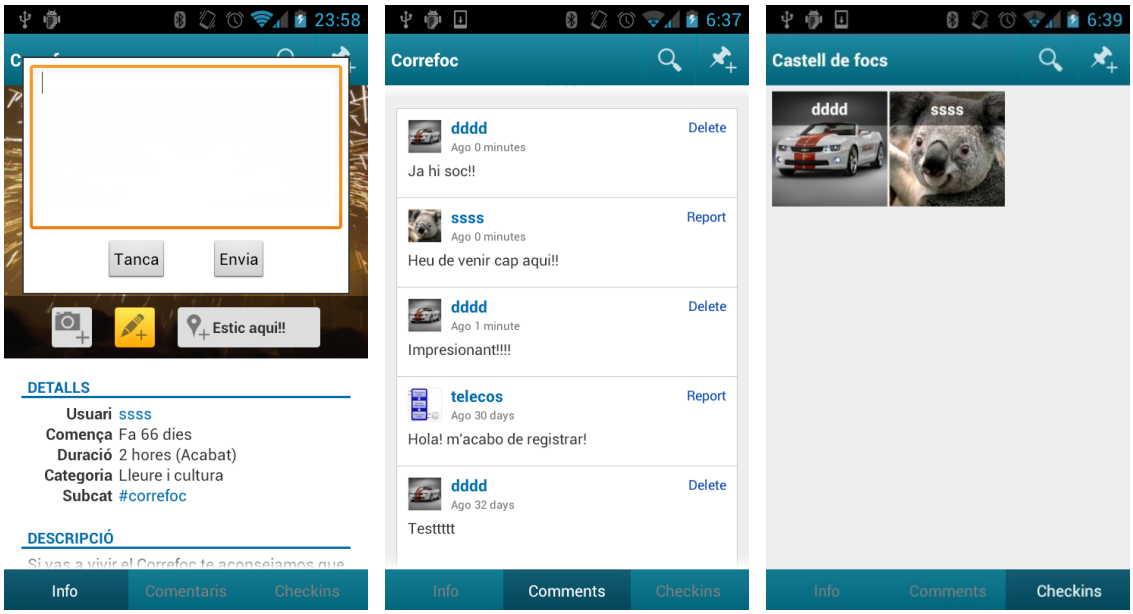

Figure V.6 Screenshots of the event screen

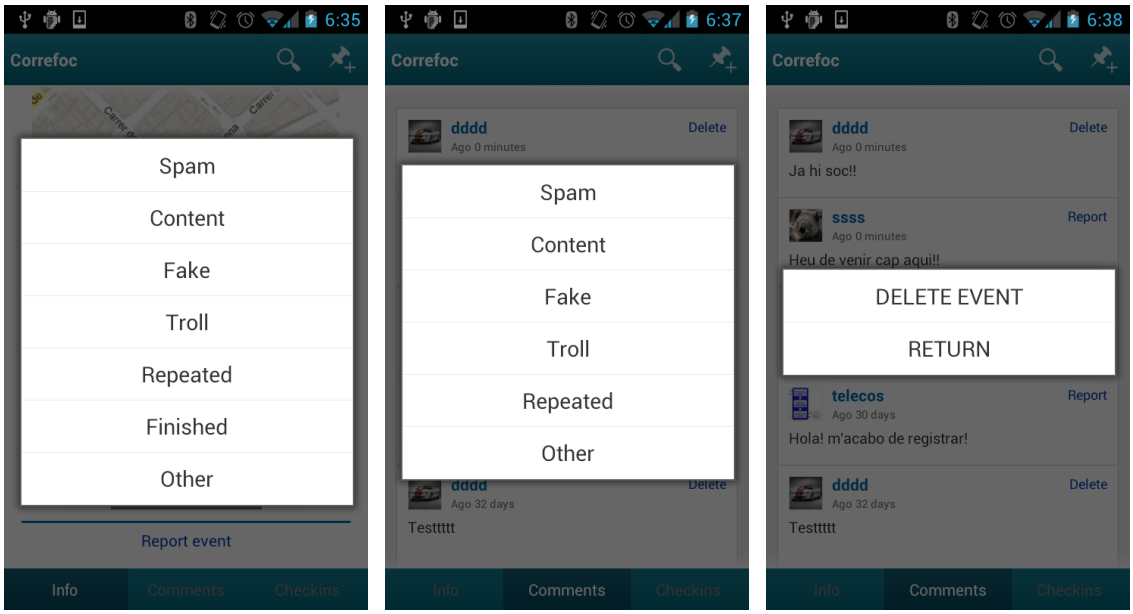

Figure V.7 Screenshots of interaction screens

In the figure above shows the pop-up menus that appear when the user selects the option of reporting an event, a comment or the confirmation when the user wants to delete an event.

The figure below shows the user profile screens, with three tabs to see the user events, its comments or its check-ins.

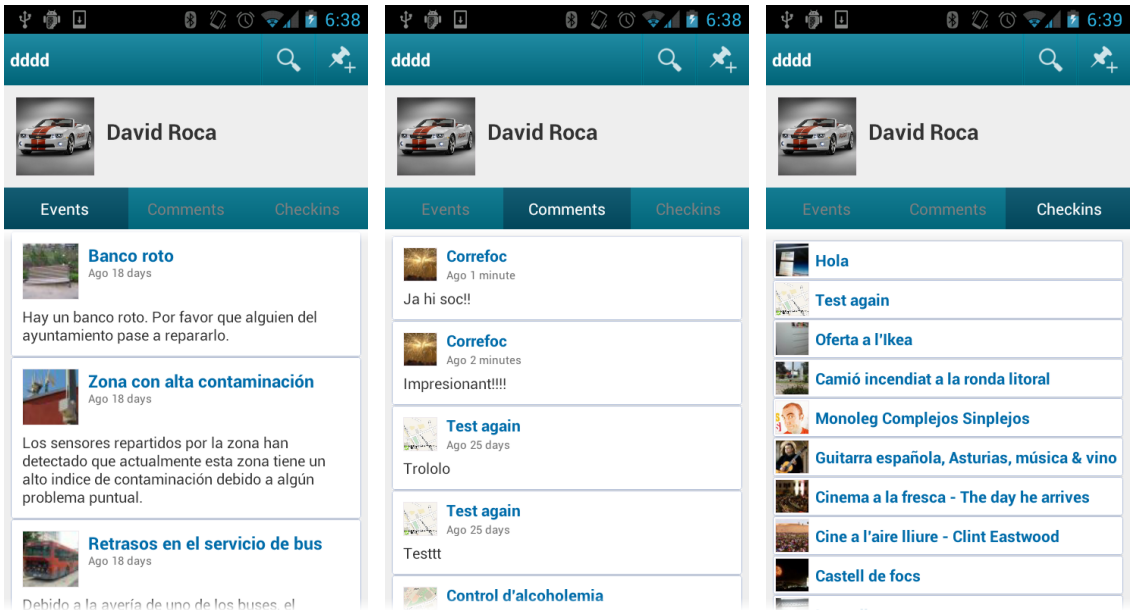

Figure V.8 Screenshots of the user profile screen

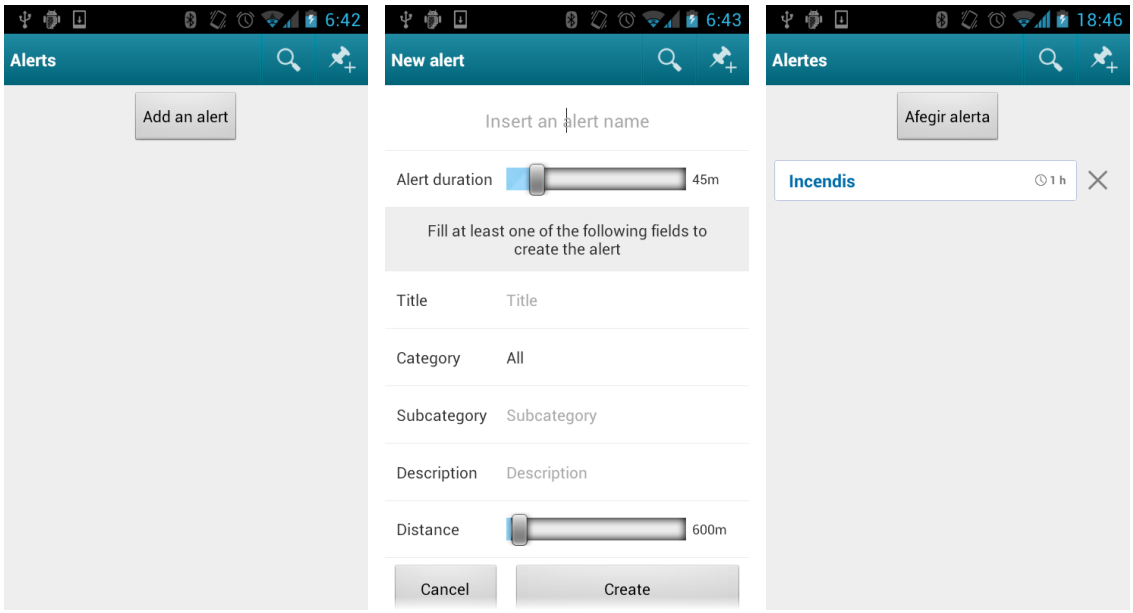

Figure V.9 Screenshots of the alert screens

The figure above shows the screen with an empty list of alerts, the form to create a new alert, and a list with a created alert without results.

