Masters in Informatics Engineering
Dissertation/Internship 2015/2016
Final Report

Moments Share: Collaborative sharing of photos

Bruno Mota Antunes da Cunha
bcunha@student.dei.uc.pt

Supervisors:
David Rodrigues
Hugo Gonçalo Oliveira
September 1st, 2016

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

Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra
Pólo II, Pinhal de Marrocos, 3030-290 Coimbra

Tel: +351239790000 | Fax: +351239701266 | **info@dei.uc.pt**

WIT Software, S.A.
Centro de Empresas de Taveiro
Estrada de Condeixa, 3045-508 Taveiro, Coimbra

Tel: +351239801030 | Fax: +351239801039 | **info@wit-software.com**

Undergraduate:
Bruno Mota Antunes da Cunha
bcunha@student.dei.uc.pt, bruno.cunha@wit-software.com

Supervisor at University of Coimbra:
Hugo Gonçalo Oliveira
hroliv@dei.uc.pt

Supervisor at WIT Software:
David Rodrigues
david.rodrigues@wit-software.com

Acknowledgements

In the first place I would like to express my gratitude to WIT Software for giving me the opportunity to work in such outstanding conditions, alongside proficient and friendly coworkers who taught me a lot.

A special consideration for my two supervisors, David Rodrigues and Hugo Gonçalo Oliveira, who were exceptional to me during these past 12 months. They were extremely helpful every time I needed.

Furthermore, I would like to thank my family for the continuous support that I was given since the very first day when I joined the university. Without them this experience would not be possible.

Lastly I would like to thank all my colleagues trainees, we had a great time during the year and it certainly helped making our internship easier.

Abstract

Mankind is steadily becoming more involved with technology in the everyday life. A significant percentage of people own a smartphone, and Internet connectivity has never had such a reach as it happens nowadays. This means that everything can be connected, at any time. Smartphones can be useful to a wide amount of tasks, and to support this reality, everyday new mobile applications arise.

Though, there is one particular use case that is not properly explored by any application as of today. When a group of friends go on vacations together, they often take umpteen photos with their smartphones. And today this scenario triggers an issue. Why does not every member of the group have access to the photos taken by everyone, immediately? Why do they have to wait, commonly, for weeks and weeks, until everyone decides to share the contents with the group? This is what happens today, but this internship intends to solve this problem with a mobile application: Momentum.

This internship will focus on developing an iOS application prototype for Momentum, with the purpose of solving the previously stated problem in an innovative way. Momentum will offer the users a private space for sharing their photos in a collaborative way.

Keywords: “Collaborative Albums”, “iOS”, “Mobile Applications”, “Momentum”, “Photo Sharing”, “Private Events”, “Public Events”, “Swift”

Table of Contents

Table of Figures

Index of Tables

Glossary

Agile Software Development	Agile Software Development is a conceptual structure for software engineering projects that presents functional versions of the product on each iteration, rather that just at the end of the project.
API	Application Programming Interface is a group of routines and protocols that makes software's functionalities available to third parties without the need to know the implementation details.
Git	Git is a distributed version control system for software development that supports distributed workflows.
GitLab	GitLab is the open source self-hosted Git repository manager used at WIT Software.
Product Backlog	The Product Backlog is the instrument that contains a project's user stories, their priority and their estimated effort.
Redmine	Redmine is the online platform that contained Momentum's backlog. It is an open source web-based project management tool.
Sprint	Sprint is an iteration of the Agile Development process. It is its core unity, and usually lasts between 1 to 4 weeks.
Spring Boot	Spring Boot offers a quick way to start a Spring project. Removes Spring configuration's boilerplate and allows the developer to start coding right away.
Swift	Swift is the latest programming language from Apple destined to Apple's operating systems.
User Story	User Story in Agile software development technologies is a phrase that represents what a certain character of the system should be able to do. User stories help identifying the system's requirements.

Acronyms

API	Application Programming Interface
APP	Application
DB	Database
HTTP	Hypertext Transfer Protocol
iOS	Apple's Mobile Operating System
JPA	Java Persistence Application Programming Interface
JSON	JavaScript Object Notation
MVC	Model-View-Controller Design Pattern
NoSQL	Non relational Databases
OS	Operating System
PDF	Portable Document Format
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
URL	Uniform Resource Locator
UX	User Experience
XML	Extensible Markup Language

1 INTRODUCTION

The present document, together with all the appendices, is part of the work done at WIT Software during this one-year internship. The internship work was supervised by Hugo Gonçalo Oliveira, PhD professor at the Department of Informatics Engineering of the University of Coimbra, and David Rodrigues, Software Engineer at WIT Software.

1.1 Context

The internship is the final course of the Master's degree in Computer Science and Engineering, from the Department of Informatics Engineering of the University of Coimbra. The work was supervised by Hugo Gonçalo Oliveira, PhD professor at the Department of Informatics Engineering of the University of Coimbra, and David Rodrigues, Software Engineer at WIT Software.

WIT Software's office of Coimbra, Portugal, is where the internship took place. WIT Software plays a big role on the development of mobile applications worldwide. The company is based in Portugal, with headquarters in Lisbon and offices in Coimbra, Porto and Leiria.

1.2 Problem

Consider this scenario: A group of friends decide to go on vacations together, and each element of the group brings its smartphone to the trip. As it is common amongst those journeys, the elements of the group start taking, but the contents are not automatically shared by any app. The excuse is often the same: "No problem buddy, as soon as I get home I will send the pictures to everyone". Thus what happens is during the holidays several people capture their own memories, but at the end not everyone has control over all the pictures that were taken. And worse than that, in some cases the group has to wait several weeks, if not even months until everything is properly shared between the group members.

There will be two different types of events: public and the private events. The aforementioned scenario would fit in the private event type, once desirably the data should remain protected from public.

The idea of naming Momentum for a private group has to do with the intention of sharing any moment on someone's life with the ones they wish. A trip to the Balkans, a summer festival in California or holidays spent in a winter resort, are examples of possible Momentums. They work like a shared folder where the group members drop their contents in a collaborative way.

The product in development allows not only the group members who go on the trip to create a Momentum and therefore publish and consume the contents posted by the others, but also their grandmothers, e.g., and family members, to be able to join the group without posting permissions. A non-publisher member of a Momentum may still comment and react to any photograph posted by the publishers. The only constraint is precisely regarding posting.

Still regarding the Momentums, there is one extra characteristic, which differentiates them from the Events. Momentums may have an expiration date. Upon creation, the Momentum's owner is in charge to decide if the Momentum will self-destruct after a chosen period of time or not.

1.3 Target Audience

The audience of Momentum will be very wide, mainly because it has the goal of not cutting off any slice of the population. Nowadays a very wide spectrum of the population is active on the internet and on social networks, and this prototype aims to capture all its width. Though, the main target audiences are teens/young adults, both male and female.

Momentum will be suitable, e.g., for a public official event, like a summer festival or a rock concert, due to having as core feature the possibility of gathering all its pictures in one place.

And the possibility of creating a temporary Momentum will be extremely handy to groups of friends, while on vacations or partying e.g.. Knowing that those publications will have a limited time frame before self-destruction may encourage some users to use this product. What supports this idea is the belief that certain photos are only posted if users know they will not last forever, which generally has to do with privacy issues. Otherwise users do not even publish the contents.

Momentum can be very convenient basically in any kind of event, e.g. a wedding. One member starts by creating an Event, which can happen before, during or after the wedding, and invites the wedding members. Furthermore, all the members may post pictures, while having access to the contents taken by every other member participant.

1.4 Motivation

This section describes the reasons why the undergraduate selected an internship concerning mobile technologies development and, most particularly, photos and videos sharing mobile applications.

According to Pew Research Center [1], as of July 2015, 68% of the United States population owns a smartphone, and that number has been growing steadily over the past few years. 46% of the United States smartphone owners say it is something "they could not live without" [2]. These numbers represent the addiction that smartphones are turning into, and affect a relatively large part of the world population. European statistics, despite presenting lower values than the American's, demonstrates that the number of smartphones is also increasing, and it is predicted to continue to rise on the next years, according to Statista [3] .

Regarding iOS apps, as of July 2015 there were approximately 1.5 million applications on Apple's App Store [4], a slightly smaller number when compared to the Android market (1.6 million applications). That confirms the influence mobile applications have over the global population.

Moreover, looking into the chart with the most downloaded applications in the United States App Store, several apps can be found whose aim involves sharing personal photos. This demonstrates the trend that is emerging nowadays. People are increasingly getting more and more interested on applications that allow innovative ways for sharing photographs. As an example, one of those applications is a direct competitor of Momentum: Moments which belongs to Facebook and was ranked 18th by the beginning of January 2016 [5].

Nevertheless, all these applications have a set of limitations, e.g., they do not ensure the privacy of their groups, they do not provide both public and private groups creation within the same app, etc. There are, in fact, some interesting applications, but none of them is able to totally fulfill user's expectations. The applications available nowadays all lack some important features that will be implemented in Momentum, and the complete analysis between Momentum and its competitors will be performed hereafter.

Finally, the problem earlier described is real, and having the opportunity to solve it by developing a mobile application that can be a success worldwide is the reason for this internship.

1.5 Goals

The main goal of this internship is to develop Momentum, an app that allows the users to share their photos in a very easy way, by creating events. The events may be public or private. The public events will be known as Events, while the private ones will be called Momentums. Momentums can be temporary or permanent.

The application is intended to be cross-platform, i.e. not exclusively targeting a concrete operating system. It will be possible for a user to use Momentum on his iPhone and on his Android device.

Nevertheless this internship's goal is to develop a prototype of the application Momentum for iOS devices. Thus, the undergraduate is expected to acquire technical knowledge on the mobile development segment, and gain experience on application prototypes production.

However there was another mark to accomplish by the end of January/beginning of February of 2016. WIT Software was represented at Barcelona's Mobile World Congress 2016 [6], the self-proclaimed world's largest gathering for the mobile industry.

In order to match the expectations set at the beginning of the academic year, the student implicitly had short-term goals, namely:

- Learn iOS and the Swift programming language
- Improve his Java knowledge, by working with Spring Boot
- Master his databases skills, due to using PostgreSQL
- Learn Riak S2, an open source alternative to Amazon S3
- Learn GitLab, a Git repository manager
- Learn to use Docker, that fixes the issues caused by deploying an application over different operating systems
- Get familiar with Agile software development

1.6 Document Structure

The document consists of 7 chapters, the introduction that is described here, and the following chapters:

- State of the art: a complete study involving Momentum's competitors and features.
- Methodology: presents the methodology followed during the internship.
- Architecture: describes the main issues that emerged upon the architecture designing.
- Implementation and evaluation: there will be a description of the code implementation and the flow of development over the year, and the tests that were performed on the prototype.
- Conclusion: exposes the student's final thoughts regarding this one-year internship.

2 STATE OF THE ART

This chapter is a summary of the complete study regarding the State of the Art.

During this stage, the undergraduate started by selecting which applications could be considered important for the analysis of the direct competitors, in the State of the Art, and they were put side by side in order to be compared and to check their similarities and differences.

During this stage, the undergraduate started by selecting which features would be considered important in order to differentiate Momentum from its competitors.

Afterwards, it was possible to determine the list of features that could be used to differentiate Momentum from its direct competitors. Each of the competitors is followed by a small description, explaining what the application does, what was well achieved by their developers and what are their negative features that can be explored by Momentum. The results are then shown in a table.

Furthermore, the set of indirect competitors is described, containing a small description too and a few reasons why they are considered Momentum's indirect competitors.

At the end of this chapter, there is a small conclusion stating how could Momentum be different from the actual applications, in order to be a better solution to the customers.

This study is focused on the following points:

- Most popular direct competitors of Momentum
- Features that influence the market
- Most popular indirect competitors of Momentum

2.1 Direct Competitors

This section describes a small group of iOS apps that provide similar features to Momentum's, and thus can be considered direct competitors.

We considered as direct competitors the applications' whose main focus was to share photos and videos with friends and family in a private way. The following iOS applications were analyzed:

- Moments [7]

- Comet [8](Previously called Crossroad)
- WedPics [9]
- Cluster [10]
- Togethera [11]

2.1.1 Moments

Launched on June 15th 2015, Moments belongs to Facebook and currently is only available in the United States.

