uc3m | Universidad Carlos III de Madrid

University Degree in Informatics Engineering

Academic Year 2018-2019

Bachelor Thesis

"Development of an application for organising cultural meetings"

Chengyi Fu

Tutor/s

Jose Antonio Iglesias Martínez

Leganés, September 23rd, 2019

[Include this code in case you want your Bachelor Thesis published in Open Access University Repository]

SUMMARY

For the purpose of this bachelor thesis, it has been developed an application in the Android platform to offer the users an easier way to assist to events and meet new people, in the world of the culture.

The application consists of a single module, which allows the users to share an event, join a group of users to meet new people and assist together to the event, and to share their experience about an event.

A study of the culture in Spain, and more specifically in Madrid, will be included in the document, stating the reasons behind the choice of developing the application mentioned.

There will be a section that explains the studies done about the mobile devices, the mobile operating systems, comparing the most relevant ones currently and reasoning the chosen system for the development.

The developing process of the application will also be described in the document, alongside the testing results and the conclusions acquired.

INDEX

FIGURES INDEX

TABLE INDEX

CHAPTER 1: INTRODUCTION

In this chapter, it will be collected a brief introduction of the project and the objectives sought by doing it, as well as the description of the next chapters that are going to be addressed in the document.

1.1 Introduction

It is known that technologies of information are in constant evolution and, nowadays, it is possible to do almost everything online, through a website or using an application, but all accessible by just using a smartphone.

The purpose of the application developed for this project is to get the people to assist to more cultural events, by providing them a tool to easier this task, allowing them to have, on a device within reach at any moment, a collection of all the cultural events around, and the possibility to meet new people who share their interests.

To develop an application that fulfils the interest of the widest range of people, it is necessary to do a study on the evolution of culture and the actual state of it, the evolution of the mobile systems and the ones available nowadays, and the different technologies available to develop the application.

1.2 Objectives

The main objective of the project is to get the knowledge to be able to develop a full workable application using Android Studio, as well as the connection with a remote database to store the data. As personal objectives, it has been set to:

- Learn about the Android Studio working environment and the possibilities it offers to a developer.
- Learn about the working of a NoSQL database.
- To know more about the state of the culture in Spain.

About the application to be developed for the project, it has been set to provide the next functionalities:

- An authentication system, using the basic email/password method.

- Allow the users to share an event, giving some details and information about it.
- A screen to show all the active events that has been shared by the users.
- To create groups inside an event, to gather people for going together to the event, and to be able to join these groups.
- A screen where the users can see the list of groups in a specific shared event.
- A section where it is possible to check all the members of a specific group.
- A comment section, where the users can ask and share details and personal experiences about the event.
- Access to user profiles, containing the information that identifies them as unique user in the application, as well as the possibility to add more information about themselves.

1.3 Regulatory framework

When developing a mobile application, it must be considered the legal aspects, and the obligation to assume the responsibilities when working with user personal data. It is necessary to inform the users, fully and clearly, about the privacy policy of the application [1].

On the 25th of May of 2018, it was implemented the new GDPR[1] to ensure a better protection of the personal data. It affects directly any storing, processing, accessing, transferring and sharing data register about an individual, and it affects any organization in the world that processes personal data of people in the European Union [2].

- **Personal data**

  It must explicitly ask the user for their consent to use their personal data when using the application, and the possibility to share them to improve the products and services offered.

  It is specified that in case that it is not given, it will not be possible to offer all the products and services.

- **Non-personal data**

[1] GDPR (General Data Protection Regulation) is a regulation in EU law on data protection and privacy for all individual citizens of the European Union (EU) and the European Economic Area (EEA)

It must inform the user about the non-personal data that can be collected and processed, aiming to learn about the customers and improve the products, services and advertisements.

- **Protection of the personal data**

It must ensure the protection of the users' personal data by encrypting when they are being transferred and, once saved, with physical security methods.

The programmer is responsible of personal data of the users when they are being exposed in the application, where it can be publicly seen by other users.

The personal data will be saved only for the time it is needed to provide the products and services, unless it has been stated otherwise by the law.

- **ARCO rights**

The users have the right to access, cancel, oppose and rectify free of charge the information about their personal data.

With the GDPR, it is added the right to limit the data treatment and portability.

- **Children and education**

It is necessary to stablish additional methods to keep the privacy and security of minors. And to ask permission from their parents if they are not the minimum age

required by the legislation (with the new GDPR, the minimum age has been changed from 14 to 13 years old).

If it has been collected information about a minor younger than 13 years old, without the permission of their parents, they must be deleted immediately.

- **Location services**

To offer location services to the user, it will be necessary to collect, use, and share data about their exact locations.

Unless explicitly asked, and given permission, this data will the collected anonymously, using them with the only aim of improving the services and products.