In the figure below, there is a screenshot of a notification received announcing results for an alert, next the alert list with a row with a green box saying that there are new results, and the next one shows the results of the alert.

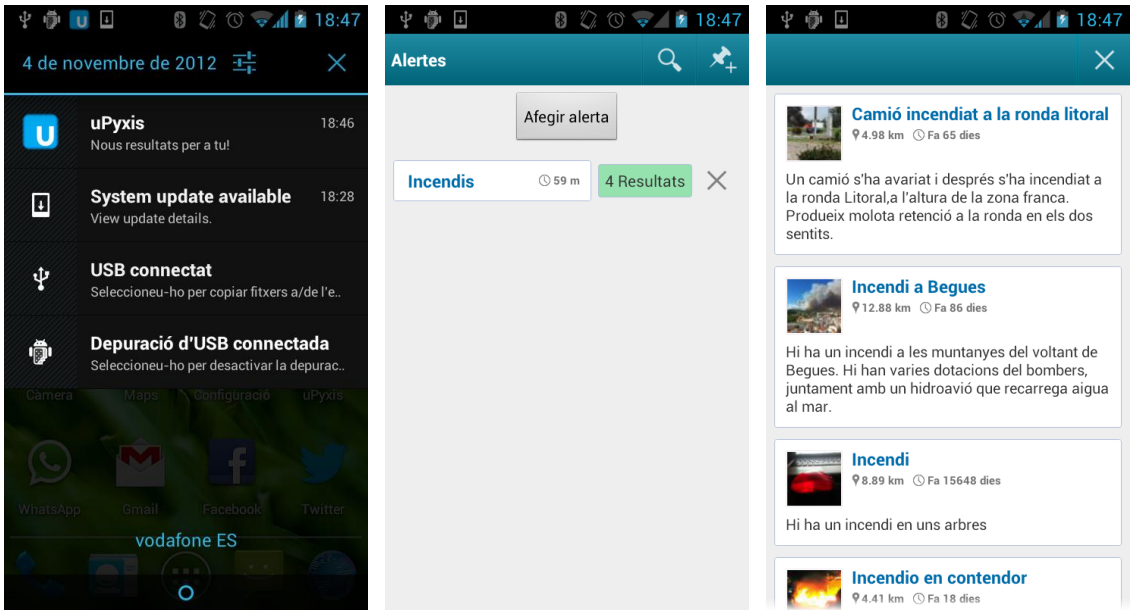

Figure V.10 Screenshots of the alert results screens

V.2 WEB

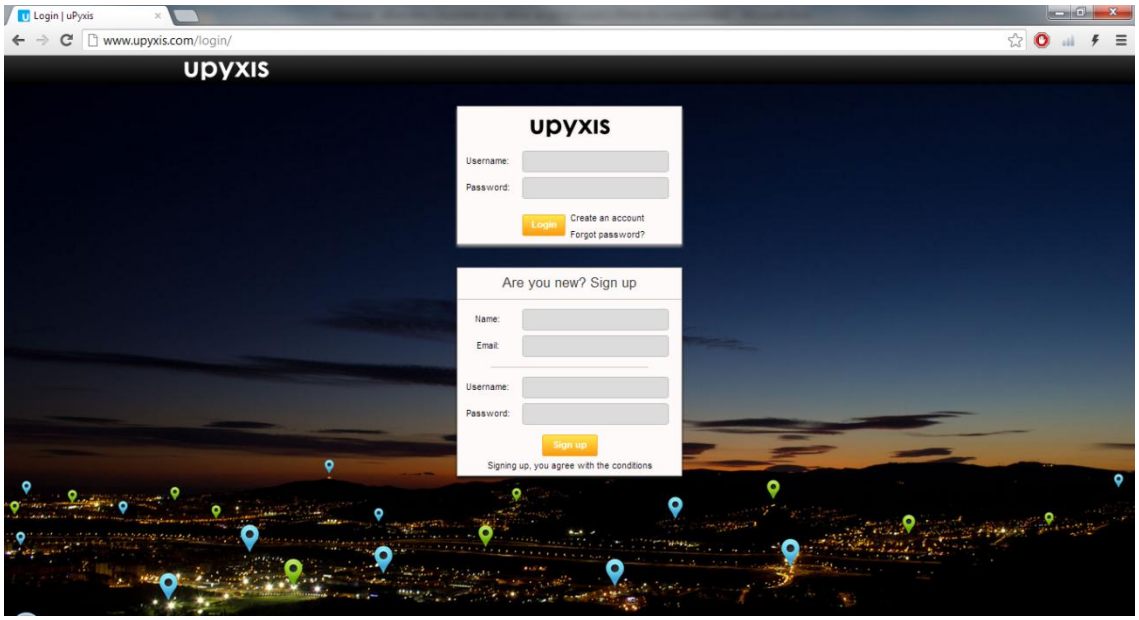

Figure V.11 Screenshots of the login screen

In the picture above you can see the login page of the website. This page also includes a form to register a new user, and another form in case that the user has forgotten its password.

The figure below shows the main page of the site, which shows the list of nearby events, with the option of choosing various filters.

