
**Treball de Fi de Grau**

**GRAU D'ENGINYERIA INFORMÀTICA**

**Facultat de Matemàtiques**

**Universitat de Barcelona**

---

# PhotoTag, a mobile application for tagging photos.

---

**Anna Casas Alarcon**

Director: Oriol Pujol Vila

Realitzat a: Departament de Matemàtica Aplicada i Anàlisi. UB

Barcelona, 27 de juny de 2015

# Acknowledgments

Finishing this project has not been an easy task. In the whole four months that I have been working on it, there have been many ups and downs but with the support of my family, my boyfriend and his family, my friends and teachers of the university, they made it easier.

Thanks to my parents, Miquel and Antonia, for always being there for me and giving me strength to go ahead not just with this project, but also with the whole career, I could not be who I am without them.

Thanks to my brother Marc, to believe in me and make me realize all I can do for myself.

Thanks to my old brother David, to teach me since I was a kid to repair computers and use them and to make me realize who I want to be.

Absolutely thanks to my boyfriend, Adrián. He has been always by my side, believing in me and making me think that I can do anything with strength and dedication, to take me somewhere when I was dazed… For everything he did for me. To his family, for making me feel like I am one of them and help me with everything they can.

Thanks to my friends, Carolina, Roger and Fernando who always have time to listen to me when I was down and also help me when I do not know how to solve a problem.

Thanks also to my friends Aitor, Joel and Rocio, for being patient and understanding that university it is not an easy road.

And last, but definitely not least, thanks to all teachers that helped me during the career, but special thanks to my tutor, Oriol Pujol, for always having time to help me with the project and having time for listening to me when I was worried during the career.

# Abstract

The application that has been developed on this project is about a mobile application that integrates a tagging photo for Android platform, to edit photos and tag them in an easy form. An easy interface has been developed to make easier to the final user to use it. Using this application, users will be able to tag any photograph with different icons and also remove them if it is necessary. Also the users can save a project file and take a screenshots of each layer with the background photograph. Currently, there are not applications like this on the play store so, it satisfies a huge user need. It is an innovate application and for surely will have a great acceptance. It is worth nothing that the project was developed with modern technology such Android that nowadays, is one of the most important operating system on the world-wide digital market.

# Index

# 1. Introduction

PhotoTag, it's an application for Android devices. You can do things such as tagging photos. But these is not all! The application is developed like the famous photo editor Photoshop to make picture manipulation and annotation easier to the user. We can work with layers, make them visible or invisible and do pan and zoom. Also, to not make this application restricted, have been implemented a plugin system that allows to expand the tool for new icons and annotations needs. At the time of writing this report, there are only three ways to mark our photos: the default one, thumbtacks, icons specifically designed for fine arts and also icons from the world famous game World of Warcraft.

Do not forget, that this application, is meant to be used in real and long work sessions. So, in order to make it easier to the user, have been developed an own file format (.acf) to save the project and continue working later on it. But, if the user has finished tagging the photo, we will be able to export data, with one click. The exportation makes a screenshot of each layer and the main photo of the background.

We have talked a lot of editing photos, tagging… but, how the user put his/her own photographs? Most devices now a days, have a rear or a front camera! So, if the user does not have any photo to be tagged the user can take the photo from the gallery of the device or take the shot with the same application!

## 1.1 Motivation

Now a days, people do not use the computer such as few year ago. What the user uses the most are mobiles and tablets. So, developing an application for that kind of devices for the project is a good choice for further exploring this field.

The first time that I did something in Android was in a subject of the university called “Projecte integrat de Software”. I really had fun doing a little game with my schoolfellows though it took us a lot of hours and a lot of long nights. Since the project was over, I always thought to spend some years of my live working on device application development and, who knows, maybe I could expend all my working live on them.

When it was time to choose the project I was not sure what to do but after talking with the head teacher (who is my tutor) I decided to do that project. I always liked working with photos and editing them, and I thought, why not? Let’s do it!

At the beginning this project was especially in collaboration with the fine arts department. But the problem was that they want it for the first semester of the year and it was impossible for me to do it with that schedule with all the subjects that I had even more while I was working in the mornings.  At the end, it seems that they were not interested and with my tutor we decided to go ahead with the project but change the specifications for a more generic tagging application.

In each step of the project, I had troubles, things that I could not fix, but with a lot of hours in front of my computer, and the support of my tutor, finally had fixed them. That is what make me think that is possible that is what I want to be in the future. I do not mind to spend four, six or ten hours trying to fix something that do not work because of the satisfaction to see it working better.

Searching on the android store, there is nothing for tagging photos as it does this project. Since we are in the information age, people has become more dependent with computers, smartphones and many electronic gadgets. For those reasons, this application is really needed for people who works for example restoring pictures but also can be used by anybody with the goal of annotating pictures.

For people who works restoring pictures, this application will be really helpful because they do not have to do the work twice. Any imperfection could be tagged and documented by a large set of icons. Also if there is any mishap or there is anything else to be tagged, the project can be saved into a file allowing the user reload the project for make the necessary changes.

## 1.2 Main goals

The main goal of this project is to design and develop an easy android application to tag photos, including load photographs from the device or taking a shoot at the moment.

The main requirements that are pretended to be done by the end of the project are:

- Design and develop an easy interface.
- Design and develop a vocabulary of gestures to zoom, pan and add a tag on the photo.
- Design and develop an algorithm to calculate the exactly point of the icon when the user apply zoom or pan on the photo.
- Design and develop an own file to save and reload project.
- Design and develop and algorithm to take a shoot of each layer with the photo on the background.
- Design and develop plugins for the application.

Secondary goals of the project:

- Design and develop tagging but not only a point, also tag an area.
- Add Dropbox and Google Drive API’s in the project.
- Put effects on the main photograph.

## 1.3 Memory structure

This report is divided in six different chapters that are going to be explained below:

- Introduction: This section motivates the project and enumerates its main goals.
- Analysis: This chapter is focused in explaining the gestures vocabulary that is used and software, hardware requirements and the methodology used to develop this project.
- Design: This chapter is focused on the features that has been developed for the application such as the interface, usability and some algorithms that are implemented on it.
- Coding: This is going to englobe the most difficult coding parts that have been done on this project.
- Testing: Explanation of all test that have been doing on the application to verify that all is working.
- Bibliography.

# 2. Analysis

When I decided to do that project the main idea was clear, to develop an application to allow fine art's people work with it. Even though they decided not to proceed with the project, my tutor and I decided to make a more generic project including their requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable and defined to a level of detail sufficient for system design.

In this project we will develop an application for tagging photographs. An important aspect is to help the user, for example fine art's people, not to repeat their job twice.  For people who work as a restorers of art, first, they have to identify if there are some stains, knockings… and annotate and write down where they are located. Afterwards, they write a report about the damages they have found.

On this chapter, is going to be explained the requirements of the application, gesture vocabulary of it, in which programming language the project has been developed, which framework has been used to do it, the methodology used and finally, how long took each issue of the project.

## 2.1 Requirements

This application will only work under android operating system. And it is thought to be used on tablets and mobile devices.

### 2.1.1 Gestures vocabulary

First of all, a set of actions has to be defined in order to make a correct use of the application:

- In the first screen:
    - If the user makes a horizontal one finger swipe to right side, it will appear the menu of the application where we can do different things such as load a photo from the gallery, load a project, get a photo from Dropbox or Drive.
    - If the user points in the middle of the center image, it will take a shoot.
- In the second screen:
    - If the user wants to zoom in on the photograph, she only has to pinch and glide the fingers apart with continuous contact.  It does not matter where the user does the movement, but it has to be on the area of the photo.
    - If the user wants to make a zoom out on the photograph, only has to move two fingers and glide them toward each other with continuous contact. It does not matter where the user do the movement, but it has to be on the area of the photo.
    - If the user wants to save the project or create a screenshots of each layer, a horizontal one finger swipe to right side, will make the menu appear.
    - If the user wants to tag a photograph, first has to create the layer, then select if want a dot or area, and then, if she makes a horizontal one finger swipe to left side, all the icons that can use to tag the photo will appear. Select one icon and touch the location in the picture that is to be tagged.

### 2.1.2 Android

This application has been developed under API 14 of Android, that it is 4.0 version, also known as Ice Cream Sandwich. I have chosen that version as the minimum requirements because, when the user wants to save the project, reload it or take screenshots of the layers, if there is a lot of information it is possible that would take a while to do it, and for old devices it is possible that the application crash because of the low RAM it has.

This project has been developed with Android Studio 1.2.2. It is the easiest way to start developing in Android because it is pre-configures with all the necessary material to do it,

unlike Eclipse IDE for example, that the user has to download plugins or update the IDE to support Android programming.

### 2.1.3 User stories

In software development and product management, a user story is a description consisting of one or more sentences in the everyday or business language of the end user or user of the system that captures what a user does or needs to do as part of his or her job function. User stories are used with agile software development methodologies as the basis for defining the functions a business system must provide, and to facilitate requirements management.

User stories describes an interaction between the final user and the system. When writing a user story, it must include some acceptance criteria, perhaps in the form of a test case or a brief description of done. The meaning of done is that code, test are finished and product owner have approved the task.

User stories defined below have two parts:

- Front: Description of the task.
- Back: Used to describe the acceptance criteria.

Finally, the user stories of the application are specified:

User story 1:

**Front**: As a user I want to take a shot with the application.

**Back**: Given the user wants to take a shot, when the photo is taken, then the user can tag the photograph she took.

User story 2:

**Front**: As a user I want to search a photograph on the gallery.

**Back**: Given the user wants to open the gallery and select a photo, when a photograph is selected, then the user can tag the chosen photo.

User story 3:

**Front**: As a user I want to save a project, so that I can retake the tagging later.

**Back**: Given the user wants to save the current project, when she select the option, the application ask for a name of the file and then, the project is saved. The project file includes all tags in the correct location, layers and the picture.

User story 4:

**Front**: As a user I want to reload a project.

**Back**: Given the user wants to reload a project, when the file is selected, then the project is loaded into the application. All tags are at the same location as in the saved project.

