Masters in Informatics Engineering
Dissertation 2014/2015
Final Report

Content Management Mobile Application using a Metadata Cloud Server

Tiago Emanuel Almeida Salvador
temanuel@student.dei.uc.pt

Supervisor Wit Software SA:
Pedro Pinto

Supervisor Department of Informatics Engineering
Jorge Sá Silva

Date: 2 of September of 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Page intentionally left blank

Abstract

The dissertation proposed subject is an implementation of a Metadata Cloud Server, which can communicate with clients and process Metadata from given contents and show this information in state of the art and innovative ways. The contents can be stored in different kinds of services, with a focus on Dropbox for this project. These types of content are crucial in people's lives in terms of culture and entertainment and how they are managed is important. By extracting Metadata that already exists in the contents or processing the Metadata in more complex ways like Smart Albums for Photos or Document Indexing, it is possible to give and present the content in different, more useful ways then simple lists or views. By building a complex Server application that communicates with the Metadata database and is accessible by REST API, it will be possible for any kind of client application to query the server and get the location of the content. Cloud Storage services don't usually support smart features or fast content browsing apart from regular Operating System folder like structure. By building a system as a complement to the Cloud Storage system it is possible for the user to access their data in more state of the art ways.

Keywords

"Cloud Services", Metadata, "Content Metadata", GPS, Geotagging, "Cloud Storage", "Exif Metadata", "Content Management", "Mobile Application", "Document Indexing", "Smart Albums", "Smart Playlists", "REST API"

Page intentionally left blank

Acknowledgements

The path traversed in this life cannot be taken alone without the support of those we love and cherish. Our lives are shaped by those around us even those who already left us and went with different paths and choices.

The importance of family cannot be understated. I give my thank you to my Parents who always tried to bring the best to me and gave me all the support in the world. They are the most important people in my life and I just want to now give them as much support they have given me as a retribution. I also thank my Grandmother, who already passed, for all the love she gave me during my childhood and how her sweetness and kindness taught me how a genuine bond can last forever. And I want to thank my Brother for finding the time to play with me and who showed me video games, being that what ultimately led to this.

Family love, I believe, lasts forever, but our Friends can make an enduring impact even though paths might be taken separately. So I thank all the people I once called or now call my friends. But especially now. Those who supported me throughout university, the internship and now my new career step. I thank them for bringing great joy and also knowledge that without a doubt shaped me in one way or another.

I want to thank my girlfriend, Gabriela Costa. These last two years have been everything I ever wanted and with you I have come great ways. You have made me a better man, and mainly, a better person. Your cheerfulness to every situation I face and how you always try to support me in any way you can, is a treasure that I cannot value enough. You have been crucial to my efforts and my work. Thank you. Also want to thank her family for accepting me and making me feel as a part of them and the help they have given me.

I want to thank Pedro Pinto for his patience and guidance in the best possible way he could. And also Jorge Sá Silva for his support when starting the internship and his care for seeing it finish well.

At last I want to thank Wit Software SA for giving me the opportunity to work in a good enterprise environment and their flexibility and, especially, Marta Coelho for her advices, conversations and support.

Page intentionally left blank

Table of Contents

List of Figures

Page intentionally left blank

List of Tables

Page intentionally left blank

Glossary

Android SDK	The software development kit for Android platform.
API	Application Programming Interface is a set of functions given by software to be accessed by any number of other external parties without those parties knowing the internal workings of the software.
Cloud	The term used for content or services that are stored in some physical server and are accessible to users.
Cloud Storage	Data stored in multiple servers physically but to the user seems like unique location where all his data is stored.
Development Team	A group of developers that implement the project.
Document OCR	Extract computer process able data from scanned images of documents.
Face recognition	An algorithm that takes into account known persons and tells which person is by detecting its face in a photo.
Full Text Search	Ability of quickly finding a search term in a fairly large database of documents by taking the advantage of indexing them previously.
Geotagging	Inserting location data like GPS in a resource.
Metadata	Information about information. Contains descriptive or technical fields about a given resource.
Metadata Cloud Server	The Metadata Cloud Server that is the main module of this project. It is accessible via REST API and it can manipulate a MySQL database for saving and fetching the Metadata from the content.
Product Backlog	Group of User Stories of the project.
Product Owner	The owner of the project in Scrum.

Relational Database	A database that can model relationships between its tables.
REST	It relies on a stateless, client-server, cacheable communications protocol and in almost all cases the HTTP protocol is used. REST is an architecture style for designing networked applications.
Scrum	Agile methodology for Software projects.
Scrum Master	Provides guidance to the Development Team in Scrum.
Smart Playlist	A playlist of music or videos that has special criteria defined in it to agglomerate content given those criteria.
Spring	An interval of time where development occurs in Scrum.
Spring Framework	A framework for building web applications.
Text-to-Speech	Algorithm to convert text to understandable human voice.
User Story	A feature or requirement specified according the user's point of view.

Acronyms

API	Application Programming Interface
CRUD	Create Read Update Delete
CS	Cloud Storage
DCMI	Dublin Core Metadata Initiative
DRM	Digital Rights Management
Exif	Exchangeable image file format
GPS	Global Positioning System
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ID3	IDentify an MP3
IMDB	Internet Movie Database
IPTC	International Press Telecommunications Council
IPTC	International Press Telecommunications Council
JSON	JavaScript Object Notation
MVC	Model View Controller

OCR	Optical Character Recognition
ORM	Object Relational Mapping
PLUS	Picture Licensing Universal
PNG	Portable Graphics
POJO	Plain Old Java Object
RDF	Resource Description Framework
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
TTS	Text-to-Speech
XML	Extensible Markup Language
XMP	Extensible Metadata Platform

Page intentionally left blank

Page intentionally left blank

1 Introduction

WIT Software SA located in Taveiro, Coimbra, purposed the dissertation Metadata Cloud Server and this document, along with all the appendixes, represents the work accomplished. This work was done with the support from WIT Software SA's Software Engineer Pedro Pinto giving direct support about any subject during the project and the support of Department of Informatics Engineering's PhD Professor Jorge Sá Silva for supervising the overall aspects of the dissertation and to provide guidance.

This document is structured by:

- **Introduction**: explains what was the context of the work and what are the objectives of the proposed Dissertation;
- **State of the Art**: presents the study done of various applications for the various contents;
- **Approach**: explains what was the approach to the project in terms of software methodology, the technologies used and the development tools. The complete specification of the Product Backlog and Sprints can be referred in the *Appendix A – Approach*;
- **Architecture**: shows the design of the architecture of the system, individual modules and the data model;
- **Results**: the look into the work done with considerations for each feature and the UI/UX of the application;
- **Evaluation**: tests for backing up the results obtained with detailed specifications;
- **Conclusions and Future Work**: the last chapter explaining what was achieved and what was not achieved and what can be done in the future regarding the project.

1.1 Context

This project enters the fields of Metadata, Cloud Storage and Smart Features. The usage of special information acquired by processing the content can lead to the development of smart features that enable the user to find, consume and share their content in ways that bring value to the experience.

Metadata is information about content and can be described in different terms in different contexts but mainly it exists to allow "resources to be found by relevant criteria, identifying resources, bringing similar resources together, distinguishing dissimilar resources, and giving location information."[1] The title, author or date created are examples of simple Metadata that can be used to identify content, of any type, whether it is inside the content itself, sometimes not possible, or stored in an external database and linked to the content. With even the simplest Metadata it contributes a long way in organizing information and finding that information which could turn out to be impossible otherwise.

There are 3 classes of Metadata[1] :

- Structural metadata which indicate how compound objects are put together;
- Descriptive metadata that serves to describe the contents using captions or titles which serves mainly for discovery and identification;
- Administrative metadata, which is for licensing and rights usage and also technical details like file type.

Any part of the content can be described using Metadata whether the content is a single object or a compound object. Metadata can describe, for example, the whole contents of a book and the book itself, as well the individual chapters of the book and even the paragraphs in each chapter.

Allied with the possibilities Cloud Storage enables (syncing of content between devices, high availability), the Metadata of contents can be enriched by processing the files and extracting custom data and save that data in an external secure Database. With this Database accessible also from everywhere and with high availability, it is possible to construct a supporting platform for the Cloud Storage applications.

1.2 Motivation

With mobility ever so increasing thanks to the incredible number of smartphones and tablets sold in the global market [2] a large focus on a mobile application to serve as the primary way to access content is justified. People travel, move around in the cities and have access to all kinds of applications and content on the Web. Users of Internet outside home are growing constantly as seen by the next graph comparing 2012 to 2014 growth in European Union countries [3] .

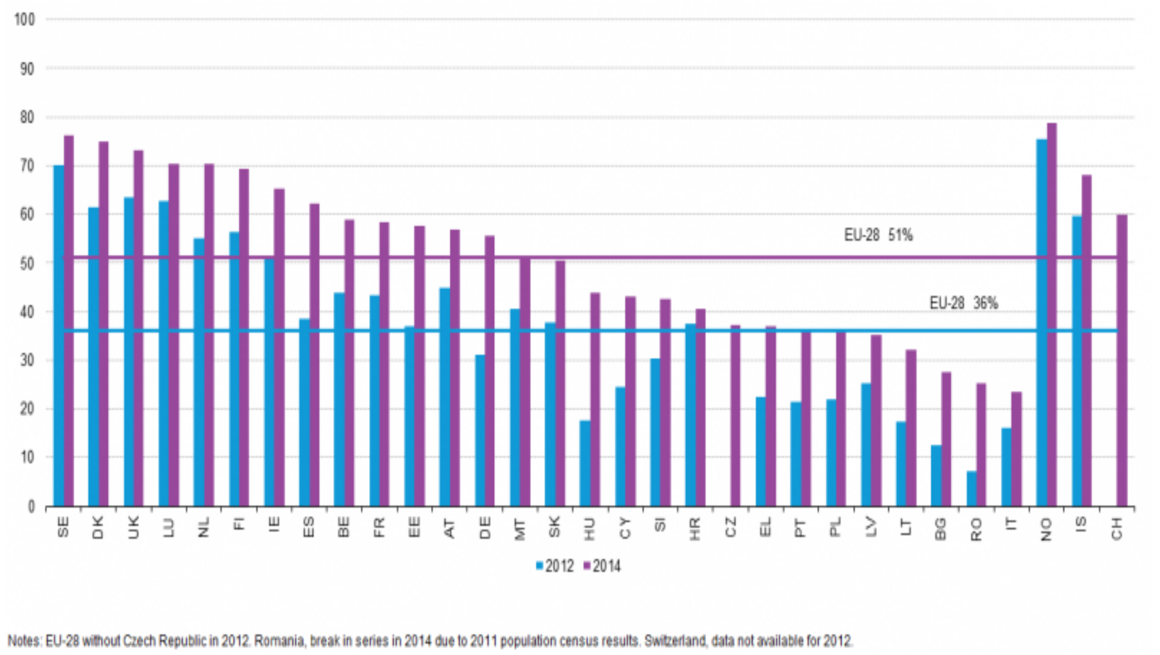


Figure 1 Growth of Internet users outside home

Numerous content types can be enriched by enabling the users to find them quickly and in interesting new ways. Photos and images are one of the most common of user-generated contents on the web and one of the focus of this project thanks to the interesting ways photos can be managed and discovered. Music can also be organized using Metadata about the file, like Author, Album title and even the Beats per Minute. Finally, Documents that can be indexed with their whole contents enabling Full Text Search retrieving the files fast and scanned documents can have their text extracted and searchable.

If we look into the most shared contents and their types between users in the European Union in 2014 [2] , we can see how this project perfectly fits in the current market trends.

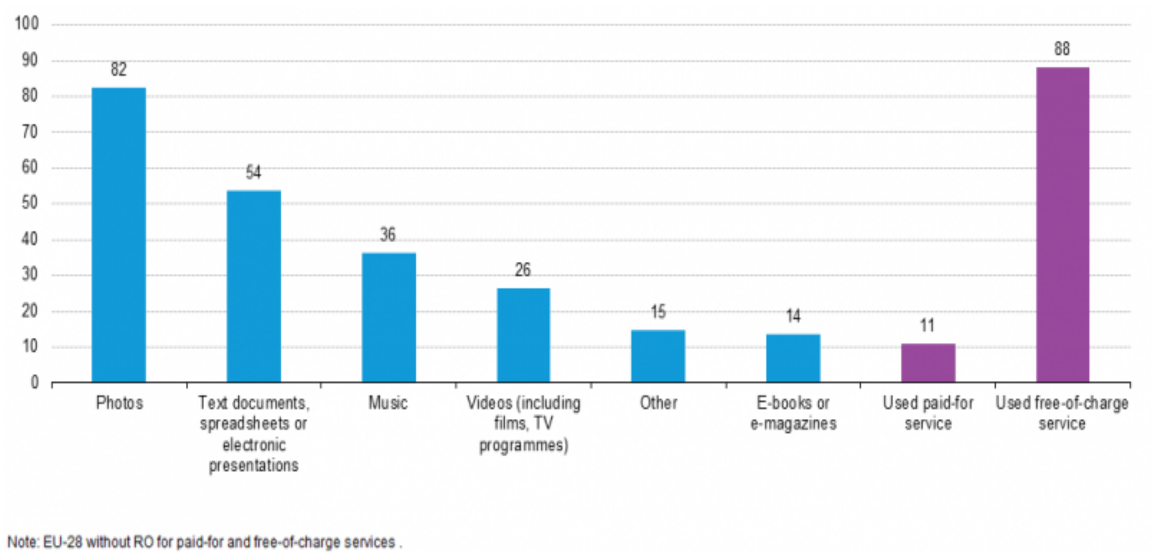


Figure 2 Usage of paid and un-paid services as well contents shared in the Cloud

This figure shows how the contents that this project wants to deal with are among the most shared, with attention to Photos, which is the most shared content between users.

Cloud Storage providers don't usually have good search support being mainly like an OS file system with very strict ways to find the content. Having a support platform that stores the data about the content and knows where it is and gives the users fast ways to access it is the main motivation for this project. If a user stores photos, for example, in their Cloud Storage provider, then uses the Metadata Cloud Server to ask where they are by using data indexed and readily available about the very content it is trying to access, the user can get photos from specific times, locations, with certain objects or people.

1.3 Goals

The goals can be divided into two categories: trainee scope and project scope. In terms of trainee scope, it is important for the trainee and WIT Software SA to get as much value from the project as possible, meaning, that the trainee can learn new concepts and important concepts for enterprise environment and put to practice what was learnt in the best way possible. For WIT Software SA the internships it offers are an investment in talent and work of students that are finishing their Master's Degree. WIT Software SA wants mostly that the trainee can bring great value to himself and to the company while doing this project so that the trainee can have a future in the company and that the internship was worth the effort.

In terms of scope of the project, the project has a good level of ambition and deals with several software stacks and tools that have bring great value to the trainee by giving him a year's worth of valuable knowledge of software development.

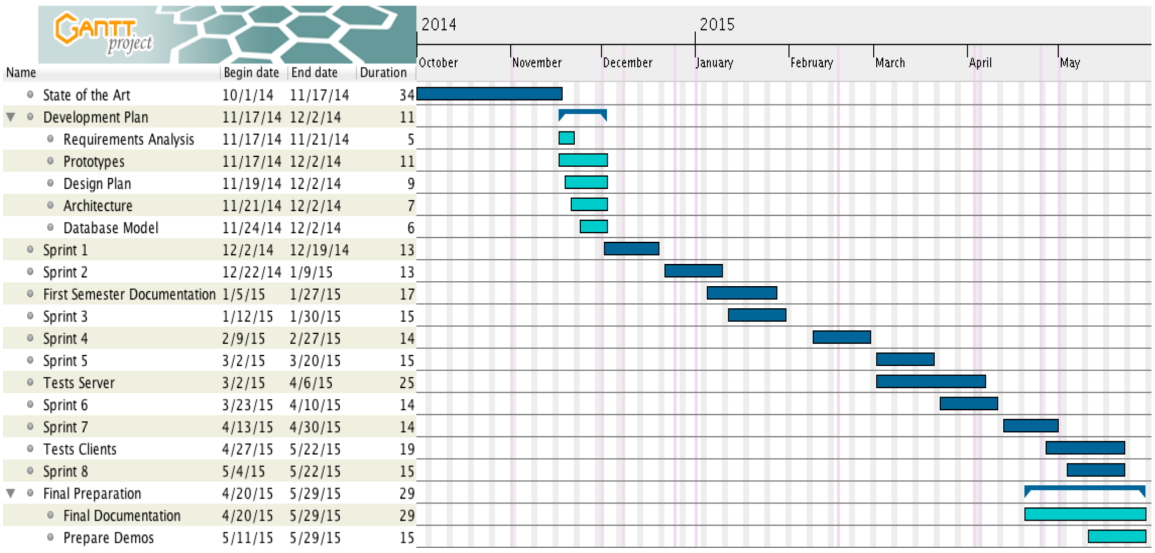
The goals for this project are:

- Have a robust understanding of what content means to the user and how to enrich it;
- Master the technologies used everyday by Software Engineers to develop the best applications in the market;
- Create a robust application that can show the interesting features that use Metadata to enrich the experience;

- Build a Cloud Server to enable the access everywhere to the contents Metadata and the contents themselves stored in Cloud Storage applications;

1.4 Planning

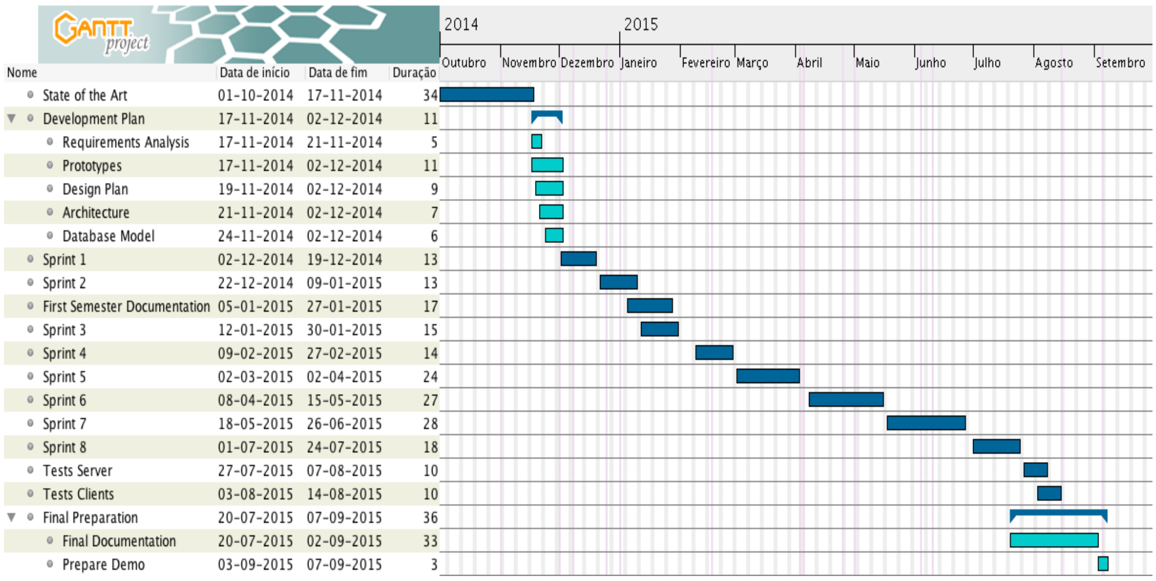
The planning of the project was always roughly connected with the software development methodology, meaning, there was a focus on planning Sprints, which are software development iterations during a certain amount of time with the main objective of building features that can bring value to the user. With this in mind, the planning that was elaborated at the end of the first semester is the one next presented.