- **Websites and third-party services**

The application may contain links to external websites, products and services.

The collection of data from third parties, will be guided by their respective privacy policies.

- **Information and cookies**

It is necessary to inform the user about the regulated legal aspects, showing information about the creator/s and the people behind the application.

It must warn the users about the cookies, which must also be accepted.

In this application it does not access directly any data store in the device, it does not use cookies and does not have any advertisement.

It will be explained to the user about the possible uses of their name, email and telephone number, asked when registering in the application. And that some of them may be exposed publicly to other users of the application.

It will be provided to the user the possibility to hide the information he is not willing to show publicly to all the users.

1.4 Socio-economic aspects

The world of culture in Spain has suffered very much from the financial crisis of 2008, being still not recovered from it nowadays.

Some years later from the crisis, the government decided to raise the taxes of the cultural activities, from an 8% to 21%, the biggest increase in the history. This, added to the financial crisis, caused a great decrease in the consumption of cultural activities, as it is not considered as a good of first need, so the people tend to reduce their consumption when having financial difficulties.

One of the main objectives of the application developed is to make the people to consume more cultural activities, and another, to allow them to meet new friends by sharing the same activities.

There are many people that does not have many friends to go this kind of events, or many of them that cannot assist to these activities often due to their financial situation. With the application, the users can join bigger groups where they can meet new friends to share the experience together, and at the same time, opting for group discounts from the business when going in a big group.

This will increase the sales of the tickets, allowing the businesses to offer different kind of the activities and at lower prices. And at the same time, the potential users that like the culture activities will increase and they will try to go to more events and bring more friends with them.

If the use of the application increases and more and more people consumes cultural activities, the economy of the culture will be able to rise once again and recover its previous state, before the financial crisis started.

1.5 Structure of the document

In this section, there will be a brief description of the content of each of the chapters, to give a summarized view of the entire document:

- Chapter 1. Introduction: A brief introduction to project and the objectives sought are described, including also a summary of the content of the document.
- Chapter 2. State of art: It is exposed the studies done about the mobiles devices and their operating systems and the evolution of the consumption of culture in Spain and its current state.
- Chapter 3. Designing of the technical solution: It is explained the technologies used to develop the application, and the use and implementation of each of the functionalities.
- Chapter 4. Tests and results: It is shown the tests done to evaluate the application and the results obtained.
- Chapter 5. Project management: It contains the planification of the project, shown in a Gantt chart, and the estimated budget for the development of the project.
- Chapter 6. Conclusion and future work: It is explained the conclusion reached after finishing the project, and the improvements that can be done to the application to get closer to the objective exposed.

CHAPTER 2: STATE OF ART

In this chapter, there will be a study of the mobile devices and their operating systems, and an analysis of the consumption of culture in Spain.

2.1 Mobile devices and their operating system

2.1.1 Mobile devices history

Mobile device is a general term for any handheld computer of smartphone. Their main characteristic, as its names says, it is their mobility, their reduced size and the ease to transport them.

Nowadays, all mobile devices have similar characteristics:

- Wi-Fi or cellular access to the internet
- A battery that powers the device for several hours
- A physical or onscreen keyboard for entering information
- Touch-screen interface in almost all cases
- The ability to download data from internet
- Wireless operation

But it has not always been like this, the first mobile phones were not really mobiles phones at all. They were two-way radios that allowed people like taxi drivers and emergency services to communicate.

The wireless phones were first created for military use. During the First World War, the German military test them on military trains running between Berlin and Zossen. Later, in 1924, wireless phones were tested on trains running between Berlin and Hamburg.

Starting in 1940, hand-held radio receivers had been widely available, opening communications in battlefields around the world.

Motorola were the first company to mass produce the first handheld mobile phone in 1973, referred to as 0G mobile phones, Zero Generation mobile phones. Most phones today rely on 3G and 4G mobiles technology, being seen more recently also 5G mobile technology.

1G refers to the first automated analogue cellular networks around the world, it was first deployed in Tokyo in 1979 and would spread throughout the of Japan in 1981. Northern European countries received it that same year.

The DynaTAC mobile phone, world's first cell phone, was launched on the 1G network on Match 6th, 1983. The network still suffered from major security issues, but the people did not care about the security at that point yet. The phone was priced around $4000 and it lasted 30 minutes of talk before dying [3].

The second generation of the cellular networks, 2G, emerged in the 1990s, along with a new generation of mobile devices. European and American networks started to split apart and compete against each other, reigning GSM standard in Europe, and CDMA in the United States.

Both systems used digital transmission technologies, improving the security and networking speed. It allowed also basic SMS communication.

3G networks first rolled out around the world in 2001, launched in Tokyo. The main advantage compared to 2G was that 3G used packet switching instead of circuit switching to transmit data, allowing faster data transmission speeds. This improvement opened the door for media streaming over mobile networks.