User story 5:

**Front**: As a user I want to have different types of tagging.

**Back**: Given the user wants to have different types of tagging, when she select tag mode option, then a sub menu to select between dot and area style tagging appears.

User story 6:

**Front**: As a user I want to change the tagging mode from dot to area and put them together into the same layer.

**Back**: Given the user wants to change the tagging mode, when the option is selected, then a sub menu appears and the user can select between dot and area.

User story 7:

**Front**: As a user I want to have different types of icons for tagging.

**Back**: Given the user wants to have different icons, when plugin option is selected, then a pop-up appears and she can select between icons from fine arts and icons from world of worldcraft, then on the left menu, appears icons of what the user selected.

User story 8:

**Front**: As a user I want to put some effects on the photographs.

**Back**: Given the user wants to put some effects on the main photograph, when menu option is selected, appears a sub menu with different options of effects (nothing, black and white, and focus).

User story 9:

**Front**: As a user I want to write some extra information about the photograph.

**Back**: Given the user wants to write extra information, when the option menu is selected, then a pop-up appears and she can write some extra information.

User story 10:

**Front**: As a user I want to take a screenshot of each layer with the background photograph.

**Back**: Given the user wants to take a screenshot, when the option is selected, then a pop-up appears asking for a name for the shots and the type of the photo.

User story 11:

**Front**: As a user I want to download a photo from Dropbox.

**Back**: Given the user wants to download a photo from Dropbox, when the option is selected, then she can navigate for all directories that has on cloud and select a photo.

User story 12:

**Front**: As a user I want to download a photo from Google Drive.

**Back**: Given the user wants to download a photo from Drive, when the option is selected, then she can navigate for all directories that has on cloud and select a photo.

User story 13:

**Front**: As a user I want to have a menu options that appears when swipe to right side.

**Back**: Given the user wants to have a menu, when she swipe to right side, then the menu appears.

User story 14:

**Front**: As a user I want to have a menu options that disappears when swipe to left side.

**Back**: Given the user wants to hide the menu, when she swipe to left side, then the menu disappears.

User story 15:

**Front**: As a user I want to share my project files on Dropbox.

**Back**: Given the user wants to upload the project files to Dropbox, when the option is selected, then the user can select all files to upload.

User story 16:

**Front**: As a user I want to share my project files on Google Drive.

**Back**: Given the user wants to upload the project files to Drive, when the option is selected, then the user can select all files to upload.

User story 17:

**Front**: As a user I want to be notified if changes on the project have not been saved when I want to quit the application.

**Back**: Given the user wants to have a save quit, when quit option is selected, a pop-up appears asking to save the project.

### 2.1.4 Development methodology: Scrum

In this project we have used Scrum methodology. Scrum is an agile software development methodology. Scrum is a process with a set of best practices for working in teams for obtain the best result. In scrum, each team make a partial functional deliveries between 1 and 4 weeks.

The process is simple, as it is shown on the picture below:

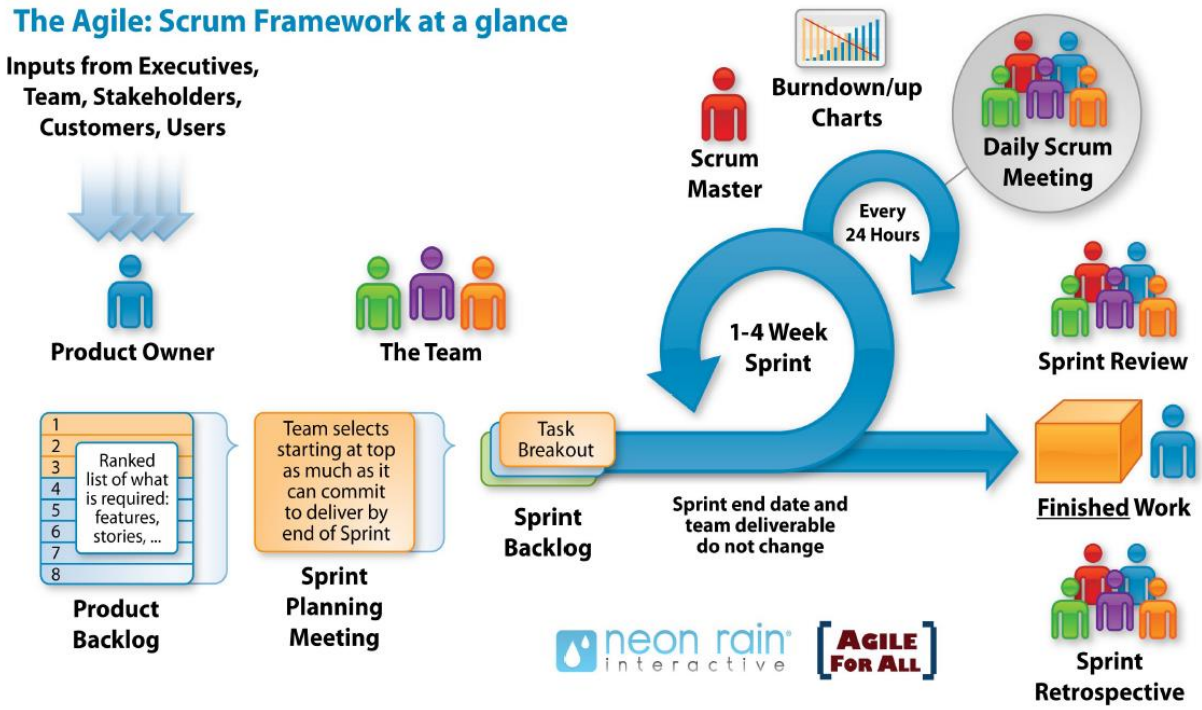


Figure 1

Figure 1 shows the Scrum cycle. First of all the product owner delivers the information obtained from costumers for example, by writing the product backlog of all requirements. When the backlog is update it, each team with the scrum master, have to decide which issues are going to be done on this sprint and how would take to do it. The scrum master, who is accountable for removing impediments to the ability of the team of the team to deliver the product goals and deliverables.

Next step is start working with issues. Each team has a daily scrum meeting each 24 - 48 hours with the scrum master who at the end of the sprint, has to do the burn down/up charts that shows a plot of work to do over time. It lets extrapolate if the team can complete all issues in the estimated time.

At the end of the sprint time, teams present to product owner all the finished work in a functionally application.

Finally, each team analyzes how they worked and what problems might prevent an adequate progress of the issues in a continuously and productive way.

The way I worked with my tutor was quite simple than all of these terms. We decided to meet one day per week to see how the work is doing and also to define more issues or search for a solution if something it was not done.

As in time table I explained, all figures there, are all the issues during the whole project. Observe that there are several issues that took me more than expected, for example the interface, which is so ridiculous how to center a simple button could become such a difficult task.

### 2.1.5 Timetable

On this subchapter, I show all issues that have been done during the period of developing this huge project. There are serval issues that took me more much time than I expected and I’m going to explain them more specifically in the coding chapter.

The timetable of the whole project is as follows:

| Activity | Start | End | Days |
|---|---|---|---|
| Storyboard | 10/02/2015 | 16/02/2015 | 6 |
| Interface | 17/02/2015 | 02/03/2015 | 14 |
| Graphics APP | 25/02/2015 | 10/03/2015 | 6 |
| Navigation drawer | 11/03/2015 | 24/03/2015 | 13 |
| Pop-up info photo | 11/03/2015 | 17/03/2015 | 1 |

| | | | |
|---|---|---|---|
| Pop-up share Dropbox and Drive. | 11/03/2015 | 17/03/2015 | 1 |
| Message when button is pressed | 11/03/2015 | 17/03/2015 | 2 |
| Create layers | 18/03/2015 | 24/03/2015 | 6 |
| Load photo from gallery | 18/03/2015 | 24/03/2015 | 3 |
| Toolbar | 18/03/2015 | 30/03/2015 | 12 |
| Rotate image | 18/03/2015 | 30/03/2015 | 1 |
| Create an own file | 18/03/2015 | 30/03/2015 | 12 |
| Class Diagram | 07/04/2015 | 14/04/2015 | 1 |
| Refractor code structure | 18/04/2015 | 28/04/2015 | 2 |
| Modify read/write file | 18/04/2015 | 28/04/2015 | 2 |
| Add thumbtacks | 18/04/2015 | 28/04/2015 | 1 |
| Create tags with Photoshop | 18/04/2015 | 28/04/2015 | 1 |
| Resize photo | 18/04/2015 | 28/04/2015 | 1 |
| Drag & drop | 18/04/2015 | 28/04/2015 | 6 |
| Change button Toolbar when dot or area is selected. | 18/04/2015 | 28/04/2015 | 1 |
| Make to persist information on pop-ups. | 18/04/2015 | 28/04/2015 | 1 |
| Create navigation drawer for icons | 29/04/2015 | 12/05/2015 | 1 |
| Create buttons on Photoshop | 29/04/2015 | 12/05/2015 | 1 |
| Set photo on the center of the screen | 29/04/2015 | 12/05/2015 | 2 |
| Use internal file search to load a project. | 29/04/2015 | 12/05/2015 | 2 |

| | | | |
|---|---|---|---|
| Zoom in & out | 29/04/2015 | 12/05/2015 | 6 |
| Change position of tags when zoom and drag | 29/04/2015 | 12/05/2015 | 6 |
| Refractor code structure | 13/04/2015 | 31/05/2015 | 2 |
| Add icon for plugins on Toolbar | 13/05/2015 | 31/05/2015 | 1 |
| Create icons plugins on Photoshop | 13/05/2015 | | 1 |
| Make a screenshot of each layer with the photo | 13/05/2015 | 31/05/2015 | 4 |
| Create trash with Photoshop | 13/05/2015 | 31/05/2015 | 1 |
| Drag & drop tags | 13/05/2015 | 31/05/2015 | 2 |
| Destroy a tag | 13/05/2015 | 31/05/2015 | 2 |
| Write the report of the project | 01/06/2015 | 25/06/2015 | 25 |
| Refractor read project | 15/06/2015 | 23/06/2015 | 3 |
| Refractor save project | 15/06/2015 | 23/06/2015 | 3 |
| Refractor screenshot algorithm for areas. | 15/06/2015 | 23/06/2015 | 1 |
| Refractor toolbar options. | 15/06/2015 | 23/06/2015 | 1 |

# 3. Design

Software design is the second step while a software is in development. After analyzing all requirements of the software, it is time to start thinking about the internal architecture of the application and how will be developed.