Name	Begin date	End date	Duration
State of the Art	10/1/14	11/17/14	34
▼ Development Plan	11/17/14	12/2/14	11
Requirements Analysis	11/17/14	11/21/14	5
Prototypes	11/17/14	12/2/14	11
Design Plan	11/19/14	12/2/14	9
Architecture	11/21/14	12/2/14	7
Database Model	11/24/14	12/2/14	6
Sprint 1	12/2/14	12/19/14	13
Sprint 2	12/22/14	1/9/15	13
First Semester Documentation	1/5/15	1/27/15	17
Sprint 3	1/12/15	1/30/15	15
Sprint 4	2/9/15	2/27/15	14
Sprint 5	3/2/15	3/20/15	15
Tests Server	3/2/15	4/6/15	25
Sprint 6	3/23/15	4/10/15	14
Sprint 7	4/13/15	4/30/15	14
Tests Clients	4/27/15	5/22/15	19
Sprint 8	5/4/15	5/22/15	15
▼ Final Preparation	4/20/15	5/29/15	29
Final Documentation	4/20/15	5/29/15	29
Prepare Demos	5/11/15	5/29/15	15

Figure 3 Dissertation Plan

Since the the date of the final defense was in July, there were some changes to the overall plan. A large focus was on building a robust Android client application. This was the most area of interest for the trainee and since this was an internship for software engineering it made sense to have some consideration about the future. The features for the Android application were given priority, even more, because the most important aspect of the project agreed was to show the smart features. In the end the project defense was moved to September and the additional time was used to polish and write the report with more time.



Nome	Data de início	Data de fim	Duração
State of the Art	01-10-2014	17-11-2014	34
▼ Development Plan	17-11-2014	02-12-2014	11
Requirements Analysis	17-11-2014	21-11-2014	5
Prototypes	17-11-2014	02-12-2014	11
Design Plan	19-11-2014	02-12-2014	9
Architecture	21-11-2014	02-12-2014	7
Database Model	24-11-2014	02-12-2014	6
Sprint 1	02-12-2014	19-12-2014	13
Sprint 2	22-12-2014	09-01-2015	13
First Semester Documentation	05-01-2015	27-01-2015	17
Sprint 3	12-01-2015	30-01-2015	15
Sprint 4	09-02-2015	27-02-2015	14
Sprint 5	02-03-2015	02-04-2015	24
Sprint 6	08-04-2015	15-05-2015	27
Sprint 7	18-05-2015	26-06-2015	28
Sprint 8	01-07-2015	24-07-2015	18
Tests Server	27-07-2015	07-08-2015	10
Tests Clients	03-08-2015	14-08-2015	10
▼ Final Preparation	20-07-2015	07-09-2015	36
Final Documentation	20-07-2015	02-09-2015	33
Prepare Demo	03-09-2015	07-09-2015	3

Figure 4 Final Dissertation Plan

2 State of the Art

The State of the Art research in a specific field serves to create a solid knowledge of the latest and most important initiatives in the given field in terms of real life uses and also to create an innovative starting point to build upon the advances already made by other authors and creators. As such, the analysis presented in this document, will show various applications from various content types that either competes directly or indirectly with others and the features they provide.

As previously stated, the objectives of this project are to build a Server that can store Metadata from various content types and build a client application that is able to query the Server to fetch Metadata from local and Cloud Storage content. With this in mind the applications studied were among the most useful and with the most interesting set of features for each content type regarding its organization. Most of the focus was done on Mobile applications but some desktop or web applications are present since they have features of interest.

It is important to note that there was not a clear competition approach to this analysis. The emphasis was done in what already exists in common between applications and can be considered a group of fundamental features for the user and also different features and implementations that some of the applications might have that could prove important in the system to be designed. In other words, it means that there was no clear distinction made between competitors or non-competitors at all against the Metadata Cloud server and clients. The data is shown and studied in a way that can provide advantages to the system to be implemented which, in itself, will provide an edge to the Cloud Server and clients compared to the other applications.

Regarding server considerations, a list of the non-functional requirements, which are critical for the system to work, are taken into account and discussed the state of the art for each of them.

Since the focus is in the Metadata and how the information from the various files types can be presented to the user a brief explanation of the various Metadata types and how Metadata can be used in this project will be presented next.

2.1 Metadata

Custom Metadata is actually one of the main reasons for this project since storing content in Cloud Storages like Dropbox or Google Drive, usually Metadata is either lost or those applications don't support in any way custom Metadata and custom usage of that Metadata. With the Metadata Server it is possible to extract interesting, non standard Metadata like Face Recognition or emotions in Music and that Metadata be accessible from everywhere and be used in interesting ways. With those interesting ways in mind, the following part shows some of the ideas.

2.1.1 Metadata usage ideas

Some of these ideas are State of the Art of current applications out in the market. Other ideas can include the usage of the GPS for managing content in a location meaningful way. Each of the contents has unique information and users could find interesting the following features:

Regarding photos:
- Face recognition, having friends and family lists for easy access and queries. This is already presented in some applications like Picasa or Facebook;
- Night/day, summer/winter recognition;
- Good/bad photos, meaning, defocused, shaky or perfect photos and automatically editing them and getting rid of duplicates. This might be easier to achieve if the Metadata presented in both files is exactly equal.

Regarding music:
- Emotional footprint, organize by mood. This could be achieved by gathering some Metadata information from an API or simply researching the Internet to create a application specific emotional footprint of the music genres;
- Define quality standard in a simple way without the user having to understand what is MP3 320kb/s or AAC quality. For example enabling the user to discover that it has a specific music in a better format somewhere in the library;
- If a music is a single, if it is a famous one, which charts it topped, things that can be obtained via website scrapping or through an API;
- Recommendation system by using Natural Language Processing and Semantic Web ideas.

Regarding documents:
- Organize letters and bills and other important documents by using specific data only obtainable if the user scans the document then a deployed app in the cloud can get its Metadata;
- Document indexing to enable Full Text Search on all documents present in the Cloud Storage.

2.1.2 Storing Metadata

Usually Metadata is stored inside the files but it is also possible to use sidecars that accompany the file. This can enable that the Metadata extracted and processed be used in other applications and this can prove to be an advantage for the dissertation application. Having all photos in some Cloud Storage account and getting them and then modifying their Exif for example and re-upload is no good because it will be very heavy for the service. It must work with existing photos in that Dropbox or Google Drive account for example so Exif or Dublin Core or any other will be used and not modified in the files but instead will be modified if necessary and together with custom Metadata will be stored in the server database. Also, even if all photos are uploaded using the server it should be always better, if not possible to have a sidecar with standard Metadata, to store the Metadata separately, mainly because special metadata that is not standard will not be read by other applications.

2.2 Cloud Server considerations

The Metadata Cloud Server will need to have the standard non-functional requirements that every server should have which are: availability, performance, security, capacity and continuity. If the server is expected to work well it needs to have the best possible implementation with these requirements in mind.

2.2.1 Availability

Since we are dealing with data that is very important to the user, being files like personal photos or music the user listens to every day or important documents they just can’t find anywhere without the help of the Metadata server, then the server should be available 100% of the time except when maintenance is necessary and users are notified. The developer should take notice of single-point of failures and should make sure that no crashes would halt completely the server functionality. The server should be modular so that changes to a given content type Metadata requests don't affect all the others.

2.2.2 Performance

It is expected that many users will use the server at any given time. Heavy operations like synchronization of files, face recognition and OCR will be implemented on the Cloud and these must have the best performance possible so that the user does not give up on the client applications that are waiting for a response. Although it is out of the scope of this dissertation, performance should always be something to take in mind.

2.2.3 Security

Metadata can be a very fragile thing to deal with. A photo can be compromising or even the location in the Metadata can lead to faults in security if the information lends on the wrong hands. So users should feel safe trusting their precious private information with a Cloud service. This is a non-functional requirement that will always need to be taken very seriously and all efforts done to accomplish it. Data integrity is also very important since machine failure can lead to wrong information in files. But new users, with these measures, should not feel that it is hard to start using the application, so security should not be too much invasive.

2.2.4 Capacity

Due to having various users simultaneously the system might not be able to cope with all requests. So the code instances should be able to run on another machine or run multiple instances of code in the same machine. It is beyond the scope of this dissertation but it should be taken in mind when designing the architecture.

2.2.5 Continuity

This requirement is relative to being able to cope with failures in the server. If a disaster occurs or if a crash occurs in some part the server should be able to have code that can handle it and continue to serve as much as possible. Error messages should be sent to the user, but messages that are well understood. If data is corrupted the server should be able to understand it and stop corrupting the database for example.

2.3 Critical Features

Next it will be discussed the features that were found to be more critical for each given content type. Most applications only handle a single content. If we look at the most critical features for a given content it is possible to build a single application that can bring those features together with other contents. It is also important to see what kind of features are innovative or rare and build upon those. Sometimes when building a feature, it is also important to know what is being done best relative to that and incrementally add or simplify so that it gets better.

2.3.1 Photo content

Photos are very important to users and most users nowadays have thousands of photos laying around in their devices and Cloud Storage applications. The following group of features represents most of the photo/image organizers in the market:

- **Organize by folders**: ability to replicate the Cloud Storage folder system or show the local photos by folders;
- **Organize by albums**: ability to organize photos by creating albums with titles. Does not reflect on the file system since photos can belong to more than one album;
- **Insert keywords for a photo (tagging)**: ability to insert tags in photos;
- **Filter by keywords (tags)**: ability to search by tags;
- **Search by fields**: ability to search by any field present in the photos Metadata;
- **Face detection**: ability to detect faces in photos;
- **Face recognition**: ability to gather as much samples as possible from photos of a given person and suggest to the user to add a name to identify that person;
- **Search by faces**: ability to search for photos only of a given person or multiple persons in the same photo;
- **Sharing**: ability to share via email and social applications;
- **Read Metadata**: ability to read the standard Metadata inside the content;
- **Edit Metadata**: ability for the user to be able to edit Metadata present in the photo;
- **Location detection**: ability to know where the photo was taken by means of extracting GPS Metadata (Geotagging) present in the photo;
- **Location shown**: ability to integrate a maps application and show the location to the user;
- **Location organizing**: ability to organize photos by location similar to albums;
- **Integration with Cloud Storage**: ability to get and send photos from to applications like Google Drive and Dropbox.

2.3.2 Music content

Users normally have thousands of music files that are normally organized by folders or simple playlists that the users have to make. Applications that offer new ways to organize music can prove successful. The features set discussed next will have smart playlists for its focus:

- **Filter music by Metadata**: ability to filter music by using the various fields available in ID3v2 or another type of Metadata. Standard fields are a must which can be part of Dublin Core;
- **Organize by Metadata**: ability to organize music by artist, album, genre, etc.
- **Insert keywords for a music (tagging)**: ability to insert tags in music;
- **Filter by keywords (tags)**: ability to search by tags;
- **Search by fields**: ability to search by any field present in the music Metadata;
- **Get beats per minute information**: ability to extract the beats per minute value from a music;
- **Use BPM to create smart playlists**: ability to use the BPM extracted and create a playlist for a given special criterion;
- **Sharing**: ability to share via social applications;
- **Create smart playlists**: ability for the user to specify special criteria for a given playlist;
- **Integration with Cloud Storage**: ability to get and send music from to applications like Google Drive and Dropbox.

2.3.3 Documents

Apart from Photos this is one of the most interesting contents, which is indexing of documents and OCR of documents. Reading the information from documents, images or scanned documents and extract information from them and organize all the important documents for the user so that he can find critical information regarding his responsibilities for example is very interesting. There are various applications that do this and a feature set is presented next:

- **OCR the document**: ability to recognize the characters present in the document and extract them as accurate as possible;
- **Full Text Search**: ability to search through all documents in the database by using indexing capabilities for fast search.
- **Edit documents Metadata**: ability to edit Metadata fields;
- **Insert keywords for a document (tagging)**: ability to insert tags in documents;
- **Filter by keywords (tags)**: ability to search by tags;
- **Search by fields**: ability to search by any field present in the documents Metadata;
- **Sharing**: ability to share via email and social applications;
- **Reading standard Metadata**: ability to read and understand standard Metadata for documents like PDF;
- **Integration with Cloud Storage**: ability to get and send photos from to applications like Google Drive and Dropbox;
- **Read various file formats**: ability to read most used image formats like PDF and JPEG scanned images.

2.4 Comparison of features between applications

In this section it will be done a description for each of the applications that were found to have relevance to this project. The features used for comparison are the ones that were presented before.

2.4.1 Photo applications

The photo/image applications were found aplenty. The focus was done on Android applications and there were many applications that gave features that are considered standard nowadays like geotagging and organizing.

Sony Album [4]
One of the best applications for managing photos. The gallery is very well done with a great look showing a featured photo on top and then the gallery disposed in grid manner. The application has Favorites so that users can add favorite photos to be quickly seen and also shows the photos by folders. Regarding smart features, there is Geotagging with Places showing a map with clusters of photos and the user can check each of them for individual photos. There is also Faces which basically has face recognition and detection. Sometimes it works well and others no, but overall the face recognition works well enough. Finally, the application can connect to Facebook to get all the albums, profile and shared photos and also Picasa and Flickr. Finally it is possible to add geotagging to a photo that doesn't have and also add tags for fast searching.

Current version: Album 4.2

Main features: beautiful gallery; geotagging; Places; Faces; connection with Social Networks.

QuickPick Gallery [5]

Application for Android created by Cheetah Mobile Cloud. It has between 10 million to 50 million downloads and is considered one of the best Pictures managing application. It enables the connection to several Cloud Storage providers including Dropbox and Google Drive. The pictures are then put in that category and potentially can be set inside timelines. The view of photos from Cloud Storages are based in folders like the Cloud Storage application itself. The application can also process local photos. The photos can be organized, moved around and data about them is available, mainly in Exif format with location available through opening the Maps application.

Current version: QuickPic 4.5.2

Main features: simple and fast photo browsing; many Cloud Storage providers; some Exif data available; timeline.

PictureLife [6]

Developed by the homonymous company, PictureLife is a photo cloud service that gets all the photos from devices, social networks and cloud storages and puts it all together in one place so organizing and finding is faster and easier. It can sync also with other apps like Aperture and iPhoto. The user can send the photos from the places that were described before. The uploading can take some time but then users can edit the photo and the photo's Metadata online. It has clients for Android and iOS and also for desktop OS. The photos keep their Metadata and with it PictureLife can organize the photos in interesting ways like Memories which has a timeline with the various photos and also each day PictureLife can send an email to the user to remind him of a photo of that day in the previous year. It is possible to search by various methods including tags and Exif data. It is possible to favorite a photo, share directly to social networks, rate the photo, edit it with some simple filters and download the original. There are privacy settings since the photos can be accessed if they are public. There is face recognition and location map view. But in the iOS app there is only a filter named Faces, which show any photos with faces on them, without knowing, who is who.

Current version: PictureLife 3

Main features: timeline; memories; integration with numerous devices, social networks and cloud storage applications; all photos in one place meaning faster searching; face recognition; location map view.

Photos for iOS 8 [7]

Developed by Apple, now the standard application for iOS devices is Photos. It has smart search with the screen being pre populated with several default options like nearby, one year ago, favorites and home. Recent searches and also new search will try matching with months, city and other geographic names and names of albums. Photos that are imported can be titled, labeled, sorted and organized into groups known as events. The application has numerous options for sharing. Albums can be made into dynamic slideshows; can be shared via iMessage, Mail, Facebook, Flickr and Twitter. The user can create and share iCloud Photostreams since all photos are saved in the iCloud. Simple filters and smart retouches to photos are possible.

Current version: No information was found. It is a new app to replace the iPhoto.

Main features: simple and fast photo editing; unified library with iCloud; fast image browsing.

Dropbox Carousel [8]

This application developed by Dropbox came out in 2014 and serves as support for the photos users might have stored in their Dropbox account. Focusing on the Android application, it enables memories, albums and direct sharing with other Dropbox users. The application organizes photos by date and location and also as a Flashback feature that shows photos from the same day but from years ago. It can mainly serve as a backup service by having photos automatically taken by the device's camera being uploaded to Dropbox and local photos deleted to save space in the device while having the original stored in Dropbox.

Current version: Carousel 1.13.3

Main features: gallery with location and date organization; shared albums and photos; flashback; backup sync.

Google Photos [9]

Developed by Google initially only for Android KitKat and now available for all 4.0 plus devices, it enables backup in Google Cloud services and smart features like auto tags for objects in Photos and Albums automatically created. It also has various sharing, editing and montages and slideshows features. Photos are organized either by collections, date or types like People or Places.

Current version: Not detailed

Main features: gallery with various filters; backup; smart features enabled by processing photos in Google's servers.

Next is presented the table with the features described above and the applications. Each of the applications might or not have a feature and this information was gathered from various sources for each application. Sometimes no information for a given feature was encountered. If this was a shortcoming of the research or if no information is actually available, it is not possible to be sure. But the effort was done to gather as much information about each application. The table is presented next.

	Album	**QuickPic**	**Picture Life**	**Photos for iOS 8**	**Carousel**	**Picasa**	**Google Photos**
Organize by folders	✓	✓	✗	✓	✓	✓	✓
Organize by albums	✗	✗	✓	✓	✓	✓	✓

Insert keywords for a photo	✓	✖	✓	✖	✓	✓	✖
Filter by keywords (tags)	✓	✖	✓	✖	✓	✓	✖
Search by fields	✖	✖	✓	✖	✓	✓	✓
Face detection	✓	✖	✓	?	?	✓	✓
Face recognition	✓	✖	✓	?	?	✓	✓
Search by faces	✓	✓	✓	?	?	✓	✓
Sharing	✓	✓	✓	✓	✓	✓	✓
Reading Metadata	✓	✓	✓	✓	✓	✓	✓
Edit Metadata	✓	✓	✓	✓	✓	✓	✖
Location detection	✓	✓	✓	✓	✓	✓	✓
Location shown	✓	✖	✓	✓	?	✓	✓
Location organizing	✓	✖	✓	✓	?	✖	✓
Integration with Cloud Storage	✖	✓	✓	✓	✓	✓	✓

Table 1 Photos applications and features

✓ - The feature is implemented.

✖ - The feature is not implemented.

? - No information was found.