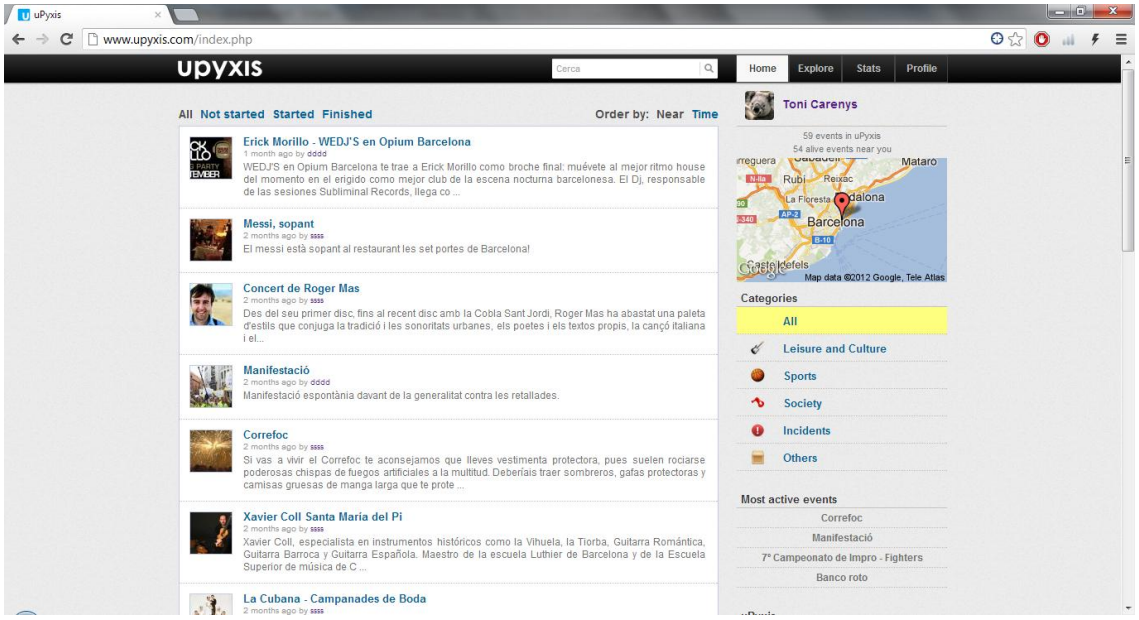

Figure V.12 Screenshots of the list screen

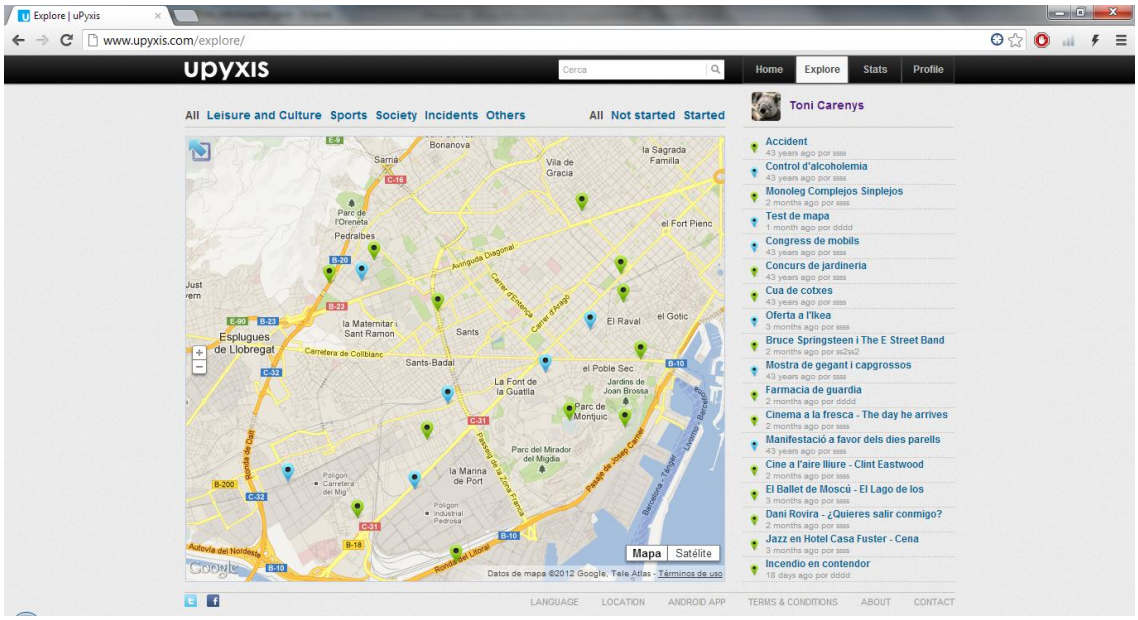

Figure V.13 Screenshots of the explore screen

The figure above shows the other way to see events. In this case is showing icons on a map. These icons have a different colour depending on the status of the event. On the right there is a list with the events that are on the map. Here, the user can see up to 18 events. When the user moves the map, he can see the nearest events to the centre of the map.

The figure below shows another way to display the map. In this case the map fills the entire screen, and then the user can see up to 100 events.

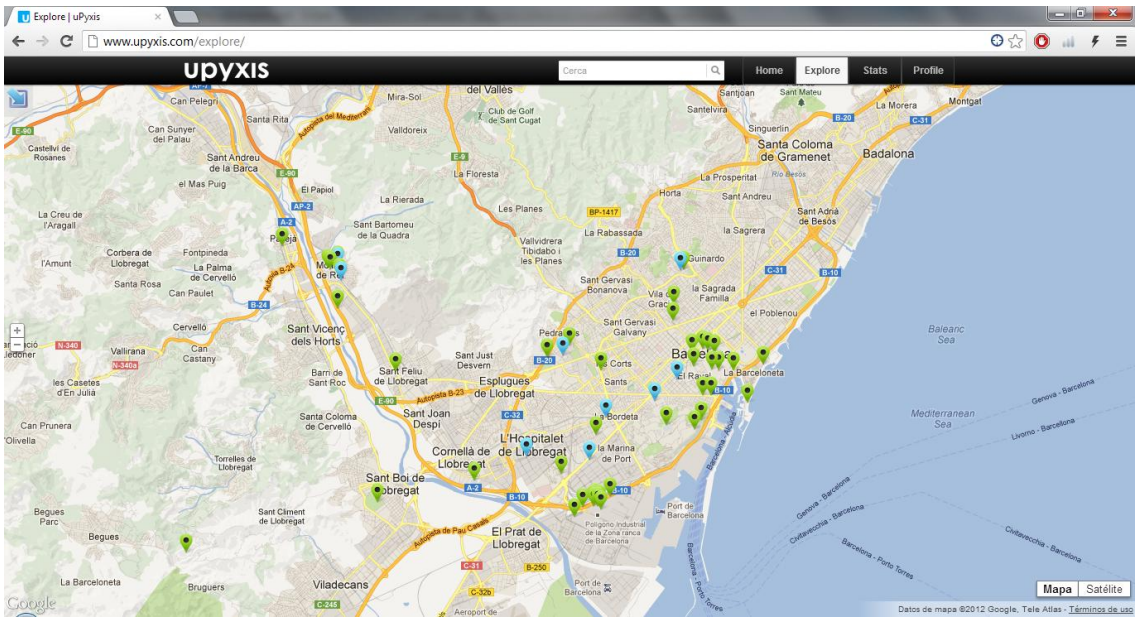

Figure V.14 Screenshots of the explore screen

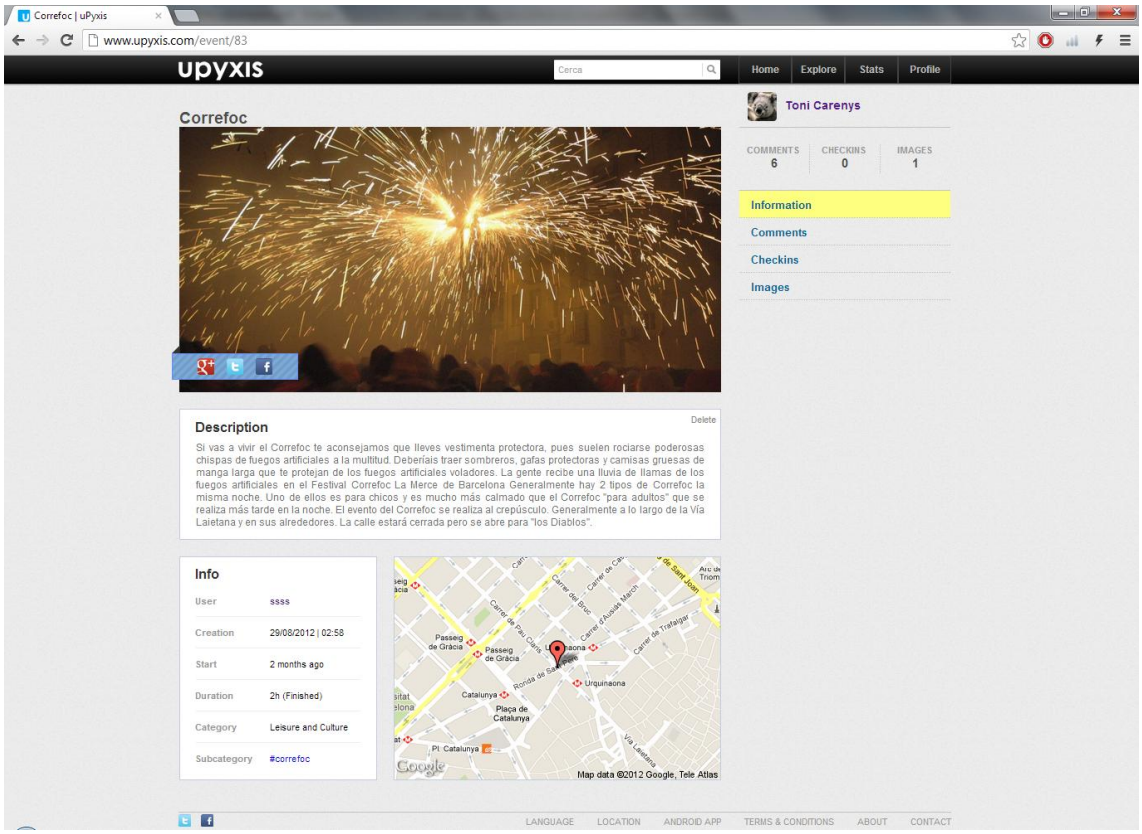

Figure V.15 Screenshots of the event info screen

The figure above shows the main page of the event information. It shows the main picture, sharing options, and its details.

The figure below shows the page with the comments regarding to an event, and there is a field to insert a new comment.