On this step, developers have to solve problems and planning software solutions about all requirements found before, and also can include the user experience design like a storyboard to help determine those specifications.

Once have been explained the gesture vocabulary and the user stories, on this chapter, is going to be introduced different aspects of the design of the application such as, the story board, the usability report, the class diagram and finally, some algorithms that had to be developed for the applications.

## 3.1 Storyboard

A storyboard is a graphic organizer in the form of illustration or images displayed in sequence for the purpose of pre-visualizing a motion picture, animation or interactive media sequence. Storyboarding is used in software development as part of identifying the specifications for a particular software. During the specification phase, screens that the software will display are drawn, either on paper or using other specialized software, to illustrate the most important steps of the user experience. The storyboard is then modified by the engineers and the client while they decide on their specific needs. The reason why storyboarding is useful during software engineering is that it helps the user to understand exactly how the software will work, much better than an abstract description. It is also cheaper to make changes to a storyboard than an implemented piece of software.

In this section will be shown a comparative between the storyboard and the final interface of the application.

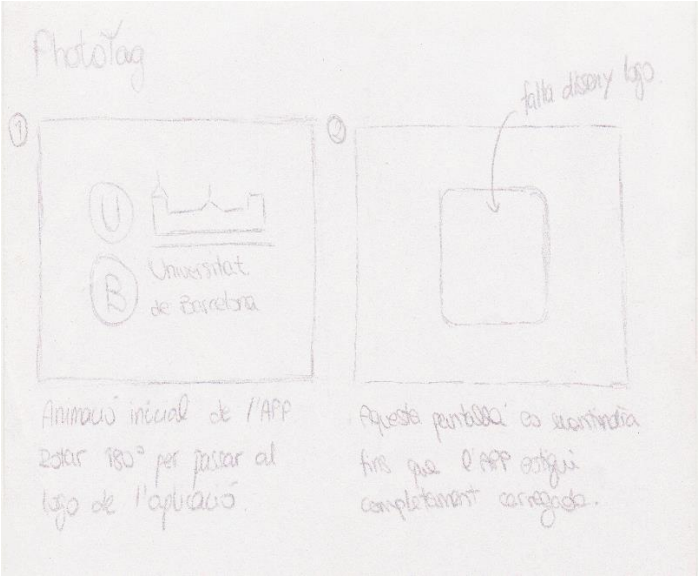


Figure 2

Figure 3

The application starts showing the university logo and after rotating 180 degrees, then the project logo appears (see figure 2). But finally, only the application logo appears (see figure 3). After the splash screen has ended, the first screen of the application where the user can do different things such as take a photo, select a photograph from the gallery, load a project, know who developed the application or download the photograph from Dropbox or Google Drive (see figure 4) is shown. On figure 5 and 6 you can see the final appearance of the first screen of the application.

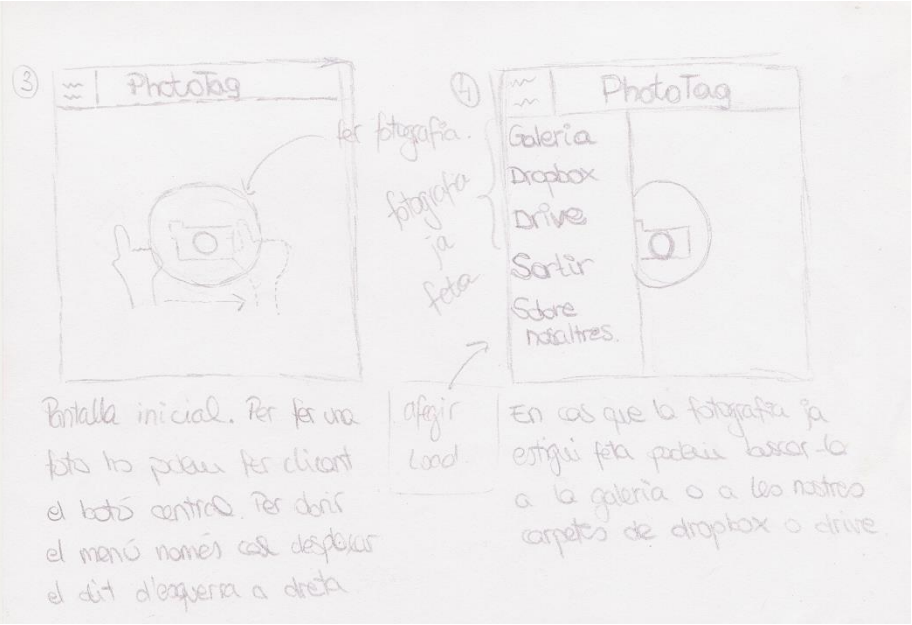


**Figure 4**


**Figure 5**

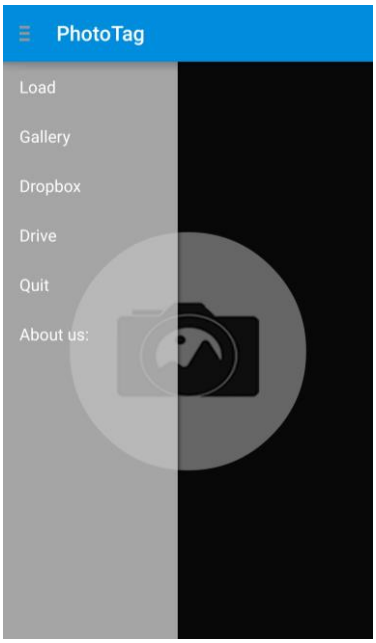


**Figure 6**

On this swipe menu we have different options:

- If the user touches the central button of the screen, the application ask the user witch program wants to use if on the device there is more than one application. If only there is one, the application of the camera will be executed (see Figure 7).
- If the user select “load” on the menu, a file system manager pop-up and the user can navigate through the different folder and search the project file she wants to load (see Figure 8). The project’s folder is on the root directory (see Figure 9).

- If the user select “gallery” if on the device there are several applications installed, will ask for open the gallery with one of them (see Figure 10), and if there is only one, will open the gallery (see Figure 11).
- Dropbox (see Figure 12).
- Drive (see Figure 13).
- If the user select “quit” a pop up appears asking the user if she really wants to leave the application (see Figure 14).
- If the user select “About us” a pop up appears showing some information about the developer of the application (see Figure 15).