From this table with a reduced amount of applications it can be seen that almost all features are present in all of them. But all of the features can be considered critical since also these that are not very common can prove to be important for the project. A more detailed perspective with all applications is given in the *Appendix A – State of the Art*.

2.4.2 Music applications

There are a lot of music players that offer some kind of organization, even if simple. Some players like iTunes and Windows Media Player offer great features to organize via smart playlists and a good handling of Metadata. Other Cloud applications or desktop applications can also offer features that can prove important for the dissertation application.

iTunes [10]

One of the most used music applications, developed by Apple Inc. it offers numerous functionalities in terms of Metadata and music organization. In itself offers a lot of smart playlists functionalities and these all use ID3 standard metadata. But doing this kind of smart playlists requires input from the user by choosing the criteria for the playlist. It is possible to use a third party application to read the beats per minute of the music and that Metadata can be used by iTunes. The iTunes Match can match songs in the devices that are being uploaded to the iCloud to songs in the web services so that redundancy is not a problem and users can fetch the music from a repository. All metadata from iTunes libraries can be uploaded through iCloud.

Current version: iTunes 12.

Main features: smart playlists with user defined criteria; various Metadata filters; iTunes Match.

Windows Media Player [11] Erro! Fonte de referência não encontrada.

Standard player present in Windows OS from Microsoft, it has numerous albums, covers and general music Metadata functionalities. It has smart playlists, which are called Auto Playlists. With it the user can add various criteria to the list. Like iTunes it doesn't read BPM data but that data can be extracted from other applications and in Windows Media Player it is possible to use it with custom Metadata fields.

Current version: Windows Media Player 12

Main features: Custom Metadata fields; various filters; Auto Playlists.

MyCloudPlayer [12]

A music player in the Cloud developed by SoundCloud that has various functionalities like sorting by album, genre, artist, and various others. It has playlists also with smart filtering. The Android application does not have caching and download due to copyright problems. It has various sorting, by Title, length, play count, likes, repost, date, genre, BPM, also featured, trending, recent and suggested which are fetched using various APIs present in the web. It is also possible for the user to create playlists and search for genres with preset values or custom. The Android application has integration with XMBC and Chromecast for continuous playback. It is possible also to search users by followers or favorites.

Current version: Android application 14.8, no information for web app.

Main features: search by users and favorites; various filters including BPM; continuous playback with XBMC and Chromecast.

Xbmc media center [13]

Directly quoting the XBMC website, "XBMC is a free and open source media player application developed by the XBMC Foundation, a non-profit technology consortium. XBMC is available for multiple operating systems and hardware platforms, featuring a 10-foot user interface for use with televisions and remote controls. It allows users to play and view most videos, music, podcasts, and other digital media files from local and network storage media and the Internet." It has tag reading, smart playlists with genres, artists, albums, singles, songs, years, top 100, recently added, recently played, compilations, playlists and the users can use smart playlists with rules and limits. It has innumerous and all the important Metadata fields for music. XBMC rely on various third party websites to get the best possible Metadata for the users content. The websites are: fanart.tv; themoviedb.org; thetvdb.com; theaudiodb.com. Note: XBMC has changed its name for Kodi but very recently so for sake of other applications that communicate with Kodi and also since most users knew Kodi with its former name, this document uses XBMC still.

Current version: Kodi 14

Main features: communicates with numerous Internet databases for Metadata; numerous filters; multi-platform.

AudioBox [14]

Developed by iCoreTech Inc. it is a Cloud service for music with integration with its own desktop application, Dropbox, Skydrive, Box, Google drive, YouTube, SoundCloud and also allows offline playing. It is possible to edit metadata, to sort music, and drag and drop music to upload. It can filter collections by genres, artists and albums. AudioBox will find best artwork for matching artist/album and the user can put its own images. It has third party for sharing and get real time data and favorite artists. The web application has a HTML5 Upload panel. For playback it has prefetching and leverages browser cache so it starts streaming with no pauses. For preferences and editing it has various color schemes for cloud player. Finally it can fetch lyrics and Last.fm API.

Current version: No information for web application, Android app still in beta 0.9.22, iTunes app 1.3.2.

Main features: it has integration with multiple Cloud storage services; good web app written in HTML5; can fetch Metadata using APIs; playback has prefetching and caching for fast streaming without pauses.

Amazon Cloud Player [15]

Amazon own music player. It has interactive lists of tracks, categories and playlists. Prime members have access to millions of songs and hundreds of playlists, unlimited ad-free streaming and infinite playbacks. It has direct integration with the music purchases done on

Amazon and has instant search and play with smart search to find music quickly and play it. It has integration with iTunes and Windows Media Player but this is exclusive to the Amazon Music for PC application. The user's music library is always organized even if bought from iTunes or ripped from a CD. It is also possible to export music bought from Amazon to play in iTunes and Windows Media Player. It has numerous filters like artists, albums, songs and genres.

Current version: no information found.

Main features: integration with iTunes and WMP; all music libraries in one place all organized.

Google Music [16] Erro! Fonte de referência não encontrada.

A Google service, it provides functionalities similar to Amazon Cloud Player in terms of number of songs and ad-free content. It allows backing up of iTunes and most other music files like MP3, AAC, WMA, OGG and FLAC. Ability to upload and download music files and it is possible to store the user's songs in Google Cloud servers. It has integration with sharing for Google Plus including sharing an entire album. There is also some kind of support offline with caching of the latest heard songs. It has radio stations for mood and activity chosen by music experts and the subscription includes YouTube Music key. The service was free before, but now it has 30 days trial and then €9.99 per month.

Current version: No information found.

Main features: allows backing of user's songs in the Cloud; integration with Google Plus.

The table for each of the features for each application is presented next.

	iTunes	Windows Media Player	My Cloud Player	Xbmc Media Center	AudioBox	Amazon Cloud Player	Google Music
Filter music by Metadata	✓	✗	?	✓	✓	✓	✓
Organize by Metadata	✓	✓	✓	✓	✓	✓	✓
Insert keywords for a music (tagging)	✓	✓	✗	✓	✓	✗	✗
Filter by keywords (tags)	✓	✓	✗	✓	✓	✗	✗

Search by fields	✓	✓	✓	✓	✓	✓	✓
Get beats per minute information	✗	✗	✓	?	?	?	?
User BPM to create smart playlists	✓	✓	✓	✓	?	?	?
Sharing	✗	✓	✓	?	✓	✓	✓
Create smart playlists	✓	✓	✗	✓	✓	✗	✗
Integration with Cloud Storage	✓	✗	✓	✓	✓	✓	✓

✓ - The feature is implemented.

✗ - The feature is not implemented.

? - No information was found.

Table 2 Music applications and features

It is possible to see in this table that the desktop and other devices applications have more functionalities regarding editing Metadata from files. Since Cloud services normally get music from deals with recording companies it is hard to enable Metadata editing. But it is also possible to see that there is a strong degree of integration with Cloud or other devices music and that applications enable smart playlists and filtering of Metadata.

2.4.3 Document applications

This type of applications is proving to be very useful for the users. There is a set of applications that are very robust and that are presented here. Most can read well the characters from images and PDFs and can enable search on its contents.

Neat [17]

Neat enables various search methods for document title, category, date, type and action type. It has various lists with expenses and other categories. Depending on document type users can filter and search for documents via receipts, vendor, category, date, payment type, sales tax, total amount, tax category or reimbursable. The Neat Android application can scan receipts, business cards and any other document. It has Intelligent Text Recognition that reads and extracts key information such as names and addresses from business cards and tax and totals from receipts. All Metadata can be edited. It is possible to create expense reports right from the phone or tablet and it is possible to sync between all devices. There are sharing features and enables comments from users in the documents. Metadata can be exported to Excel or CSV and others.

Current version: Android application varies with device.

Main features: great OCR engine; enables various data from documents; various categories; can scan directly with Android device camera.

Evernote [18]

A very successful application created by Evernote Corporation. It doesn't focus only in OCR of documents and more on notes but it has OCR capabilities and features. Users can take photos with their device of documents and the images and PDFs are sent for processing in servers in queues. Premium users are put in front of queue. It is possible to have many matches possible for each word present in documents scanned so each document has a PDF attached with the words, although not available to user, to facilitate search. Enables photos of documents and creation of notes and tagging. PDFs must not contain selectable text, only bitmap image. Scans of documents are qualified for OCR. Can index 28 typewritten languages and 11 handwritten.

Current version: The information given is that it varies with device.

Main features: overall an extremely useful application with OCR features; OCR is done in server so no heavy processing in devices; tagging and notes in documents.

Adobe Acrobat [19]

Created by Adobe, it scans documents and OCR them and make them into PDFs. It makes text searchable via the OCR. PDFs in themselves have Metadata that the user can use to input title and other fields data. It is possible to OCR during scanning process or already scanned documents. There are two options for OCR, which are, searchable image that allows keeping of the scanned image and adds a hidden layer on top of the image of searchable text, or the other option is clear scan if the user wants to keep the look of the document the same but still want to convert the scanned image to text so as to reduce the file size. OCR custom settings with lossless compression for color/grayscale and CCITT Group 4 for monochrome images and this will ensure that OCR gets to work on the highest quality image thereby improving the OCR accuracy. This is useful in low resolution scans (<=150DPI). Relative to already scanned documents, if the image format is TIFF, can be converted to PDF and run OCR to improve accuracy.

Current version: Adobe Acrobat 11

Main features: OCR configuration for better results; searchable text; editable Metadata.

ScanMoutain [20]

A document organization application, it backs in the cloud the documents the user uploads. It makes documents searchable by keywords and enables access of documents online. There is no desktop or smartphone application, just online web application. It is possible to upload files by scanning, sending as an email, faxing files into

ScanMoutain or uploading them to the website. Then data is extracted as searchable text, allowing keyword searches that make finding files instantaneous.

Current version: No information was found.

Main features: web application only; easy uploading; all work done by ScanMoutain servers enabling the searchable text.

Abbyy [21]

It has various OCR applications but regarding mobile it has an engine that enables users to take photos and OCR them. It also enables transfer of business cards to mobile address book. It can capture words and translate them on the go and get necessary data with help of barcode recognition. It can read text in various languages easily. With correction of image skews, it can improve OCR accuracy. Other features include: document orientation detection; hyphenation support; confidence level indicator. Abby has improved data analysis algorithm, which discards unnecessary information in the image. It can recognize business cards in 23 languages with fields: first and last name, position of the cardholder, various types of phone numbers, email, company name, web site and postal address of the company. Barcode recognition types 1D and 2D. 61 recognized languages. Preserves multi column text and preserves character fonts like properties bold, italic and underlined. It has two recognition modes, one fast for express with images of good quality and full mode for accurate recognition for low quality images.

Current version: No information was found.

Main features: strong OCR engine; OCR for mobile; various capabilities and features that are useful in a daily basis like scanning of business cards or language translation.

CamScanner [22]

CamScanner helps scan, store, sync and collaborate on various contents across smartphones, tablets and computers. The phone camera can scan receipts, notes, invoices, whiteboard discussions, business cards, and certificates among others. It is possible to optimize the OCR quality by smart cropping and auto enhancing that makes text and graphics look clear and sharp. OCR documents are then possible to be searched by any keyword, title, note or images. It is possible to export text from images to text files for sharing and it is possible to share documents in PDF and JPEG format via social media, email or sending the document link. Also regarding social features, it is possible to invite friends and colleagues to view and comment scans in a group. There is editing of documents possible with watermarks, and security like using a password for important documents. Syncing between platforms is possible also. The third party storage applications are: Box, Google Drive, Evernote, Dropbox and OneDrive.

Current version: CamScanner Android 3.6.1.

Main features: powerful sharing capabilities; optimizing OCR; multi-platform.

Google Drive [23]

Google Drive is a Cloud Storage application but while uploading documents it is possible to do OCR. Image formats include JPEG, PNG and GIF or multi page PDF documents. Some types suitable for

OCR are: image or PDF files obtained using flatbed scanners and photos taken with digital cameras or mobile phones. For best results: high resolution with each line of text with at least 10px height; only horizontal left to right documents are recognized; only Latin character sets are working; sharp images with even lighting and clear contrasts will work best. File size limitations, maximum size 2MB and only 10 pages per pdf. Preserves text style but not always because it is hard to recognize and keep it in OCR. Structuring elements such as bulleted and numbered lists, tables, text columns, footnotes or endnotes are likely to get lost. OCR takes longer than normal upload, up to 30 seconds for images and 1 minute for pdfs.

Current version: No information was found.

Main features: Cloud Storage with OCR capabilities; images and PDFs supported; images from mobile cameras are suited also.

The table with the features and applications discussed is presented next.

	Neat	**Evernote**	**Adobe Acrobat**	**ScanMouta in**	**Abbyy**	**CamScann er**	**Google Drive**
OCR the document	✔	✔	✔	✔	✔	✔	✔
Full Text Search	✔	✔	✖	✔	✔	✔	?
Edit documents Metadata	✔	✔	✔	✔	✔	✔	✖
Insert keywords for a document (tagging)	✔	✔	✖	✔	✔	✔	✖
Filter by keywords (tags)	✔	✔	✖	✔	✔	✔	✖
Search by fields	✔	?	?	✔	✔	✔	?
Sharing	✔	✔	✖	✖	✔	✔	✔
Reading standard Metadata	✔	✔	✔	✔	✔	✔	✔

Integration with Cloud Storage	✓	✓	✓	?	✓	✓	✓
Read various file formats	✓	✓	✓	✓	✓	✓	✓

Table 3 OCR applications and features

✓ - The feature is implemented.

✖ - The feature is not implemented.

? - No information was found.

The applications studied have most of the features from the important feature set. Adobe Acrobat is lacking some of the features it is a widely used application.

2.5 Summary

What can be seen from this study is also how presentation plays a role in how Metadata is perceived. Some applications present lists and more lists of fields that are not decipherable by all users, for example Exif Metadata which is very technical. Most users that take photos, unless they are enthusiasts or professionals will not do anything with that kind of Metadata. It is then a way to innovate in this field or at least do things in a different way which is use this kind of technical Metadata, study it and agglomerate it with other types and enable filters that are not common but that can be entertaining and interesting for the user.
Some of the features present in the applications are common between media types and this can be taken into advantage to get a group of features that are common between all applications. It is important to continue to test and use these applications to learn how they interact with the user in a deeper way and how it can be transported to the clients that will query and read the Metadata Cloud server. The State of the Art study is a continuous work because new ways to handle Metadata can be found at any time and innovative ways to present that Metadata is also appearing every time a new application is released.
With this, the major conclusions taken from this work were:

- **Metadata role in files**: the Metadata that is present in various files types is very important and should be always preserved, even if custom fields are added. All applications should respect the existing Metadata and not infringe upon the existing standard in that file. But it should not stop the addition of new Metadata fields, which can bring value;
- **How applications present the Metadata**: it is very important to know which kinds of filters are interesting for the user. Maybe the user will find it entertaining to have a Faces filter which shows all the pictures with faces, or maybe a location map integrated or location organizing the files. Or even extracting new data from Music that is not standard like BPM;
- **Cloud storage capabilities**: in a ever so more connected world it is important to have the files available any time at any place with any device;
- **Content consumption**: most organizers are not even organizing focused. It is very important for an application to prove useful to also consume the files that it organizes or not having this feature will lead the user to have to use more than one application to deal with its files.

3 Approach

This chapter discusses the approach to the development of the project. An Agile framework was used, Scrum, which consists mainly in having a focus on bringing value to the user in each of the iterations, called Sprints. A Product Backlog is used to specify User Stories, which are descriptions of the features of the system, by using the user's perspective. Risks were analyzed and potential solutions specified and are presented later. Finally, the frameworks and tools chosen to build the system are detailed. For a more detailed analysis whether of the Product Backlog with the User Stories or the finished Sprints and planned ones, please refer to the *Appendix A – Approach*. To show a more detailed look into this chapter, the sections are:

- **Methodology**: section to show what were the development methods used for this project;
- **User Stories**: section to explain how the User Stories were essential to build the features;
- **Product Backlog**: an overview of what the backlog does with all the details in the *Appendix A: Approach – Product Backlog*;
- **Sprints**: explanation of what Sprints are and how they served this project with details in the *Appendix A: Approach – Sprints*;
- **Technologies**: the choice of technologies in the project from the server to the authentication method;
- **Development tools**: the libraries, IDEs, servers and other tools that were chosen to build the system.

3.1 Methodology

"Scrum is a way for teams to work together to develop a product. Product development, using Scrum, occurs in small pieces, with each piece building upon previously created pieces. Building products one small piece at a time encourages creativity and enables teams to respond to feedback and change, to build exactly and only what is needed." [24] Scrum is the methodology chosen for this project since it is an iterative process methodology and for this complex project it suits well because of the nature of constantly learning how to do, trying new things and making it work continuously. It is then possible to constantly deliver value with the project.

There are various roles in Scrum [24] :

- Product Owners determine what needs to be built in the next 30 days or less;
- Development Teams build what is needed in 30 days (or less), and then demonstrate what they have built. Based on this demonstration, the Product Owner determines what to build next;
- Scrum Masters ensure this process happens as smoothly as possible, and continually help improve the process, the team and the product being created.

For this project it was defined:

- **Product Owner**: Supervisor Pedro Pinto defines the requirements and goals of the project and supervises the progress of the Development Team;
- **Scrum Master**: Supervisor Pedro Pinto gives support to the Development Team creating opportunities for discussion and meetings for giving support;

- **Development Team**: comprised only of Tiago Salvador with responsibilities for all the implementation of the project.

3.2 User Stories

For elaborating the User stories, it is taken into account the Features that were found to be most important for each content manager application that were studied in the State of the Art.

User stories consist in translating features and requirements to the commercial user perspective. With this it is possible to check the real value that the feature can bring to the user because it is possible to detail the What, the Who and the Why of the feature.

Usually it is the Product Owner and Scrum Master that define the User Stories, but for the sake of the Dissertation and how it is important for the author to develop such skills, it will be the Development Team that will define the User Stories.

Since the project is aimed to be able to manage various types of contents and also there are different types of concerns of the user, the User Stories were divided in several categories.

Category	Description
Photo Content	This category belongs to the photo content in the client that will communicate with the server and show photos along with other features.
Music Content	This category belongs to the music content in the client that will communicate with the server and play music along with other features.
Documents Content	This category belongs to the Documents contents in the client that will communicate with the server and show the documents that the user uploaded along with other features.
Sharing	This category belongs to sharing features in social networks and via other methods.
Cloud Storage Applications	This category belongs to the communication with Cloud Storage applications.
Authentication	This category belongs to the authentication in the server, Cloud Storage applications and Social Networks.
Privacy	This category belongs to the privacy concerns that need to be taken care of.