By 2007, there were 295 million users using 3G around the world, but it was only 9% of the total number of mobile users.

By the end of the first decade of the 21st century, 4G networks was launched, improving the data optimization and promising speeds up to 10 times faster than the existing 3G technologies. Two different 4G technologies were developed, including the WiMAX standard and the more popular LTE standard.

Nowadays, at the end of the second decade of the 21st century, it has been released the 5G network, restricted for a few smartphones for the moment, but offering speeds up above 1 Gbps.

2.1.2 Mobile operating systems

A mobile operating system (or mobile OS) is an operating system for phones, tablets, smartwatches, or other mobile devices. It combines features of a personal computer operating system with other features useful for mobile or handheld use; usually including, and most of the following considered essential in modern mobile systems; a wireless inbuilt modem and SIM tray for telephony and data connection, a touchscreen, cellular, Bluetooth, Wi-Fi Protected Access, Wi-Fi, Global Positioning System (GPS)

mobile navigation, video- and single-frame picture cameras, speech recognition, voice recorder, music player, near field communication, and infrared blaster.

Among all the operating systems, the most relevant ones, and thus, the ones that has the biggest market share, are Android and iOS, which are detailed next.

- Android: it is a operating system based on Linux. It was created by Android Inc, found in 2003 and acquired by Google Inc in 2005, and launched in 2007 to the market.

  Android is an open and free software, allowing all the manufacturers to use it without acquiring a license. Besides, the manufacturers can modify them as wished, personalising them and adjusting them to what they consider it is best for the consumer.

  Android application are, in general, developed in Java programming language, but it can also be developed using C or C++. Nowadays, there is a new emerging language, Kotlin.
- iOS: it is a operating system property of Apple Inc. It was developed in 2006 to be launched together with their first iPhone in the next year.

  iOS is a proprietary code software platform, being used only by their own brand devices.

  The applications are developed using the Objective-C programming language and using the XCode developing environment.

2.1.3 Why Android?

To develop the application for the project, it has been chosen to use the Android platform. The main reason is simple: a much wider range of users have access to an Android system device. In the next table there is a comparative of the aspects between Android and iOS, explaining later the others reason for why it has been chosen to use Android:

Platform	**Android**	**iOS**
Environment	Android Studio	XCode
Market Share [4]	78.35% (Spain, August 2019) 76.23% (World, August 2019)	21.36% (Spain, August 2019) 22.17% (World, August 2019)

Costs	License of 25 euros/year	License of 100 euros/year
Code	Open source	Proprietary
Security	Easier to find vulnerabilities, as it is open-source code	Harder to find vulnerabilities

Table 1: Comparative Android vs iOS

As already mentioned before, it has been chosen to develop an Android application because it has a much bigger market share, the cost of keeping it on the store is four times lower and it can be developed using Java programming language, a widely known language as it is one of the basic programming language that is taught.

2.2 Consumption of culture in Spain

It has been made a study and analysis about the consumption of cultural activities in Spain, among the different aspects, it has been focused on the relation between financial aspect and between the level of studies.

2.2.4 Financial analysis

The level of consumption of the culture in Spain has suffered since the global financial crisis of 2008, and the posterior increase of the taxes of the culture from 8% to 21% in 2012 [5], being the highest rise in the cultural sector in the history of Spain (*Figure 1*).