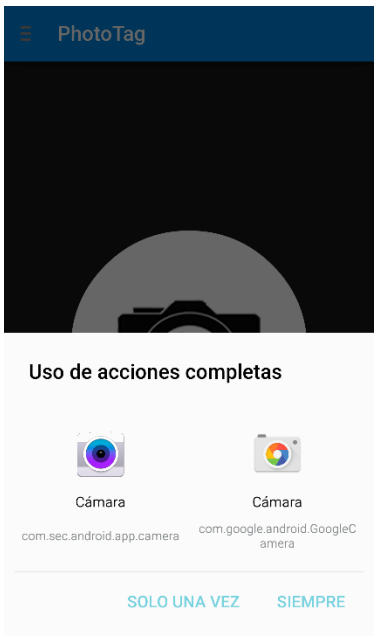


Figure 7

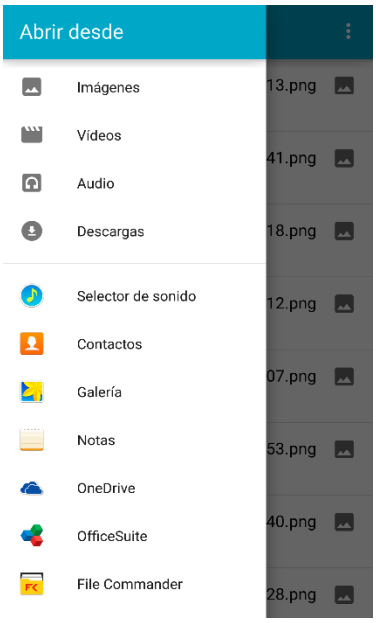


Figure 8

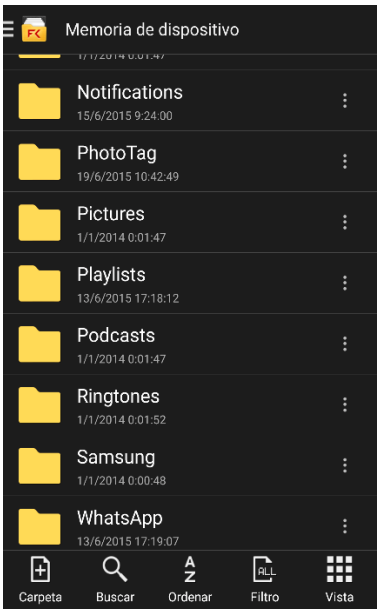


Figure 9

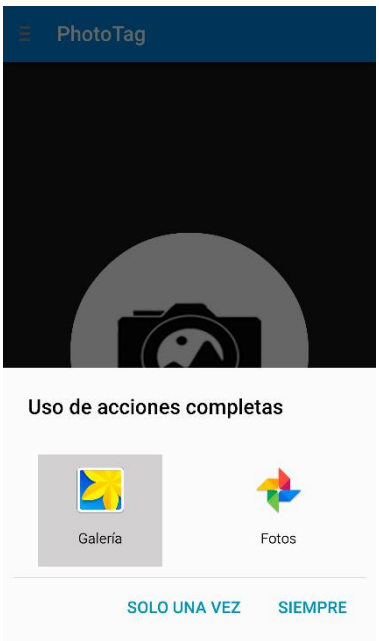


Figure 10

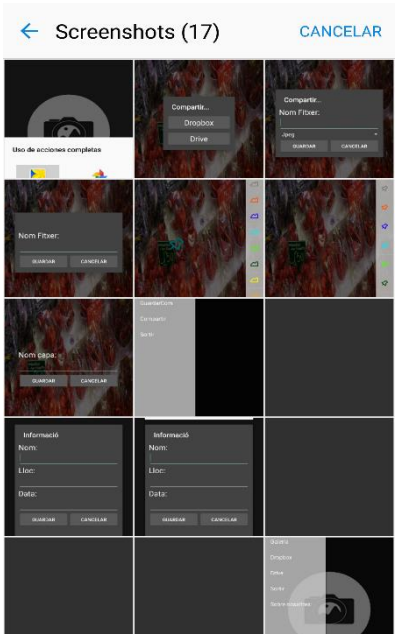


Figure 11


Figure 12


Figure 13

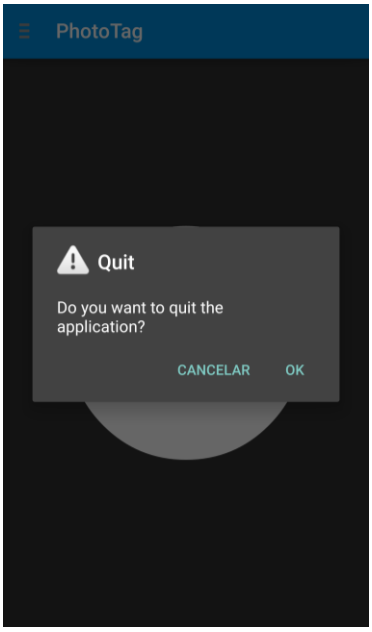


Figure 14

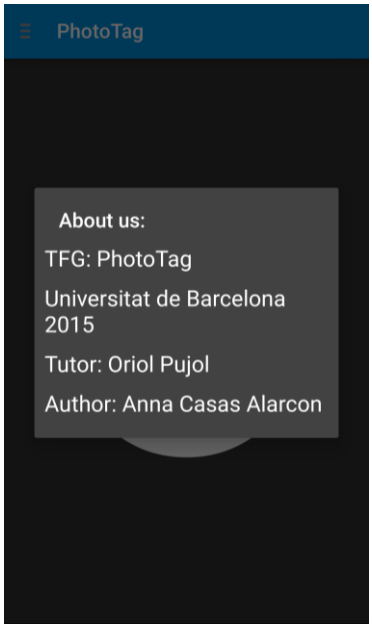


Figure 15

When the user has selected a photograph or load a project, the second screen appears with different options on the bottom to edit the image (see Figure 16 and 17).

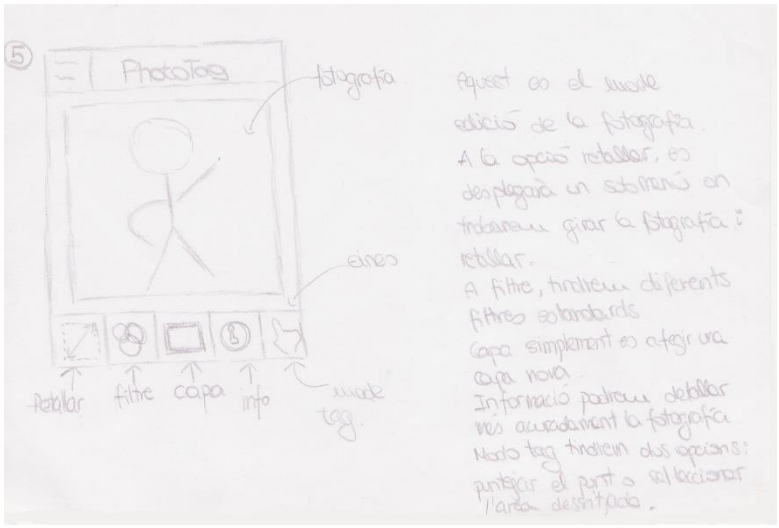

Figure 16


Figure 17

Let's explain a little about the options the application has on the toolbar. While explaining the toolbar you can see some options that have been added or removed between the storyboard and the final application. While explaining them, the reason because are added or removed is going to be explained.

The option cut of the menu (see Figure 18 a) has been removed because at the end of the project did not have any utility. The second option that appears on the toolbar is effects that has three sub options: nothing, black and white and focus (see Figure 18 b and 19). Third option, add layer, does not only add a layer but also shows all layers created before. When the user wants to create a layer, the applications asks for a name to have it identified (see Figure 20, 21 and 22). Then, we have information option, that when is selected, a pop up appears, and ask for some general information that can be useful for the user to write it down (see Figure 23a and 24). Tag mode option, when selected, appears a sub options asking the user if wants to tag a single dot or an area (see Figure 23b and 25). When one of them is selected, the user can swipe to left side, and a menu appears with all the different icons that can use to tag the photograph (see Figures 26 and 27). Last one, plugin, when selected, appears a sub options with all plugins loaded and an option to add more (see Figure 28). If the user touch add plugin, appears a pop up with all plugins that can be loaded (see Figure 29 and 30). The icons of the last plugin selected, will appear on the left swipe menu (see Figures 31 and 32).