Table 4 User Stories Categories

3.3 Product Backlog

The Product Backlog is the group of User Stories for the project. Each row of the table plays an important role of describing the whole User Story:

- **ID**: the identification number for a User Story to be easily consulted later in each of the sprints;

- **Category**: one of the categories explained before;
- **Tasks**: the individual tasks to complete the User Story. These are technical tasks which help in the development process to complete the User Story;
- **Acceptance Criteria**: this is the group of criteria that when met means the User Story can be closed;
- **Progress**: can have the values, Closed, On-going, New and Discarded.

The Product Backlog has all the User Stories of the project and it is constantly updated throughout the project. Some User Stories can be discarded all together, other divided into more granularity and the ones that were closed will be updated here so it is possible to have a full view of what has been achieved in the project.

An example of the full Product Backlog found in the *Appendix B – Approach* is presented next.

ID	Category	Description	Tasks	Acceptance Criteria	Progress
PH-001	Photo Content	As a commercial user, I want to be able to check my photo's Metadata in order to have more knowledge of my content.	Develop a list of data; Get data from server; Present data to user.	Integrity of data and availability; Presented in a good way.	Closed
PH-002	Photo Content	As a commercial user, I want to be able to filter the list of photos by Metadata field ascending or descending so I can check information better.	Make the list of photos filterable ascending or descending; Present the data.	All photos should be organized in the list by ascending or descending order by the field; Should be fast the sorting.	Discarded

Table 5 Product Backlog Example

3.4 Sprints

After the Product Backlog and the rest of the system is minimally planned, Sprints began. These consisted in an interval of time that the Development Team develops and implements the User Stories. The main objective of the Sprints is to bring as much value as possible to the user, or Product Owner in this case, and each need to be planned in advance, taking the User Stories from the Backlog that the Scrum Master and Development Team think can be completed in the given Sprint.

For this project, in each Sprint the tasks were defined in order to fit in the duration of the Sprint and estimated by the developer in terms of the expected time it should take to implement them and consequently their User Stories.

In each closed or on-going Sprint the planned User Stories and the closed User Stories are described, along with the motives for not completing them if there are unclosed Stories for a given Sprint. For the planned Sprints a table of planned User Stories and their tasks are shown, which basically consist of a subset of Stories from the Product Backlog.

The tasks duration served the main purpose of controlling the available time compared to the expected time the Development Team has each day. For this it was used a Burndown chart, which is a great tool Scrum methodology, has in showing the real progress of the Sprint. It is compared the expected time with the real time that the Development Team managed to spend in developing the tasks and finishing them. Please refer to the *Appendix A: Approach – Sprints* for the Sprints specification that occurred during the project and their Burndown charts.

3.5 Technologies

This section describes the technologies used and why they were chosen.

3.5.1 Java Programming Language

Due to the use of the Spring framework and Android client, Java was the obvious choice as the main programming language in this project. Even if the Spring framework was not the one chosen, Java by itself has a lot of support with numerous existing libraries and it is the language the developer has the most experience with.

3.5.2 REST API

The API that will handle the requests from clients, it relies on a stateless, client-server. Since there is no need to save state, it is a great architectural style for building high performance, scalable servers. Each request is independent from the previous and the next one since all information needed is always sent in a single request.

3.5.3 Relational Database

The developer's experience in Database resumes to Relational Databases. Not only that but there is use of Metadata for each content and that Metadata has relations between them and the content like Photo's albums and folders.

3.5.4 Web Server

The whole system is built to just rely in CRUD HTTP requests, even more because of using REST API, and for that a Web Server is enough. There is no need to serve business logic to applications like the server having a connection with the Android application and exposing business logic via various protocols not exclusive to HTTP.

3.5.5 Secure Socket Layer / Transport Layer Security

HTTPS protocol that uses SSL/TLS is still one of the most secure ways to deal with privacy of network requests. Basically both ends of the connection have Public and Private keys and a use the Public key to confirm their identity. Messages can be encrypted with the Public key but only decrypted by the corresponding Private key enabling communication between two entities that don't have prior knowledge of each other over a open insecure connection because there is need of both Public and Private keys to do the encrypting and decrypting of messages. Using a Certificate Authority enables the confirmation from a third party that an entity is who it says it is. For this project a self signed certificate was used to enable HTTPS since for proof of concept there is no need to have an external CA confirm the identity of the server, the application or the content service.

3.5.6 Bcrypt

It is the password derivation library used using the Blowfish algorithm. It expands the password times the cost defined and generates a new salt each time a password is hashed. All this information is present in the final hash which means it doesn't need to have always the same salt which would need to be stored in the database. Some hashing methods hash the password using a pre-computed salt with this salt stored along the hash in the database. If an invader gets his hands on the salt also, it is easier to use rainbow tables to find the correct password. Bcrypt disables the use of rainbow tables while enabling the server to always hash the passwords with all the information available, for example with the Hash: $2a$10$vI8aWBnW3fID.ZQ4/zo1G.q1lRps.9cGLcZEiGDMVr5yUP1KUOYTa

- 2a identifies the Bcrypt algorithm version that was used;
- 10 is the cost factor; 2^{10} iterations of the key derivation function are used;
- vI8aWBnW3fID.ZQ4/zo1G.q1lRps.9cGLcZEiGDMVr5yUP1KUOYTa is the salt and the cipher text, concatenated and encoded in a modified Base-64.

3.5.7 OAuth 2.0

One of the privacy issues when dealing with mobile devices is to store passwords and emails in the device, even if encrypted. A stored password even if encrypted needs to be decrypted so it cannot use a one-way hash because that would defeat the purpose of having a stored password that could be used again. With this in mind the OAuth 2.0 protocol was chosen to enable login in the Metadata Server without the need to send username or email and password in every request. Even with HTTPS sending the password that many times could be dangerous and storing it in the device would enable someone who steals it to use it and then change it to have complete control over the account. The OAuth 2.0 token generated when the user logins for the first time is then used to authenticate the user in every request. Even if the token was to be stolen, it is something that can be revoked quickly and the duration of the token itself is of 30 days and needs to be refreshed using a Refresh Token.

3.6 Development Tools

Choosing the best development tools was easy due to well-established and excellent existing frameworks and SDKs that enable fast and simpler development cycles, future possibility of integration with existing WIT Software projects and also the choice of the type of server-side applications and client applications.

3.6.1 Spring Framework

A very popular framework to develop Web applications, it has a very robust structure using the Model View Controller pattern. It was the choice of main framework to develop the server due to already existing projects in WIT Software SA using Spring and in a similar area to this project which can enable an easier integration of the modules of the server on existing applications. The Ultimate IntelliJ IDE with student license from JetBrains was the IDE of choice because of its smart features and incredible support for Spring and Hibernate.

3.6.2 Android SDK

The Android SDK has a huge support from the community with a huge amount of resources due to the extreme popularity of the Android platform. The developer already has a good amount of experience in Android development, which contributed to the decision that the client application to be developed in this project would be for Android. The Android Studio, the official IDE for Google for Android development, was the choice of IDE. There were numerous Open Source libraries used that greatly helped to improve the application including:

- **Universal Image Loader** [25] for handling image loading from network;
- **EventBus** [26] to enable simple communication using POJOs between components in Android;
- **Android Design Library** [27] with Material Design complaint views and practices;
- **OkHttp** [28] with the best and fastest network client enabling gzip by default.

3.6.3 Dropbox SDK

Dropbox is currently one of the best Cloud Storage providers and their SDK is simple to use. It also enables the use of OAuth 2.0 tokens to be passed to third parties to allow for processing of data. If the user has Dropbox application already installed it is even simpler to connect Dropbox to third party apps with no need to login.

3.6.4 Tomcat Web Server

A Web server can only handle HTTP requests. Due to the nature of simple requests from the client applications to the server to send files, Metadata extracted or queries to the server, the server doesn't need to be an Application server. There will be no need to expose business logic to the clients, only receive and send data, so a Web server is sufficient. Also Tomcat works great with Spring, and if there is need for any Application server functionalities, Spring can handle it. The developer had experience in Tomcat so it facilitated the choice.

3.6.5 Google Cloud Messaging

When dealing with network requests in Android if a connection is maintained using a Service for example, it can drain the battery fast. Having the application request something to be processed in the server asynchronously and then receive Push notifications from Google's servers with progress or a small message to sync with the server, removes the necessity of having a constant network connection and also the need for the device to be turned on to not lose any notifications since push notifications are only sent when device is on.

4 Architecture

This section presents the architecture of the system as well the more detailed architecture of each of the modules of the system. First the system outline, explaining what are the main scenarios of usage and how each of the modules will communicate and do, is presented. Then the architecture of each of the modules is presented and the details about specific components and choices in terms of implementation. The sections present are:

- **System Outline**: the overview of the system architecture by looking at each model and their highest level action;
- **Metadata Server Architecture**: the architecture of the Metadata Server. It has also details about authentication and how it handles data from the Content Service and Android application;
- **Content Service Architecture**: this section explains how the cloud service works and syncs data from Dropbox with the Metadata Server;
- **Android Application Architecture**: this is the largest section and it has details from the Android platform to implementation details regarding high level components.

4.1 System Outline

The system comprises the Metadata Cloud server that communicates with a Database, web application running on the Cloud called Content Service to process the Metadata from files and create smart features like Smart Albums and the client application to show the Metadata results and enable features.

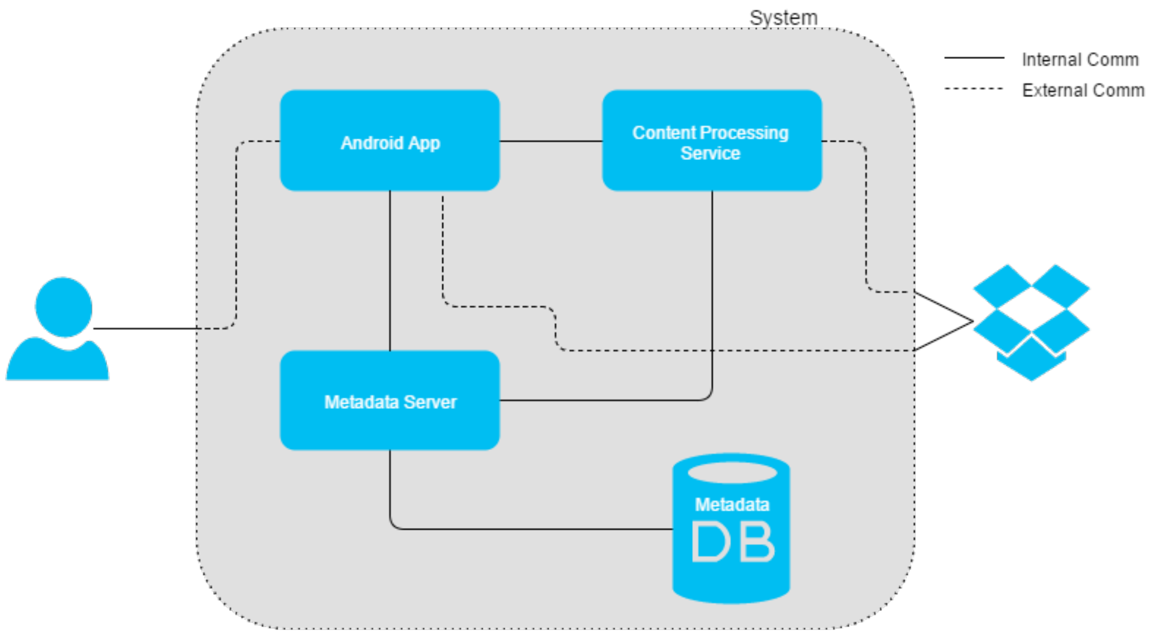


Figure 5 System Architecture

These are the components of the system while communicating with Dropbox and the User. In specific their actions are:

- **Getting data from Metadata Server**: client application will make a request to the Server to get some specific kind of content. The Cloud server will receive the request by means of a REST API. With the request the Server will query the Database that contains all the locations of all the content that the user owns in the 3rd party Cloud

Storage applications. The Cloud server will then, via elaborate queries get the Metadata of the content and return it to the client application;

- **Syncing information in files with the server**: the user can send a Dropbox token to the Content Service and enable it to download the photos, process, delete them and then send the information to be saved in the Metadata server. Each client that sends a request has a folder generated with a Base64 random generated name to permit files with the same name to be processed and not confused between clients. The data in local photos can be directly extracted from them in the Android application and sent to the Metadata server without passing through the Content Service;
- **Communicating directly with Dropbox**: the Android application can get the files directly from Dropbox as to start the whole system and the smart features since using the Delta endpoint of the Dropbox Core API should be in the client so that the whole files are stored in a local SQLite database so that requests to Dropbox are kept minimum.

Next is presented a more specific description of how each of the components work.

4.2 Metadata Server

The server is built using Spring Framework and uses the MVC model. Basically the Android application or the Content Service send HTTP POST requests to the server using JSON as the data format and the server MVC dispatcher dispatches the requests to each Controller that is mapped to a given path.

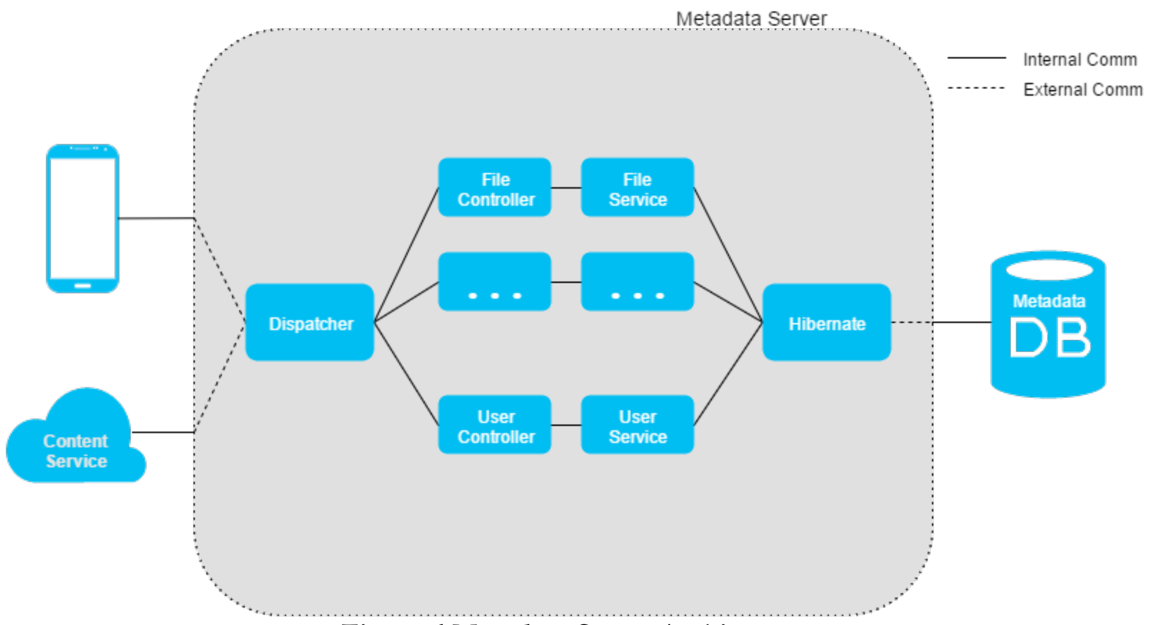


Figure 6 Metadata Server Architecture

4.2.1 Handling requests

In Spring it is possible to use annotations to configure all aspects of the server. It is also possible to use XML configuration or Java configuration. For this project it was chosen to have annotations for Controllers, Services and Async tasks, but for configurations it was chosen XML because of its simplicity and clean look.

The JSON data is modelled using Java beans in all of the components to facilitate the requests and in the server Jackson is used to un-map or map the POJOs to JSON data. It is possible to see by the architecture figure that the dispatcher can dispatch the requests to each File type controller. In Spring it is advised to not have the Controllers communicate directly with the Model, but instead use a Service.

Each of the Controllers methods are annotated with their path, the HTTP method accepted and the accepted parameters. Basically they all receive the requests, call the Service method and return the result.

Services are declared as beans that can be injected in the Controllers and their methods invoked. In the services methods they are annotated with Transactional annotations and set attribute to rollback the transaction if there is any exception. Also each Transactional method can be set to be just a read transaction or a write transaction which changes how Hibernate handles locking the database.

Using Hibernate it is possible to mostly use Java objects in the server which greatly simplifies the code and the length of the code. Along with the annotations and XML configuration, the server's code is fairly well modeled.

4.2.2 Authentication

The chosen authentication method is OAuth 2.0. As previously stated, in mobile devices, having the password stored in plain text or even encrypted by not a one-way hash function like Bcrypt, it can be very insecure since mobile devices are frequently lost.

With this in mind, the OAuth 2.0 protocol was used to secure the access to the data in the server along with HTTPS using a self signed certificate.

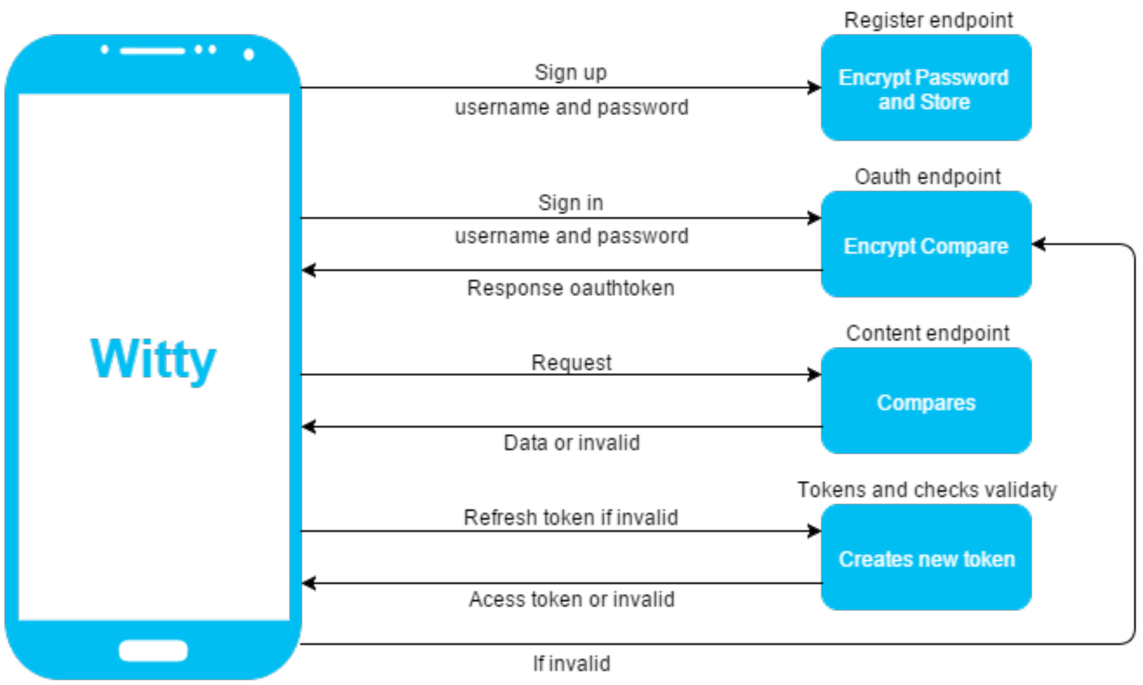


Figure 7 Authentication Flow

When the user first signs up the password is sent plain text via HTTPS to the server. The password is not encrypted client side because if someone was to gain access to the contents of the connection the encrypted password could still be used as the password to gain access. HTTPS is secure enough to have the password sent plain text. The important part is the encryption server side. If a malicious third party was to gain access to the database, if all passwords were plain text then he would simply be able to access the data. By encrypting the password server side using Bcrypt with a strength of 11 and by the very nature of Bcrypt generating a new salt each time, it is a very secure, at least compared with most other hashing methods, way to store the hashed password.