Moments currently figures in the top 20 of App Store free apps [5], and has been rising steadily in the charts in the past weeks. There is a trend emerging around this sort of applications nowadays and these numbers serve as an example. People are showing their interest in this market segment and thus it must be better explored.

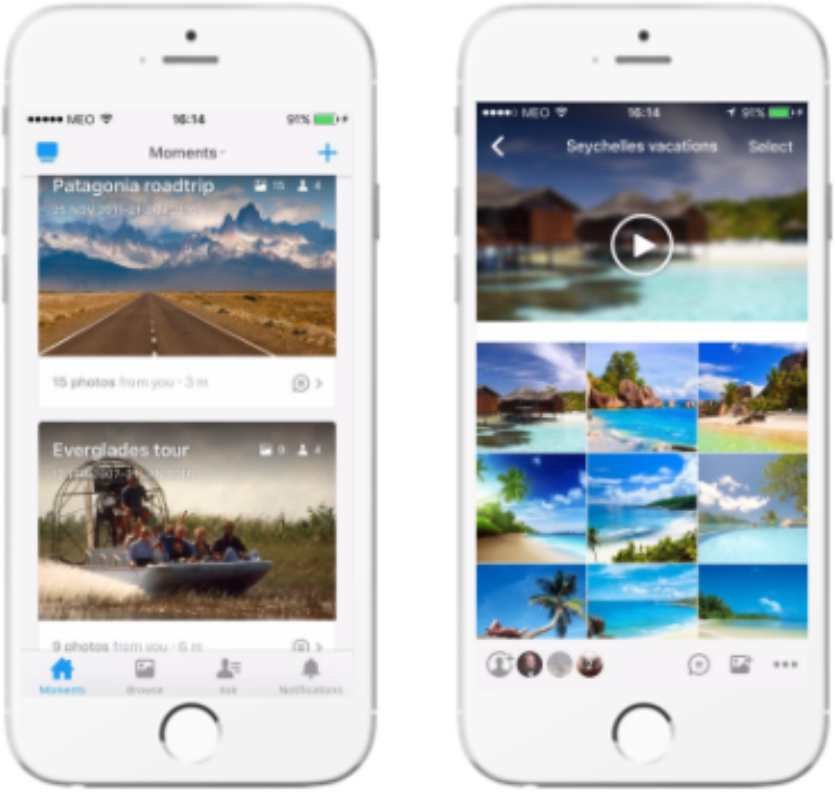


Figure 1 - Moments Screenshots

The application consists in albums (moments) shared by a group of people. Upon creation, the moment creator selects which friends they want to invite, by accessing his Facebook contacts. Afterwards, any moment member can post. The application has a very interesting feature that is to compile the moment's pictures and adding a customizable soundtrack. The outcome stands out for its originality. Another feature that should be emphasized is the face detection mechanism, original from Facebook.

It is promised that each of our friends will be automatically tagged, but at the moment it is not that effective. And the application presents us the photos sorted by location and by date.

To summarize, what makes this application stand out from the crowd is its extremely simple and original design, combined with the other features.

Comet (former Crossroad)

This competitor was launched on January 2nd 2014 in Paris by Mathieu Chabasse, Aurélien Sibiril, Florent Hobein and Mathieu Spiry, originally under the name Crossroad.

For the first 18 months the app was exclusively available in France, and since last September 2015 it is on App Store worldwide.

Figure 2 - Comet Screenshots

Comet has a lot of similarities with the other competitors, and focus on allowing a user to create an album and share it with his friends. The contacts list is obtained by accessing the device's contacts, and the photos are then displayed by chronological order, starting from the oldest or from the newest.

Furthermore it is also possible to export any album to the user's device, and share straight from the app to other services. The design is also very appealing and the simplicity is once more a key attribute on this application.

2.1.2 WedPics

WedPics aims into a particular target: weddings. A member of the engaged couple creates the private group and sets his/her fiancée/fiancé. They will become admins of the event, which means they can set guest's permissions, remove guests, delete contents and change the wedding details. The event is intended to remain private although by sharing it with the guests it may not work that way, since one public link is generated. Other options of inviting people are available such as sending email requests, sms and Facebook invites.

Then all the guests are free to post contents to the group, and the app made a good work on giving users the possibility to, during the upload, define the album which the uploaded contents should be related to, e.g., reception, ceremony, engagement, etc.

The app has no advertisement at all, but it has some paid features. A user can pay inside the application to physically print some photos, according to its region, after choosing one of the suggested formats.

Figure 3 - WedPics Screenshots

The idea of the app is great, although its main target is too limited. Why not developing such an application for different kinds of event besides weddings?

The company, Deja Mi, Inc. has headquarters in Raleigh, North Carolina. The application received nearly 10M of funding by the end of 2015, and was launched in December's 1st 2010 [12].

2.1.3 Cluster

Cluster Labs, Inc. is a San Francisco based company that owns several similar iOS applications: Cluster, TripCast [13], ChurchSnaps [14], Daily Kiddo [15] and Homeroom [16]. Despite having different names, the apps' architecture is precisely the same and the decision of having different names is nothing more than a smart marketing move to reach a wider audience.

The core app is Cluster, launched in 2013 in San Francisco, California. Brenden Mulligan and Taylor Hughes were its founders.

The company received 1.6M seed from 6 investors in August 2013 (Baseline Ventures, First Round, Freestyle Capital, Google Ventures and Sherpa Capital) [17].

Cluster has a very appealing design. A user can belong to several different groups, as almost all the other competitors here enumerated. In the groups menu we can search for their names, although this is the only searching feature inside the application. Each group has the members invited by the group owner or by other user with such privileges.

Figure 4 - Cluster Screenshots

Groups have only a name and a cover photo, which by default is the first photo uploaded.

Contents are displayed in a timeline feed and in a custom photos and videos gallery. In this last section contents can be sorted according to different properties such as date that they were posted, newest date first, oldest date first, person who posted and most liked contents. Those are basically the only metadata associated to the pictures and contents, since they have no hashtags and no geotag property. Thus no intelligent mechanism offers the possibility to organize the data in any different way.

Users of a group can invite friends by sending invitations. Invitations may arrive by email, sms or generated public link.

The data the user posts comes from the user device's gallery or from the device's camera. Once the contents are uploaded into the groups they may be exported by generating an external public link.

2.1.4 Togethera

The responsible company is called Generic Ventures Limited and it is based in London, UK. It was founded in 2013 by Sokratis Papafloratos and Matt Dempsey, by the time Togethera was launched. At the beginning the application was exclusively for iOS, then Android and Web versions followed.

Until September 2015 Togethera raised roughly $ 715k in seed financing, mostly from European Entrepreneurs.

Togethera allows creating an event and immediately sharing photos, videos and notes. Then we can invite our friends to join and we have a small private circle where any member can share contents within the group.
There are a few points that distinguish this app from the competition. To start, the possibility of auto-backing up the contents of a group into a Dropbox account should be highlighted, even though it is a premium feature. Notice that this requires not only having an account on Dropbox, but also having the necessary available space in this third party account.

Furthermore, this app has a premium account that enables the user to upload 100 photos and videos per post, extends video's maximum duration to 5 minutes, auto backs up all the data into Dropbox, and removes all the advertisement.
The app's design let's us quickly and easily move between our close friends and all our created or joined groups, simply by tapping a button in the top part of the screen.

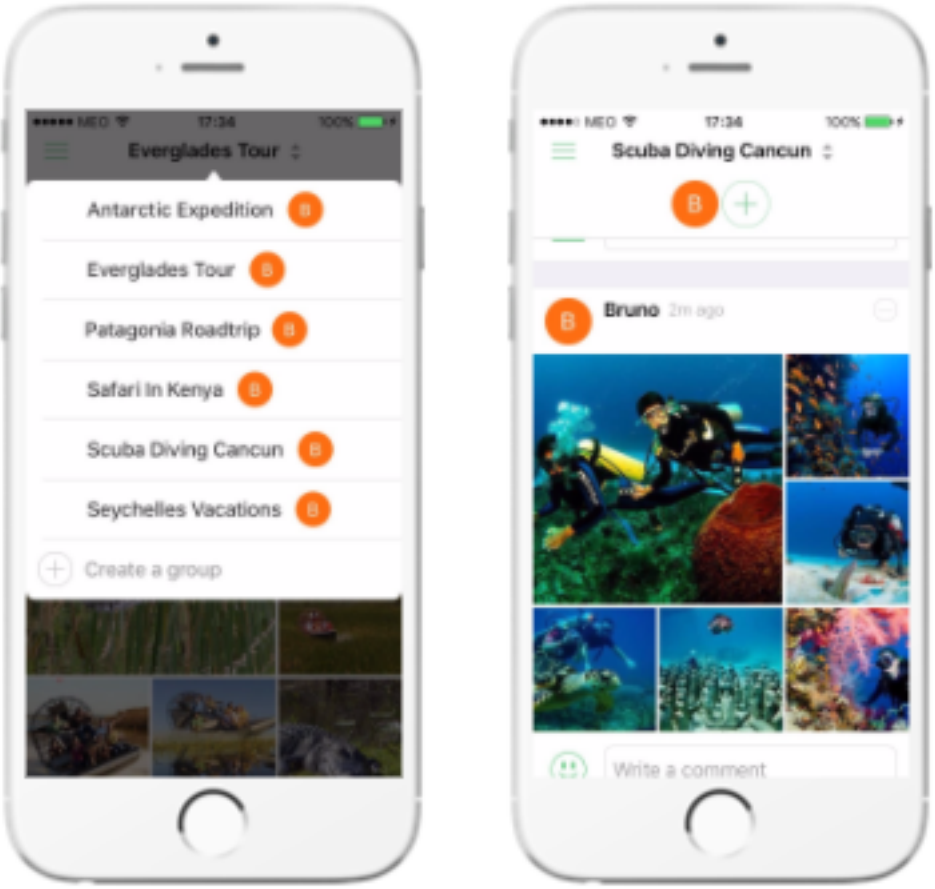


Figure 5 - Togethera Screenshots

It is possible to invite friends in order to participate in a group by generating a public link. Anyway this is not as bad as other app's public links since this one allows the user to reset it. Although in the student's opinion, other ways of member's invitation would constitute a safer approach.

The auto-refreshing property of this application cannot remain unnoticed, as well as the great work performed by the design team, who achieved an extremely simple and clean user interface. Moreover the possibility of posting contents to multiple groups at the same time is a good point.

On the other hand, one of the drawbacks of the Togethera application is the lack of a searching field.

2.2 Features

This section will enumerate the features that are believed to enrich an application who share the same purposes of Momentum.

After analyzing the biggest direct competitors of Momentum, it was possible to select a group of features, which allow a differentiation between all the involved applications.

Those features try to cover all the important tasks that an application should have implemented. In order to share our photos privately among family and friends, e.g., a few set of properties are required to the app. On the other hand, in case of public events, the concerns have to be slightly different.

Below are the features that were considered:

- **Public events** - Events that are created in order to host photos about a certain occurrence. Every user may join.
- **Private events** - Events that are created in order to host photos about a certain occurrence. Users are only able to join with personal invitation.
- **Temporary events** - Events that are created in order to host photos about a certain occurrence for a limited period of time. After that period, the events and its contents disappear.
- **Cross-platform** - Application that will exist on several platforms, e.g. iOS or Android.
- **Like/React to a post** - Liking or expressing other feelings for a photo.
- **Comment a post** - Append a text commentary to a photo.
- **Login with other services** - Chance of using the app by login in other service, like Facebook or Google, e.g..
- **Share to other services** - Export the pictures to other platforms, such as Facebook or Instagram.
- **Download photo** - Possibility of downloading a photo to the user's device.
- **In-app camera** - Access the camera inside the own application to take photos.

- **Thumbnails of photos** - Visualize thumbnails of the contents, to provide a faster user experience.
- **Photo in full-screen** - Display the photos in full-screen mode.
- **Update event automatically** - Refresh the event without any manual action
- **Search** - Text input that offers the user a search mechanism.
- **Tag people** - Tag users on photos.
- **Access Facebook contacts** - Connect with Facebook to get user's list of friends.
- **Zoom in photos** - Zoom in/out the pictures, by pinching in/out.
- **Hashtags on photos** - Relate the photos with topics.
- **View-only permission in events** - Inside the same event there may be users with posting privileges and others without them.
- **Find nearby events** - Access device's location to find public events nearby.
- **Receive notifications** - Receive notifications from the application.
- **Sort events by location** - List events based on location.
- **Sort events by date** - List events based on a specific time frame.
- **Favorites events** - Display a list with the user's favorite events.
- **Map view** - Show a map view of the events.
- **Receive suggestions for events** - Be invited to join events based on the user's preferences.