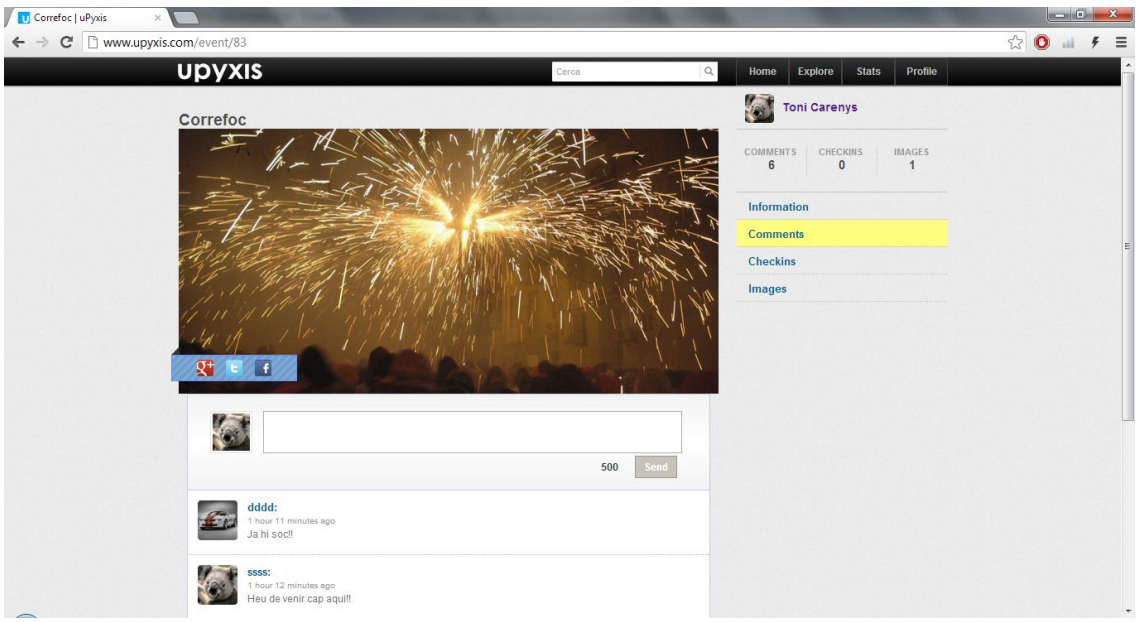

Figure V.16 Screenshots of the event comments screen

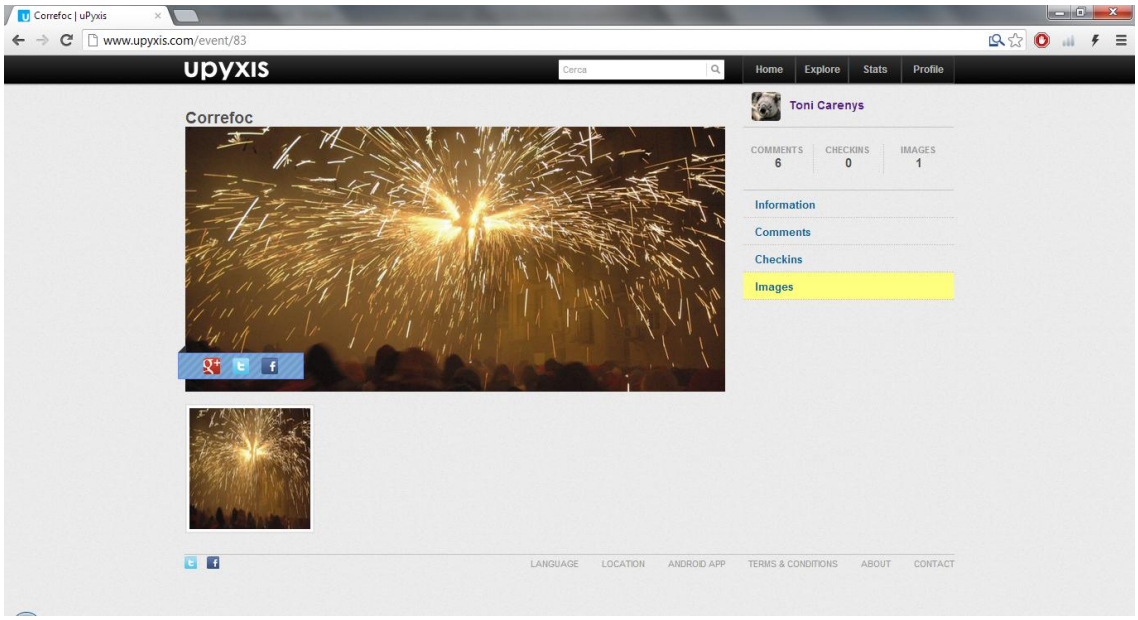

Figure V.17 Screenshots of the event images screen

The figure above shows the page with the images that have been inserted into an event. When the user click on an image, it open a popup Windows that shows the image with a big size

The figure below shows the page with the check-ins that has been made to an event. When the user clicks on a check-in image, it opens the related user profile page.

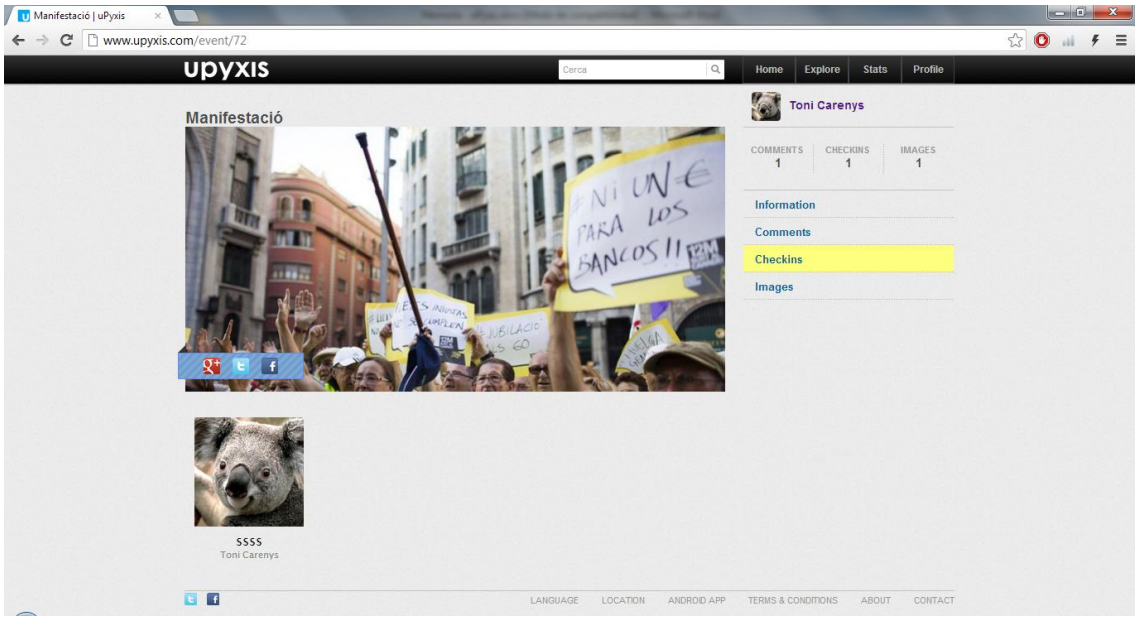

Figure V.18 Screenshots of the event check-ins screen

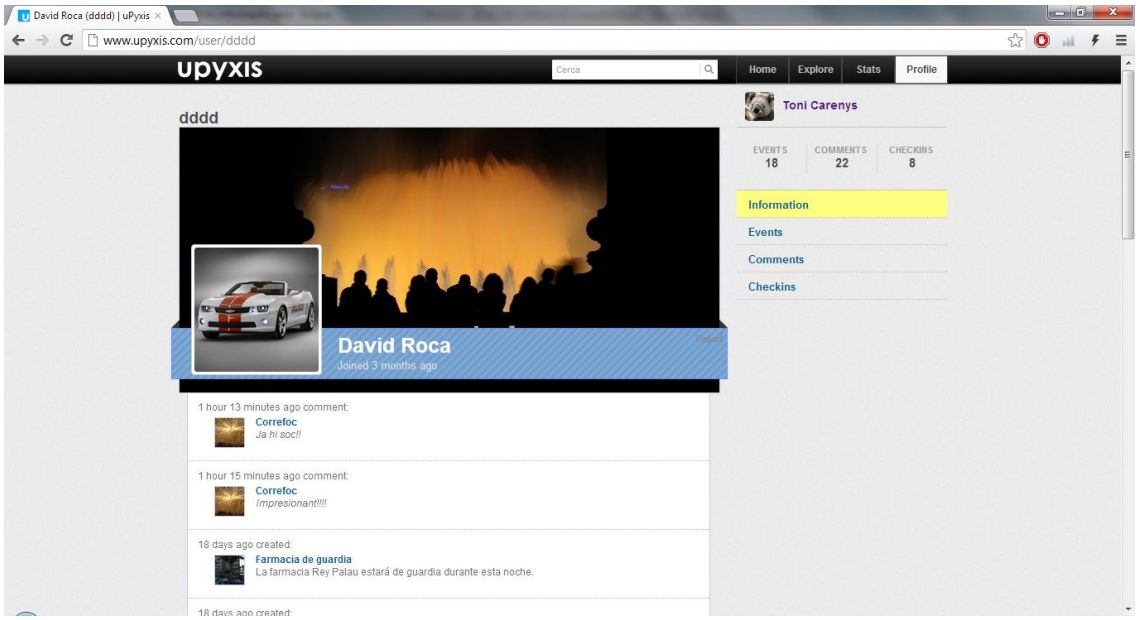

Figure V.19 Screenshots of the user profile screen

The figure above shows the profile page of a user. It displays its personal information, its avatar and its recent activity.

The figure below shows the settings page of a user. On that page it can change its background photo, its avatar image, its password and its name.