After the sign up, when the user signs in, the password entered is again encrypted and compared with the hash stored in the server. The request sent to the token endpoint has the grant type, in this case password, the username and password encrypted Base64 in the header and the URL encoded has the parameters for client id which represent the known trusted client and the client secret. If they are the same, then the user gains access to the OAuth 2.0 token endpoint and a new access token along with a refresh token are generated and sent back as a response.

Then with each request the access token is sent to the server in the Authorization header of the request encrypted Base64 and if the token is valid then the user gains access. Each 30 days the token expires and the refresh token must be used to get a new access token. If not possible, the user will have to authenticate again.

4.3 Content Service

The web service to process content is also built in Spring Framework. The requests received can only be from the Android device. The dispatcher then forwards to the sync Controller and processing starts.

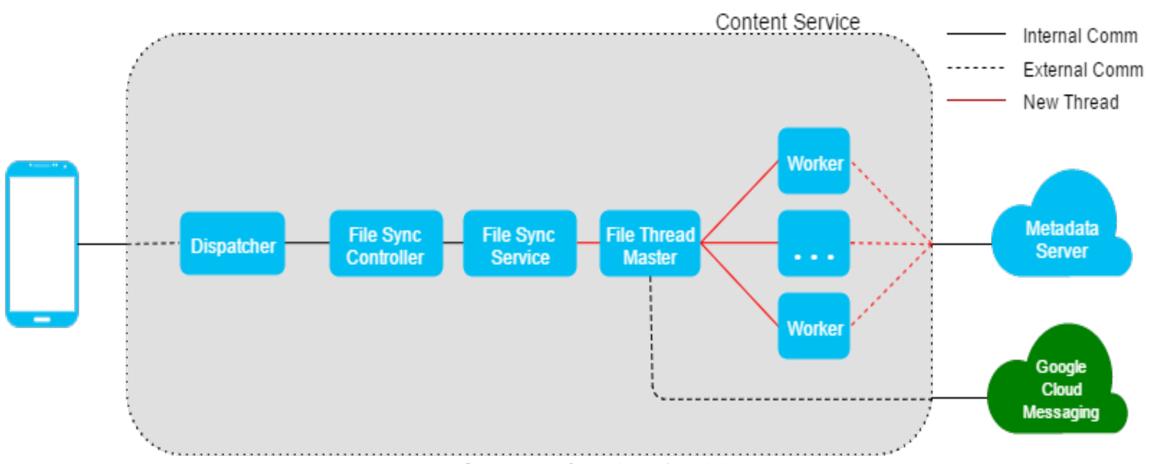


Figure 8 Content Service Architecture

4.3.1 Asynchronous Processing

The user application shouldn't have a persistent connection with any other entity as such would drain the battery of the device. With this in mind the Content Service was built in a way that the request is received, confirmed to be valid, and a response returned if there is any processing started. The Content Service requests to the Metadata Server all the files names and locations and then compares with the request from the application, which also forwarded to the Content Service all the names and locations of all files currently in Dropbox.

The Content Service eliminates from the list all repeated files and proceeds to create a new Master Thread for processing and the Main Thread just responds to the application stating processing started. The application can then confirm to the user that a synchronization process has started.

In the Master Thread the list to be processed is divided at most to 5 other Worker Threads. Each Thread is created and sent their part of processing just one time. The Master Thread continues to loop waiting for the Worker Threads to finish and creates a Singleton that handles the sending of Push Notifications requests to the Google Cloud Messaging Server and is used by all Worker Threads.

Each of the Worker Threads download the files between them asynchronously and process the Metadata from the files. The files are immediately deleted after the Metadata is extracted for privacy concerns. The data is then sent to the Metadata Server to be stored.

From time to time, the Singleton is used for sending the Push Notifications request. The Push Notifications requests are not made for every single file processed because there regarding GCM most messages would be skipped and potentially too many notifications would be received in the application device. Also with the heavy load, most would reach late.

4.3.2 Files protection and authentication

When sending the request from the client application, the OAuth 2.0 token for the Metadata Server authentication and also the token for Dropbox authentication are sent to the Content Service. Each of the tokens are used to say to the other entities that it has permission in behalf of the user to process information. HTTPS with self signed certificate is used to protect the connection. No passwords are sent.

Regarding files protection, all files downloaded from Dropbox are immediately deleted. A folder is generated in the server's local file system with a Base64 randomly generated name so that uniqueness can be preserved as much as possible so no conflicts between users' content's occur.

4.3.3 Performance

Several optimizations were done throughout development. At first the Content Service had as many threads spawned as files there were to be processed. Quickly it proved to be inefficient and also Dropbox did not like so many requests per second.

So a major change went underway with creating the Master – Workers thread model which quickly proved to be much more efficient, faster, and no errors whatsoever occurred with all the numerous testing trails, with more than 30 runs with zero errors relative to Threads handling and processing.

4.4 Android Application

The focus of the development process was done on the Android Application. A concise, robust architecture was what was tried to achieve. There are various Open Source libraries that proved very useful and make integral part of the architecture thanks to that.

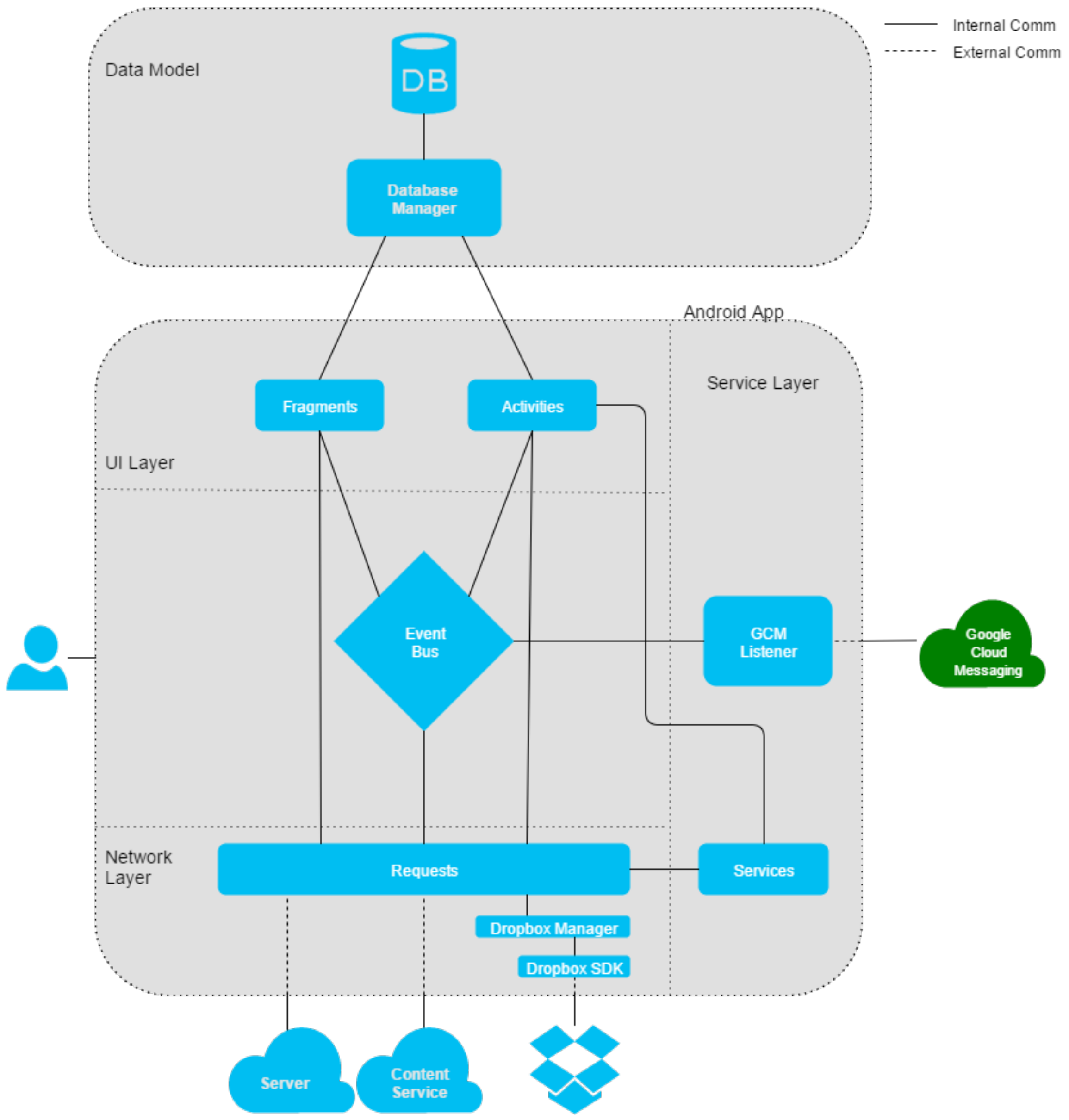


Figure 9 Android Application Architecture

There are several components in the Architecture so there is need to separate them and explain each one.

4.4.1 UI Layer

The Android UI component is comprised of Activities and Fragments. Usually Applications make heavy use of Activities which are basically large UI components that represent a single screen with its own lifecycle methods:

- **onCreate**: when the Activity is created with its view inflated;
- **onResume**: when the Activity is ready to be shown to the user whether it was created now or if it resumed from processing another Activity;
- **onPause**: when the Activity is paused while another Activity is being started;
- **onDestroy**: the Activity is then destroyed for Garbage Collection and cannot be used.

These lifecycle methods are heavy and showing an activity or starting another activity can have serious predicaments regarding UI and UX.

Fragments were created to be individual reusable UI components and multiple can live in the same Activity. Having multiple Fragments inside an activity makes the code more modular, thanks to the reusable nature of Fragments and also makes the transition between screens smoother, with overall better UI/UX.

One of the disadvantages of having a single Activity for a group of Fragments is that it can lead to bloated Activities with lots of responsibilities and basically what is called a Fat Activity. This does not happen in this project because there are Activities for the Introduction, Authentication, Preparation, Synchronization and Photos parts and apart from lifecycle methods and some Toolbar responsibilities, the only responsibility of the Activity is to manage the Fragments and transition between them.

The Services starting is of the responsibility of the Activity because of its nature of providing a Context for the application and being the uppermost component in the application apart from the Application class. So the Services can be started from Activities but they do not manage it, and only a reference is sent so that the Services know their context.

The Fragments basically have responsibility over a single screen unless there are more granular Fragments that just handle a UI component like the Navigation Drawer. The Fragments starts requests, show the information, handle their own lifecycles similar to the Activities and communicate with their parent Activity by Events. Having an Activity reference used in a Fragment can lead to Null Pointer Exceptions if the Activity is getting destroyed at the moment and the Fragment is still running code.

Regarding special views like the Gallery or Lists, Adapters are used. Adapters is a well known Android component that serves to adapt different types of data together. The RecyclerView.Adapter is a new component that makes use, by default, of the recycling of views, and presents the data given any type of data list. Recycling of views consists in re-using a component multiple times in a list or grid for no performance penalty of creating a new view for each data object. For large lists this would completely halt the application. One of the problems with recycling views is that the data showed in that view is still present there when re-used so there is need to manage what should be shown at any given time for each data object.

4.4.2 Event Bus

This component is based on the EventBus Open Source library. It uses reflection or in the newest version, annotations, to call onEvent methods in any listener class. An Activity or Fragment that wants to listen to events should register specific event types by having onEvent methods or annotate any given method with the Subscriber annotation. In onResume and in onPause lifecycle methods of either the Activities and Fragments, the methods should be registered or unregistered.

Any other class, Activity or Fragment or any other component can then post events and only classes registered to events will receive them. If no components are listening right now, the event is simply discarded.

Comparing with the default Android communication system of using Intents the objects passed around can be simple POJOs, there is no IPC involved and no need to instantiate

IntentFilter or BroadcastReceiver objects. Also sending thousands of events has no performance penalty.

The best advantage is that there is no need to hold references of objects in another class to enable communication between them. One of the biggest issues in Android is having an Activity reference hold in a Asynchronous task or Thread and if the Activity needs to be destroyed or finish for any reason, there will be a leak because of the reference in the Thread. The Thread doesn't even know this and when it tries to call a method of the Activity to send data it will cause an NPE.

EventBus simplifies all the communication process between all components and it is a technology that I cannot recommend enough. The only disadvantage I found was that if no Annotations are used, it can be a bit tricky to manage the code and it can become confusing to know which events correspond and where it can be found the subscribers. A way to overcome this is to have a single class for a group of events, for example UI Events and have several static inner classes with the specific Event type.

4.4.3 Network Layer

The Volley Open Source library is used to handle most of the network requests with Gson Open Source library to handle the mapping of Java objects to JSON and vice-versa. Volley has some nice features like tagging requests to be later cancelled and good asynchronous processing. The Universal Image Loader Open Source library is used to get images from network and cache them.

There are several configuration changes made to Volley to retry connections with a certain timeout and Login request is different from the other requests thanks to being URL form encoded, and the others are body encoded. Requests are cancelled when a Fragment that started them is paused because it doesn't make sense to waste resources in requests that the result will not show up in the screen.

Regarding loading of images, the library gets the images and caches them but the whole Gallery view is designed around the recycling of views. Also a smaller, with less quality, version of the images is loaded first for fast showing and then a more detailed version is downloaded. The handling of threading is done by the library. At first a custom implementation was created but it soon proved to not have the robustness of the UIL library although good results were achieved. Another library was also tried, Picasso, but it didn't have the caching capabilities required for the application.

The communication with Dropbox is done via a Singleton Manager that handles the connection for all Activities and Fragments and calls the Dropbox SDK methods using a single instance. In some applications an instance of the Dropbox SDK object is created, but this is something that was not considered to be best practice.

4.4.4 Data Model

This local database created using SQLite is used to store the information about Dropbox so that requests to Dropbox are kept at a minimum. The data contained in the tables are:

- fileType: the type of the file in Dropbox;
- filename: the name of the file;
- path: the path in Dropbox to the file;
- synced: if the file has already been synced in the server;

- id: the primary key of the record.

Also there is another table that contains a hash which is used by the Delta process in Dropbox that is explained later.

The access is always by the Singleton class DatabaseManager and it has the responsibilities of creating the Database and querying it.

4.4.5 Service Layer

In Android an Application if terminated all of its requests should also be terminated. This is normal behavior, but the problem is that in mobile devices with their restrictive memory all processes are subject to be terminated by the OS at any given time. So for long running or critical operations a Service should be used. For the application services are used to:

- Download files from Dropbox;
- Upload files to Dropbox;
- Get all info about the files from Dropbox;
- Listening to Push Notifications.

The Services are usually started by Activities using their reference and they are basically long living components that hold requests being made to the Metadata Server, Content Service or Dropbox. Communication from Services can also be made using the EventBus.

4.4.6 Local Preferences

There are several options in the Settings menu as well as small data that can be saved using the Preferences utility. Preferences in Android are basically settings data that is unique to the Application with also unique fields that can store from Integers to complex objects.

A Singleton class is used to manage all Preferences. In Preferences the authentication tokens are used as well as flow relative data like if the user already watched the Introduction or if already started syncing the data.

4.4.7 Dropbox Core API Delta

The Dropbox SDK is used to communicate directly with the Dropbox API. The Core API is used and basically grants access to getting information about files and downloading or uploading them. There are no Sync capabilities as that is restricted to the Sync API, recently deprecated and unified with the Core API. There are no Sync features because using the Core API is enough to get all information about the files in Dropbox. This is possible thanks to the Delta endpoint.

The Delta endpoint enables the request to get all list of files present in Dropbox and their path. The request does not return sequential files, just gets a subset of files each time along with a hash. This hash can then be sent in the next request so that the Dropbox server knows which files already returned and to return new ones.

The Delta asynchronous task in the application does several Delta requests to Dropbox while storing the files data in the local database along with the hash. Each time the user does a refresh or the application does it by itself, the hash is used and new files are returned.

4.4.8 How data behaves in Dropbox and the server's database

In Dropbox no data is re-indexed when moved. If a file is moved, it is deleted and then a new equal file is created in the new location. Having to re-index can be extremely expensive so the same happens in the Metadata Server. When a file is deleted or moved, the data is copied to a new record and deleted from the original one. Since files are uniquely identified with the account id, path and file name, it would be too much complicated and bloated to try to re-index data.

4.4.9 Logout from Application

When the user logs out of the application there is no need to keep the authorization tokens. The token from the Metadata Server is deleted and when signing in again a new token is generated, as well as with Dropbox token. If the user has the Dropbox application, deleting the token in the Application will not sign out the user from the Dropbox app.

4.5 Data Model

The data was modeled using SQL and a database model defining the tables and using an entity relationship diagram to define the relationships. The focus was on photos content, the modeled tables were the Accounts, Photos, Exifs, which corresponds to Exif Metadata, Folders, which correspond to folders in Cloud Storage, Albums, Tags, Statistics and also tables for OAuth authentication.

The development of an application that uses Metadata must follow some set of standards so that other applications make use of the Metadata that the application can create. Metadata is a simple specification overall since it normally just exposes various fields with different data types with no relation between them. This can be easily transferred to tables in a non-relational database since no relations between the Metadata are seen. But since there is a need for more complex structures where the Metadata can reside and be presented, the choice for a relational database proved more logic.

The Metadata that is extracted from the contents is still defined in the same standard fields names and types in the MySQL database, but then there is need for more complex structures like Albums and Folders for photos. Folders for example are also present in all the other contents since there is a need to also access the information via normal Folders views whether the information is in the local device or the Cloud Storage. Since also the user needs to authenticate in the server, and since the information always belongs to someone, there is need for relationship between all the Metadata, whether standard or not, and the user's account.

Albums and Folders are not the only tables that need to exist for accessing the Database in interesting ways. There is also need to separate all the Metadata types in different tables similar to Object Oriented Design so that it is clearer and simpler for navigating through the data.

4.5.1 MySQL Database

The MySQL database was modeled using MySQL Workbench, an open source tool for modeling databases and querying them. Each of the tables has a single primary key, which constitute an integer value that is completely unique to each record in each table. The modeled tables were defined with various columns. The Accounts table has the data for each user registered in the system and this table is also queried when the user tries to login along with the tables relative with OAuth 2.0 when all other requests are sent to server. These tables are advised to have those names and fields by Spring Framework to enable OAuth 2.0 authentication.