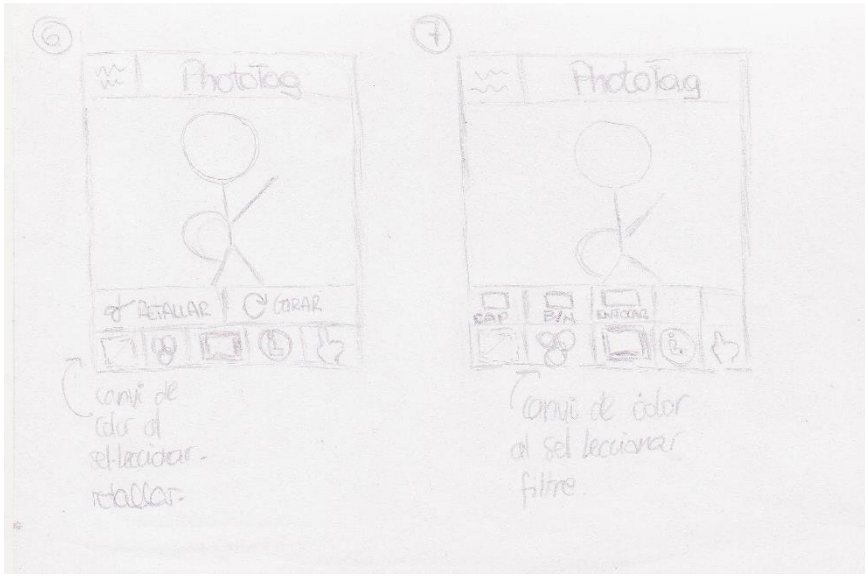


**Figure 18**

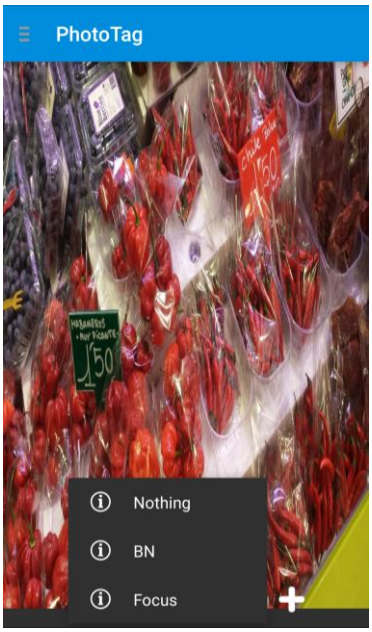


**Figure 19**


Figure 20

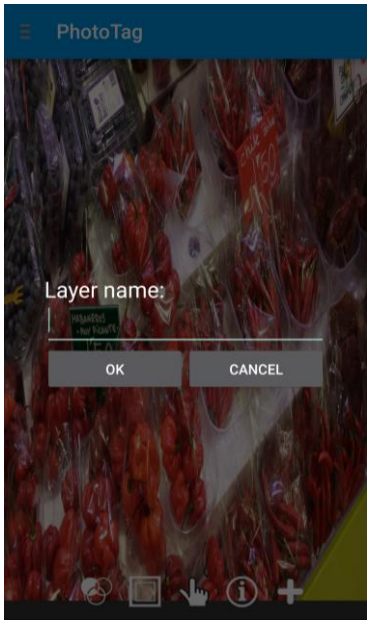


Figure 21


Figure 22

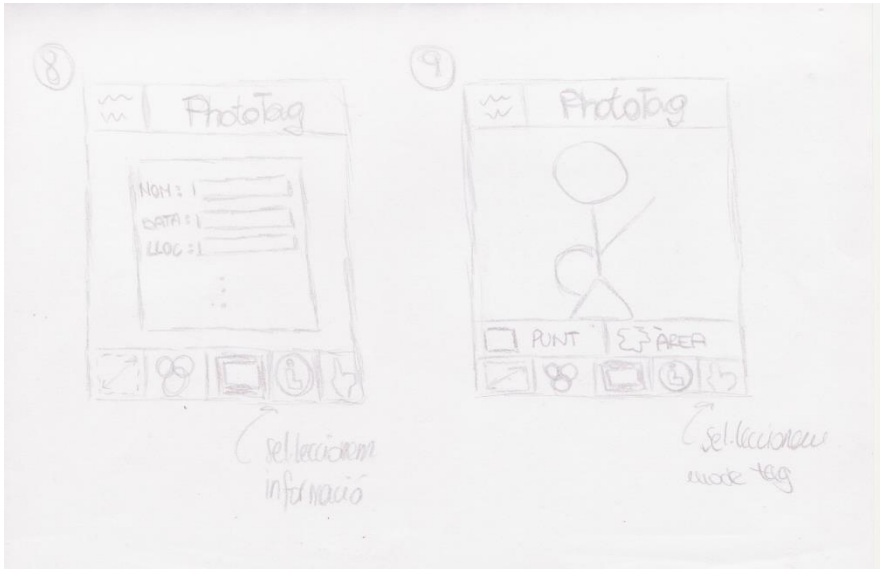


Figure 23

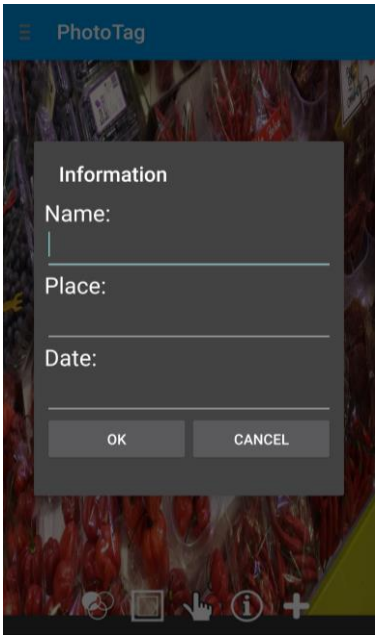


Figure 24


Figure 25

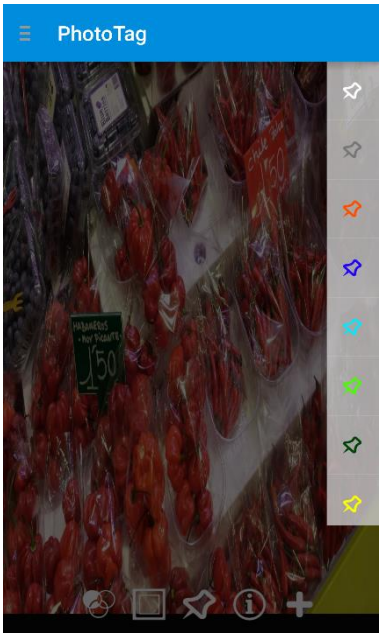


Figure 26

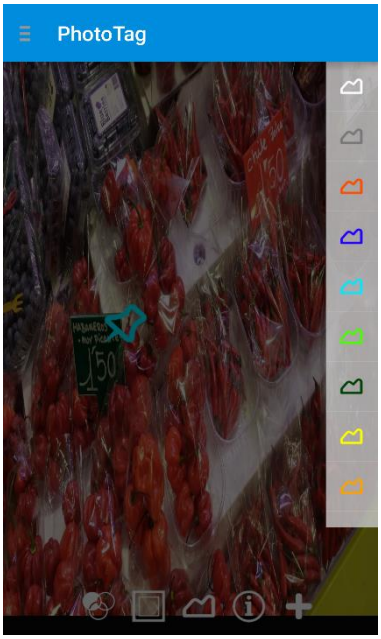


Figure 27


Figure 28

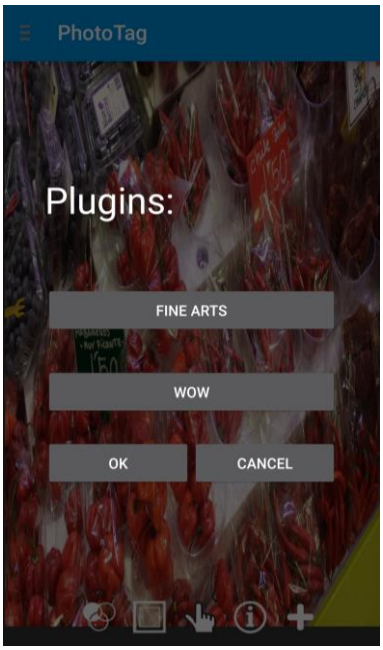


Figure 29

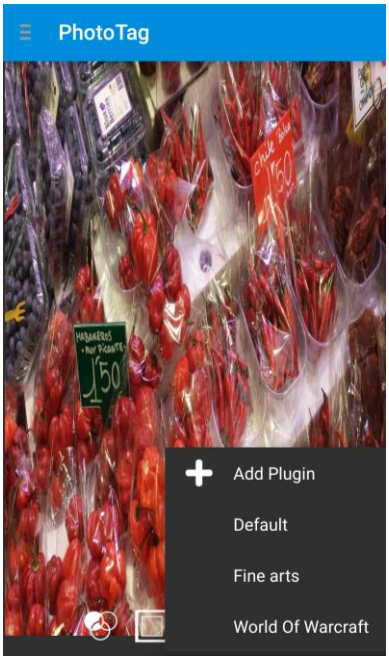


Figure 30


Figure 31

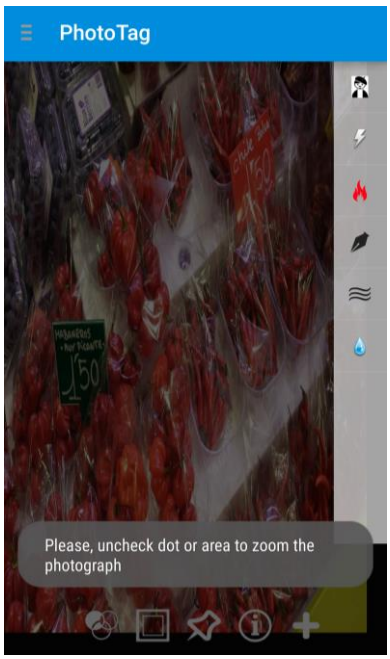


Figure 32

If the user swipe to right side, another menu appears with different options than the first one (see Figure 33 and 34). To do it simple, all options are going to be explained in the same order that appear in figures 33 and 34.
First one, save, when the user select that option, a pop up appears (see Figure 35 and 36) asking for a name for the project file. Second option, save as, appears another pop up (see Figure 37 and 38), asking for a name for the screenshots that the application is going to take of each layer and the format of the photographs that the user prefers. When share is selected, another pop up appears, waiting the user to select one of the two options of the pop up (see Figure 39 and 40).The appearance that have the APIs of Dropbox and Google Drive is shown on figure 41. Finally, quit, before closing the project, ask the user if she wants to save it (see Figure 42).