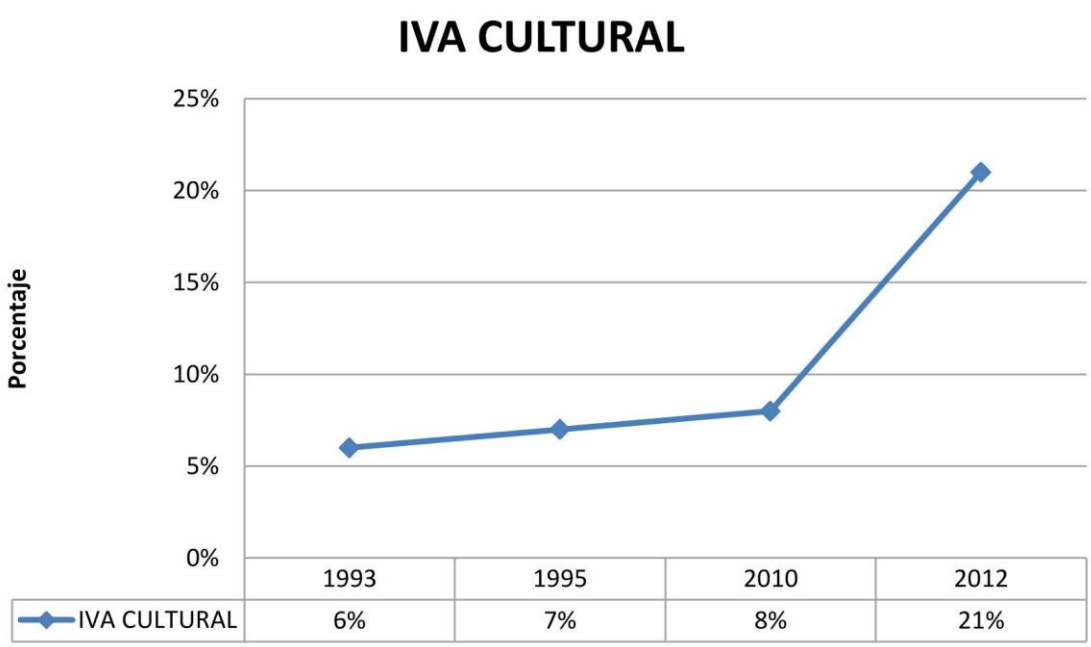


Figure 1: Culture taxes in Spain

Not all the activities inside the culture has been equally affected. Some has kept the same (libraries, museums, books…) but others, such as live shows, has been affected the most, explaining the decrease in the next months, the third quadrimester of 2012, of about 55% of consumers and about 61% in the net incomes of the activities [6].

The culture is one of the most precious good any society can have, it is what differentiates one society from another, and what identifies them. But the problem is that it is not considered as a first necessity, so the people tends to reduce the consumption of it when having financial difficulties.

In 2014, 2017 and 2018, the government has decreased the taxes on art [7], lives shows [8] and the cinema [9], respectively, from 21% to 10%, but these reductions do not have to affect directly the consumers. Since many businesses have assumed the increase of the taxes in 2012, without applying them to the final consumer, the prices maybe be kept the same.

To study the effects of the previous reductions, it has been consulted the annual directory of cultural statistics [10] from 2007-2017, to see the evolution of the consumption of culture, focusing on the cinema, shows, concerts, theatres…

Shows (Cinema, theatre, shows, concerts, dancing…)	Absolut values (Millions)	Average spending per family	Average spending per person
2007	€ 2,065.80	€ 126.90	€ 46.30
2008	€ 1,846.30	€ 110.30	€ 40.70
2009	€ 1,863.80	€ 109.20	€ 40.80
2010	€ 1,857.50	€ 108.20	€ 40.50
2011	€ 1,654.10	€ 95.40	€ 36.00
2012	€ 1,512.50	€ 83.60	€ 32.70
2013	€ 1,416.50	€ 77.80	€ 30.70
2014	€ 1,532.20	€ 83.70	€ 33.30
2015	€ 1,519.30	€ 82.70	€ 33.10
2016	€ 1,751.80	€ 95.00	€ 38.10
2017	€ 1,678.60	€ 90.70	€ 36.40

Figure 2: Culture consumption from 2007-17

From the table above, it is possible to see that starting from the year of the crisis (2018), there has been a decrease of around 32% on the total income per year until 2013, the year in which the recession of the crisis reached its peak, having the lowest value, but starting to grow again from then. It is said that the financial crisis reached an end on 2014 [11], because it was starting to recover, but even nowadays, it has reached the values before the crisis.

The reduction of the taxes on the live shows has been applied on 2017, but on the contrary, the values of table shows that the income has been decreased from 2016. It has been mentioned before that the reduction on the taxes does not affect the prices of the tickets directly, and it might be the reason that there was not an increase of the consumption.

2.2.5 Level of studies

Other aspects that affects the consumption of culture is the level of studies, the higher the more consumption.

In a study done by “La Caixa” shows that the education is most important factor in the consumption of culture. Directly, the higher the education, the more interest for the culture. And indirectly, the higher the education means also higher incomes, resulting in a higher consumption of culture [12]. In the next graphic (*Figure 3*) it is shown the

relation between the income and the annual assistance to cultural activities, differentiated by the level of education:

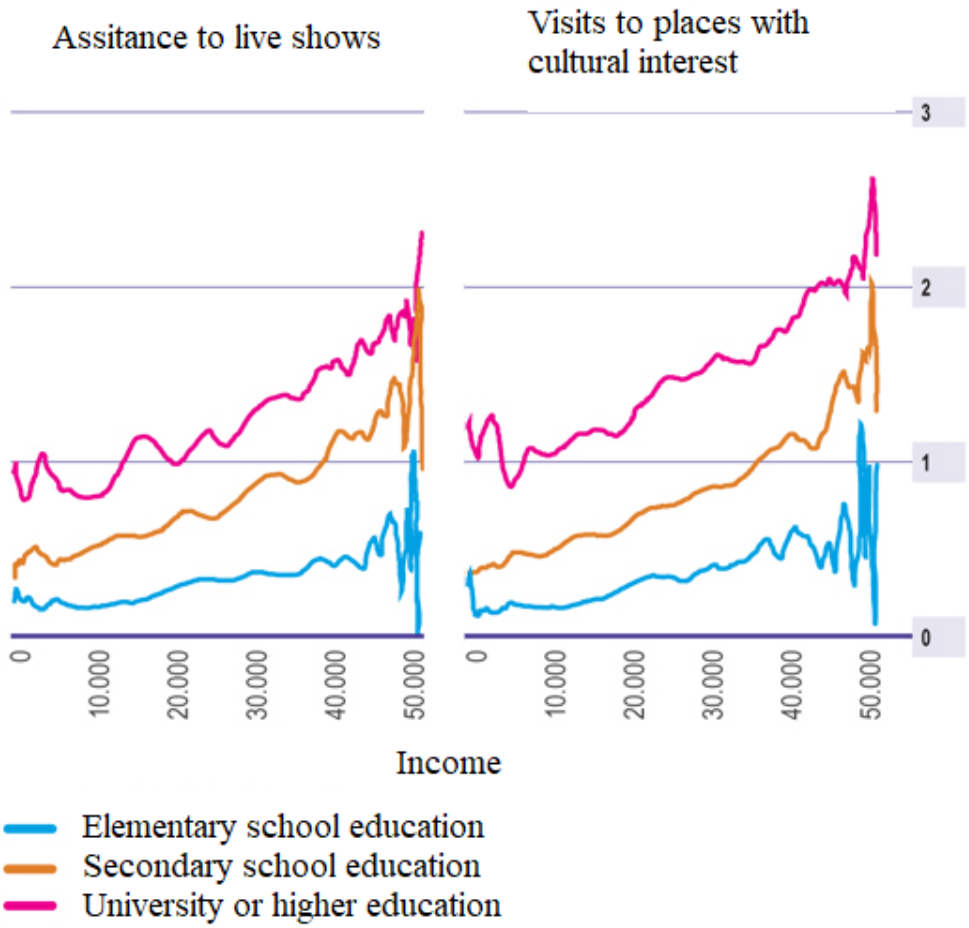


Figure 3: Annual assistance by income and level of studies

From the graphics, it is possible to see that the assistance increases as the level of studies increases, for any level of income. Likewise, the higher the level of income relates to an increase participation in activities, being strongly influenced on higher education levels.

For the people that does not assist to cultural activities, they have been asked about their possible reason, shown in the next graphics:

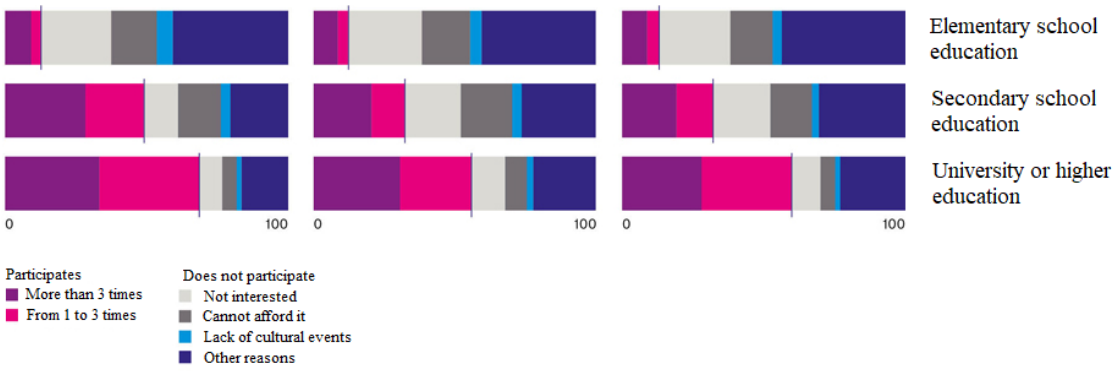


Table 2: Non-assistance reasons

Most of the answers has been "other reasons", so it is not possible to know what is the main reason they do not assist to more cultural activities. From the rest of answers, most of the people are not interested or cannot afford the activities. The pattern about the education can also be seen in these graphics, the lower the education the more people that does not participate, and the higher the education the more people that does participate in cultural activities.

From the results of this researches, concluding that one of the main reasons is still financial problems, but also the interest in the culture, it has been decided to develop an application to help them in these aspects, allowing them to get possible discounts on the prices when assisting in groups (and future agreements with the event companies), and the possibility to meet new people alongside their friends, giving them one more reason and encouraging them to assist.

CHAPTER 3: ANALYSIS AND DESIGN

In this chapter it will be explained the analysis of the solution to be implemented and the design of the system. First, it will be exposed the functional and non-functional requirements of the system. Then, the use cases, explaining them one by one in tables and a global diagram to have an overview of how everything is connected. And lastly, the architecture of the system.