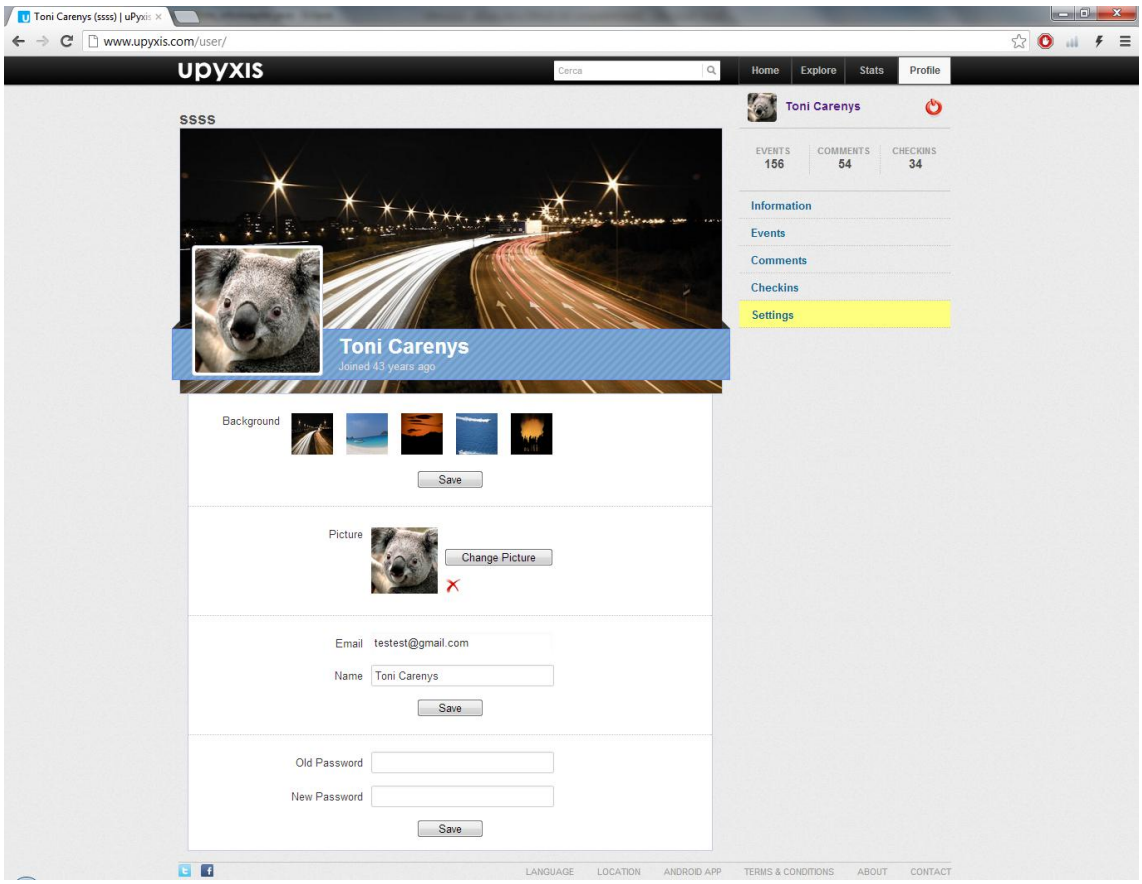

Figure V.20 Screenshots of the user settings screen

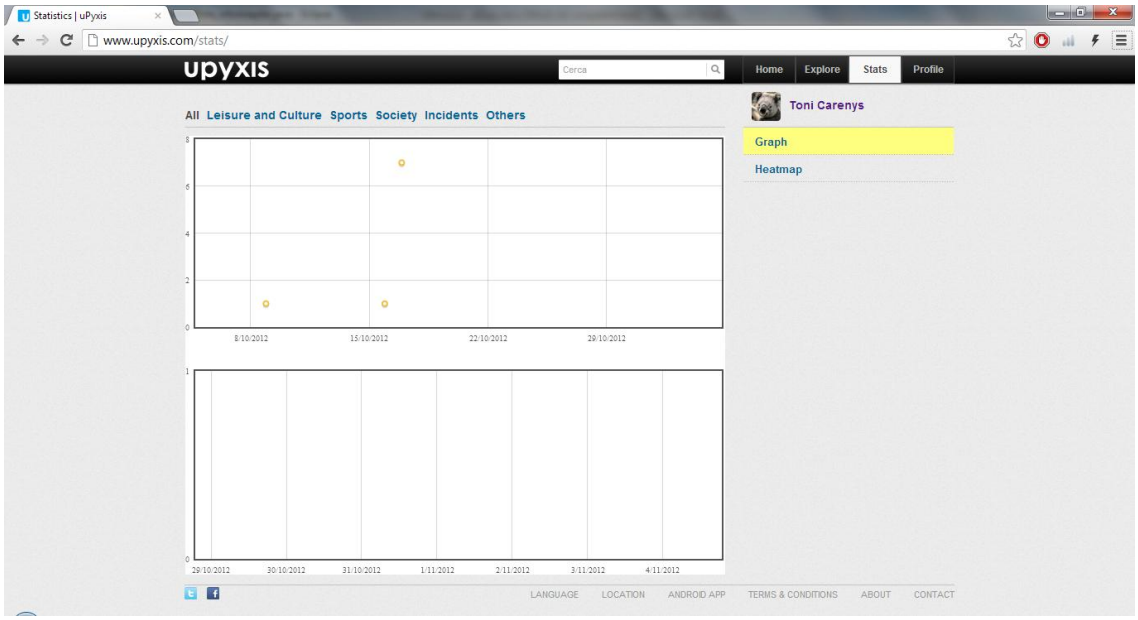

Figure V.21 Screenshots of the statistics screen

In the picture above you can see the statistics page with graphics of events generated during the last month and last week.

In the figure below, it shows another option of the statistics page. This option displays a heat map colorized depending number of events created on a same zone.

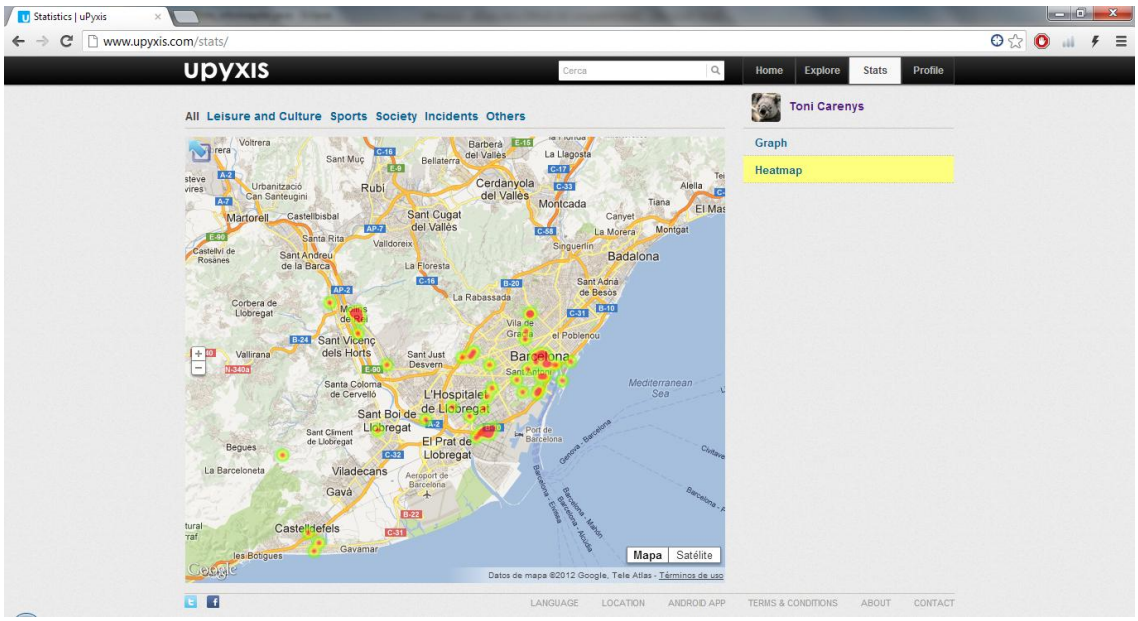

Figure V.22 Screenshots of the heat map screen

VI SURVEYS

VI.1 Survey

PERSONAL

1- Age: [ ] **2-** Gender: [ ] **3-** Use of web and new technologies: [ ] 1 -> Very low 5 -> Very high

APP

4- The application is easy to use……..…… [ ] 1 -> Strongly disagree 5 -> Strongly agree

5- It's quick to learn its operation……..…… [ ] 1 -> Strongly disagree 5 -> Strongly agree

6- It is easy to find the desired features ……..…….. [ ] 1 -> Strongly disagree 5 -> Strongly agree

7- The general design of the application is appropriate …….. [ ] 1 -> Strongly disagree 5 -> Strongly agree

8- The organization of the elements is appropriate …..... [ ] 1 -> Strongly disagree 5 -> Strongly agree

9- The application has all the expected functionalities ……..… [ ] 1 -> Strongly disagree 5 -> Strongly agree

10- Test: Start the APP and register……..… [ ] seconds

11- Test: Start the APP and create a new event ……..… [ ] seconds

WEB

12- The website is easy to use……..……...…. [ ] 1 -> Strongly disagree 5 -> Strongly agree