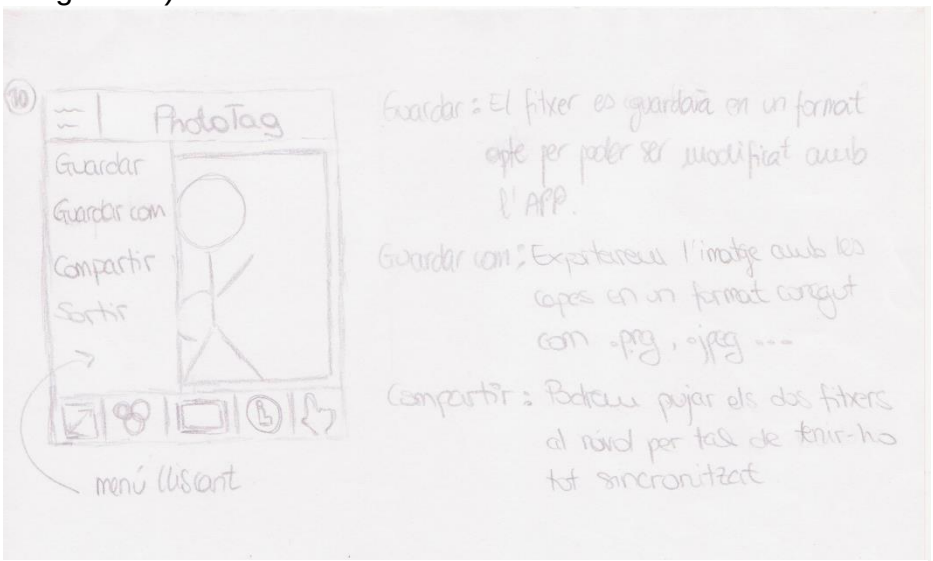


Figure 33

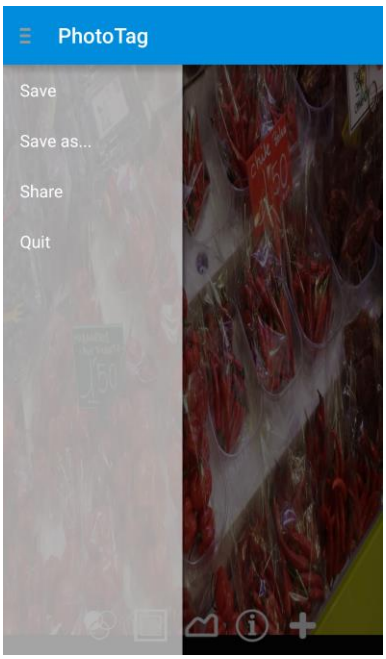


Figure 34

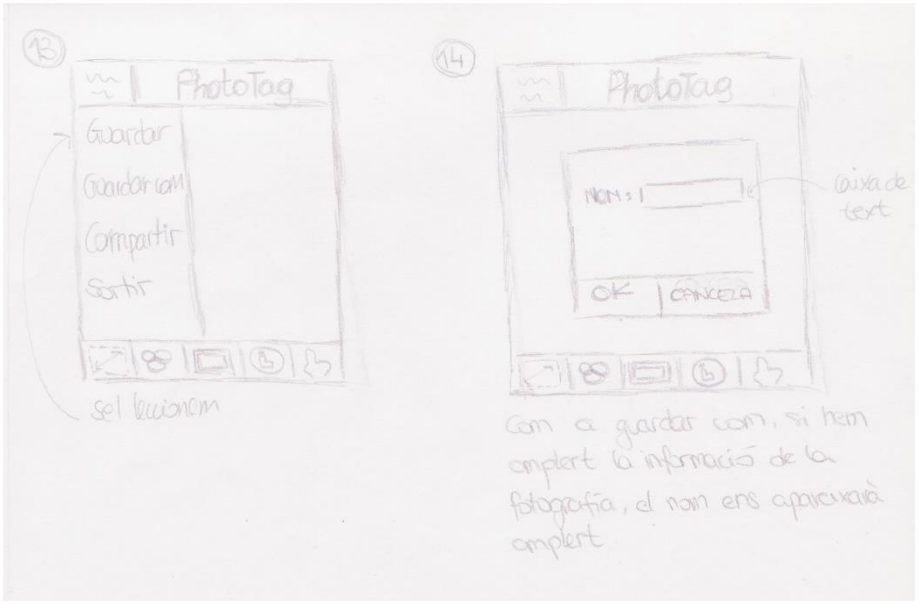


Figure 35

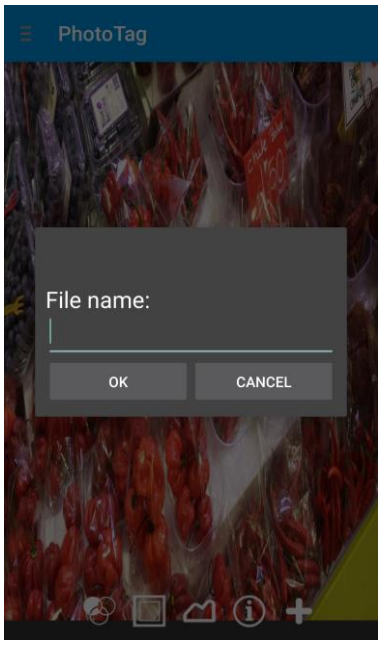


Figure 36

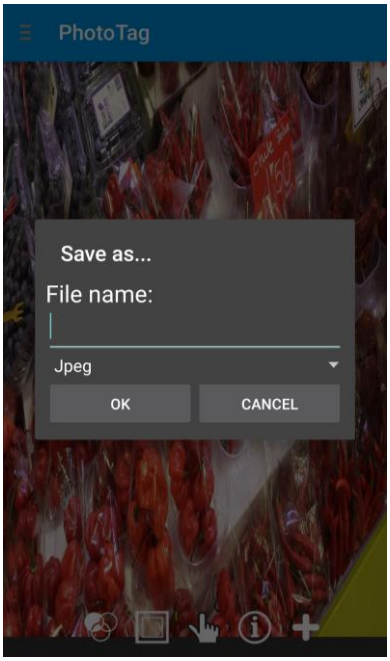


Figure 37

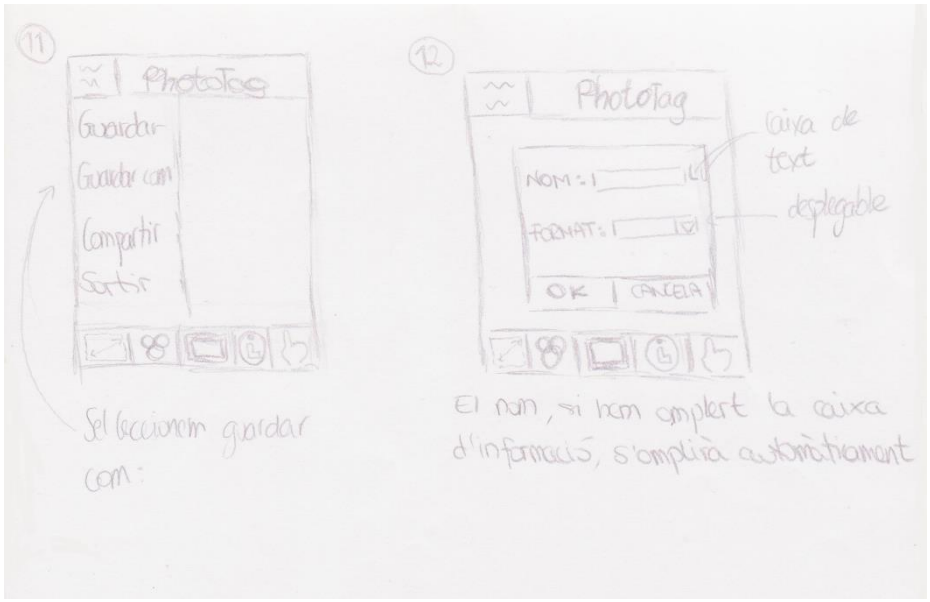


**Figure 38**

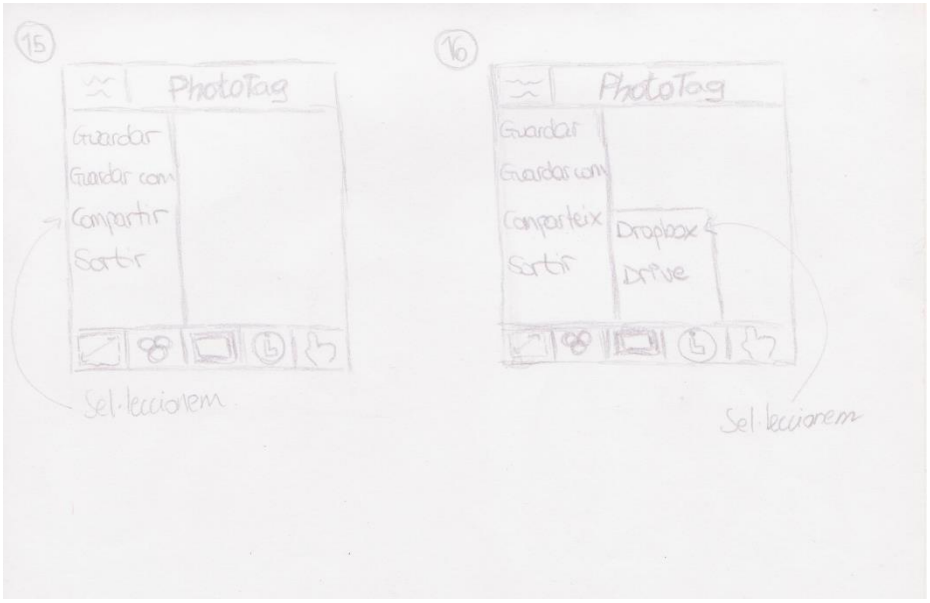


**Figure 39**

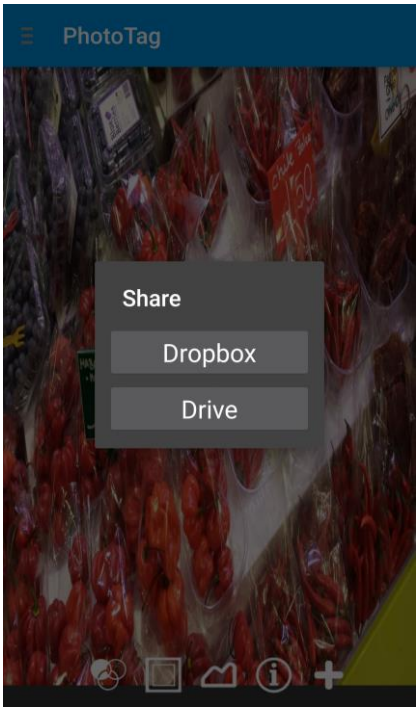


**Figure 40**

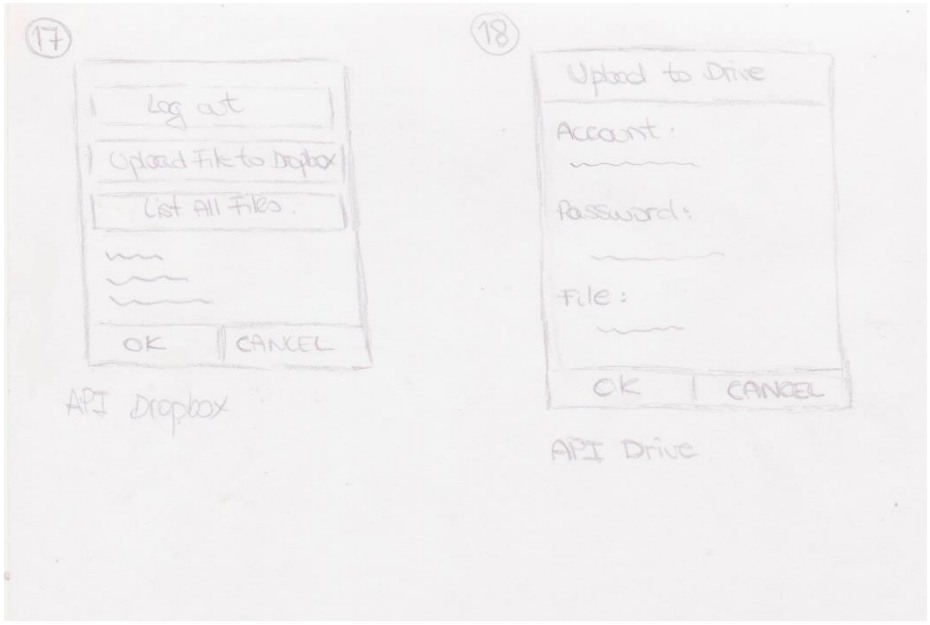


Figure 41

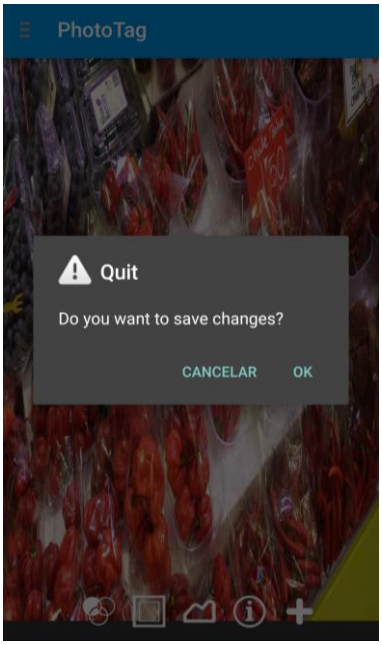


Figure 42

## 3.2 Usability

We want to introduce the user in our app for Android. We'll show each screen of the application to allow the users understand how it works.

First of all, the first screen we have the chance to take a photo with the camera of our device. To open the menu of the application the user has to do with one finger swipe to right side like. It doesn't matter where, just making the movement left to right, the menu will appear.

In our menu, we have different options:

- **Load**: We can search a project that we started before, but we haven't finished yet. So, the user can modified what did.
  If the user selects that option, the application will go immediately to file system and search the folder of the application.

- **Gallery**: If the user has taken the picture before, she can search it with that option.
  If the user selects that option, the application will go immediately to the gallery. If has more than one gallery, will ask which app to use.