2.2.1 Comparison

The following table represents the comparative analysis between Momentum and its direct competitors. To distinguish the applications, the previously described list of features was used in order to highlight the differences.

Features	Momentum	Cluster	Comet	Togethera	Moments	WedPics
Public events	✔	✖	✖	✖	✖	✖
Private events	✔	✔	✔	✔	✔	✔
Temporary events	✔	✖	✖	✖	✖	✖
Cross-platform	✔	✔	✔	✔	✔	✔
Like/React to a post	✔	✔/✖	✔/✖	✔	✔/✖	✔/✖
Comment a post	✔	✔	✔	✔	✔	✔
Login with other services	✔	✔	✔	✖	✔	✔
Share to other services	✔	✖	✔	✔	✔	✖
Download photo	✔	✔	✔	✔	✔	✔
In-app camera	✔	✔	✖	✖	✖	✔
Thumbnails of photos	✔	✔	✔	✔	✔	✔
Photo in full-screen	✔	✔	✔	✔	✔	✖
Update event automatically	✔	✔	✔	✔	✔	✖
Search	✔	✔	✖	✖	✔	✖
Tag people	✔	✖	✖	✖	✔	✔
Access Facebook contacts	✔	✖	✔	✖	✔	✔
Zoom in photos	✔	✔	✔	✔	✔	✖
Hashtags on photos	✔	✖	✖	✖	✖	✖
View-only permission in events	✔	✖	✖	✖	✖	✖
Find nearby events	✔	✖	✖	✖	✖	✖
Receive notifications	✔	✔	✔	✔	✔	✔
Sort events by location	✔	✖	✖	✖	✔	✖
Sort events by date	✔	✖	✖	✖	✔	✖
Favorite events	✔	✖	✖	✖	✖	✖
Map view	✔	✖	✖	✖	✖	✖
Receive suggestions for events	✔	✖	✖	✖	✖	✖

Table 1 - Momentum's direct competitors

2.3 Indirect Competitors

This section describes a small group of iOS apps that do not rival directly with Momentum, but have the power to steal customers from WIT Software's application by offering similar services.

Every application from the list below includes at least one feature that Momentum provides, but none implements them all. In addition, most of the applications below described often offer a much wider set of functionalities.

Unlike our app, some of these apps do not allow the user to create private events in order to share contents with a group of friends, while others do not support public events. Regarding the temporary event feature that Momentum provides, only one application from this list does something similar, Snapchat.

The indirect competitors are:

- Facebook [18]
- Snapchat [19]
- Dropbox [20]
- Google Photos [21]
- Instagram [22]
- Flickr [23]

2.3.1 Facebook

Facebook is the all time most downloaded app for iOS [24]. The social network itself is also the biggest of all time, with over 1.55 billion monthly active users as of the third quarter of 2015 [25]. The set of functionalities they provide is huge. With no surprises, creating private and public groups that allow sharing photos and videos are some of them.

Nobody really questions the efficiency of this application. And naturally it is not in a direct fight with Momentum, since the features we focus and the features Facebook covers are not the same.

2.3.2 Snapchat

Snapchat does not look like a rival at first glance. But it may become one, considering some of Momentum's features. In Momentum, users may create an event that self-destructs at a certain date, which is precisely the core feature of Snapchat, where photos and videos are stored temporarily.

Snapchat was released by three students from Stanford University: Evan Spiegel, Bobby Murphy and Reggie Brown in September 2011.

According to the app's statistics, we figure out Snapchat has, by May 2015, 100 million daily active users worldwide [26], and as of January 2016 it is the fifth most popular application for iOS [27]. This product was worth $15 billion by the last time it raised money [28].

2.3.3 Dropbox

With 400 million users and 300.000 apps built on the Dropbox platform, as of June 2015, Dropbox is one of the big players in the online photography storage segment [29].

This application saves user's photos, videos, and other files, and displays them sorted by date or alphabetic order. There is also the possibility to create shared folders with our friends. These folders then become private and are only accessible through invitations or through a public link. Dropbox also has an auto-backup option, but is not possible to set it up in order to automatically share the photos to a shared folder.

Dropbox launched a few months back a standalone application (Carousel) with the purpose of presenting user's photos in a new and innovative way, but the solution did not run as expected and it will be discontinued on March 2016 [30].

2.3.4 Google Photos

Google Photos is very similar to Dropbox. It connects with user's Google account to retrieve their stored contents and display them in a very original way too. Once more, how the screen presents the user its photos and videos is definitely a plus sign, and the app allows the private groups creation too, just like Dropbox.

Although, following the Dropbox footprints, there is not the possibility to create public events, nor temporary events.

2.3.5 Instagram

This one is slightly different from the previous competitors. It does not provide private events creation, nor temporary events creation, but there is something similar to the concept of a public event.

Let's suppose a summer festival will occur, and the organization wants to gather all future photos taken by the crowd in one place. Nowadays it happens in Instagram,

through the use of Hashtags. Each element of the crowd just has to append the hashtag to its post and every content will be related. But this mechanism originates some issues: what if the people use different Hashtags like “SummerFestival15” and “SummerFestival2015”? And what if the hashtag has been used for other purposes at the same time? Those are the questions that Momentum answers.

2.3.6 Flickr

Flickr looks like Instagram and Facebook somehow. It is a social network where users can follow and be followed by others. Additionally they can create groups. These groups may be public or private, although, as it happens in Facebook, the application does not center its design on the groups.

Flickr is a social network. Its main focus is publishing photography for the community, not to share contents in groups publicly or privately.

2.4 Conclusion

After a deep study over the main direct and indirect competitors that are available on the App Store, there is a true conviction that Momentum can stand out from the crowd.

First of all, Momentum will provide the possibility of creating private and public events, and a very few set of competitors do the same (Facebook, Flickr). The difference these two applications is, Facebook and Flickr are not appealing enough to encourage the user to use these two features, at least not as appealing as Momentum. Momentum will have the public and private events screen always, at maximum, two taps away from the user’s position in the app.

So it is believed that the simplicity of Momentum’s design will play a big role on this competition. For example in Facebook, to create an Event and upload a photo it requires 14 taps to complete this action, not considering the user login. On Flickr the experience is quite similar to Facebook. The result is that Facebook and Flickr are not centered on this action so their users do not get intuitively attracted to perform this behavior. Momentum will focus on both creating public and private events, with a proper user experience directed to these two features.

Another difference is that none of the applications studied offer the possibility of creating temporary events. Snapchat offers something similar: to share pictures and videos temporarily. Momentum will take this idea and implement a slightly different concept: to create private events temporarily, i.e., events that will only last for a

period of time, before being destructed. As previously mentioned, this can be a very interesting feature from Momentum.

Comparing with the direct competitors, none of Momentum direct competitors allow creating public events. And regarding the private ones, Momentum will implement an innovative and creative user experience to provide a smoother UI to its users, in order to be a successful application.

3 METHODOLOGY

In this chapter, the methodology used by the undergraduate during the internship will be explained.

3.1 Agile

The internship follows an agile software development. Agile development works as a conceptual structure for software engineering projects.

Contrarily to the Waterfall model, where software teams separate the software development through several stages (conception, initiation, analysis, design, construction, testing, production/implementation and maintenance), those stages run as a cascade, i.e., after finishing one stage the process will not rollback to a previous stage, on the Agile model the development phases are iterative.

The development team usually presents multiple intermediary versions of the final product to the client, since this way it is easier to identify if something is not meeting the client's expectation. In some cases, even the requirements initially set by the client are not those that they wished to see implemented. In the waterfall model, that would be severe because the first time the client would see the working prototype it might be already too late to change something. On the other hand, in an agile approach modifications are much easier to handle.

The Agile approach that was followed during the internship allowed the student, e.g., to rollback his first version of the application in February, after the first usability tests, and start again. In a Waterfall model this procedure would be harsher.

3.2 Sprints

Each iteration is called a sprint, and is like an own software project inside itself. A sprint usually lasts between one and four weeks, and often contains the same stages: planning, requirements analysis, design, coding, unit testing and acceptance testing. At the end of the sprint the outcome is presented to the project's stakeholders. The purpose is having an available release at the end of each sprint.

During the development of Momentum a sprint took, on average, 10 working days, i.e., two weeks.

3.3 Sprint Meetings

At the end of each sprint, a meeting occurs. During Momentum's development, the characters of the meeting were essentially the undergraduate and its supervisor at WIT David. Occasionally, a third person intervened: Momentum's Manager, WIT Software's Engineer Luis Guilherme. The sprint intention is to evaluate the previous sprint, figure out what went wrong and why, and plan the next sprint.

3.4 Planning

At the beginning of the internship, one of the goals of the student was to, together with his supervisor at WIT, define what features would be implemented during the internship and what features would be left for future implementation. It is important to enlighten that the purpose of the internship in WIT Software is not to develop a fully functional product ready to be commercialized, but rather to implement a prototype of that product. With this taken into account, a set of user stories was planned.

The user stories were formerly assigned a priority and a theme, and later a t-shirt size. Finally, each user story originated a set of tasks that are closer to what the developer would implement.

3.5 User Stories

One of the most important parts of the Agile paradigm is choosing the user stories. Those descriptions consist of a sentence divided by three parts: the first represents the character, the second demonstrates the action to be performed and the third is the purpose of the action, even though this last one is optional.

One example of a user story would be: "As an user, I want to create an event to share my photos with my friends".

Since the internship has followed the Agile paradigm, the user stories changed throughout the route.

The initial full set of user stories, as well as the final's, are reported below:

Initial User Stories

Theme	ID	User Story
Sign Up	SU01	As a visitor, I want to sign up using my Facebook account.
Authentication	AU01	As a visitor, I want to sign in using my Facebook account.
	AU02	As a user, I want my session to be automatically renewed.
	AU03	As a user, I want to log out.
Account	AC01	As a user, I want to delete my account and all my content.
	AC02	As a user, I want to control if mobile data can be used to upload and download content.
	AC03	As a user, I want to control if photos and videos taken in the in-app camera should be kept locally in the device's gallery.
	AC04	As a user, I want to export my contacts to the service to quickly share content with my colleagues, friends or family.
Transfers	TR01	As a user, I want my uploads and downloads to continue when the application goes background.
	TR02	As a user, I want to check the progress of uploads and downloads.
Moment: Management	MM01	As a user, I want to create a new moment from a selection of photos already taken in the past to share with my colleagues, friends or family.
	MM02	As a user, I want to create a new moment to aggregate all photos taken in the future for a certain time frame.
	MM03	As a user, I want to see whom, of my contacts, is already registered in the service to share a moment with them.
	MM04	As a user, I want to see whom, of my contacts, is not registered in the service to invite them to share a moment with me.
	MM05	As a user, I want to create groups of contacts to help me share content with more than one person easily.
	MM06	As a user, I want to be notified when someone invites me to join a moment.
	MM07	As a moment's member, I want to see who has joined the moment.

	MM08	As a moment's member, I want to see who has a pending invitation.
	MM09	As a moment's member, I want to be notified when a member joins the moment.
	MM10	As a moment's member, I want to leave a moment.
	MM11	As a moment's owner, I want to give a name to my moment.
	MM12	As a moment's owner, I want to invite more contacts to join my moment.
	MM13	As a moment's owner, I want to remove contacts and all their content from my moment.
	MM14	As a moment's owner, I want to delete my moment.
Moment: Visualization	MV01	As a user, I want to see the moments that I have created and also the ones I have joined.
	MV02	As a moment's member, I want to visualize a list of thumbnails of photos and videos belonging to the moment.
	MV03	As a moment's member, I want to visualize photos and videos in full screen.
	MV04	As a moment's member, I want to visualize photos and videos sorted by date.
	MV05	As a moment's member, I want to visualize photos and videos grouped by place.
	MV06	As a moment's member, I want to visualize new content inside a moment in real time without the need to manually refresh.
	MV07	As a moment's member, I want to be notified when someone adds new content and the app is not currently running.
	MV08	As a moment's member, I want to visualize who posted a photo or video.
	MV09	As a moment's member, I want to see a geographic view of all the photos and videos.
Moment: Interaction	MI01	As a moment's member, I want to add new photos and videos from the in-app camera to the moment.
	MI02	As a moment's member, I want to add new photos and videos from the device's gallery to the moment.
	MI03	As a moment's member, I want to delete any photo or video added by me.
	MI04	As a moment's member, I want to download photos and videos.