13- It's quick to learn its operation…….……...…….…. [ ] 1 -> Strongly disagree 5 -> Strongly agree

14- It is easy to find the desired features……..…… [ ] 1 -> Strongly disagree 5 -> Strongly agree

15- The general design of the website is appropriate ……..... [ ] 1 -> Strongly disagree 5 -> Strongly agree

16- The organization of the elements is appropriate …..... [ ] 1 -> Strongly disagree 5 -> Strongly agree

17- The website has all the expected functionalities ……..…… [ ] 1 -> Strongly disagree 5 -> Strongly agree

18- Test: Log-in, search a fire event, and comment it……..… [ ] seconds

19- Test: Open profile and change your avatar image ……..…… [ ] seconds

GENERAL

20- The service offered can be useful …..…….....…… [ ] 1 -> Strongly disagree 5 -> Strongly agree

VI.2 Results

	Person 0 (Me)	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6
Q1	26	25	26	21	57	33	23
Q2	Male	Female	Male	Female	Female	Male	Male
Q3	5	4	4	4	2	4	4
Q4	5	5	5	5	4	4	4
Q5	5	5	4	4	5	5	4
Q6	5	4	4	4	4	5	5
Q7	5	5	5	4	4	4	4
Q8	5	4	4	3	5	3	4
Q9	5	5	3	4	5	4	4
Q10	40	90	95	112	206	84	101
Q11	100	165	173	156	277	170	166
Q12	5	4	5	4	5	4	4
Q13	5	5	4	4	4	4	4
Q14	5	5	5	4	4	4	3
Q15	5	5	5	4	4	5	4
Q16	5	5	4	3	5	3	4
Q17	5	5	4	4	5	4	4
Q18	25	75	62	67	130	64	77
Q19	17	70	81	61	165	77	70
Q20	5	5	4	4	5	4	4
Avg	5	4,77	4,31	3,92	4,54	4,08	4,00

	Person 7	Person 8	Person 9	Person 10	Person 11	Average all	Average without me
Q1	22	26	55	49	35	33,17	33,82
Q2	Female	Male	Male	Female	Male		
Q3	4	3	3	3	4	3,67	3,55
Q4	4	4	4	3	4	4,25	4,18
Q5	5	4	4	3	4	4,33	4,27
Q6	4	4	5	4	5	4,42	4,36
Q7	5	5	4	5	5	4,58	4,55
Q8	4	5	4	5	5	4,25	4,18
Q9	4	4	5	4	5	4,33	4,27
Q10	105	143	177	221	92	122,17	129,64
Q11	163	192	180	250	169	180,08	187,36
Q12	5	4	5	4	4	4,42	4,36
Q13	4	3	4	4	4	4,08	4,00
Q14	4	3	5	5	4	4,25	4,18
Q15	5	4	4	5	5	4,58	4,55
Q16	4	3	4	4	4	4,00	3,91
Q17	5	4	4	4	4	4,33	4,27
Q18	71	102	105	121	69	80,67	85,73
Q19	83	93	111	161	71	88,33	94,82
Q20	4	4	4	4	5	4,33	4,27
Avg	4,38	3,92	4,31	4,15	4,46	4,32	4,26

Figure VI.1 Survey results

On the Figure VI.2, can be seen the distribution of the surveyed people depending on its age, use of web and new technologies, and its gender. The surveyed people, as more adult are, less related to the new technologies.

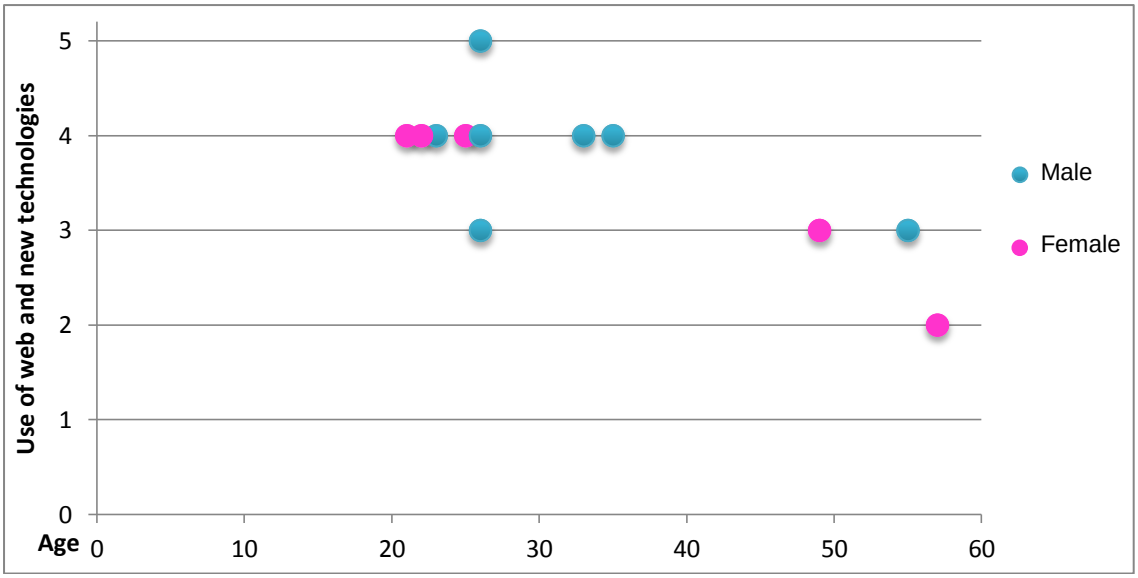

Figure VI.2 Characteristics of surveyed people

In general, the survey results of the *Figure VI.1* are very good. The average rating is 4.26, that corresponds a mark between “Agree” and “Strongly agree”. The aged people, has more problems to interact with the service, taking longer to complete the different tests, but after a while of practice became more agile, and its final valuation of the service, the website and the application is good.

Also, the people surveyed have commented mainly two topics.

- People commented that in the APP, on some phones the button used to open the option to register a new user is too hidden when the keyboard is displayed.

- On the other hand, people do not like that in their profile are displayed their activity (events created, comments, check-ins), and would like to have an option to change the privacy of their profile.

On the next figures, there are some graphics with the results of the different questions. The *Figure VI.3* shows the results for the questions related to the APP, the *Figure VI.4* shows the results for the questions related to the Web, and the *Figure VI.5* shows the results of the test realized at the APP and the Web.