3.1 Requirements

The system must comply several requirements, classified in two categories:

- Functional requirements: those that defines the functionalities of the system.
- Non-functional requirements: those that defines the characteristics of the system, that is, how the system should work

All the requirements are shown, one by one, using the following tables:

3.1.1 Functional requirements

ID	**FR-01**
Description	There must an activity that register the user into the system, with an email and a password, and some additional information
Dependencies	-

Table 3: FR-01

ID	**FR-02**
Description	There must an activity that authenticates a user in the system, using an email and password.
Dependencies	FR-01

Table 4: FR-02

ID	**FR-03**

Description	There must be an activity that shows a list of all the active events, ordered by the starting date of the event, using the designed layout for an event item, with their corresponding data.
Dependencies	FR-02

Table 5: FR-03

ID	**FR-04**
Description	There must be an activity that allows the user to create a new event, accessible from the main activity, and checking that the data filled is valid.
Dependencies	FR-03

Table 6: FR-04

ID	**FR-05**
Description	When clicking on an event from the list, the user must be taken to a new activity, containing detailed information about the event.
Dependencies	FR-03

Table 7: FR-05

ID	**FR-06**
Description	There must be an activity that shows the list of groups that has been created in an event, ordered by the date of the group, using the corresponding layout for a group item, and each of them showing its information.
Dependencies	FR-05

Table 8: FR-06

ID	**FR-07**
Description	There must be a button in each of the group items that allows the user to join/unjoin the group.
Dependencies	FR-06

Table 9: FR-07

ID	FR-08
Description	When clicking on a group item from the list, the user must be taken to a new activity, containing detailed information about the group.
Dependencies	FR-06

Table 10: FR-08

ID	FR-09
Description	The information of an event must be editable, from the detail activity of the event.
Dependencies	FR-05

Table 11: FR-09

ID	FR-10
Description	There must be an activity that shows the list of the users that has joined a group, using the designed layout for a user item, with their corresponding data.
Dependencies	FR-08

Table 12: FR-10

ID	FR-11
Description	The information of a group must be editable, from the detail activity of the group.
Dependencies	FR-08

Table 13: FR-11

ID	FR-12
Description	There must be an activity that shows a commenting section belonging to a group, which allows the users to communicate within the application.
Dependencies	FR-08

Table 14: FR-12

ID	FR-13

Description	The list of comments of the group must the shown in the designed comment section, ordered by the date in ascending order. Each of the comment item must use the layout designed.
Dependencies	FR-02

Table 15: FR-13

ID	**FR-14**
Description	A comment must be removable by the user that has created it.
Dependencies	FR-14

Table 16: FR-14

ID	**FR-15**
Description	There must be an activity that shows the profile of the user
Dependencies	-

Table 17: FR-15

ID	**FR-16**
Description	The profile of a user must be editable, from the activity of the profile.
Dependencies	FR-15

Table 18: FR-16

ID	**FR-17**
Description	There must be an activity that shows the list of events the user has created, and the ones the user has joined a group from it.
Dependencies	FR-04, FR-07

Table 19: FR-17

ID	**FR-18**
Description	The user must be able to log out from the application

Dependencies	FR-02

Table 20: FR-18

ID	**FR-19**
Description	The user must be able to see another user's profile by clicking on the user item from the list, but they must not be able to edit them.
Dependencies	FR-10, FR-15

Table 21: FR-19

ID	**FR-20**
Description	The user must be able to create a new group and add it to the list of groups of the event.
Dependencies	FR-06

Table 22: FR-20

ID	**FR-21**
Description	The expired events must be hidden from the users, except for the creator, and the users that has joined any group of it.
Dependencies	FR-03

Table 23: FR-21

ID	**FR-22**
Description	The expired groups must be hidden from the users, except for the ones that has joined it.
Dependencies	FR-06

Table 24: FR-22

ID	**FR-23**
Description	The user must be able to load pictures from their phone gallery, when choosing the event picture or their profile picture.

Dependencies	-

Table 25: FR-23

ID	**FR-24**
Description	The user must be able to locate, using an external map application, the address of the event, from within the application.
Dependencies	FR-05

Table 26: FR-24

ID	**FR-25**
Description	The user must be able to refresh the list of events in the activity, for showing the new events that has been added.
Dependencies	FR-03

Table 27: FR-25

ID	**FR-26**
Description	There must be an activity where it shows the events the user has created, and the events he has joined a group of it.
Dependencies	FR-04, FR-07

3.1.2 Non-Functional requirements

ID	**NFR-01**
Description	The loading of the list of events, when the user enters the application, must be fast.
Dependencies	-

Table 28: NFR-01

ID	**NFR-02**
Description	The new events added by the user, must be shown at the bottom of the list at first.