	MI05	As a moment's member, I want to comment photos and videos.
	MI06	As a moment's member, I want to like photos and videos.
	MI07	As a moment's member, I want to add a reaction to photos and videos.
Event: Management	EM01	As a user, I want to create an event publicly available to gather all content regarding a certain occurrence.
	EM02	As a user, I want to add an event to my favorites to have an easy access to it.
Event: Visualization	EV01	As a user, I want to search events available.
	EV02	As a user, I want to see nearby events based in my current location.
	EV03	As a user, I want to visualize a list of thumbnails of photos and videos belonging to the event.
	EV04	As a user, I want to visualize photos and videos in full screen.
	EV05	As a user, I want to visualize photos and videos sorted by date.
	EV06	As a user, I want to visualize photos and videos grouped by place.
	EV07	As a user, I want to visualize new content inside an event in real time without the need to manually refresh.
	EV08	As a user, I want to visualize who posted a photo or video.
Event: Interaction	EI01	As a user, I want to add new photos and videos from the in-app camera to an event.
	EI02	As a user, I want to add new photos and videos from the device's gallery to an event.
	EI03	As a user, I want to delete any photo or video added by me.
	EI04	As a user, I want to download photos and videos.
	EI05	As a user, I want to comment photos and videos.
	EI06	As a user, I want to like photos and videos.
	EI07	As a user, I want to add a reaction to photos and videos.
Camera	CA01	As a user, I want an in-app camera to publish a picture or a video directly in a moment, flash or event.

	CA02	As a user, I want an easy way to select the moment, flash or event to where I want to publish the picture or video.
	CA03	As a user, I want to see a different layout to easily distinguish if I'm posting to a flash or not.

Table 2 - Momentum's user stories

Final User Stories

Theme	ID	User Story
Sign Up	SU01	As a visitor, I want to sign up using my phone number.
Authentication	AU01	As a visitor, I want to sign in using my phone number.
	AU02	As a user, I want my session to be automatically renewed.
	AU03	As a user, I want my session to expire after a period of time.
Account	AC01	As a user, I want to delete my account and all my content.
	AC02	As a user, I want to control if mobile data can be used to upload and download content.
	AC03	As a user, I want to control if photos taken in the in-app camera should be kept locally in the device's gallery.
	AC04	As a user, I want to export my contacts to the service to quickly share content with my colleagues, friends or family.
Transfers	TR01	As a user, I want my uploads and downloads to continue when the application goes background.
	TR02	As a user, I want to check the progress of uploads and downloads.
Event: Management	EM01	As a user, I want to create a new event from a selection of photos already taken in the past to share with my colleagues, friends or family.
	EM02	As a user, I want to create a new event to aggregate all photos taken in the future for a certain time frame.
	EM03	As a user, I want to see whom, of my contacts, is already registered in the service to share an event with them.

	EM04	As a user, I want to see whom, of my contacts, is not registered in the service to invite them to share an event with me.
	EM05	As a user, I want to create groups of contacts to help me share content with more than one person easily.
	EM06	As a user, I want to be notified when someone invites me to join an event.
	EM07	As an event's member, I want to see who has joined the event.
	EM08	As an event's member, I want to see who has a pending invitation.
	EM09	As an event's member, I want to be notified when a member joins the event.
	EM10	As an event's member, I want to leave an event.
	EM11	As an event's owner, I want to give a name to my event.
	EM12	As an event's owner, I want to invite more contacts to join my event.
	EM13	As an event's owner, I want to remove contacts and all their content from my event.
	EM14	As an event's owner, I want to delete my event.
Event: Visualization	EV01	As a user, I want to see the events that I have created and also the ones I have joined.
	EV02	As an event's member, I want to visualize a list of thumbnails of photos belonging to the event.
	EV03	As an event's member, I want to visualize photos in full screen.
	EV04	As an event's member, I want to visualize photos sorted by date.
	EV05	As an event's member, I want to visualize photos grouped by place.
	EV06	As an event's member, I want to visualize new content inside a event in real time without the need to manually refresh.
	EV07	As an event's member, I want to be notified when someone adds new content and the app is not currently running.
	EV08	As an event's member, I want to visualize who posted a photo.
	EV09	As an event's member, I want to see a geographic view of all the photos.

Event: Interaction	EI01	As an event's member, I want to add new photos from the in-app camera to the event.
	EI02	As an event's member, I want to add new photos from the device's gallery to the event.
	EI03	As an event's member, I want to delete any photo added by me.
	EI04	As an event's member, I want to download photos.
	EI05	As an event's member, I want to comment photos.
	EI06	As an event's member, I want to like photos.
	EI07	As an event's member, I want to add a reaction to photos.
Camera	CA01	As a user, I want an in-app camera to publish a picture directly in an event.
	CA02	As a user, I want an easy way to select the event to where I want to publish the picture.

3.6 Product Backlog

All the user stories were saved into the product backlog, which is an online platform that contains all the dynamic specifications of the project. Each user story has a category and number of points assigned. Story points correspond to the estimated time required to implement a certain user story. To estimate this value in Agile software development, one of the most common methods is using t-shirt sizes. T-shirt sizes are values from XS to XXL that follow the Fibonacci sequence in the way that they refer with the number of days, hence points, that they are assigned:

- XS is the smallest measurement unit and corresponds to 1 working day, i.e., 8 points.
- S corresponds to 2 working days, and 16 points
- M relates to 3 working days, and 24 points
- L points to 5 days, and 40 points
- XL means 8 working days, and 64 points
- XXL is called Epic, and means this user story has to be split into smaller task

4 ARCHITECTURE

This chapter specifies the architecture was designed for Momentum's prototype and the decisions that had to be taken before the development phase.

The architecture planning took place after the creation of the user stories, and right before the start of the development stage. During this phase, many questions emerged:

- Where should the photos be stored?
- Which kind of database should be used: relational or non relational?
- Should the iOS application be developed in Objective-C or Swift?
- In the back-end, which framework should be used? And, e.g., in Java or in Python?
- How should we handle user authentication?
- To deploy the application should we use Docker?

The undergraduate did not have any real life experience in choosing the architecture for a real system until the beginning of this academic year, so this task was an interesting challenge since its first moment.

4.1 Decisions

The prototype that is being developed focuses mainly on the mobile application, although the back-end and the database management are also taken care.

We will start the section by describing the front-end solution at first place.

4.1.1 Objective-C vs. Swift

At the very beginning of the internship, both the student and his supervisor at WIT discussed whether should they use Objective-C or Swift to develop the iOS application. Once more, the front-end component of the system was the biggest focus of the internship, so this was an important decision.

On the other hand, there is Objective-C, iOS's and the OS X's main programming language since the 1980's. It is a very stable language, and is completely compatible with his successor. Due to its long existence, this programming language gathered a large support community on the Internet, which constitutes an advantage comparing to Swift. Furthermore, most of the applications on the market were developed in Objective-C.

On the other hand, Swift is the brand new programming language by the Apple developer's team, and is expected to last for quite a few decades, just like its

predecessor. Although both languages can be interpreted by the same iOS compiler, they have strictly different syntaxes.

Swift's syntax is similar to Java and Python, languages that the student worked with during the university. Thus, Swift would represent a smoother learning curve to the student, since he did not have any experience in iOS systems as of the beginning of the internship. Oppositely, Objective-C language is harder to learn for beginners in iOS, and does not have similarities with the programming languages that the student was familiar with.

For those reasons, right at the beginning of the internship, the undergraduate and his supervisor at WIT agreed on taking Swift as the chosen programming language for implementing the Momentum's mobile application.

4.1.2 Back-end framework

The back-end of the product was never intended to be the main focus of the internship. Though, it was implemented too in order to give support to Momentum's mobile application. WIT Software gave the student total freedom to choose the programming language he felt most comfortable with. Due to the student's experience during his degree, Java, alongside Python were the two first possibilities to emerge.

The student asked for his coworkers' opinions, spoke with more experienced people about the project and ended up being advised that Spring Boot would fit perfectly in his model. The support community in the Internet is great, and since the student was already familiar with the Java language there were no reasons not to choose Spring Boot.

The back-end was therefore developed in Java using Spring Boot.

4.1.3 Persistence

In any back-end service, one of the core features has to do with persisting the data.

First of all, it is important to take into account several important requirements related to our data structure, such as the kind of data to store. The contents to be persisted were essentially: photos, videos, their metadata, user profiles and the events. So the data was divided into: contents and metadata.

4.1.3.1 Contents persistence

For the pictures and videos the doubts were: should we persist them in databases or in distributed data stores?

Initially, distributed data stores and simple databases were compared. The student realized that storing pictures or videos on non-distributed databases overloads them. And databases are often the first parts of a system to fail when scaling occurs. Additionally, in case database migration is necessary, distributed data stores proved to be the best solution since it is easier to move paths to the pictures and videos from table to table than moving whole contents.

Afterwards, the goal was to find good distributed data store candidates. The idea was to store this data on cloud providers, since there are interesting quality/price ratios on some of the players of this area.

Amazon S3 [31] was one of them. In fact, the student had already had the opportunity to try this service previously. The feedback was great. Amazon S3 guarantees at least 99.99% of availability for the user's objects, provides great scalability, is low-cost compared to its competition and can be integrated with a list of other services.

But the solution had to be open source. That was a constraint set by WIT Software.

Then, another very interesting solution emerged, Riak S2 has almost all Amazon S3's advantages but is open source [32]. This specification would ensure a cost saving solution, at least at first glance.

Another particularity of Riak S2 was that its migration to Amazon S3, once needed, would be very simple due to both apps sharing a similar API.

The undergraduate realized that the most accurate solution for this scenario would be storing the photos and videos on Riak S2.

Riak S2 is an open source service, and provides high availability and scalability.

4.1.3.2 **Metadata persistence**

Next step would be deciding where to persist the remaining data that would not fit in Riak S2, like the photos and videos metadata, the users profiles, and so on.

At this point, we could follow two paths: the relational and the non-relational databases. Both of them have their pros and cons, and the decision of which one to use naturally caused a big impact on the system development.

We knew in advance Momentum would require a lot of relationships between the entities such as: user to photo, user to event, event to photo, etc. A user may belong to several events, and an event may have multiple users enrolled. In this situation, a relational database would be the best fit. Likewise, on the relational model enables

join operations over the database, and data integrity is ensured, as well as atomicity, which would be very important, e.g., for transactions, or when the system wants to delete an user from two different lists at once.

Relational databases would always be a good solution since they are in the business for many decades and achieved great success since its very beginning. The non-relational model only emerged recently.

An alternative would be MongoDB [33], that would offer flexibility over the data structure, better scalability, it would cut out the overhead caused by the other model indexes, and a few more advantages. But the disadvantages were more difficult to handle. The many-to-many relationships between the user, the event and the photo entities, e.g., would not become so clear in MongoDB while compared to PostgreSQL. MongoDB does not provide join table operations, as well as the ACID properties (atomicity, consistency, integrity and durability) that MongoDB does not support and PostgreSQL does. The join table operations revealed very handy during the Spring Boot implementation, and the atomicity and data consistency, e.g., will be important when dealing with the databases.

The decision was to use PostgreSQL [34], which can be considered a hybrid solution, since it gathers the best of both the relational and the non-relational worlds. It provides the flexibility that characterizes so well the non-relational model, once PostgreSQL supports JSON fields, and has way less overhead than the relational model. It is a relational database that embraces the NoSQL best attribute.

4.1.4 Summary

This prototype would consist of a mobile application, developed in Swift [35], and a back-end application developed in Java, taking advantage of Spring Boot [36]. The back-end part will also connect to a database, PostgreSQL [34], and a distributed data store, Riak S2 [32], which means this prototype will play the role of a cloud provider too, even though it may not be the case of the Momentum's final application. The final version of Momentum will have two options though, to store the data in its own servers or, instead, work with external cloud providers such as Dropbox [20] or Google Drive [37].

4.2 Spring Boot Fundamentals

Spring Boot [36] is a simple way to create Spring-powered applications and services. Developers can start coding right from the first moment, without wasting too much time in setting up application's configurations.