- **Dropbox**: Now a days, people have most of their documents in the cloud, so, we have implemented the option to upload or download, the picture we want from it.
  If the user selects that option, the application will ask for the user and password of the account of Dropbox. After that, the user can navigate for his/her folders and export the picture that the user wants (is under development).

- **Drive**: That's the same case as Dropbox, but it's another way to have our documents, images, videos… in the cloud.
  If the user selects that option, the application will ask for the user and password of the account of Google Drive. After that, the user can navigate for his/her folders and export the picture that the user wants (is under development).

- **Quit**: As the name says, we'll close the application but, before going out, will show a message to verify that the user really wants to quit.

- **About us**: Here we'll explain who developed that application.

In the second screen, the user can edit the photograph and put the information within different layers. As you can see in the picture (see Figure 17), it is simple and the user will have a lot of space for edit the picture and work with it.

At the bottom of the screen, we have a simple toolbar where we can cut, put some filters on the picture, add a layer, put some information of the picture and choose witch type of tag we want to use. We'll explain carefully each option to make sure the user understands it all:

- **Filter**: We can modify a little our picture with some basic filters (is under development):
    - **No filter**: We can turn off if the user select any filter (is under development).
    - **B/W**: Turn the picture in a scale of grey (is under development).
    - **Focus**: If the picture is a little bit burred, the user can focus it with that filter (is under development).
- **Layer**: We add a transparent layer above the picture and put some tags on it. To make the work easier, the user can set layers visible or hide them. So, when we create the first layer, when the user click on the icon, a sub menu will appear showing the other layers that the user have created before and also, create a new one.
- **Information**: The user could need to add information from the picture for other people or him/herself. So when it is pressed, will appear a popup that the user can fill things as name of the picture, date, site, and other information that the user thinks that will be useful for other people.
- **Tag Mode**: In one picture the user can have a specified point that needs to be tag or can be a huge area, for that reason, I've decided to implement two types of tagging.
    - **Dot**: Singular point on the picture. If the user select that option, will appear an extra toolbar on the right of the screen with different types of tagging a dot.
    - **Area**: Huge area on the picture that the user wants to be tagged. If the user select that option, will appear an extra toolbar on the right of the screen with different colors to tag an area.
- **Plugin**: As I mentioned before, to make this application for all users, I have developed three types of tagging.
    - **Default**: The user can tag with thumbtacks.
    - **Fine arts**: The user will find different icons to tag the photograph, such as, stains, knockings…
    - **WOW**: On this one, will find enemies, missions, items, clans… that can be tagged with an icon.

Also, we have and extra menu on the left side of the screen like the other one, but with different options in it:

- **Save:** The user can save a project that isn't currently finished and wants to open it later.
- **Save as …**: The user can choose different types of formats to save the work and also, save the project if there's some mistakes and the user has to modify it.
- **Share**: Will appear a popup asking the user if wants to upload the project and the picture on Dropbox or Google Drive (is under development).
- **Quit**: Will appear an advertisement to make sure that the user wants to quit the project. Will ask if wants to save it or not, and after that, will go to the previous screen.

## 3.2 Class diagram

All the classes of the project are shown in figure 43, including methods and attributes that the app needs to work. The largest class corresponds to *EditTag* class. This class has all the logical to tag photographs, put effects, save the project… But the diagram is carefully explained.

As said before, *EditTag* has all the logical of the application and obviously to make use of the information expert pattern. In general aspects, all classes only have the information they need. This means that they only have information about their object and not the others. Is a good way to have all information ordered and capsuled on each class.

Using fragments for the right menu in both screens, most of the classes are generated internally by Android: *NavigationDrawerFragment*, *NavigationDrawerFragmentEdit*, *PlaceholderFragment* and *NavigationDrawerCallbacks*.

To make the code easier to understand, I create a class called *LayerParent* which has all layers saving them into an *ArrayList*. There is a differences between dot and area, so, to make the operations with components a little bit more faster, there are two *ArrayLists* one for each type of tagging. As we can see, it has class called *LayerChild* that inherits from it. The reason to develop that class it is because to put any icon above the main photo it needs an *ImageView* and for example when the user zoom, I cannot operate directly with the icon, I have to work with the *ImageView*. Also, the application has another two classes called *Dot* and *Area* that inherit from *LayerChild*. The only difference between both classes, is that in *Dot* only has to know one position on X coordinates and one position on Y coordinates and as opposed, in *Area* as its name indicates, all dots belonging to the same subset of points determine a region. The objective with that is if for some reason there a huge zone that has to be tagged, it is better to point this out with that type of tagging.

Finally, as I want this application to work like Photoshop, I developed the choice to save and reload a project to keep working with it. As it is a costly work, it is better that this two methods, has their own class to work.

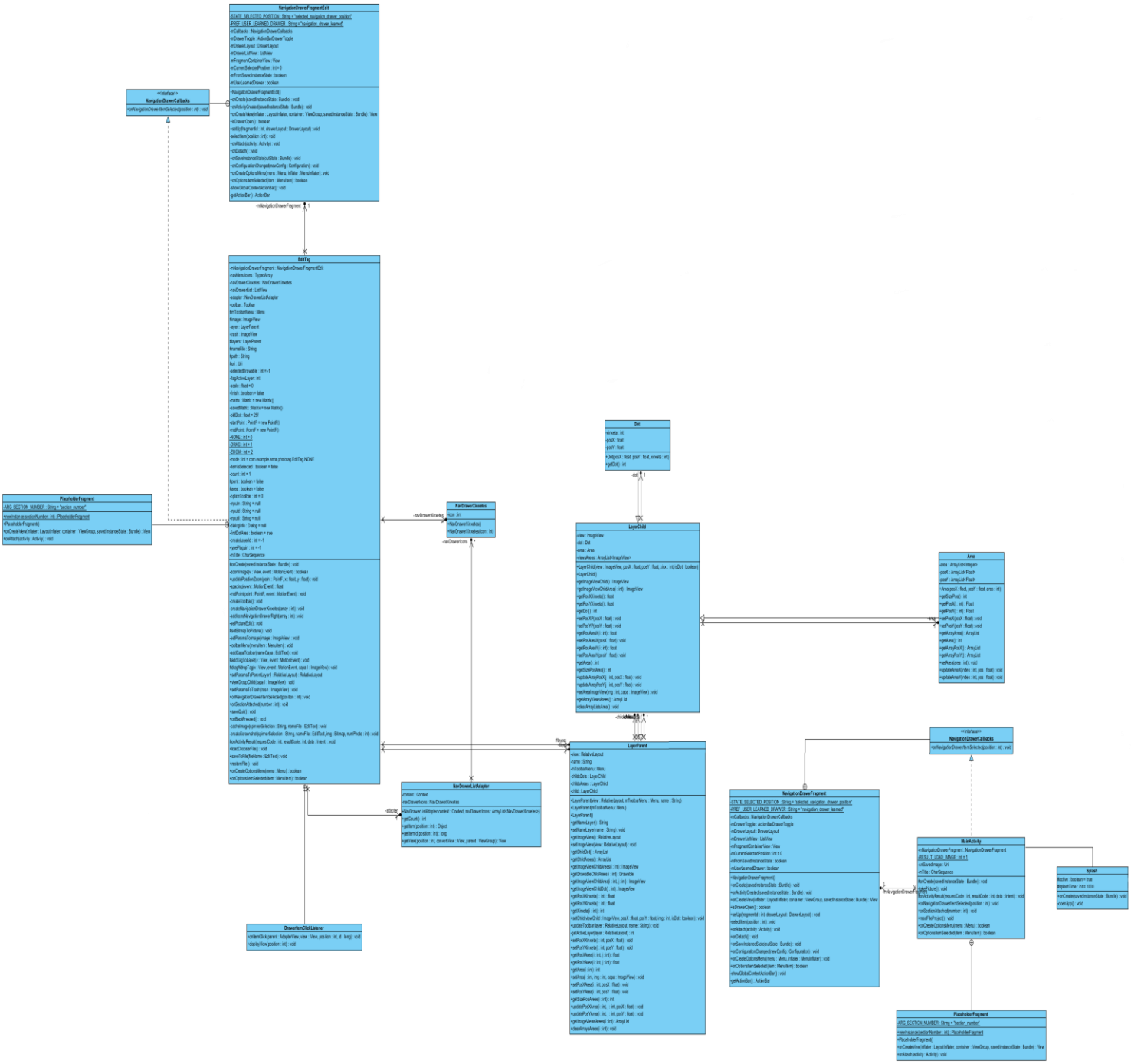

**Figure 43**

## 3.3 Algorithms

In this section the most important algorithms used on this application are described. They are: zoom, drag & drop, read/write the project file and take a screenshot of each layer.

- Zoom: Working with zoom in and out, not only on the photograph but also with tags, to make them persist at same position an affine matrix has to be applied. Figure 44 shows the translation matrix in homogeneous coordinates.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S & 0 & T_x \\ 0 & S & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

**Figure 44**

When translated to the picture domain, including the picture coordinates and the device coordinates it is better to explicity work with the following equations:

$$x' = scale * x + \left(\frac{dx_o + dx_f}{2}\right) * (1 - scale)$$

$$y' = scale * y + \left(\frac{dy_o + dy_f}{2}\right) * (1 - scale)$$

Where x, y are the pixel coordinates, scale is the distance of the pinch, dx0 and dy0 are the start coordinates of the pinch and dxf and dyf are the final coordinates of the pinch. This equation calculates the exactly position of the icon when the user pinch.

Scale defines the amount of zoom to be applied and it is computed as follows:

$$scale = \frac{dist_f}{dist_o}$$

Where $dist_f$ are the last coordinates of the pinch and $dist_o$ are the start coordinates.

Distf and distо defines the distance of the user's pinch and it is computed as follows:

$$dist = \left(\frac{dx_o + dx_f}{2}\right), \left(\frac{dy_0 + dy_f}{2}\right)$$

*Scale* and *dist*, will be recalculated until the user finish pinching the photograph. When it is finished, the translation equation is applied to recalculate each tag position.