Dependencies	-

Table 29: NFR-02

ID	**NFR-03**
Description	A user must be able to edit only his own profile.
Dependencies	-

Table 30: NFR-03

ID	**NFR-04**
Description	An event must be edited only by its creator.
Dependencies	-

Table 31: NFR-04

ID	**NFR-05**
Description	A group must be edited only by its creator.
Dependencies	-

Table 32: NFR-05

ID	**NFR-06**
Description	A new group added by the user to an event, must be shown immediately in the list after being created.
Dependencies	-

Table 33: NFR-06

ID	**NFR-07**
Description	A new comment must be shown to the users immediately, without the need to refresh the activity.
Dependencies	-

Table 34: NFR-07

ID	NFR-08
Description	The application must have an attractive and intuitive design.
Dependencies	-

Table 35: NFR-08

ID	NFR-09
Description	A user must be able to stay logged in the application, until he explicitly uses the function to log out.
Dependencies	-

Table 36: NFR-09

ID	NFR-10
Description	A user cannot have the same username as another user existing in the database
Dependencies	-

Table 37: NFR-10

3.2 Use cases

In this section, there will be a diagram with all the use cases of the application and the relation with the actors, and then, a more detailed description of them using tables.

3.2.3 Diagram

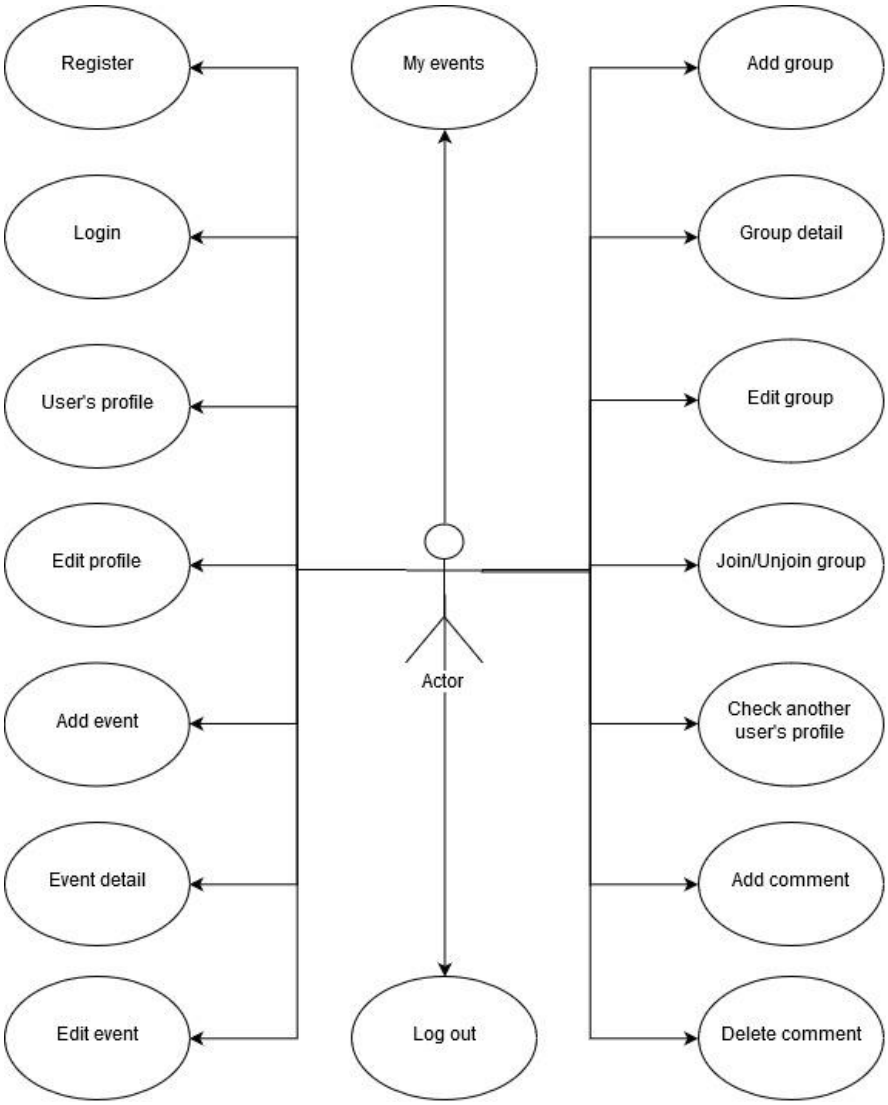


Figure 4: Use cases diagram

3.2.4 Table description

ID	UC-01
Name	Login
Description	The user authenticates in the application

Actors	User
Pre-conditions	The user is not logged in
Post-conditions	The user is authenticated
Success	The list event is loaded from the database and the user is taken to a new activity
Fail	The user stays in the same activity and a login error message is displayed