Using Spring Boot initializer [38], the developer just selects which dependencies they want to include, and Maven [39] (which is part of Spring Boot projects) takes care of the rest. All the project configurations are on a file named *application.properties*, so there is no need to maintain any xml file.

The JPA [40] (Java Persistence API) and the Hibernate [41] dependencies were used in order to establish the connection with the PostgreSQL database, and annotations to avoid the tedious and old-fashioned XML used to wire the Spring Boot's beans. The code was organized according to the following layers: service layer, domain layer, controller layer and the configurations layer.

A simple architecture diagram containing the main entities of this project can be visualized in Figure 6.

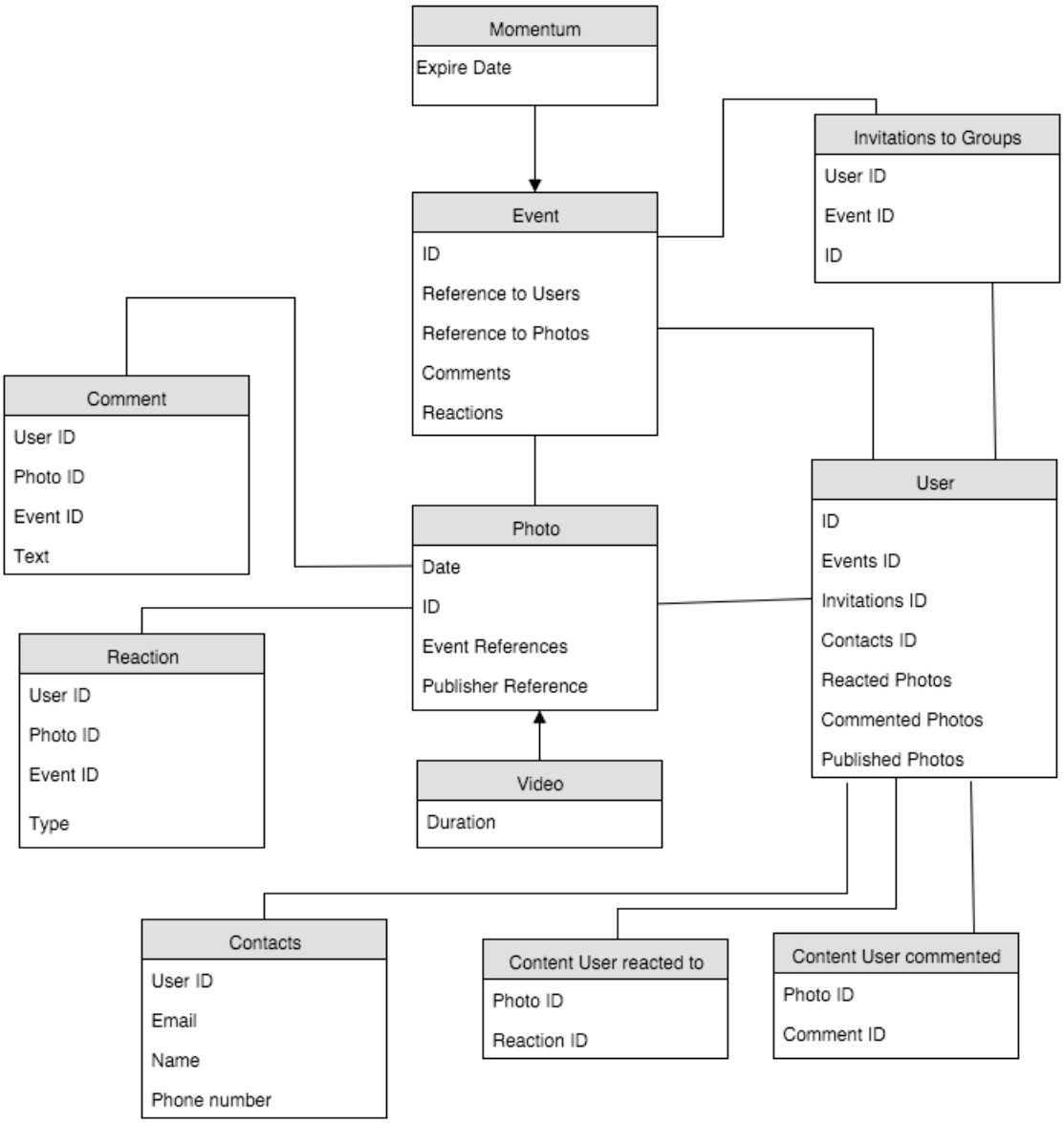


Figure 6 - Architecture diagram

4.3 iOS Fundamentals

In this section the iOS will be detailed, including some of its best practices, design patterns and developing techniques.

4.3.1 Application life cycle

In this section the iOS application life cycle will be described.

An application can have one of the following states: not running, inactive, active, background and suspended.

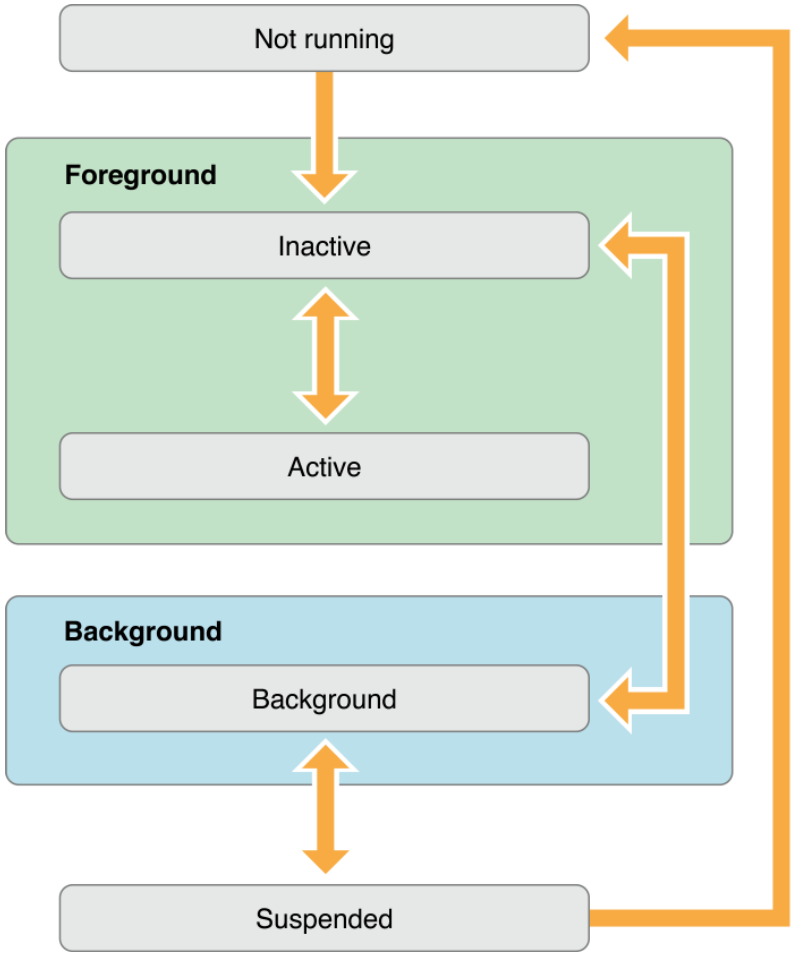


Figure 7 - iOS App life cycle [42]

- **Not running**

Once in this state, the application does not consume any computational resources. The application, either was not launched by the system or was already terminated.

- **Inactive**

When the application is in an inactive state it means that although the app is running in the foreground, it is not receiving events. Usually the application remains in this state for a very short period of time, immediately before a transition to another state.

- **Active**

In this state the application is running smoothly in the foreground. The application is visible to the user and its events are being received and handled. This is the most frequent state of an app running in the foreground.

- **Background**

This is the state that an application has when performing a task while being hidden from the user. If code is being executed and the application is not running in the foreground, then this is the state. The operating system limits the activity of such apps usually by 180 seconds of execution. Most applications however only pass though this state very briefly, in order to become suspended.

- **Suspended**

Oppositely to the previous state, in this one the application is in the background but not performing any code. After the time frame allowed by the operation system to run in background expires, the application reaches this state. It remains in memory until the operative system needs to reallocate space for the apps running in the foreground, in case a low-memory condition occurs.

4.3.2 iOS design patterns

In this section the iOS Design Patterns used during the internship will be detailed.

The design patterns are a topic that should not be underestimated by a programmer, since they help developers to write better code, code that is easy to reuse and easy to understand. Keeping the code well structured, clean and easy to read is halfway to achieve the developer's goals and deliver a solid and robust project to the client.

Thereon, the most common design patterns on iOS can be included on three different categories: Creational, Structural and Behavioral.

On the course of the internship, the following design patterns were adopted by the undergraduate:

- Creational: Singletons

- Structural: MVVM, Delegates and Protocols
- Behavioral: Observer, Notifications, Persistence

MVVM

The Model-View-ViewModel is a small variation of the most widely used on iOS, Apple MVC model. The biggest difference is that contrarily to the Apple MVC model, on MVVM the presentation logic remains separated from the view. The four components of the MVVM are the model, the view model, the view controller and the view.

The view model owns the model, and consequently is updated by it. The view model also updates and is own by the view controller. Lastly, the view controller manages the view and implements all its logic. Comparing to the Apple MVC model, the major difference on this scenario would be that the view and the view controller would be one single element. The following image illustrates the above-mentioned behavior.

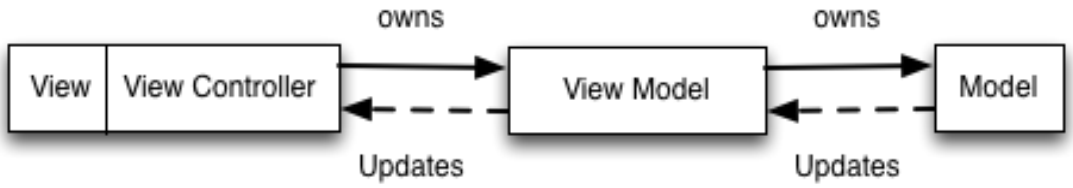


Figure 8 - MVVM model [43]

Some important advantages of using this model are that the views become more reusable and the app is easier to test.

Delegation

Delegation is …

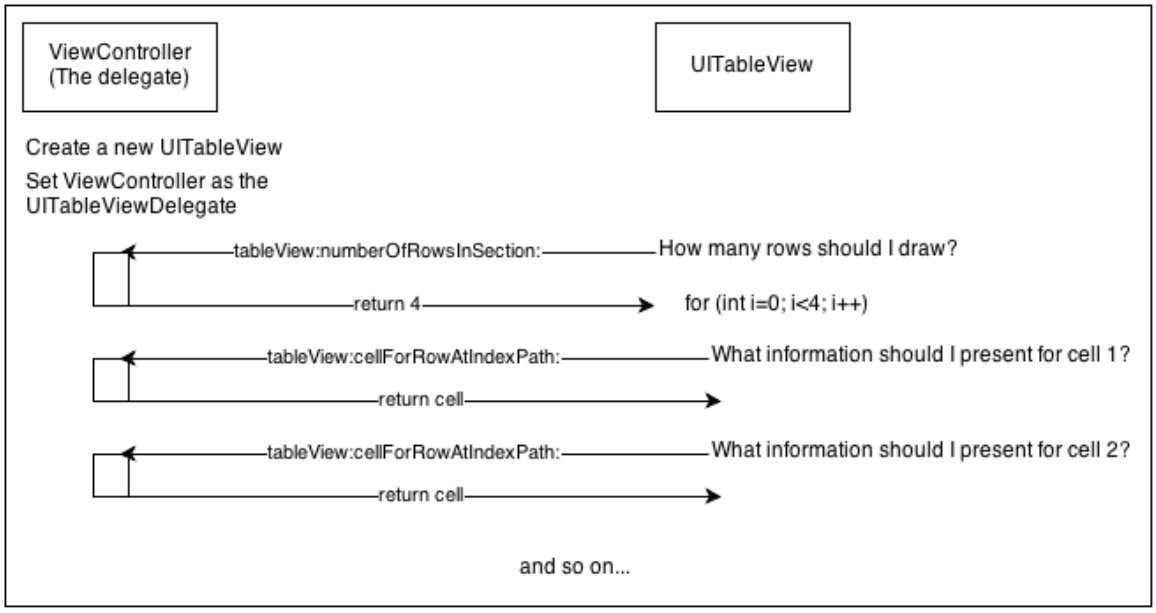


Figure 9 - Delegation example [44]

Singleton

Singleton is a creational design pattern [45] that ensures that a class has only one instance. That instance can be accessed from anywhere in the application, in a thread-safe way once initialized with the word *let* (which means that the instance is immutable), and supports lazy initialization since Swift lazy initializes class constants and variables.