All of the other tables have a reference to the table Accounts since a user owns all the contents recorded, except for the Exifs table since it doesn't need to be connected to an account but only to a photo.

The Exifs table exists only for simplifying the Photos table. Since Exif Metadata has numerous fields the Photos table would be enormous and confusing since other types of Metadata might exist in a photo and also the path, name and account foreign key would be in the middle of all the Exif Metadata columns.

An Object Relational Mapping (ORM) named Hibernate is used along the Spring framework to model the database in Java POJOs using annotations. Using an ORM it is possible to map incompatible objects like a database to Java objects, use those Java objects in development and simplify the accessing of data.

In a last note, there are no Identifying Relationships in the database since these complicate a lot the development code using an ORM like Hibernate and there is no real necessity since the same behavior can be replicated using Foreign keys that do not allow NULL value in the relationship tables.

4.5.2 Entity Relationship Diagram

Next is presented the ER diagram that can show the tables and columns as well the relationships between each table in the database.

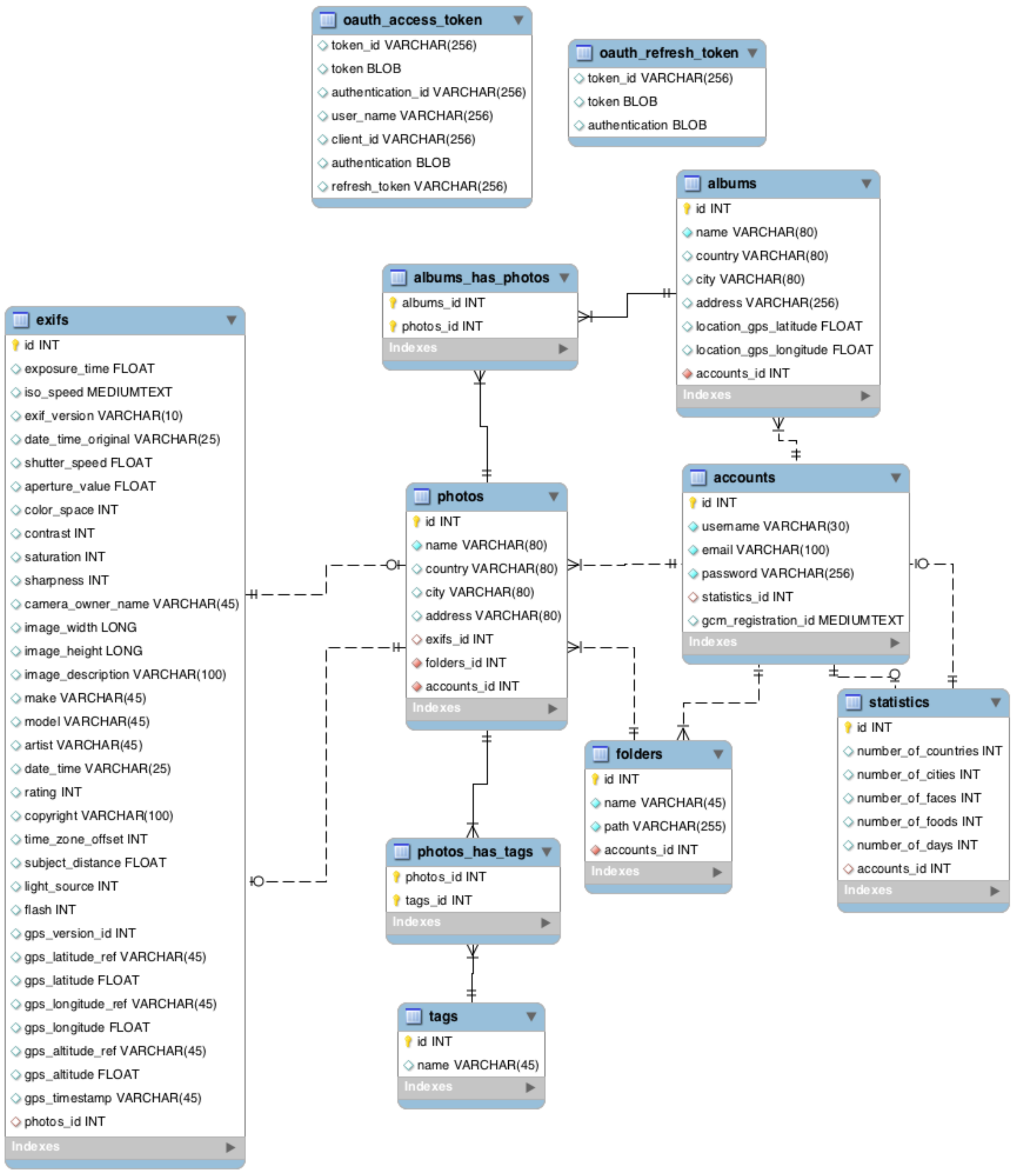


Figure 10 ER Diagram Metadata Database

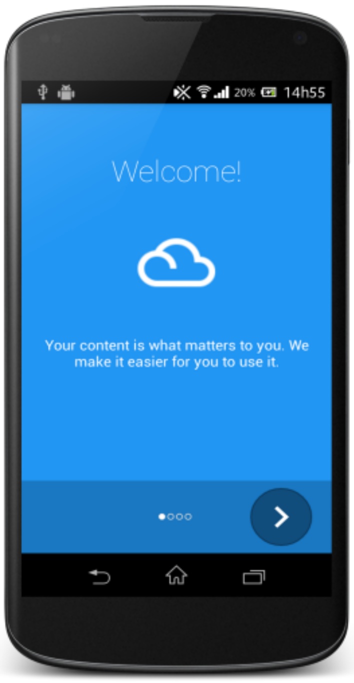
5 Results

The application was the center stage of the project and this chapter is to present the results achieve in the whole project from the perspective of the Android application. The features will be described and a screenshot or two will be presented for each of them. The next list is used as a reference to the most relevant features, the description for each are presented in their section:

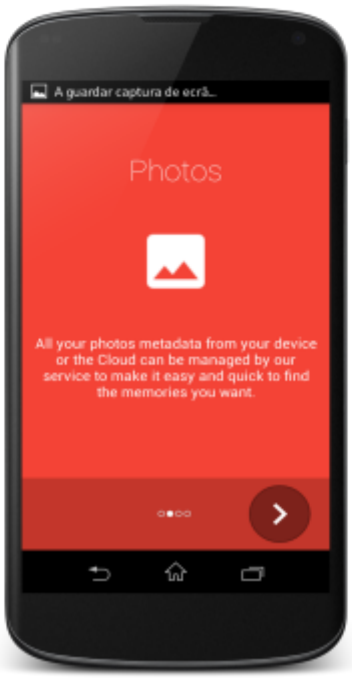
- **Introduction**
- **Sign in and Sign up**
- **Dropbox connection and sync**
- **Syncing with Metadata Server**
- **Gallery with order photos by path, relative location and date**
- **Individual photo view and pager with details and location with change**
- **Albums automatic creation**
- **Folders view and local photos**
- **Places**
- **Clustering view of photos**
- **Itinerary view**

5.1 Introduction

This feature serves to give an introduction to the user about what the application does. It consists of a pager with description about the application, the photos, music and documents. The user can go back and forth as he likes.



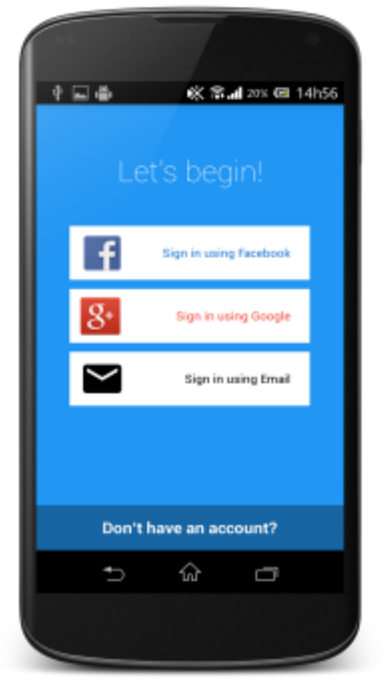
Your content is what matters to you. We make it easier for you to use it.

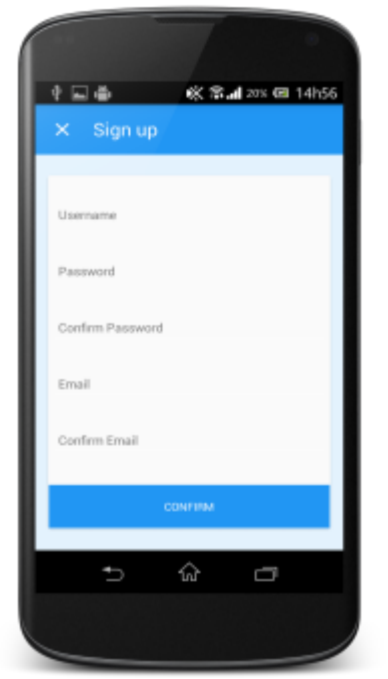


All your photos metadata from your device or the Cloud can be managed by our service to make it easy and quick to find the memories you want.

5.2 Sign in and Sign up

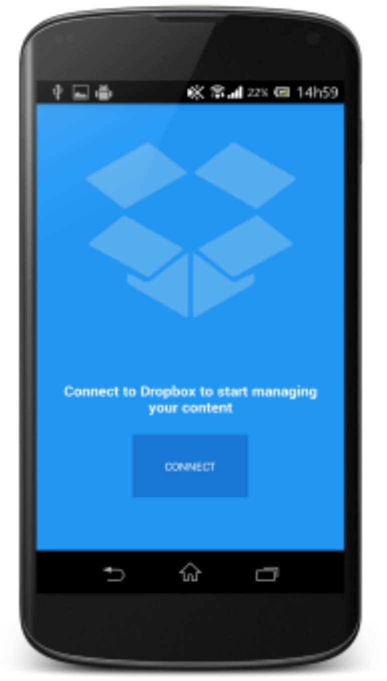
These are the authentication screens. The user can sign in using a Social Network, not implemented as the time of this writing, and email. It consists of normal screens with warnings if data is not valid.

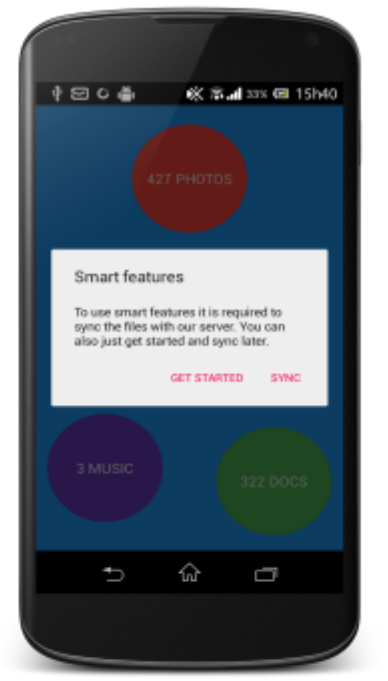




5.3 Dropbox connection and Sync

This feature consists in asking the user to connect to Dropbox, confirm the login and then wait for the synchronization task to get all files from Dropbox. It is done using the Delta endpoint. The circles showing the files have a small bouncing animation when information is added to them. Then when finished the application asks the user to sync with the Metadata Server or get started.

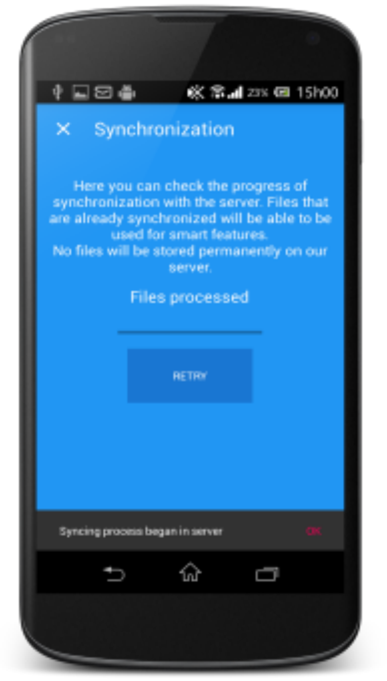

Connect to Dropbox to start managing your content


Smart features

To use smart features it is required to sync the files with our server. You can also just get started and sync later.

5.4 Syncing with Metadata Server

This feature is well explained throughout this document. It consists in having the files Metadata processed in the Content Service and stored in the Metadata Server. The user can keep up with the progress and periodically receives Push Notifications saying the process. The user can then press the Notification to go to the Synchronization screen.



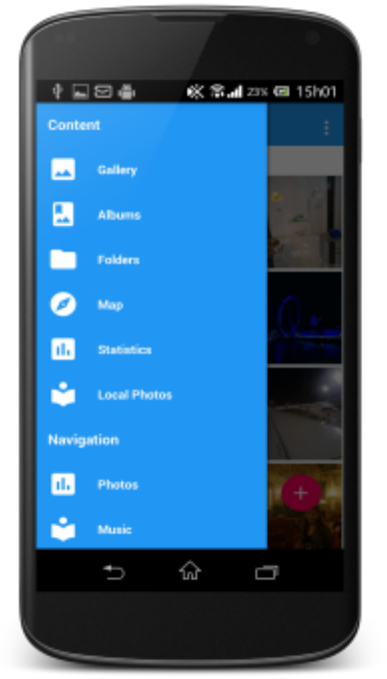
Here you can check the progress of synchronization with the server. Files that are already synchronized will be able to be used for smart features.
No files will be stored permanently on our server.

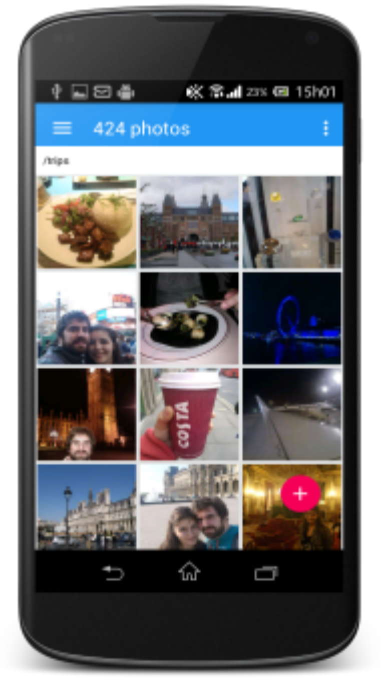
Files processed

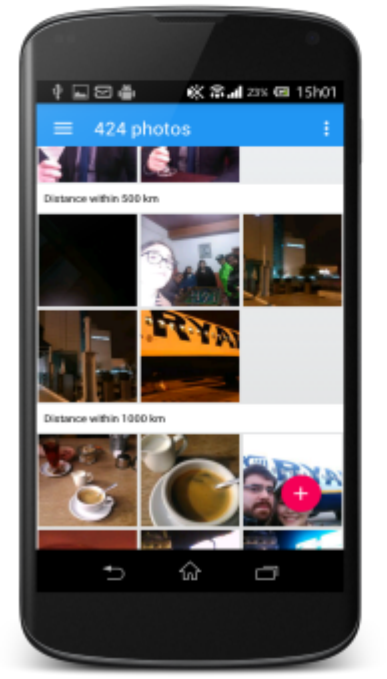
Syncing process began in server

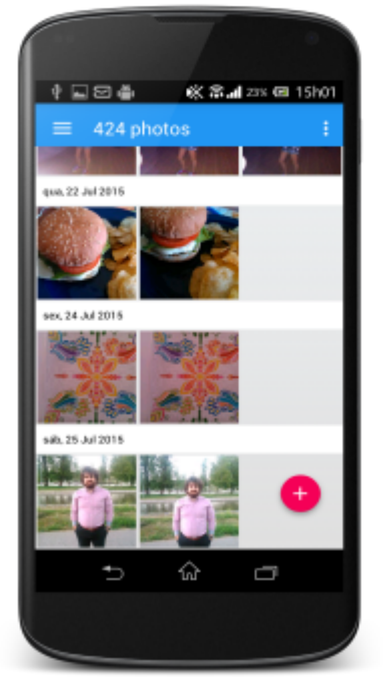
5.5 Gallery

The gallery shows all the pictures with various options regarding ordering. The order by location uses GPS to get current location and then calculates the distance to the location of the photos. The date organizes by day, month or year.



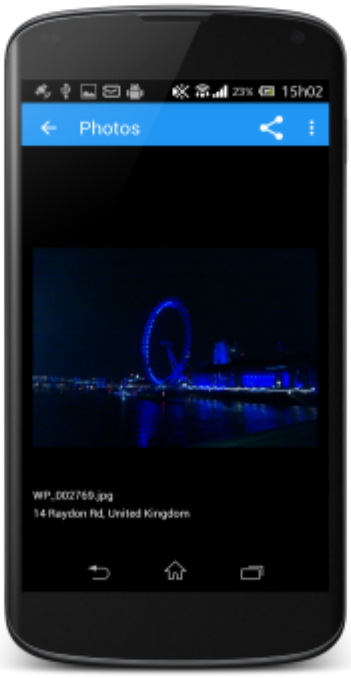




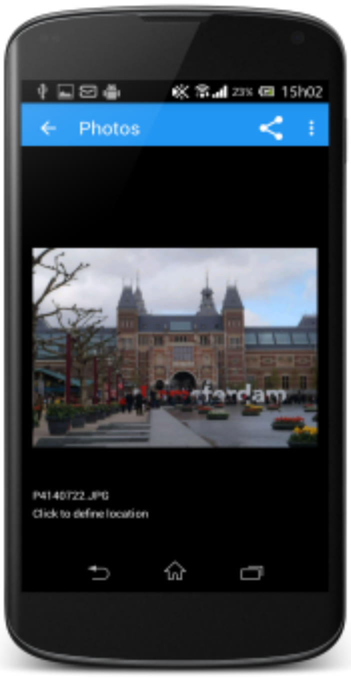


5.6 Individual Photo View with pager

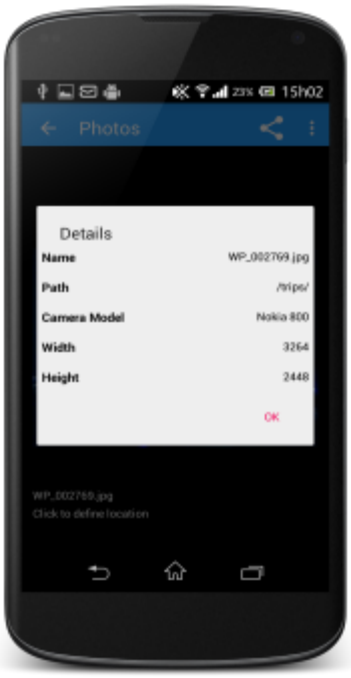
This feature consists in showing the photos and being able to swipe left and right between the available ones. It is possible to see details, see location, and zoom in and out and also to set new location.