In case of the photo, when the user is pinching it, there is a function called *postScale* native of the library of *Matrix* that zoom in or out depending on the pinch is doing the user.

- Drag & Drop: In case of drag & drop, it is quite different. I use this equation to recalculate the point but in this case, scale will be 0.

$$x' = scale * x + \Big( x + \big(dx_f - dx_0\big)\Big) * (1 - scale)$$
$$y' = scale * y + \Big( y + \big(dy_f - dy_0\big)\Big) * (1 - scale)$$

That is the same as applying this equation:

$$x' = \Big( x + \big(dx_f - dx_0\big)\Big)$$
$$y' = \Big( y + \big(dy_f - dy_0\big)\Big)$$

Where x and y are the pixel coordinates, dxo and dyo are the first drag coordinates, and dxf and dyf are the last drag coordinates

In case of the photo, when the user is dragging it, there is a function called *postTranslate* native of the library of *Matrix* that moves the photograph depending on the movement is doing the user.

- Project file: In order to create an own project file, first of all, ask to user to input a name to identify it. With that name, we create a folder inside the main folder of the application to save all data there. Next step is create the file. The structure that I choose to save it is shown below (see Figure 45):

```
content://media/external/images/media/9435
1
ImageView
Hola
1
Child Dot
1
2
Drawables
2130837591
510.0
434.0
Child Area
0
End
```

**Figure 45**

1. Uri from the main photograph
2. Descriptor text of what is going to save, *ImageView*
3. Save the name of the layer
4. Save the id of the layer
5. Descriptor text of what is going to save next, *Child Dot*
6. Save how many dots has that project
7. Save the id of the *Child Dot*
8. Descriptor text of what is going to save, *Drawables*
9. Save the id of the icon
10. Save the x coordinates of the icon
11. Save the y coordinates of the icon
12. Descriptor text of what is going to save, *Child Area*
13. Do again the steps 8 to 11 for *Child Area*
14. If user put some information about the photo will look like:
    1. Descriptor text of what is going to save, Info
    2. Save the name that the user input of the photograph
    3. Save the date that the user input before
    4. Save the place that the user input before
15. Descriptor text of the end of the file, *End*

This is the structure used in order to save a file project. To read this file, the same order if followed and each attribute is created accordingly.

- Take a screenshot of each layer: The algorithm is simple. First ask to the user in which extension wants the screenshots. Next step, is going throw both *ArrayList* and save all icons into a *canvas*. Finally, save the main photo and the canvas into a picture with the extension selected by the user.

# 4. Coding

As I mentioned in other chapters, there are several issues that took me more time that I expected at the beginning. On this chapter I want to explain which were the difficulties I found that made me spend more time on them.

- Interface: It is ridiculous how difficult could this turn out to be when wants to put a simple button on the center of the scene and android does not let you. This happen also with the interface color and size of any component. All these, depends extremely of which layer you are using as a container. Android has seven different main types: *FrameLayout*, *LinearLayout* (Vertical), *LinearLayout* (Horitzontal), *TableLayout*, *TableRow*, GirdLayout and *RelativeLayout*. My error was to use in one activity a *LinearLayout* and in the second one use a *GirdLayout*.
  *LinearLayout*, works like a table. This layer is recommended to use it when the developer wants to show a list, not to use it like a container of all components that are on the scene.
  *GirdLayout*, divide the screen into small boxes taking the shape of a grid without giving the chance to resize each box. What I had in mind at the beginning of the project, is to have the biggest space possible for the photograph to allow the user work more comfortable.
  At the end, I changed both layers and put a *RelativeLayout* because remove the problem of screen sizes on each device.

- Toolbar: First, the development notes of the official web site of Android (see 5 bibliography link), tell us that *Action Bar* can be positioned at the bottom of the screen. But this functionality has been deprecated for Android 5.0 *Lollypop*. If the developer wants the *split Action Bar* at the bottom has to use a specifically theme called *Holo.* Until I found this extra official information I spent so much time trying different coding ways to make it works.
  Finally, the solution was to take the standard that supports *Action Bar* functions (see 6 bibliography link).

- Zoom and Drag & Drop: On the internet, there is a lot of information of how zooming and dragging a photograph but there is not any information of how recalculate the position of a component that is above of it. After talking about it with my tutor, he found the solution. On graphics and data visualization, exist a geometrical transformation called affine transformation that help us on that problem. In our case, we need a translation and scaling.
- Screenshot of layers: To do the screenshot saving, I need to talk about first what is canvas in the environment of Android developer. Canvas represent an area where we can draw. To draw something, we need four basic components: a bitmap, one canvas,

a drawing primitive, and a paint. In my case, I only needed two: a canvas and different bitmaps, depending on the number of icons has each layer.

About this, also there is a lot of information on internet about creating a canvas and put different images on it, but in my case, I need to put the icons on the exactly position where they were placed. Spending time trying to set the images on the correct position, I found the solution (see 2 bibliography link).
First of all, I had to create a bitmap with the background image. Second step, create the canvas with this bitmap I just created before. Finally, this is the hard step for me, create a second bitmap for each icon on the layer and then, add it into the canvas. Now it seems obvious how does it works, but to add it, the function needs the second bitmap and coordinates X and Y.
Canvas has different types to add a bitmap on it, and on the internet, only talks about matrix, drawing primitive and paints. And the reason that icons does not work, for example, with matrixes it is because when the user zoom, the icon turns bigger or smaller and I do not want that.

# 5. Testing

On this chapter is going to show and explain the different types of testing that has been done to prove that the application have all requirements and works fine and also, the specifications of the devices Sony Xperia z2 and Samsung Galaxy S6 Edge where we have tested the application.

## 5.1 Testing videos

1. The user can take a shot whit the application.

   https://youtu.be/cCoRmr9EgkI

2. The user can search a photograph on the gallery.

   https://youtu.be/FeuPR6C_hto

3. The user can save the project.

   https://youtu.be/o_AqHGqNM9A

4. The user can reload the project and continue working.

   https://youtu.be/Cteaz7FzulQ

5. The user has different types of tagging.

   https://youtu.be/AjzJ-VUFxHQ

6. The user have different types of icons for tagging the photograph.

   https://youtu.be/NR2TVnqytO4

7. The user can't put some effects on the photo that is going to be tagged.

   https://youtu.be/0xY57pB-YqA

8. The user can make use of the different plugins and icons developed for the application.

   https://youtu.be/P5UGAdCT1hI

9. The user can put some extra information on the information dialog.

   https://youtu.be/oDeHqfucP_0

10. The user can take a screenshot of each layer that has created.

    https://youtu.be/A-nfK0aE-NE

11. The user can't download a photograph from Dropbox or Google Drive.

    https://youtu.be/e0ZSgSBoWn8

12. The user can’t share the project files on Dropbox or Google Drive.

https://youtu.be/5KhbFHZp2AQ

13. The user can make zoom on the photograph.

https://youtu.be/KpEtD30tBxk

14. The user can make pan on the photograph

https://youtu.be/imZGSwEJkMI

15. The user can drag and drop icons.

https://youtu.be/puGUVo_kbkU

16. The user can delete an icon of the screen.

https://youtu.be/p0EFOkqIz2w

17. The user can create layers.

https://youtu.be/p4ov5Pg9_Pk

18. The user can hide layers.

https://youtu.be/BisdX3o1J_8

19. The user can make layers visible.

https://youtu.be/dS3heVMr0rU

20. The user is notified before exiting the application.

https://youtu.be/94KndWDHAk0

## 5.2 Testing devices


Figure 45

Camera resolution (Front): CMOS 5MP
Camera resolution (Rear): CMOS 16MP
Internal memory: 32 GB UFS 2.0
RAM: 3 GB
Main display size: 5.1”
Main display resolution: 2560 x 1440 (Quad HD)
OS: Android 5.0, Lollypop


Figure 46

Camera Resolution (Front): 2.2 MP
Camera Resolution (Rear): 20.7 MP
Internal memory: 16 GB
RAM: 3 GB
Main display size: 5.2”
Main display resolution: 1920 x 1080 Full HD
OS: Android 4.4.2, Kitkat

# 6. Conclusions

In the whole project, I have not stop learning more and more about Android and what developers can do with it. On the first weeks working on the project, I thought it was just a simple application that the user can put tags on it, but it has not been as easy as I expected.

I think, one of the aspects that Android has to improve for the developers is to create the scenes more easily. As I mentioned before, I spend a lot of hours trying to center a simple button or put any component where I want.

The individual work on the whole project, has been a very important aspect to me. During all these four years, all practices had to be done collaborating between other students. But in this case, most of all had developed and learn it alone. So, my biggest motivation to make this works and grows was to reach success.

At the end, it has been a large project and I am satisfied with of all work I have done.

All main goals explained on previous chapters have been implemented. Some secondary goals have not been fully developed due to the lack of time. During testing process, I realized that there are several things that can be improved on the application such as define different languages, rotate the image and adding more plugins.

# 7. Bibliography


Android icons converter:

1. http://romannurik.github.io/AndroidAssetStudio/icons-generic.html#source.space.trim=1&source.space.pad=0&size=24&padding=8&color=33b5e5%2C100&name=ic_example

Canvas:

2. http://developer.android.com/reference/android/graphics/Canvas.html

Translation matrix:

3. https://en.wikipedia.org/wiki/Translation_%28geometry%29

Scrum:

4. https://en.wikipedia.org/wiki/Scrum_(software_development)

Toolbar:

5. https://developer.android.com/reference/android/widget/Toolbar.html
6. http://commonsware.com/blog/2014/11/18/android-5p0-deprecation-splitactionbarwhennarrow.html

Bitmap:

7. http://android-er.blogspot.com.es/2013/08/combine-bitmap-side-by-side.html
8. http://about-android.blogspot.com.es/2011/05/combine-two-images-in-android-java.html

Path from a photography or file:

9. http://stackoverflow.com/questions/2789276/android-get-real-path-by-uri-getpath

Pan and zoom:

10. http://code.cheesydesign.com/?p=723