An example of its usage on the Momentum app is when recording important data regarding the user session or the user account, e.g., the account id, the user name, the authentication tokens, the account email or the account phone number.

Observation and Publish-Subscribe

The observer software design pattern consists on an object maintaining a list of dependents. The object then notifies its dependents, called observers, usually by calling one of their methods.
In Swift, the two most common popular ways of implementing the Observer pattern are by using the Notifications or the KVO Key value observer.

Notifications are based on the Publish-Subscribe model, i.e., one object plays the publisher role and sends messages to other objects, which are the listeners or subscribers. The publisher does not need to know anything regarding the subscribers. An example of where this technique is used on the project is, e.g., when launching the feed screen. The application sends an HTTP request to the backend requesting the list of events where the user belongs, and continues running its code.

Later, when the response arrives, the object that ordered the HTTP request is notified and some code is triggered.

A simple code snippet is illustrated below:

```
NSNotificationCenter.defaultCenter()
    .addObserver(self, selector: #selector(FeedViewController.gotEvents(_:)), name:"feedEvents", object: nil)
```

Figure 10 - Adding an observer

```
NSNotificationCenter.defaultCenter().postNotificationName("feedEvents", object: nil, userInfo: nil)
```

Figure 11 - Posting a notification

```
func gotEvents(notification: NSNotification) {
    NSNotificationCenter.defaultCenter().removeObserver(self)
    // some code ..
}
```

Figure 12 - The code that will be triggered

The notification above illustrates two classes, the class of the observer that contains the figures 8 and 10, and the class of the publisher, which includes the figure 9. Once the observed has been added and the notification is sent, the specified method of the observer class is triggered.

4.3.3 Persistence in iOS

Other important aspect of the Momentum iOS app was its persistence. If relevant data can be stored in the user's device, plenty of network data transfer is avoided as well as a potentially significant amount of time is saved, which results in a better user experience. Being able to store relevant data on the device revealed to be crucial.

In order to implement persistence there were several options to choose from, and the most popular for the iOS are the following:

- Property Lists. Colloquially referred to as *plists* [46], a property list is a structured data representation used in iOS as a convenient way to store, organize and access standard types of data. They are essentially an abstraction for representing simple hierarchies of data. The data represented in a plist can be either a primitive type or a container of values. A primitive type of data means that the item is a string, a number, binary data, a date or a Boolean value. Alternatively, containers of data can be either dictionaries or arrays of primitive items.

- User Defaults. User Defaults [47] is a property list that is used to store simple data, such as settings to determine the application's default state at startup

or its behavior by default. This option is meant to store small pieces of information like single values, user preferences or settings. Those objects must be a property list, i.e., instances of type NSData, NSString, NSNumber, NSDate, NSArray or NSDictionary. A defaults database is created automatically for each user and can store, e.g., user's high score.

- SQLite. The most popular database engine in the world [48]. It is open source, cross-platform and consists in a simple transactional SQL database engine. This relational database is small and light, requires no previous configuration and is safe against concurrent accesses.

- CoreData. CoreData is an object relational mapping framework created by Apple. Unlike SQLite, CoreData focuses more on objects instead of traditional table database methods. CoreData [49] fetches records faster than SQLite, but it also uses more memory and storage space. This framework allows developers to store and retrieve data in an object-oriented way. The mapping of the objects into database does not require any SQL coding by the programmer, as this procedure is hidden. This solution was introduced in 2009 in the iOS 3.0 version, fact that helped creating a robust community around CoreData. Support is easy to find online, as well as external libraries that were built around this framework. CoreData requires a bit of pre-configuration such as creating the managed object model and building and initializing the CoreData stack, factors that can be avoided by adopting some third party CoreData based libraries.

- Realm. Realm is a cross-platform mobile database especially dedicated to mobile applications. It is extremely simple to implement in an iOS project and does not rely either in CoreData or in SQLite. This proprietary data storage solution was released in July 2014, and there is not a consistent support community yet. Even though, according to Realm's developers, its product provides a faster solution than the competitors.

After a deep study of all the possibilities that the iOS has, and after facing up each competitor's aforementioned advantages and disadvantages, the undergraduate decided for the CoreData. However, instead of the genuine, robust framework created and maintained by Apple, he chose an alternative third party library: CoreStore.

CoreStore is a wrapper library for CoreData, that allows the developer to take advantage of the CoreData model without all its boilerplate code and with Swift's language elegance and type safety. The library is often updated by its author (usually once per week at least), and has a big support community over the Internet, reasons that make CoreStore one of the most recommended CoreData based libraries available to the developers.

4.3.4 Concurrency

In this section it will be explained how does the concurrency work in the iOS environment.

Concurrency is, naturally, one of the most important topics in the process of iOS development. As in any computational resource nowadays, often emerges the need for performing multiple tasks simultaneously. To accomplish that, iOS provides Grand Central Dispatch, a C language based API that offers support for concurrent code execution in iOS.

The most typical procedure of running a piece of code, or task, concurrently using GCD is by sending it a closure. The closure can be seen as passing a function as an argument to another function running in a different thread, i.e., asking the new thread to run a certain task during or after its execution.

GCD executes tasks in two different ways, serially or concurrently. When tasks are executed serially it means that one task only starts when the antecessor task finishes. On the other hand, concurrent tasks may execute at the same time.

Another issue to bear in mind is the difference between synchronous and asynchronous functions. The former only returns at the end of its execution, while the latter returns immediately, continuing its execution on a different thread.

Furthermore, concurrent code can run with or without parallelism. On multi core devices, generally GCD provides parallelism, i.e., different threads executing at the same time. If the device is single cored then instead of parallelism, GCD uses context switch. In this case the one single thread often changes its context, consequently storing and loading back the different contexts. Regarding parallelism, it is not the developer's job to decide how it will be carried out, it is Grand Central Dispatch's responsibility.

The mechanisms used by GCD to execute its tasks are queues. Queues can be serial or concurrent. Serial queues guarantee that its tasks are executed and finished on a FIFO order (first come first served). Oppositely, concurrent queues only guarantee that its tasks start their execution in a FIFO order, i.e., a task can start first than another but finish for last. The outcome of this last scenario uniquely depends on the operating system, and can result, e.g., in race conditions [50].

Natively, the GCD provides 5 different queues to the developer environment:

- The main queue, that is responsible for updating the app's views. Any change on the application's views must be performed in this queue. Also, post notifications should be executed here.

- The user interactive queue. This queue is most suitable for tasks that require very low latency, such as the handling of events or anything that guarantees a good user experience.

- The user initiated queue, which is intended to perform tasks that the user created but can continue running on backgrounds.

- The utility queue. Designed for long tasks, such as involving networking or long computations.

- The background queue. The background queue should be used for, i.e., prefetching or querying data, or tasks that do not require user's interaction in real time.

Any other queue can be easily created with a few lines of code, either serial or concurrent [51].

4.4 User Authentication

Authentication is an important property in almost any service, nowadays. And so it is on Momentum. It is important to keep track of who is using our application, what permissions do they have and which events and momentums do they belong to, for example.

Authentication
In this section the process of authentication will be described. This process is composed by four major steps.

Initially the app sends an HTTP post request to the back-end containing the msisdn [52]. After receiving the msisdn phone number, the server generates an OTP code [53] that subsequently is sent by SMS to the target mobile device. The system is implemented in such a way that the user has 30 seconds to send the OTP code back to the back-end. After this threshold, or in case the code sent by the user was not the correct one, the user must repeat the previous post request, containing the msisdn, in order to receive another code and conclude the login process. After sending the correct OTP code, the app will receive the user id, an access token and a refresh token, encapsulated in a JSON Web Token [54].

The access token is used on each iteration of the communication between the app and the back-end, and has a validity of 24 hours. The refresh token is required so that the access token can be renewed. Every time that a refresh token request is sent the back-end returns two new tokens.

4.5 Docker

Docker [55] is a software container, i.e., a complete filesystem that has everything set to run. This component receives the developer's code, optionally the system libraries or other features that can be installed on a server, wraps it up, compiles and runs. Using Docker ensures that our code will run always the same way, regardless of the environment.

Docker was a solution that the student's supervisor at WIT suggested in order to avoid program's deployment errors and to quicken its process of deployment.

4.6 Mockups

Another important attribute of the internship's architecture was the mockups. They were provided by WIT's design team after the student's submission of the system user stories, and they served as guidelines for the undergraduate's implementation.

The first screen would be reached right after the user authentication. The user opens the app, signs in and will see the list of Events above.

Figure 13 - Mockup Events screen

The mockup placed above shows the Momentum expected output for the available events enumeration. The numbers refer to:

- 1: this screen will have a search field.
- 2: there will be a connection to the user Momentums.
- 4: the final prototype will access the map view.
- 5: there will be a shortcut to the app's settings.

This view contains a list of Events, each with a name, a description, a logo and a background image. When tapping into one of the displayed events, the user moves to the Event view.

Figure 14 - Mockup Event view

The previous mockup shows the screen view of an Event. The numbers refer to:

- 2: add a new content to the Event from the user device's gallery.
- 3: post a photo or video from the device's gallery.

This screen contains the list of contents that belong to this Event. Tapping on each of them will open the content in full-screen mode, and the arrow on top-left takes the user back to the previous screen.

5 IMPLEMENTATION AND EVALUATION

This chapter will describe the code implementation and the flow of development over the year, and the tests that were performed on the prototype.

5.1 First semester

This section describes how the development evolved from the day the internship program started, September 14th of 2015, until the end of the third sprint of development, on January 8th of 2016.

During the first 3 weeks of the internship the basics of iOS systems were learnt, as well as Swift's programming language. In order to accomplish this the student attended to the "Developing iOS 8 apps with Swift" online course by Paul Hegarty from Stanford University. This course contained all the classes recorded at Stanford University, and all its materials in pdf format.

Afterwards, some changes on the initial internship's program occurred but everything was set clear by the middle of October. The following assignment was to perform a deep study over the state of the art. The company provided the necessary tools to fulfill this job, which required having an iOS device. The student downloaded about 20 applications and tested all their features. Some of the applications were not even available on the Portuguese App Store but he managed to find an alternative to try them. The competitor applications were all compared and, during this analysis, the student retained a list of great ideas to put in practice on his prototype. The result of this analysis was shared with the student's supervisor at WIT and, at the end of this stage, the path to follow seemed to be clearer.

The following step was to identify the user stories that would fit in our system.

Several versions of the user stories were drafted, by both the undergraduate and his supervisor at WIT. Additionally, a theme and a t-shirt size were appended to each user story.

As stated in the methodology section, the t-shirt size is an important component of the Agile programming methodology and lets the workers estimate an idea of how long a specific task will take.

The theme of the user stories only helped us to categorize each user story.

After the user stories selection, another very important step took place: the architecture decisions. The project was quite complex, compared to the ones the undergraduate used to be involved in during college, so this time an extra effort was given. A lot of implementations questions arose on the student's mind as it is stated on the architecture chapter. Some of the questions involved concepts that were partially unexplored by the student, and thus raised deep research work.

After facing all these architectural decisions, it was time to start coding. The first development sprints took place.

Later, by the end of the year of 2015, we could say that the development had run remarkably well. The first goal of the internship was to have a functional prototype to present at Barcelona's World Mobile Congress in February 2016, and work has been done in that direction.

By the time this document was written, there were already three development sprints done, following the Agile programming methodology.

5.1.1 First sprint

The first sprint marked the beginning of the development stage. The planning was performed previously and all the tasks were assigned to the product backlog, on the Redmine platform. Thus, the first sprint consisted in the following tasks:

- Implementation of the back-end server using Spring Boot
- Implementation of the PostgreSQL database
- Connect the server with the database
- Create an iOS app
- Use Facebook SDK to login and logout from the app

From the mobile application's point of view, this sprint was quite simple. The first screen of the app contains a login button that the user should tap. Afterwards the application requests a Facebook login. If successful, the user is redirected to Momentum's logged in screen. Momentum would get some data from the user's Facebook account such as email, name or birthday, e.g.. This screen had a logout button that would log the user out and get him back to the former screen, where he would have to repeat the login procedure.