WP_002769.jpg
14 Raydon Rd, United Kingdom



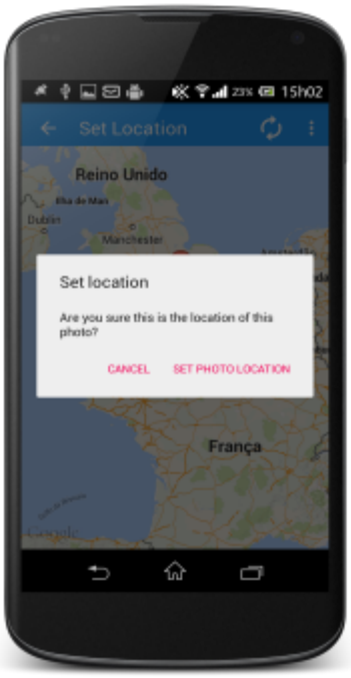
P4140722.JPG
Click to define location



Details

Name	WP_002769.jpg
Path	/tripe/
Camera Model	Nokia 800
Width	3264
Height	2448

WP_002769.jpg
Click to define location

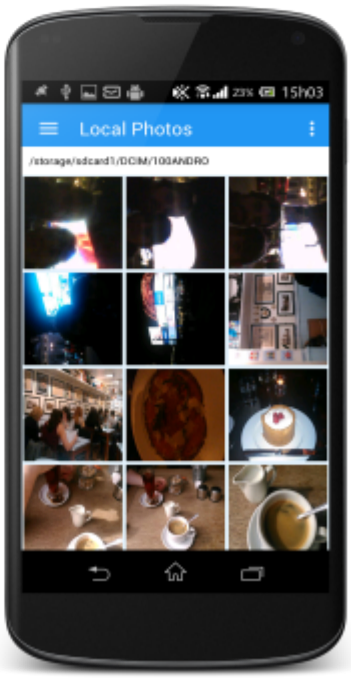


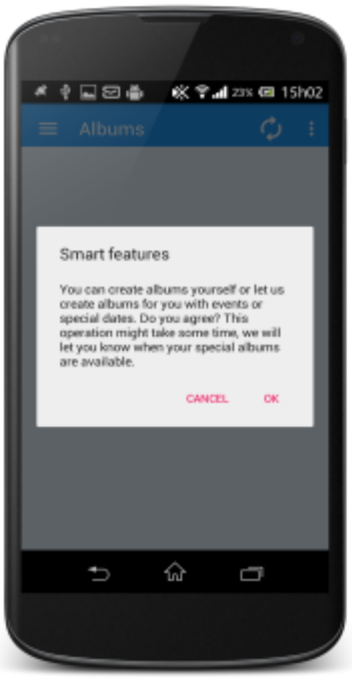
Set location

Are you sure this is the location of this photo?

5.7 Automatic Albums creation, Folders and Local Photos

The albums can be generated in the server. They are created based on special dates like New Year 's Eve, Christmas, Spring, Summer, etc. The Folders show the folders that have photos in Dropbox and the Local Photos show the photos in the device and it is possible to upload all of them to Dropbox and sync in the server.

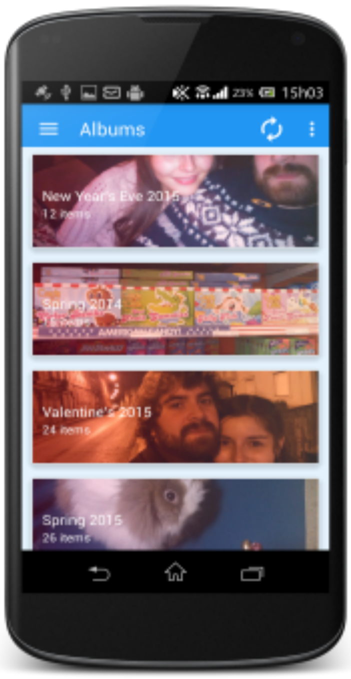


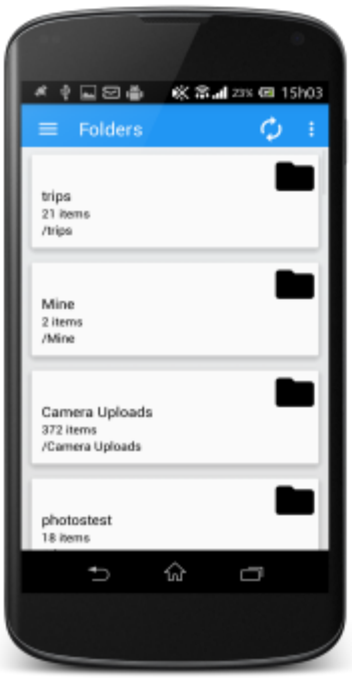


Smart features

You can create albums yourself or let us create albums for you with events or special dates. Do you agree? This operation might take some time, we will let you know when your special albums are available.

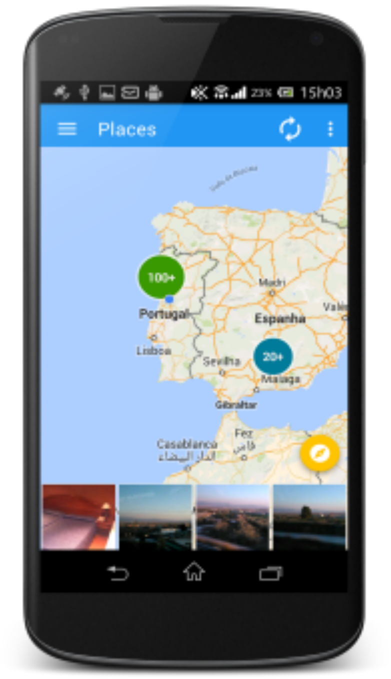
CANCEL OK

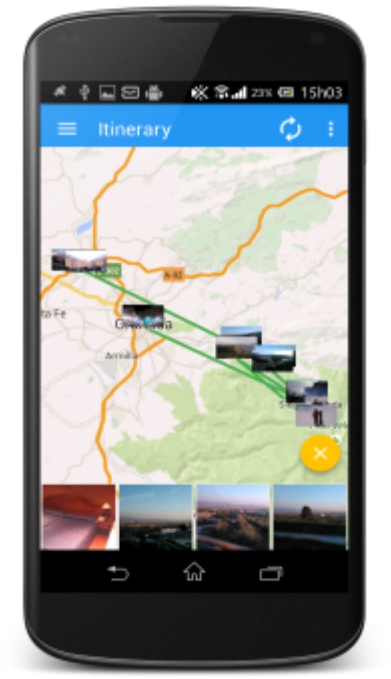




5.8 Cluster of photos and itinerary

The cluster of photos is done using Google Maps Util library and the photos are shown based in the zoom level. The Clusters can be pressed and a list of photos show up with all the photos from that cluster. The user can then click on a photo and go to the pager which has all the features present in the Gallery. The itinerary feature works by clicking in a cluster, pressing the button to start the itinerary, and the cluster is zoomed, the photos are loaded in each position and a connection between them based on the date is shown basically meaning the user can see where did they went through a trip for example.





The features done have many more details and options available than what is shown here. Due to time constraints all the options could not be shown. It is advised to go to the Evaluation chapter, next, to have a full view of the features and see the results of users actually interacting with the application.

6 Evaluation

This section details the tests done to provide reference for what was achieved. Testing is fundamental to assess the success of a product in terms of productivity, value and results. There are two testing methodologies being:

- **Black box testing**: the internal system design like structure of the program is considered and only Functional and Non-Functional tests are considered;
- **White box testing**: tests the inner workings like code paths and branches, it requires expert knowledge of the inner system and how it is designed.

With these two methodologies in mind there are then various specific types of software testing and generally they are divided in granularity:

- **Unit testing**: tests for individual classes, functions or modules in a program. It serves to see if the software code does what it is intended to do well without bugs and unexpected results and should be done by the programmer since it requires knowledge of the inner workings;

- **Functional testing**: checks if the function, group of functions or whole feature does what it should be doing, not requiring knowledge of the code but instead of the client requirements;
- **Integration testing**: tests to check if modules and parts of the whole system communicate well with each other, for example a login system with a database;
- **Acceptance testing**: taking into account all clients requirements, tests with the client are made to see if he accepts how the system behaves;
- **Usability testing**: tests to see if the system is user friendly, like if the users can understand what they should do in each feature without outside help.

There are more types of testing but usually these are the most used. For this project Unit testing was planned but due to time constraints and the focus in features instead of code perfection it was discarded. Functional testing was done and presented next as well as Usability testing. The Functional testing is used to show how the app is behaving correctly and does what it is supposed to do and the Usability testing serves to check the work done in terms of Android application development and if in terms of perceived performance, flows and overall features if potential users could be happy with the results.

Finally, the tests have several objectives in terms of successfully checking if the system works as intended[29] :

- meets the requirements that guided its design and development;
- responds correctly to all kinds of inputs;
- performs its functions within an acceptable time;
- is sufficiently usable;
- can be installed and run in its intended environments;
- achieves the general result its stakeholders desire.

Next is presented the Functional testing.

6.1 Functional Tests

Functional testing basically follows a simple model of having a group of functions inside each feature and then the feature as a whole and checks if with a given correct input or incorrect input if the application behaves correctly.

This distinction between correct and incorrect input can be designed as positive Functional testing and negative Functional testing, essentially meaning that positive testing means providing valid input to the application and negative testing providing incorrect and potentially breaking input to the application. Next is presented the positive Functional tests that were created. They try to test all the available features of the application at the time of writing of the tests but some features of the end product might be missing due to time constraints.

6.1.1 Positive Functional Tests

The tests were done multiple times during development and also just for testing. There might be code branches or conditions that can lead to bugs, but these can only be avoided with Unit Testing or other low-level testing.

	Input	**Output**	**Result**
Introduction	Pressing back and forth in the button to skip page.	There isn't any unexpected results, the pages are shown without errors and at any time the user can continue with the app.	**Passed**
Sign up	Going to sign up screen. Inputting valid values and pressing in button to confirm.	The account is created and a Snackbar is created with a message showing the result.	**Passed**
Sign in	Going to sign in screen having an account and inputting valid information.	Valid sign in, no problems whatsoever.	**Passed**
Dropbox connection and sync	Clicking in Connect button. Going to external page or screen to connect to Dropbox. Waiting for getting information to finish.	Successful login to Dropbox and successful counting of files. Shows the number of files in colored circles.	**Passed**
Syncing with our server	Accepting to sync. Waiting for sync and seeing the progress.	There is nothing wrong, progress is shown and files can be used in smart features.	**Passed**
Push Notifications for syncing	Google Play services are installed so push notifications can be received.	Push notifications are received either for progress or for saying that progress is finished.	**Passed**
Gallery	Looking at gallery and scrolling through content.	Gallery loads correctly. Photos numbers are correct and options that are available work.	**Passed**
Individual photo view and pager	Choosing a photo, clicking. Swiping between photos. Zooming in and out photos.	No problems, no crashes. Zoom also works ok.	**Passed**
Individual photo details and location	Viewing Metadata details and location being available or not.	Details are shown in a Dialog without crashes and weird data. Location is either available or not, or doesn't exist and can be set.	**Passed**
Location change	Scrolling the map, and choosing a location. Then confirming.	Usually it works, but sometimes due to server issues it can not work and give a error. Markers set while the error occurred are	**Failed**

		not automatically cleaned.	
Sharing individual photo	Sharing via any means, email, social network, etc. Resuming application.	Share works well. There isn't any unexpected results, just an image being shared.	**Passed**
Downloading individual photo	Pressing download and getting the photo.	Notification is shown. Exiting the application does not stop the download. In general it works well.	**Passed**
Selecting photos to delete	Selecting multiple or a single photo. Deleting it.	Photos are deleted from Dropbox and server and when deleted they disappear from the gallery immediately.	**Passed**
Selecting photos to move to album	Selecting multiple or a single photo. Adding to album.	In general works well.	**Passed**
Order photos by path	Choosing option to order by path.	The photos are ordered well and shown in the gallery with titles showing the path.	**Passed**
Order photos by relative location	Choosing option to order by location. GPS is enabled.	The photos are ordered well and shown in the gallery with titles showing relative distance to current location.	**Passed**
Order photos by date	Choosing option to order by date.	The photos are ordered well and shown in the gallery with titles showing the day, month or year.	**Passed**
Albums automatic creation	Confirming the album creation in server.	Notification shows up when albums are created or by refreshing the album list they show up. The albums make sense given the dates.	**Passed**
Albums view	Scrolling the albums list and also the gallery and clicking in photo.	In general it behaves like the normal Gallery.	**Passed**
Folders view	Folders list is scrolled, gallery checked and photo clicked.	There is sometimes a crash when getting folders. The rest works well.	**Failed**
Tags	Setting tags in photos and searching for them.	It shows the correct photos, no problems.	**Passed**

Local photos	Having photos in the device and no invalid images.	The local photos are loaded, and the user can look around.	**Passed**
Uploading local photos	Choosing to upload to Dropbox, having a valid Dropbox account and session as well as space available.	Notifications for each file are shown as well as progress. Files are correctly uploaded.	**Passed**
Places	Google Play services in the device. Maps are fetched correctly. All photos locations request is done.	Photos are set in map correctly and the user can look, zoom and choose photos.	**Passed**
Clustering view of photos	Photos in a specific place are put in clusters.	Numbers are correct and locations also.	**Passed**
Clicking in cluster and having access to list	Choosing a cluster shows a list in the bottom of the screen.	Photos are correctly shown and locations are the correct ones as verified in server database.	**Passed**
Viewing photos from a cluster list in a pager	Choosing one of the photos from the cluster list and swiping between them.	There isn't any unexpected behaviors and only photos from the cluster chosen are in the pager.	**Passed**
Itinerary view	The itinerary option is chosen.	The cluster is zoomed in, and photos are set as well the connections between them.	**Passed**
Viewing photos in itinerary	Clicking in a photo from a itinerary.	The photo is shown in the viewer and has the features required.	**Passed**

Table 6 Positive Functional Tests

In general, the results are satisfactory. Some of the issues stand to be corrected. Most of the features are robust and present the expected output given a correct expected input.

6.1.2 Negative Functional Tests

For these tests bad input was tried to be provided. It leads to some unpredictable results. These are bugs or lack of protection that can be fixed until the end of the project and presentation.

	Input	**Output**	**Result**

Introduction	Exiting the application and getting back in.	The introduction is reset from the beginning but that is expected.	**Passed**
Sign up	Bad emails, not confirmed passwords.	Inputting bad data gives out Snackbars warning against it. It doesn't let bad data to be registered.	**Passed**
Sign in	Inputting wrong username/password combination.	Warning is given out.	**Passed**
Dropbox connection and sync	Clicking in Connect button. Not connecting to Dropbox. If connecting then exiting the application while syncing.	Without connecting to Dropbox, nothing happens, just returns to Connect screen. If connected but exits application, when getting back it shows the syncing progress without problems.	**Passed**
Syncing with our server	Not accepting to Sync. Exiting the application while syncing.	Not accepting leads to not using Smart features which is warned throughout the application. Accepting and going out of the Sync screen or exiting the application does nothing to stop the sync and progress is shown.	**Passed**
Push Notifications for syncing	Google Play services is not installed.	Push notifications will not work. A warning is given out for this.	**Passed**
Gallery	Exiting application, refreshing multiple times, scrolling aggressively, going to other screens.	Nothing critical happens. No files are corrupted, and photos are shown correctly.	**Passed**
Individual photo view and pager	Choosing a photo and then swiping aggressively. Choosing a photo while another option is processing.	Nothing really happens expect when disconnecting from Dropbox. But disconnecting from Dropbox immediately hides gallery, unless there is delay users wouldn't be able to click in a photo while doing that.	**Passed**
Individual photo details and location	Swiping aggressively between photos then trying to set location and seeing metadata details.	There can be possibility of sometimes not working well. Not very safe.	**Failed**

Location change	Going back and forth setting location screen to pager. Not cancelling but instead pressing outside the set location dialog.	There aren't memory problems. Not cancelling doesn't clean markers.	**Failed**
Sharing individual photo	Going back and forth. Pressing back while in share screen.	It can break the navigation drawer.	**Failed**
Downloading individual photo	There isn't much to do except to remove SD while doing it.	There is protection against not having a valid path.	**Passed**
Selecting photos to delete	Unselecting and selecting again. Scrolling around while selected. Exiting from gallery while deleting.	The behavior is good and nothing happens. There is dialog showing it is being deleted so it is not possible to exit to another menu. But the user can exit the application and it can potentially harm the process.	**Failed**
Selecting photos to move to album	Trying to add to a non existing album.	Unless there is error in album from server, nothing bad will happen.	**Passed**
Order photos by path	Going to another menu or screen while the order is in process.	There isn't any unexpected behavior. When user returns to gallery the data is ordered again if processing didn't finish.	**Passed**
Order photos by relative location	Going to another menu or screen while the order is in process.	There isn't any unexpected behavior. When user returns to gallery the data is ordered again if processing didn't finish.	**Passed**
Order photos by date	Going to another menu or screen while the order is in process.	There isn't any unexpected behavior. When user returns to gallery the data is ordered again if processing didn't finish.	**Passed**
Albums automatic creation	Not confirming the creation.	A warning message saying no albums yet if no albums exist. Nothing special happens, just no smart albums created.	**Passed**

Albums view	Trying to break the albums view. Going back and forth.	It behaves as well as the Gallery.	**Passed**
Folders view	No folders in Dropbox.	There is always the root folder.	**Passed**
Tags	Setting repeated tags.	They are not created, simply added.	**Passed**
Local photos	Having invalid images or no SD card.	Protection against invalid paths is set.	**Passed**
Uploading local photos	No Dropbox space available.	No protection against it but the app does not crash.	**Passed**
Places	No Google Play Services. Going back and forth, trying to break the map.	With no Play services, the map won't work but the app won't crash.	**Passed**
Clustering view of photos	Exiting the application while the calculations take place.	Nothing unexpected happens.	**Passed**
Clicking in cluster and having access to list	Trying to stress the list and pressing in multiple clusters.	Photos are refreshed, list works well.	**Passed**
Viewing photos from a cluster list in a pager	Trying to stress the photos pager.	Nothing unexpected happens.	**Passed**
Itinerary view	Going back to other screens. Exiting from itinerary quickly.	It does not crash, behaves well, although it can be a bit slow at times.	**Passed**
Viewing photos in itinerary	A huge amount of photos in a itinerary.	It can be slow but photos are loaded asynchronously. Also they are set in cache so next time it won't be slow.	**Passed**

Table 7 Negative Functional Tests

Stressing the application is an important step into having a robust and reliable product. In general, there are some considerations to be taken from these tests and some things to be improved. But it is possible to see that there are already some protections that disable bad behavior consequences from users.