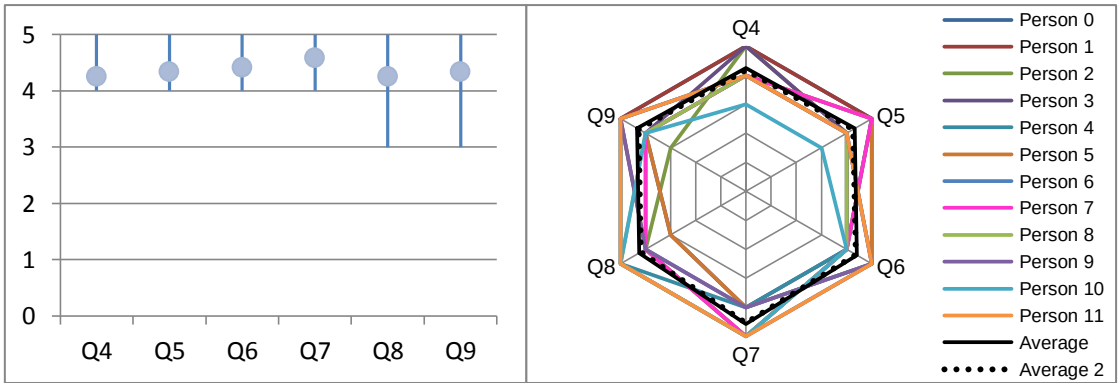

Figure VI.3 APP survey results

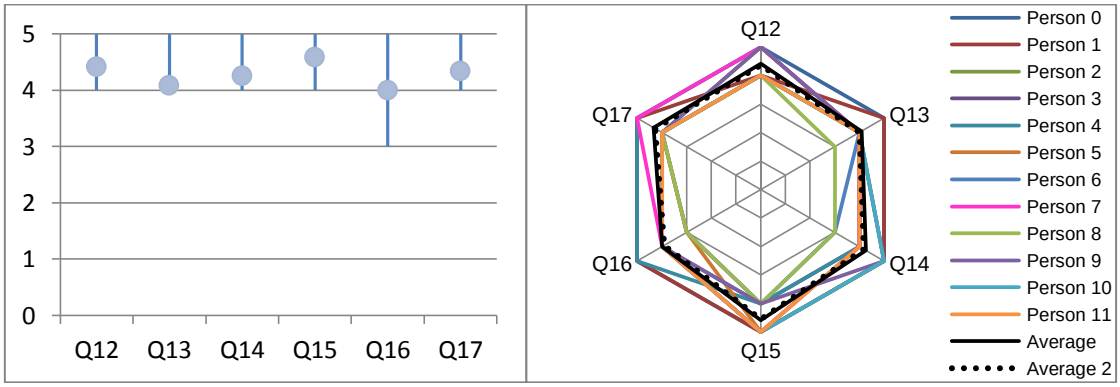

Figure VI.4 Web Survey results

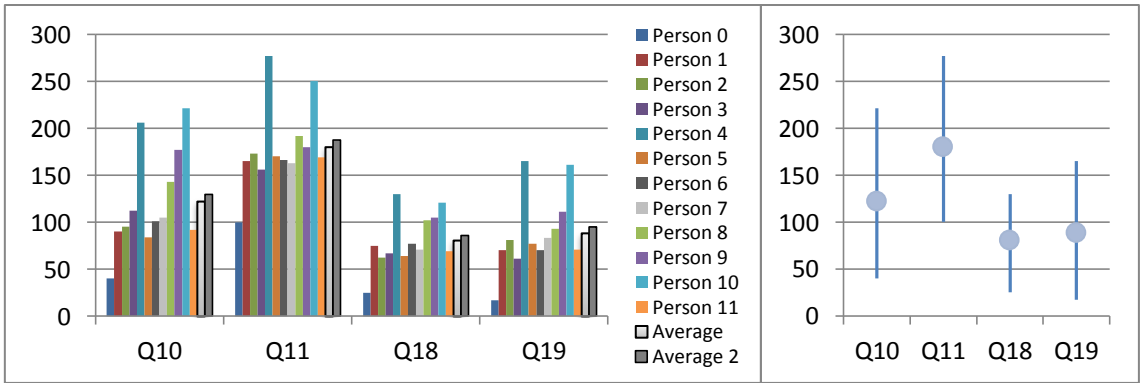

Figure VI.5 Tests results

VII WORK PLANNING

The dedication time for this project has been 920 hours. These hours equals to an average dedication of 4 hours by working day (with some weekends) during 10 months. The work planning can be distributed on four parts:

- **Idea & Previous study**. Includes the set of objectives, and the study and learning of technologies.
- **Design.** Includes the description of functionalities, and the design of the UI and the different logic parts.
- **Implementation.** Includes the install of the development environment, the development of the different parts, and the publication of the website and APP.
- **Tests.** Includes the test to verify the operation of the service, and the surveys about the service.
- **Memory Report.** Includes the realization of this report and the related PowerPoint.

The distribution of hours of each part, can be seen on the following *Figures VII.1 and VII.2*

Type	Hours	Percentage
Idea & Previous study	138	15%
Design	184	20%
Implementation	368	40%
Tests	69	7,50%
Memory report	161	17,50%
TOTAL	**920**	**100%**

Figure VII.1 Time Distribution

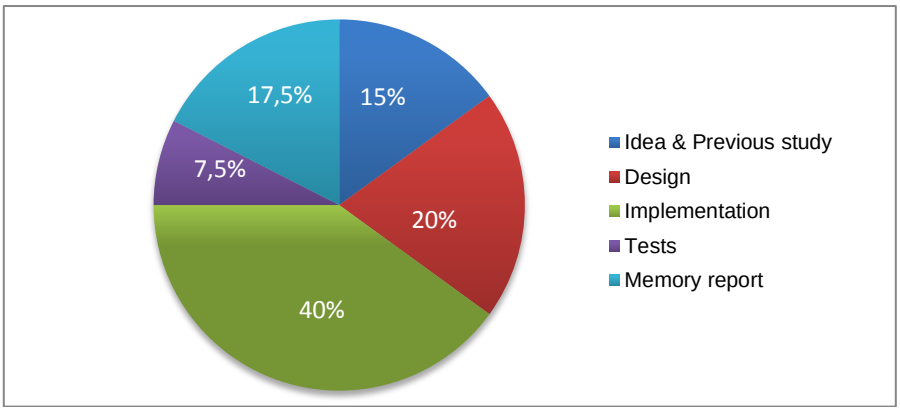

Figure VII.2 Time Distribution

VII.1 Tasks

The following Figure VII.3 shows the tasks realized on each part of the work planning, and the related dedication time to each one.

Type	Description	Hours	% type	% total
Idea & Prev. study	Description of the idea & objectives	10	7,25%	1,09%
Idea & Prev. study	Study of smartphone technologies	20	14,49%	2,17%
Idea & Prev. study	Study of web technologies	25	18,12%	2,72%
Idea & Prev. study	Learn Android basics	37	26,81%	4,02%
Idea & Prev. study	Learn PHP basics	31	22,46%	3,37%
Idea & Prev. study	Learn MySQL basics	15	10,87%	1,63%
TOTAL 1		**138**	**100%**	**15%**
Design	Design Website basic functionalities	16	8,70%	1,74%
Design	Design APP basic functionalities	16	8,70%	1,74%
Design	Design webservice basic functionalities	12	6,52%	1,30%
Design	Design databases tables	8	4,35%	0,87%
Design	Design Website UI	16	8,70%	1,74%
Design	Design APP UI	16	8,70%	1,74%
Design	Design core logic	30	16,30%	3,26%
Design	Design website logic	35	19,02%	3,80%
Design	Design APP logic	35	19,02%	3,80%
TOTAL 2		**184**	**100%**	**20%**
Implementation	Install & set up the develop environment	8	2,17%	0,87%
Implementation	Create databases tables	6	1,63%	0,65%
Implementation	Develop core logic	40	10,87%	4,35%
Implementation	Develop webservice	40	10,87%	4,35%
Implementation	Develop APP UI	50	13,59%	5,43%
Implementation	Develop Website UI	40	10,87%	4,35%
Implementation	Develop APP logic	80	21,74%	8,70%
Implementation	Develop Website logic	80	21,74%	8,70%
Implementation	Develop images and other resources	8	2,17%	0,87%
Implementation	Translations	6	1,63%	0,65%
Implementation	Buy domain, hosting and publish	8	2,17%	0,87%
Implementation	Create GooglePlay account	2	0,54%	0,22%
TOTAL 3		**368**	**100%**	**40%**
Tests	Test webservice	8	11,59%	0,87%
Tests	Test Website	16	23,19%	1,74%
Tests	Test APP	16	23,19%	1,74%
Tests	Design of the interviews	7	10,14%	0,76%
Tests	Interviews	7	10,14%	0,76%
Tests	Analyse results	15	21,74%	**1,63%**
TOTAL 4		**69**	**100%**	**7,5%**
Memory report	Memory report	126	78,26%	13,70%
Memory report	PowerPoint	35	21,74%	3,80%
TOTAL 5		**161**	**100%**	**17,5%**
TOTAL		**920**	**100%**	**100%**

Figure VII.3 Tasks distribution

VII.2 Costs

To calculate the costs of the project have been considered the three different scenarios of the *Figures VII.4 and VII.5*. As fixed costs there are the android developer license, and the price of the hosting and the domain.

- **Real.** This scenario shows the real costs that I've had. Includes the university registration fees, the 10% of my computer and smartphone cost, and others expenses like gasoline, electricity, etc.

- **Intern.** This scenario includes the costs that a company would have if it hires an intern to make this project. Includes the cost of a computer and a smartphone, and an intern at 8€/h.

- **Engineer.** This scenario includes the costs that a company would have if the project is developed by an engineer. Includes the cost of a computer and a smartphone, and an engineer at 50€/h.

	Real	Intern	Engineer
University registration fees	550	0	0
Computer	100	1.000	1.000
Smartphone	45	450	450
Android developer	20	20	20
Hosting and domain	6	6	6
Employers 8€/h	0	7.360	0
Employers 50€/h	0	0	46.000
Others	279	0	0
TOTAL	**1.000 €**	**8.836 €**	**47.476 €**

Figure VII.4 Project costs

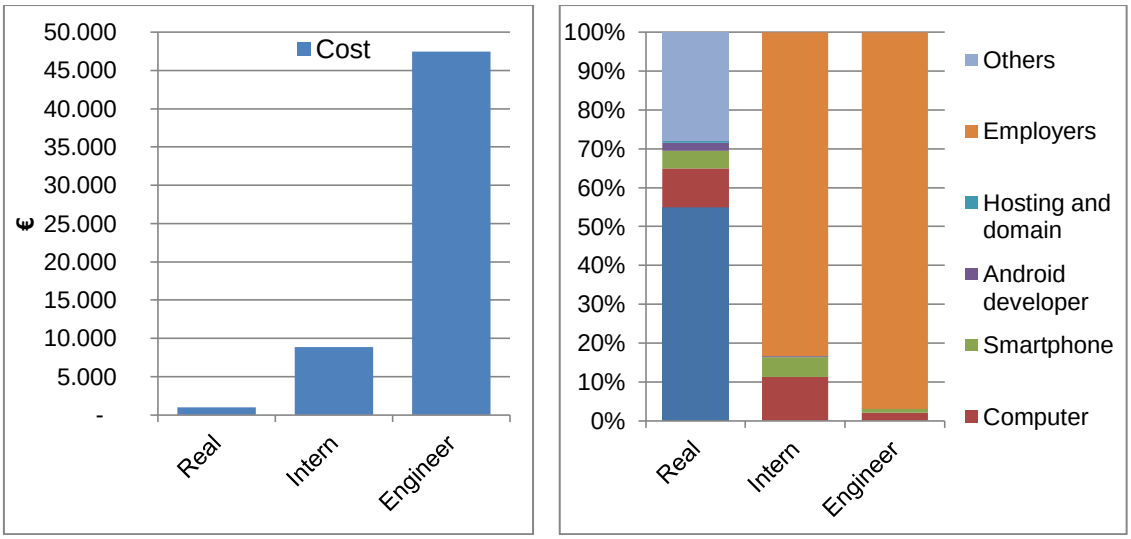

Figure VII.5 Project costs

VII.3 Estimated Earnings

In the case that would like to earn money with the service, on the following lines and figures there is a basic simulation about the earnings and costs of each scenario.