The main challenge had to do essentially with the back-end in Spring Boot, which revealed to be more complex than the Swift app in this introductory step. The

student had almost no experience in Spring Boot or Swift/iOS applications, which helped turning the experience into a very interesting challenge.

5.1.2 Second Sprint

The second sprint started right after the end of the first one. At its end, the student and his supervisor at WIT gathered in order to analyze the past sprint and discuss the next one. For the second sprint, the tasks below enumerated were planned:

- Define the overall schema of the back-end
- Implement the entity for the event and the user
- Manage the relationship between the entities
- Pass the entities between the back-end and the mobile application
- Create API for event creation
- On the app, allow simple events creation

At this point, the major issues had to do with the Spring Boot server developing and debugging. Setting the relationships between the entities and implementing the right annotations was not trivial.

On the mobile application, the logged in screen presented a button that allowed a simple creation of events for testing purposes. Still, in this sprint, the student learned how to pass objects in Swift to the back-end, by using JSON, and the opposite, converting JSON into Swift objects without using external libraries.

5.1.3 Third Sprint

The third sprint started on December's 21st and lasted the usual 10 working days, until January 8th of 2016. This sprint focused more on the mobile application than the previous ones. A document with the iOS application's mockups has been provided by the design team of WIT Software, and the aim of this sprint was to implement those mockups. The following tasks were stated on the project's backlog for the sprint number 3:

- Display the list of events on the iOS app
- Show all the pictures for each event
- Implement the missing entities and APIs on the back-end
- Fill the screen with the colors chosen by the design team

The outcome was slightly different from the expected, but the appreciation of the student's supervisor at WIT and Momentum's Manager was positive.

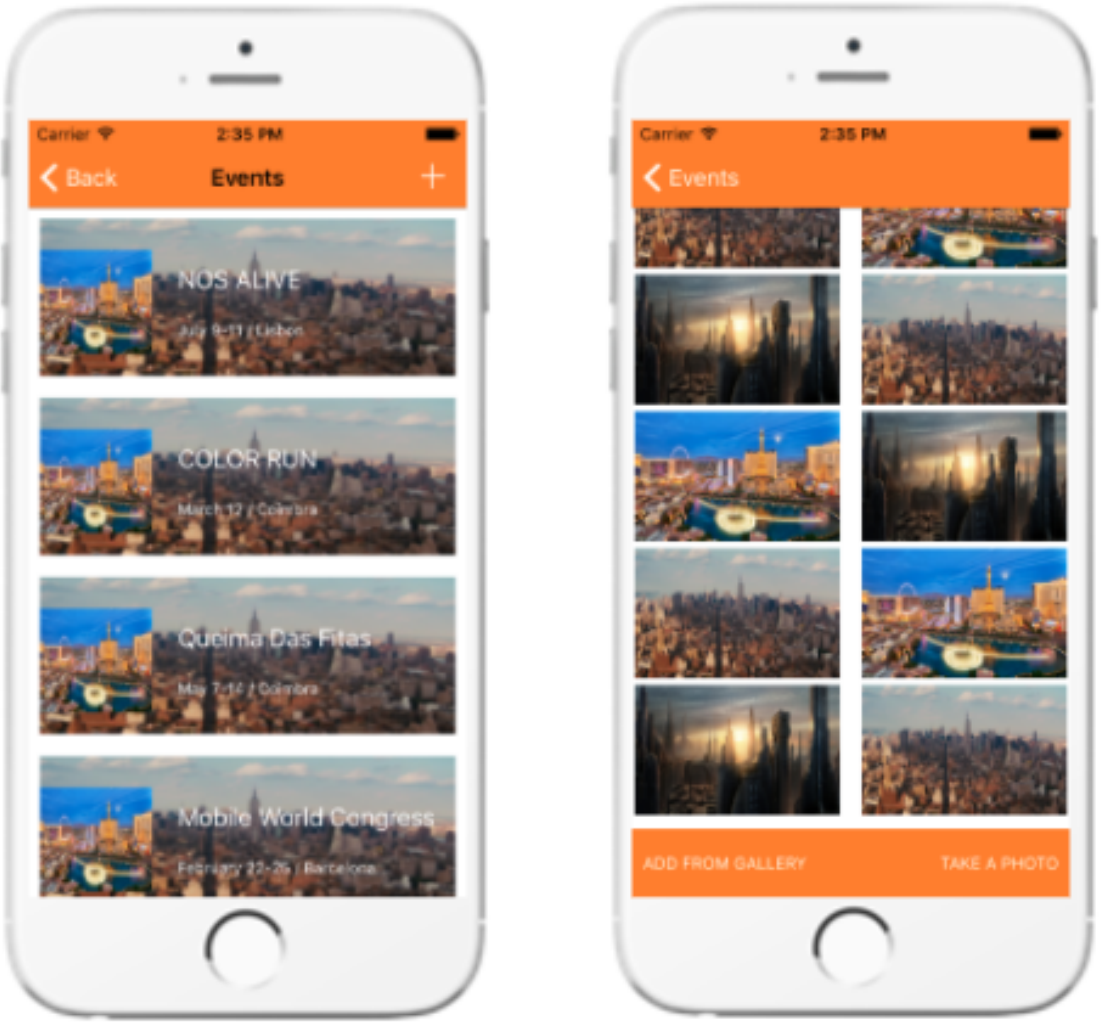


Figure 15 - Momentum Sprint 3 Screenshots

On the one hand, the feature of displaying all the pictures of an event was not fully developed, because the mobile application was not handling properly the post entity, where the picture belonged. So, instead, the application was displaying default photos just to demonstrate how the contents would be displayed on the device's screen.

On the other hand, tasks that were scheduled for future sprints were implemented, such as uploading photos from the device's gallery to the application. This happened in order to test the flow of the photos, sent from the mobile application to the back-end and vice-versa. Since it worked well and the task was useful for the future sprints, the student decided to keep it on this sprint's prototype.

To better handle the JSON sent from the Spring Boot server to the iOS app, the student made some research over some of the most popular Swift libraries, such as Alamofire [56] and Argo [57]. Alamofire simplifies the HTTP networking on Swift, with clearer syntax for each HTTP methods, e.g.. Argo provides an easier handling

of JSON in swift. They were included in the project, once they shown some advantages over the Swift native language.

The back-end had a few changes, once the entities were not implemented by the beginning of the sprint and they were needed in order to accomplish the goals from this sprint. Additionally, the necessary APIs were created to retrieve the list of events, and the contents of each event. The APIs follow the HTTP protocol, and receive mostly get and post requests.

The colors specified by the design team were put in practice, as well of most of the components presented on the document mockups. The feedback the student received from his supervisor at WIT and from Momentum's Manager was positive.

5.2 Second semester

The following section will report the work developed in the second semester of the internship, i.e. from February until August.

After the internship's interim delivery and resulting presentation at the University, the Mobile World Congress took place in Barcelona from February 22nd to 25th. As expected, WIT Software was present and the prototype developed during the first semester of the internship was shown to the fair participants. Even though the undergraduate was not present, feedback was given to him as the fair's participants found the prototype an interesting solution, despite its early state of development. By having aroused curiosity and positive discussions with several MWC's participants, the undergraduate and his supervisor in WIT Software fell confident regarding the path being tracked.

Continuously, the student projected some usability tests in order to receive some feedback regarding the current prototype. To consult them, refer to **Appendix A**.

The outcome of those usability tests had a major importance for the course of the internship.

The student analyzed the results of the usability tests and due to its unexpected outcome, further suggestions were transmitted to his supervisor, David. The student and its supervisor formerly agreed on slightly shifting a few characteristics on the application, and new mockups were requested to WIT's design team.

Having the new User Interface mockups written by the design team, the student's path was to rebuild its mobile prototype from scratch. It was not a hard setback since the major development occurred during the first semester focused on the back-end, and that work was not affected by this change.

The detailed analysis of the sprints performed during the second semester can be found in **Appendix C**.

5.3 Evaluation

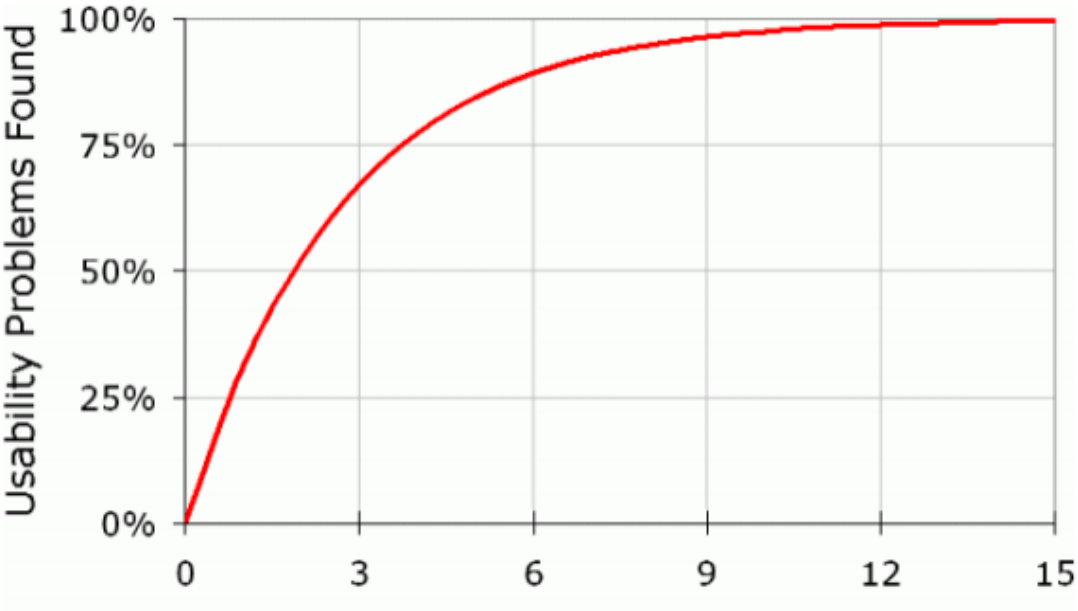
The following section will report the techniques used by the student in order to evaluate the Momentum iOS prototype.

During the internship two different types of tests were run: Usability tests and Functional tests.

5.3.1 Usability tests

Usability tests are an important technique to measure how efficient an application is in order to meet its intended purpose. The main characteristics that influence the usability of an application are the interaction efficiency, its error rates, the learnability, memorability and its subjective satisfaction.

So that the usability tests could be meaningful, the user segmentation would desirably have to be somehow familiar with the iOS system. Furthermore, due to the company's privacy police, Momentum's details should not become public, thus the testers were all performed on WIT Software's employees. The tests involved six different testers, once according to Jim Herbsleb and Jakob Nielsen this value is enough to detect more than 80% of the issues.



Jakob Nielsen's study

Summarily, usability tests let the programmer know how easy its product is to the clients. They consist on the following points:

- Select one use case from the most important features.
- Notify the tester about which use case he is expected to perform, and track its actions.
- Compare the testers results

The full set of usability tests run on the application can be found in **Appendix A.**

5.3.2 Functional tests

Functional tests are a quality assurance process that, basically, describe what the system does.

The full set of functional tests run on the application can be found in **Appendix A.**

5.3.3 Code metrics and complexity

In this section there will be shown a few statistics of the project, so that its complexity could be estimated.

The statistics focused more on the iOS application, since they were the main focus of the internship. The most noteworthy statistics were:

- The last version of the prototype counts with 37 classes of the presentation logic, i.e., directly related to the User Interface. These classes have between 150 and 200 lines of code.

- The prototype has 18 classes that belong to the application logic. These classes have between 75 and 750 lines of code each one.

- The prototype has 8 classes of the Model layer. These classes have all less than 50 lines of code.

- The prototype used 8 different external libraries, e.g. for networking communication, Json parsing, data storing, etc.

- The iOS application has around 9500 lines of code combined.

- Moreover, there is some logic on the project's Storyboard that could not be tracked, since the developing environment does not involve coding.

Below is an example of an instance of a UIViewController that was implemented using the iOS Storyboard.