6.2 Usability Tests

The goal of Usability testing is to access the quality of the user experience and how the user reacts to the flow of the application and its features. It is used to assess any potential frustrations or positives from the user experience and to use that knowledge to improve the application. There are various points to consider before doing the tests:

- The application is being tested to check if the focus on features is paid off by having built an application user friendly;
- The audience of the test is general application users with some experience and of young age, either adolescents or young adults with also a test subject of older age;
- The features of the application should be tested and the robustness to react to potential and common bad input from the user;
- The application has the Photos content part of the application done and only that part should be tested, so that the improvements can be replicated to the other contents.

Finally, the testing sessions will take between 10-15 minutes not including pre-session questions and post-session questions and the 5 test subjects are between 12 to 46 years' old. It was chosen 5 test subjects since it can comprise around 80-90% of the use cases and problems.

The users were explained what the application consisted in, the account and privacy details and what are the features present currently. Then users were asked some pre-session questions to get an idea of what the user might expect and what is their experience level with mobile applications, more specifically, Android applications. During the test, users were under attention to see what they were difficulties and how they generally communicated with the application. Details about each feature or screen were given if necessary, mainly when there was difficulty to understand.

The tests were conducted with several Android devices that are specified later and each user's Dropbox account. An internet connection is required for the application to work so a Wi-Fi connection is used with around 5mbps of speed.

As a final consideration, all tests subjects are close to the application developer so there might be some bias towards putting the application in a more positive light. But honest criticism was asked from test users anyway.

Next are presented each of the tests subjects and their results.

6.2.1 Usability Test Nº 1

The subject is Gabriela Costa of 21 years' old. The device used was a Vodafone Smart 4 with Android 4.4.4. The user has large experience with technology. The test was conducted by explaining the context of the application, privacy concerns and then asking pre-session questions. Then the test ensued and some post-questions were done.

Pre-session questions:

- **What do you expect from a Photo management application?**
  - Applications like this I only used the default Gallery that comes with the device but recently have started to use Carousel. So I only expect to see photos and have some access to the folders and some details but with

Carousel I saw some organization features that I liked like location and the backup.

- **What do you think about the features already present?**
    - I like the organization features like by Date and Location. The synchronization feature is good so that I can access it from other devices.
- **What is your Android experience?**
    - I use Android for about 1 year and I usually use Email, Social Networking apps and have been using Carousel for some time.

Test evaluation and considerations

The user reads the introduction and likes it overall, but advises that there should be a button to skip all of it immediately. When signing up, the user did not have enough time to read the view explaining the account was created and with a button to sign in screen and then the view went away. But in the end the user successfully signed in.

The user thought the Dropbox screen was nice and connected easily. Then waited for the Dropbox sync and knew it might take a while. It was noted that there should be a privacy notice regarding the sync dialog but then on the Synchronization screen, it is stated no files are stored. The user asked if it could go away from the Sync screen and if it continued, so it is something that should be stated.

The user then exited and went to the Gallery screen and scrolled around and saw the relative location to the current place. The user questioned if the most recent photos, when organized by date, if it was better to begin on top or on the bottom and found it better to start on top.

The user went to the photo view and did zoom and looked around and then shared a photo by email and tested if it worked on its own email.

In general, the user went around the app with not much difficulties. The user understood that some photos showed up later because they were still synchronizing. She also pressed the push notification, but since there was no sound in the notification, she did not notice it at first.

The user noticed that there was an option to click to define location in photos without location. Had a bit of difficulty pressing since it is a small text. The user successfully changes the location of a photo and states that some of the messages when showing location should be better. The user also tried to set tags and then saw how they basically worked.

The user then took a photo and saw that the photo was uploaded to Dropbox as shown in a computer near and saw in the Gallery short after by choosing order by places.

The user asked if the local photos are in the SD card. The user then proceeds to the albums screen and accepted that the albums would be created in the server. Then received a notification saying that the albums were created and viewed them. She enjoyed how the albums look and feel. The features of photos in albums work the same as the normal gallery.

She tried to delete photos by selecting them. Only not liked the selection color, but liked that the photo was deleted from everywhere, meaning Dropbox and application also. She also liked to create a new album and put photos there.

The user enjoyed the clusters in maps and zoomed in and out. She appreciated that it was numbered instead of showing a collage of photos because it ended up being more useful to know how many were taken in that location. The Logout screen should notice that all data is going to be deleted from the device.

In the end the application had a good look and feel, even the syncing that could continue without the user being the application is something good, although it takes some time. It had modern design and the performance was good, mainly when exiting and going back.

Post-session questions:

- **How was your overall experience?**
  - The application was pretty and some features that didn't see in other apps but to be noted I didn't have a lot of experience in other photos apps. The most interesting thing was the map, mainly the Itinerary, and to see the photos in a list bellow. The albums by special dates was also a nice touch.
- **What did you like most and least?**
  - Liked the least the syncing duration and the map was my favorite thing and liked the notifications to see how the progress went.

6.2.2 Usability Test Nº 2

The subject is Ricardo Henriques of 16 years' old. The device used was a Sony Xperia L with Android 4.2. The user has some experience with technology. The test was conducted by explaining the context of the application, privacy concerns and then asking pre-session questions. Then the test ensued and some post-questions were done.

Pre-session questions:

- **What do you expect from a Photo management application?**
  - Not much experience in Photo apps and only uses the default Gallery sometimes.
- **What do you think about the features already present?**
  - It is different than what I am used to, it is easier to see the photos and consult them later.
- **What is your Android experience?**
  - I have an Android device for a long time. Use several applications but mainly games.

Test evaluation and considerations

The user started the app and sees the introduction and knows how to go to the next page and finds it attractive. The user knows what he should do if there is no account and starts the Sign up, enters the information and confirms. The message confirming the account creation is too fast so it should have more duration.

The user found a bit hard to understand where to go after creating the account. The user proceeds after signing in, to connect to Dropbox, and finds the screen explicit.

The user looked around at the photos. Had some difficulties regarding clicking in photos because of the scrolling, but it is a user specific problem. The user likes that it is possible to Download the photo and to share photos. The user then noticed that it is not possible to order by location and date because photos are not synced. He confirms with OK the dialog asking to sync and goes to see the synchronization page and understands what the whole process entails.

He then goes to see if the already synced photos appear in order by date and location and they do. Playing with the gallery and photos pager the user saw the locations of the synced photos and how he could go through each photo and successfully set a new location to a photo.

The user then went to Albums, and it found to be practical to have photos in special events dates. Folders also work well and also the Local photos.

The user enjoyed also Itinerary because it showed a way for the several photos. In Settings, there are some good options like using mobile data or Wi-Fi only, and the Help, About and Privacy screens are the usual.

The map was the most interesting feature found and the photos seen in Clusters. Overall the application was found to be good, but given the lack of experience and interest in Photos applications, it is not an application that provoked a whole lot of excitement when using.

Post-session questions:

- **How was your overall experience?**
  - It was interesting. Some of the features would definitely use.
- **What did you like most and least?**
  - What I liked most was the map, to see the photos organized by the places and also the places in the photos pager. There wasn't anything that I particularly disliked.

6.2.3 Usability Test Nº 3

The subject is Alice Carvalho of 12 years' old. The device used was a Samsung Galaxy A3 with Android 5.1. The user has few to some experience with technology. The test was conducted by explaining the context of the application, privacy concerns and then asking pre-session questions. Then the test ensued and some post-questions were done.

Pre-session questions:

- **What do you expect from a Photo management application?**
  - I take a lot of photos but I don't have a lot of experience with Photo applications, and only recently have been using Carousel and Google Photos. I like that there are photos organized by date and time.
- **What do you think about the features already present?**
  - They are nice, I still have to see them in action. I like the backup feature.
- **What is your Android experience?**
  - I have Android devices for about 2 years. I use a lot of applications but they are mostly games. Also have large experience with social networks like Instagram and Facebook.

Test evaluation and considerations

The introduction explains well what is the purpose of the application and the look and feel is good. It seems the speed in which the message for signing up is too much fast so it should be longer.

The user signed in with not much difficulties. The Dropbox screen is explicit and connecting to Dropbox is simple, even more if there is already Dropbox app in the device.

The user clicked outside the dialog confirming to sync or get started and the dialog vanished so it was not possible to continue without a app restart.

The user had only 66 photos and started the syncing process. The gallery is well organized and looks good. None of the photos had location so it was necessary to set location for photos. When setting location, the close photos in terms of time and also taken with same camera model are supposed to be of the same place. Multiple photos were set with the same location enabling the fast use of the map feature.

There were some crashes for example when viewing folders or using the Push notifications. Application restarts seemed to fix them.

Tags were set in photos but not much used. Tried to search and it worked but not much interest in it was shown. Perhaps automatic tags are more interesting.

Photos were inserted in albums for testing and also deleted from Dropbox and everything worked ok. The map feature was clearly the most enjoyed mainly the itinerary. The albums feature is also good and shows that all photos are from Summer 2015 because the device was bought very recently and all photos were backed up from the device.

Overall the application was actually interesting to the user with curiosity shown mainly because of location and how the gallery is done. The user stated in the end that it would be nice to be able to use the application without requiring internet all the time.

Post-session questions:

- **How was your overall experience?**
  - The application is fast so it perceived to be good. A bit buggy with some errors but the location features are great. The backup is good and the albums are reliable even thought only one was shown.
- **What did you like most and least?**
  - I liked the most to be able to see all the details about photos and where it was taken. The least I enjoyed was some of the bugs when syncing or some random crashes.

6.2.4 Usability Test Nº 4

The subject is Cristiana Costa of 15 years' old. The device used was a Wiko Bloom with Android 4.4.4. The user has large experience with technology. The test was conducted by explaining the context of the application, privacy concerns and then asking pre-session questions. Then the test ensued and some post-questions were done.

Pre-session questions:

- **What do you expect from a Photo management application?**
  - To be able to organize by folders, to be able to create them. They can be more or less like albums.
- **What do you think about the features already present?**
  - They seem useful, they go in accordance with what I want from photo applications.
- **What is your Android experience?**
  - I have Android for over a year, and I use a lot of applications from games to social networks and several others.

Test evaluation and considerations

The user started the introduction. The user did not notice that there is another screen to register and tried signing in thinking it was only needed the password and username. There was some overall difficulty signing in with email. The user understood what he should do when connecting to Dropbox.

The user waited for the Dropbox counting of files and wondered if it was possible to exit. But the whole process was quick even though the user had more than 1000 photos in Dropbox.

The syncing process was accepted and it took a long time. But since so many photos were present that it was easy to continue using the application and its smart features with a subset of the photos processed.

In general, the gallery was found good and the user liked to swipe between photos as well as seeing their details. The user also found that setting the location in multiple photos at the same time was a good feature. But there were some issues when setting location, stating errors, so in the end the perception of the feature was stained.

The albums creation was fast and it showed multiple albums from special dates. There was a great number of photos so it showed how useful it could really be to divide the photos between date albums.

The map feature, as with the other users, was found to be the best feature. The user also really enjoyed how the application looked with Material Design especially the Navigation Drawer.

Post-session questions:

- **How was your overall experience?**
  - It was good. There were some bugs, but enjoyed the albums creation and the map is cool.
- **What did you like most and least?**
  - Albums and map and the navigation drawer with all the options. The least liked was that there were some issues showing the location of photos I put.

6.2.5 Usability Test Nº 5

The subject is Henrique Costa of 46 years' old. The device used was a Wiko Rainbow with Android 4.4.4. The user has few to some experience with technology. The test was conducted by explaining the context of the application, privacy concerns and then asking pre-session questions. Then the test ensued and some post-questions were done.

Pre-session questions:

- **What do you expect from a Photo management application?**
  - I would like to have photos organized by date and place. I like to take a lot of photos but usually just use a normal gallery to see them without any special features.
- **What do you think about the features already present?**
  - They seem good and different from what I am used to but I don't have much experience in photos applications.
- **What is your Android experience?**

- I have an Android device for around one year but in general I have never used a lot of technology. But now I can use Facebook, take photos and videos and do other things on my phone.

Test evaluation and considerations

The user did not have knowledge of English so it was needed to explain in each screen what should be done and where to press, more than with the other users.

The user liked how the application looked, the colors and design. Understood where to press in the introduction and when creating account help was required. The user anyway understood what he should do based on previous experiences.

The user enjoyed to see their pictures in the Gallery and how he could organize by date and location. The user went to the map and saw how his photos were disposed. The user moves around a lot during the day so it is interesting for him to see the photos in a itinerary.

Albums were also created and most showed from previous Summer and Spring. It was a nice addition but if the user had more photos from New Years Eve or Christmas it would be something he would enjoy more.

In general, the user went around trying different features and basically concentrated in the gallery and photo viewer.

Post-session questions:

- **How was your overall experience?**
  - It was good. I take a lot of pictures and if the app is published I would use it.
- **What did you like most and least?**
  - I liked to see the photos organized by dates to be remembered and the gallery. There was nothing that I particularly disliked.

6.2.6 Overall results

Each of the user's, after the test, were, along with the post-session questions, to grade each of the features of the application. The table next presented show all the scores for each user.

	Gabriela Costa	**Ricardo Henriques**	**Alice Carvalho**	**Cristiana Costa**	**Henrique Costa**
Introduction	4	4	4	5	4
Sign in and Sign up	4	4	5	4	4
Dropbox connection and sync	5	5	3	5	5

Syncing with our server	3	3	3	3	3
Push Notifications for syncing	5	5	4	4	5
Gallery	5	5	5	5	5
Individual photo view and pager	5	4	4	5	5
Individual photo details and location	5	5	5	4	5
Location change	3	4	5	4	3
Sharing individual photo	5	5	4	5	5
Downloading individual photo	5	5	4	5	5
Selecting photos to delete	4	3	3	5	4
Selecting photos to move to album	5	4	4	5	5
Order photos by path	5	4	5	5	5
Order photos by relative location	5	4	5	5	5
Order photos by date	3	4	3	5	4
Albums automatic creation	5	5	5	5	5

Albums view	5	5	5	5	5
Folders view	4	4	3	5	4
Photo tags	5	4	5	5	4
Local photos	5	4	5	3	5
Uploading local photos	5	5	4	4	5
Places	5	5	5	5	5
Clustering view of photos	5	5	4	5	5
Clicking in cluster and having access to list	5	5	4	5	5
Viewing photos from a cluster list in a pager	5	5	4	5	5
Itinerary view	4	5	5	5	4
Viewing photos in itinerary	5	5	5	5	5
Overall look and feel of the application	5	4	5	5	5
Performance perception	5	4	5	5	5

Table 8 Usability Testing Results

The results are satisfactory. The users played around with the application, found it pleasing and some of the features actually surprised them. The audience did not have a lot of experience with Photo's applications but with technology and Android is specific they had. There are some flows and details to improve and sometimes it can seem that the application might not have enough features. Also the Music and Documents part would be a great complement. But users understood the application and were able to use the features and that can only be considered a good result.

7 Conclusions and Future Work

The project in general had some level of success. It was supposed to have more features about each content but it turned out to be more focused on the Android client and the photos applications. The development of the Metadata Server and Content service had some rough spots mainly because it was a new technology (Spring Framework) but in the end the result is satisfactory with a detailed server and a Content Service that processes information fast.

The Android application had more detail and the libraries used along with the Android framework knowledge acquired, it was possible to build a robust application.

In terms of the internship itself, in Wit Software SA, Pedro Pinto supervised me well, with a lot of patience in terms of deliveries but the work was done, although, once again, the focus was not exactly the right one. There were friends made and the trip to Granada was an unforgettable experience and I will forever be grateful to Wit Software for giving me this opportunity to work there.

This project was very enriching for me in terms of software development and in terms of having contact with an enterprise environment. A career is starting in Android development and a lot of improvement was done over this year.

Regarding future work, the application can suffer from improvements in the contents processed and have more smart features even for photos like face recognition or automatic tags using image detection.

References

[1] Niso. Understanding Metadata. [Internet] Available from: http://www.niso.org/publications/press/UnderstandingMetadata.pdf

[2] Smart Insights. Mobile Marketing Statistics 2015. [Internet] Available from: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[3] Eurostat. Internet and cloud services – statistics on the use by individuals. [Internet] Available from: http://ec.europa.eu/eurostat/statistics-explained/index.php/Internet_and_cloud_services_-_statistics_on_the_use_by_individuals

[4] Sony. Sony Album. [Internet] Available from: https://play.google.com/store/apps/details?id=com.sonyericsson.album&hl=en

[5] alensw.com. Quick Pick Gallery. [Internet] Available from: https://play.google.com/store/apps/details?id=com.alensw.PicFolder&hl=en

[6] Picturelife. Picturelife. [Internet] Available from: https://picturelife.com/features

[7] Apple. Photos iOS 8. [Internet] Available from: https://www.apple.com/ios/whats-new/photos/

[8] Dropbox. Dropbox Carousel. [Internet] Available from: https://carousel.dropbox.com/

[9] Google. Google Photos. [Internet] Available from: https://en.wikipedia.org/wiki/Google_Photos

[10] Apple. iTunes. [Internet] Available from: http://www.apple.com/pt/itunes/music/

[11] Windows. Windows Media Center. [Internet] Available from: http://windows.microsoft.com/pt-pt/windows7/products/features/windows-media-center

[12] My Cloud Player. My Cloud Player v5.5. [Internet] Available from: http://mycloudplayers.com/

[13] Kodi. Xbmc Media Center. [Internet] Available from: http://kodi.tv/about/

[14] Audiobox. Audiobox by iCoreTech Inc. [Internet] Available from: https://audiobox.fm/faq

[15] Amazon. Amazon Cloud Player. [Internet] Available from: https://www.amazon.com/clouddrive/learnmore

[16] Google. Google Music. [Internet] Available from: https://play.google.com/store/apps/details?id=com.google.android.music&hl=en

[17] Neat. More ways to be Neat. [Internet] Available from: http://www.neat.com/

[18] Evernote. Evernote. [Internet] Available from: https://evernote.com/intl/pt/

[19] Adobe. Acrobat XI Pro / Features. [Internet] Available from: http://www.adobe.com/products/acrobatpro/features.html

[20] ScanMountain. ScanMountain. [Internet] Available from: http://www.scanmountain.com/?page=learnmore2

[21] Abbyy. ABBYY Mobile OCR Engine. [Internet] Available from: http://www.abbyy.com/mobileocr/

[22] Google Play. CamScanner – Phone PDF Creator. [Internet] Available from: https://play.google.com/store/apps/details?id=com.intsig.camscanner&hl=pt_PT

[23] Google. Google Drive. [Internet] Available from: https://www.google.com/drive/

[24] Scrum.org. What is Scrum? [Internet] Available from: https://www.scrum.org/resources/what-is-scrum/

[25] Universal Image Loader [Internet] Available from: https://github.com/nostra13/Android-Universal-Image-Loader

[26] Event Bus [Internet] Available from: https://github.com/greenrobot/EventBus

[27] Android Design Library [Internet] Available from: http://android-developers.blogspot.pt/2015/05/android-design-support-library.html

[28] OkHttp [Internet] Available from: https://github.com/square/okhttp

[29] Type of tests [Internet] Available from: http://www.softwaretestinghelp.com/types-of-software-testing/