Assumptions:

- 10.000 users at the end of the first year. (50.000 second year).
- 2 visits per day and user.
- 1 euro of earnings each 1.000 visits
- As more users, there is needed a best hosting.
- A person dedicated 20hours/week on each scenario.

As a result, on the following figures can be seen that:

- On the first scenario a clean balance is obtained at the end of first year.
- On the second scenario a clean balance is obtained on 2015.
- On the third scenario there is needed a lot of years to earn money, or a high increase of the number of users.

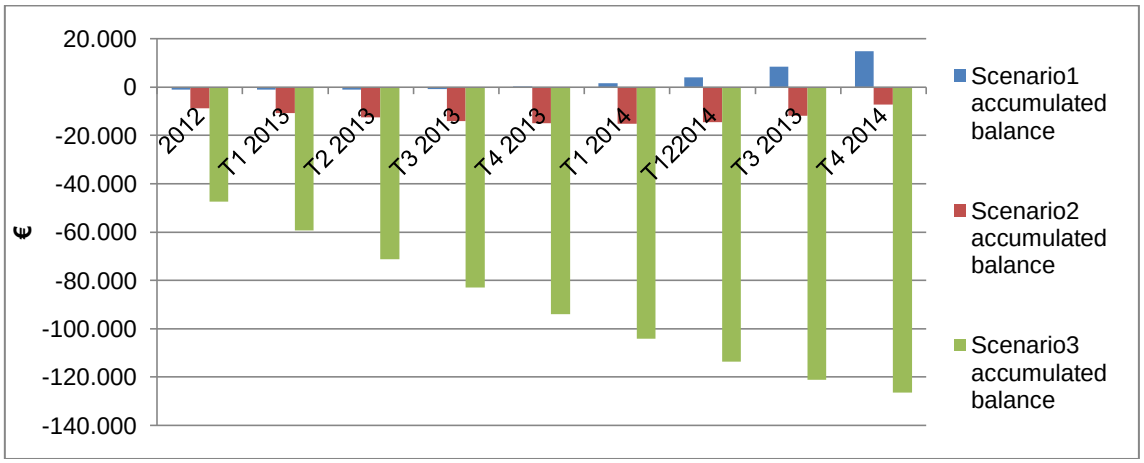

Figure VII.6 Accumulated balances

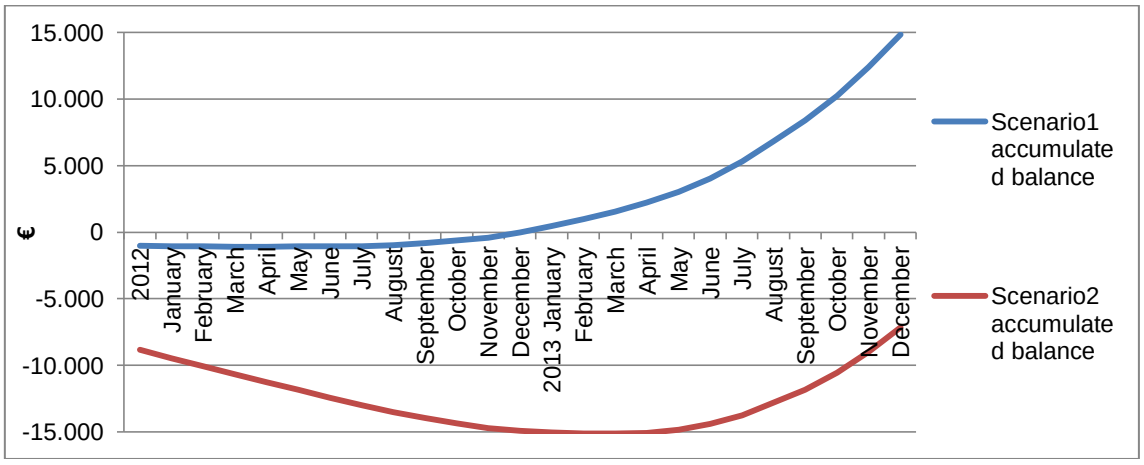

Figure VII.7 Accumulated balances

	2012	Jan.	Feb.	March	April	May	June	July	August	Sept.	October	Nov.	Dec.	2013
Total users	0	200	400	600	850	1.200	1.600	2.000	3.000	4.000	5.000	7.000	10.000	10.000
Visits	0	12.000	24.000	36.000	51.000	72.000	96.000	120.000	180.000	240.000	300.000	420.000	600.000	2.151.000
Earnings	0	12	24	36	51	72	96	120	180	240	300	420	600	2.151
Hosting Costs	0	0	0	0	0	0	50	50	50	50	50	150	150	550
Scenario1 Costs	1.000	50	50	50	50	50	50	50	50	50	50	50	50	600
Scenario2 Costs	8.836	640	640	640	640	640	640	640	640	640	640	640	640	7.680
Scenario3 Costs	47.476	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	48.000
S1 Result	-1.000	-38	-26	-14	1	22	-4	20	80	140	200	220	400	1.001
S2 Result	-8.836	-628	-616	-604	-589	-568	-594	-570	-510	-450	-390	-370	-190	-6.079
S3 Result	-47.476	-3.988	-3.976	-3.964	-3.949	-3.928	-3.954	-3.930	-3.870	-3.810	-3.750	-3.730	-3.550	-46.399
S1 accum. balance	-1.000	-1.038	-1.064	-1.078	-1.077	-1.055	-1.059	-1.039	-959	-819	-619	-399	1	**1**
S2 accum. balance	-8.836	-9.464	-10.080	-10.684	-11.273	-11.841	-12.435	-13.005	-13.515	-13.965	-14.355	-14.725	-14.915	-14.915
S3 accum. balance	-47.476	-51.464	-55.440	-59.404	-63.353	-67.281	-71.235	-75.165	-79.035	-82.845	-86.595	-90.325	-93.875	**-93.875**

	2013	Jan.	Feb.	March	April	May	June	July	August	Sept.	October	Nov.	Dec.	2014
Total users	10.000	11.000	12.000	13.000	15.000	18.000	21.000	25.000	30.000	35.000	40.000	45.000	50.000	50.000
Visits	2.151.000	660.000	720.000	780.000	900.000	1.080.000	1.260.000	1.500.000	1.800.000	2.100.000	2.400.000	2.700.000	3.000.000	18.900.000
Earnings	2.151	660	720	780	900	1.080	1.260	1.500	1.800	2.100	2.400	2.700	3.000	18.900
Hosting Costs	550	150	150	150	200	200	200	200	200	500	500	500	500	3.450
Scenario1 Costs	600	50	50	50	50	50	50	50	50	50	50	50	50	600
Scenario2 Costs	7.680	640	640	640	640	640	640	640	640	640	640	640	640	7.680
Scenario3 Costs	48.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	4.000	48.000
	0													
S1 Result	1.001	460	520	580	650	830	1.010	1.250	1.550	1.550	1.850	2.150	2.450	14.850
S2 Result	-6.079	-130	-70	-10	60	240	420	660	960	960	1.260	1.560	1.860	7.770
S3 Result	-46.399	-3.490	-3.430	-3.370	-3.300	-3.120	-2.940	-2.700	-2.400	-2.400	-2.100	-1.800	-1.500	-32.550
S1 accum. balance	1	461	981	1.561	2.211	3.041	4.051	5.301	6.851	8.401	10.251	12.401	14.851	**14.851**
S2 accum. balance	-14.915	-15.045	-15.115	-15.125	-15.065	-14.825	-14.405	-13.745	-12.785	-11.825	-10.565	-9.005	-7.145	-7.145
S3 accum. balance	-93.875	-97.365	- 100.795	- 104.165	- 107.465	-110.585	-113.525	-116.225	-118.625	-121.025	-123.125	-124.925	-126.425	**-126.425**

Figure VII.8 Earnings model