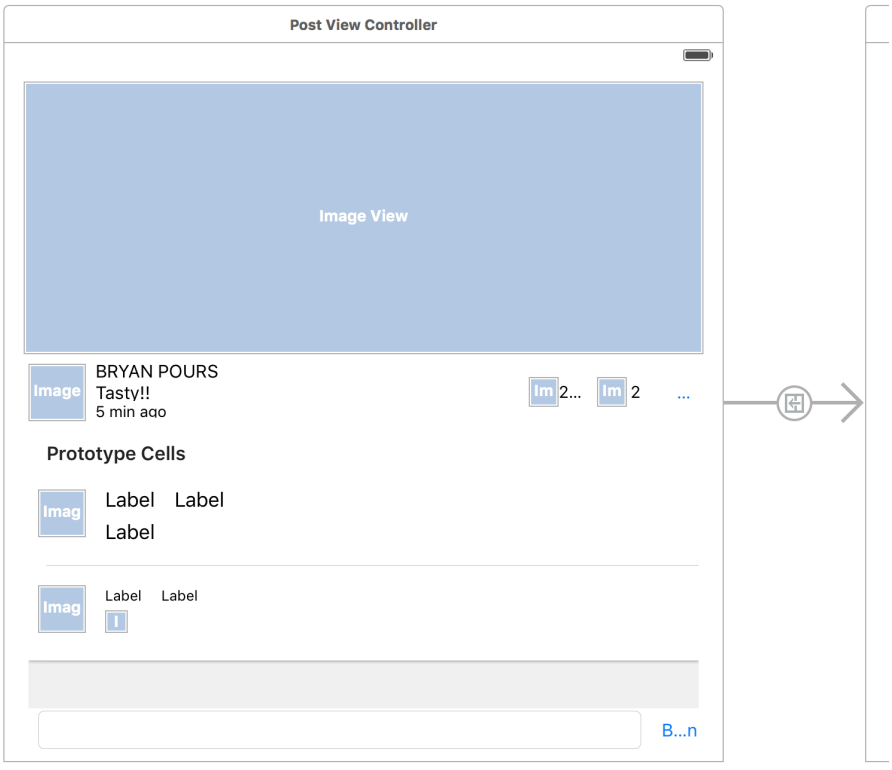


Figure 16 - example of storyboard development

6 CONCLUSION

The final chapter sets the end of the report and of this one-year internship at WIT Software.

What we have today is an iOS application that allows the users to share their favorite moments with their friends and relatives, with a commodity and ease of use that we found uncommon.

Nevertheless, the application is not going to be released in the App Store by the time of writing this report. Doing so requires a team specially dedicated to the application, ready to give support to the product whenever it is the occasion, and at the moment WIT Software cannot proceed in such a scenario.

It is time to present some personal thoughts on my internship.

I chose this internship as my number one option because I always felt interested in mobile software development and throughout the degree at the university I did not have the opportunity to master those skills. By taking this route, I managed to conciliate exploring the mobile software development field with working on a cutting edge software development company, as well as mastering my programming and computer engineering skills.

I found the experience very enriching. I faced and overcame several obstacles throughout the year, and at the end I presented a fully functional prototype, as expected at the beginning of the internship. This was my first time working on a tech company, and I met extremely committed, friendly and professional coworkers at WIT Software's office in Coimbra. I worked with technologies I have never used before, such as iOS, the Swift programming language, Spring Boot for server side development and Git for versioning control, and I found very good working conditions. Meanwhile I had the opportunity to do what I like the most, while evolving both as a professional and as a person, and contributing to the company.

While developing this prototype, I covered most of the stages of developing a real-world product, such as the state of the art, the requirements analysis, the development planning, the development, the tests and the further need to rethink the plan based on the tests results, and the documentation.

During the first semester, the main tasks that I would enhance were studying the iOS environment from the very beginning, starting the development of the back-end and developing a light version of the Momentum mobile application, which was the core of the internship. Afterwards, at the beginning of the second semester, the usability tests that were performed generated a shift on Momentum's user interface, and thus a new version of the mobile application was built. That version is the final one that is presented at the end of the internship.

The skills I acquired during my internship were important for me, for my career and for my company as, on July 2016, I was integrated on a new team at WIT Software for developing software for iOS.

Overall, I am very proud of my internship, it made me evolve a lot.

7 BIBLIOGRAPHY

[1] Pew Research Center, 2004. [Online]. Available: http://www.pewinternet.org/data-trend/mobile/device-ownership/. [Accessed 16 1 2016].

[2] P. R. Center, 2004. [Online]. Available: http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/. [Accessed 16 1 2016].

[3] Statista, [Online]. Available: http://www.statista.com/statistics/494572/smartphone-users-in-central-and-eastern-europe/. [Accessed 14 1 2016].

[4] Statista, [Online]. Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Accessed 20 1 2016].

[5] App Annie, [Online]. Available: https://www.appannie.com/apps/ios/top/?device=iphone. [Accessed 16 1 2016].

[6] Mobile World Congress, 22-25 2 2016. [Online]. Available: http://www.mobileworldcongress.com/.

[7] Facebook, Inc., [Online]. Available: http://momentsapp.com/.

[8] M. Chabasse, A. Sibiril, F. Hobein and M. Spiry. [Online]. Available: http://cometapp.io/.

[9] Deja Mi, Inc., 1 12 2010. [Online]. Available: https://www.wedpics.com.

[10] Cluster Labs, Inc., 2013. [Online]. Available: https://cluster.co.

[11] S. Papafloratos and M. Dempsey, Generic Ventures Limited, 2013. [Online]. Available: https://www.togethera.com.

[12] S. Perez, 11 2 2015. [Online]. Available: http://techcrunch.com/2015/02/11/wedding-app-wedpics-raises-6-5-million-from-shark-tanks-barbara-corcoran-and-others/#.hmodqoe:Emzd.

[13] Cluster Labs, Inc., [Online]. Available: https://tripcast.co/. [Accessed 12 1 2016].

[14] Cluster Labs, Inc., [Online]. Available: https://churchsnaps.com/. [Accessed 12 1 2016].

[15] Cluster Labs, Inc., [Online]. Available: https://dailykiddo.com/. [Accessed 12 1 2016].

[16] Cluster Labs, Inc., [Online]. Available: https://gethomeroom.com/. [Accessed 12 1 2016].

[17] Sarah Perez, Tech Crunch, 22 8 2013. [Online]. Available: http://techcrunch.com/2013/08/22/photo-sharing-app-cluster-snags-instagram-seed-investor-steve-anderson-others-to-lead-1-6m-round-launches-version-1-0/.

[18] M. Zuckerberg, Facebook, 4 2 2004. [Online]. Available: https://www.facebook.com/.

[19] E. Spiegel, B. Murphy and R. Brown, 9 2011. [Online]. Available: https://www.snapchat.com/.

[20] D. Houston and A. Ferdowsi, 6 2007. [Online]. Available: https://www.dropbox.com/.

[21] Google, 28 5 2015. [Online]. Available: https://photos.google.com/.

[22] K. Systrom and M. Krieger, 10 2010. [Online]. Available: https://www.instagram.com/.

[23] Ludircorp, 2 2004. [Online]. Available: https://www.flickr.com/.

[24] F. Van Allen, Techlicious, 8 9 2015. [Online]. Available: http://www.techlicious.com/blog/top-10-apple-iphone-and-ipad-apps-of-all-time-most-downloaded/. [Accessed 17 1 2016].

[25] Statista, Statista, [Online]. Available: http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. [Accessed 20 1 2016].

[26] C. Smith, DMR Digital Statistics, 15 12 2015. [Online]. Available: http://expandedramblings.com/index.php/snapchat-statistics/. [Accessed 19 1 2016].

[27] App Annie, 16 1 2016. [Online]. Available: https://www.appannie.com/apps/ios/top/?device=iphone. [Accessed 16 1 2016].

[28] S. Aslam, 7 10 2015. [Online]. Available: http://www.omnicoreagency.com/snapchat-statistics/. [Accessed 16 1 2016].

[29] C. Smith, DMR Digital Statistics, 7 11 2015. [Online]. Available: http://expandedramblings.com/index.php/dropbox-statistics/. [Accessed 16 1 2016].

[30] Dropbox, Inc, Dropbox, [Online]. Available: https://carousel.dropbox.com/. [Accessed 18 1 2016].

[31] Amazon Web Services, Inc., [Online]. Available: https://aws.amazon.com/s3/. [Accessed 14 1 2016].

[32] Basho Technologies, Inc., [Online]. Available: http://basho.com/products/riak-s2/. [Accessed 14 1 2016].

[33] MongoDB, Inc., 2009. [Online]. Available: https://www.mongodb.org/. [Accessed 15 1 2016].

[34] The PostgreSQL Global Development Group, 1996. [Online]. Available:

4] http://www.postgresql.org/. [Accessed 14 1 2016].

[3 5] Apple, Inc., 2 6 2014. [Online]. Available: http://www.apple.com/swift/.

[3 6] Pivotal Software, Inc., [Online]. Available: http://projects.spring.io/spring-boot/. [Accessed 14 1 2016].

[3 7] Google, [Online]. Available: https://www.google.com/intl/pt-PT/drive/.

[3 8] Pivotal Software, Inc., [Online]. Available: http://start.spring.io/. [Accessed 14 1 2016].

[3 9] The Apache Software Foundation, 2002. [Online]. Available: https://maven.apache.org/. [Accessed 14 1 2016].

[4 0] Java Community Process, 2006. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html. [Accessed 14 1 2016].

[4 1] Red Hat, [Online]. Available: http://hibernate.org/orm/. [Accessed 14 1 2016].

[4 2] Apple, Inc., 21 10 2015. [Online]. Available: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html. [Accessed 17 1 2016].

[4 3] A. Furrow, "objc.io/issues/13-architecture/mvvm/," 1 06 2014. [Online]. Available: https://www.objc.io/issues/13-architecture/mvvm/. [Accessed 30 7 2016].

[4 4] V. Ngo, 9 12 2014. [Online]. Available: https://www.raywenderlich.com/86477/introducing-ios-design-patterns-in-swift-part-1.

[4 5] Wikipedia, 23 5 2016. [Online]. Available: https://en.wikipedia.org/wiki/Software_design_pattern.

[4 6] A. Inc., 24 3 2010. [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/PropertyLists/AboutPropertyLists/AboutPropertyLists.html.

[4 7] A. Inc., 17 9 2014. [Online]. Available: https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/.

[4 8] A. Prusak, 19 2 2016. [Online]. Available: https://dzone.com/articles/ios-databases-sqllite-vs-core-data-vs-realm .

[4 9] Z. Tamim, 15 12 2012. [Online]. Available: http://www.appcoda.com/introduction-to-core-data/.

[5 0] Wikipedia, "en.wikipedia.org/," 21 05 2016. [Online]. Available: https://en.wikipedia.org/wiki/Race_condition. [Accessed 25 05 2016].

[5 1] B. Ruud, "raywenderlich.com," 7 1 2015. [Online]. Available: https://www.raywenderlich.com/79149/grand-central-dispatch-tutorial-swift-part-1 . [Accessed 11 3 2016].

[52] Msisdn, "http://www.msisdn.org/," msisdn, [Online]. Available: http://www.msisdn.org/. [Accessed 29 04 2016].

[53] Wikipedia, "en.wikipedia.org/," [Online]. Available: https://en.wikipedia.org/wiki/One-time_password.

[54] Auth0, "https://jwt.io/," [Online]. Available: https://jwt.io/.

[55] Docker, Inc., 13 3 2013. [Online]. Available: https://www.docker.com/. [Accessed 14 1 2016].

[56] Alamofire Software Foundation, [Online]. Available: https://github.com/Alamofire/Alamofire. [Accessed 14 1 2016].

[57] thoughtbot, inc., [Online]. Available: https://github.com/thoughtbot/Argo. [Accessed 14 1 2016].

[58] Go-Globe, [Online]. Available: http://www.go-globe.hk/blog/mobile-apps-numbers/. [Accessed 18 1 2016].

[59] P. Webb and D. Syer, Pivotal Software, Inc., 6 8 2013. [Online]. Available: https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone. [Accessed 14 1 2016].

[60] SwiftyJSON, [Online]. Available: https://github.com/SwiftyJSON/SwiftyJSON. [Accessed 14 1 2016].

[61] IETF OAuth WG, 2006. [Online]. Available: http://oauth.net/2/.

[62] J. Nielsen, 3 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.