Table 38: UC-01

ID	**UC-02**
Name	Register
Description	The user creates a new user in the application
Actors	User
Pre-conditions	The user is not registered in the application's database The user has not logged in
Post-conditions	The user is authenticated
Success	A new user is created in the database and the user is automatically logged in the application
Fail	The user stays in the same activity and a error messages are displayed

Table 39: UC-02

ID	**UC-03**
Name	Add event
Description	The user adds a new event in the application
Actors	User
Pre-conditions	The user is logged in
Post-conditions	-
Success	A new event is created in the database The user is taken back to the previous activity The new event is added to the list of events shown on the screen
Fail	The user stays in the same activity and error messages are shown

Table 40: UC-03

ID	UC-04
Name	User's profile
Description	The user accesses to his profile
Actors	User
Pre-conditions	The user is logged in
Post-conditions	The user arrives in his profile activity
Success	-
Fail	-

Table 41: UC-04

ID	UC-05
Name	Event detail
Description	The user clicks on an event and goes to an activity with the details of the event
Actors	User
Pre-conditions	The user is in the activity with the list of events
Post-conditions	The user arrives in the activity of the detail of the event
Success	-
Fail	-

Table 42: UC-05

ID	UC-06
Name	Edit event
Description	The user edits the information of the event
Actors	User
Pre-conditions	The is the creator of the group
Post-conditions	-
Success	The details of the event are modified

	The user is taken back to the activity with the details of the event
Fail	The details of the are not modified The user stays on the same screen and some error messages are shown

Table 43: UC-06

ID	**UC-07**
Name	Add group
Description	The user adds a group to the event
Actors	User
Pre-conditions	The user is logged in
Post-conditions	-
Success	A new group is created in the database The user is taken back to the previous activity The new group is shown in the list of groups of the event
Fail	The user stays in the same activity and some error messages are shown

Table 44: UC-07

ID	**UC-08**
Name	Join/Unjoin group
Description	The user clicks on the button of a group and joins/unjoins the group
Actors	User
Pre-conditions	The user is logged in
Post-conditions	-
Success	The user is added/deleted from the list of users of the group The event is added/deleted from the list of events of the user
Fail	-

Table 45: UC-08

ID	**UC-09**

Name	Group detail
Description	The user clicks on a group and is taken to an activity with the details of the group
Actors	User
Pre-conditions	The user is in the activity of the detail of an event
Post-conditions	The user arrives in the activity of the detail of the group
Success	-
Fail	-

Table 46: UC-09

ID	**UC-10**
Name	Add comment
Description	The user writes on the text box and sends the comment
Actors	User
Pre-conditions	The user is logged in
Post-conditions	-
Success	The comment is added to the database The comment is shown in the list
Fail	-

Table 47: UC-10

ID	**UC-11**
Name	Delete comment
Description	The user deletes one of the comments he has sent in the group by clicking on the comment item in the list
Actors	User
Pre-conditions	The user is the owner of the comment The user confirms the action in the popup shown
Post-conditions	-
Success	The comment is deleted from the database

	The comment is removed from the list
Fail	-

Table 48: UC-11

ID	**UC-12**
Name	Log out
Description	The user is logged out from the application, taking him back to the login screen
Actors	User
Pre-conditions	The user is logged in
Post-conditions	The user is logged out
Success	-
Fail	-

Table 49: UC-12

ID	**UC-13**
Name	Check another user’s profile
Description	The user clicks on a user from the list of users
Actors	User
Pre-conditions	The user is in the activity of the detail of the group
Post-conditions	The user arrives to the profile of the user clicked
Success	-
Fail	-

Table 50: UC-13

ID	**UC-14**
Name	Edit group
Description	The user edits the information of the group
Actors	User
Pre-conditions	The user is the owner of the group

Post-conditions	-
Success	The group's data is updated in the database The modified group is shown in the list
Fail	The user stays in the same activity and some error messages are shown

Table 51: UC-14

ID	**UC-15**
Name	Edit profile
Description	The user edits his profile
Actors	User
Pre-conditions	The user is the owner of the profile
Post-conditions	-
Success	User's data is updated in the database The modified profile is shown in the activity
Fail	The user stays in the same activity and some error messages are shown

Table 52: UC-15

ID	**UC-16**
Name	My events
Description	The clicks on "My events" in the application bar
Actors	User
Pre-conditions	The user is logged in
Post-conditions	-
Success	The user is taken to a new activity where it shows the events he has created and the events he has joined a group of it
Fail	-

Table 53: UC-16

3.3 Architecture

The architecture of the application has been designed based on a structure of two layers [13]: a view layer, which is responsible of handling and displaying the data in the UI[2]; and a data layer, which is in charge of retrieving/saving data from REST[3] APIs[4] and persistence data stores. In the *Figure 5* it is shown the diagram of the components of the system.

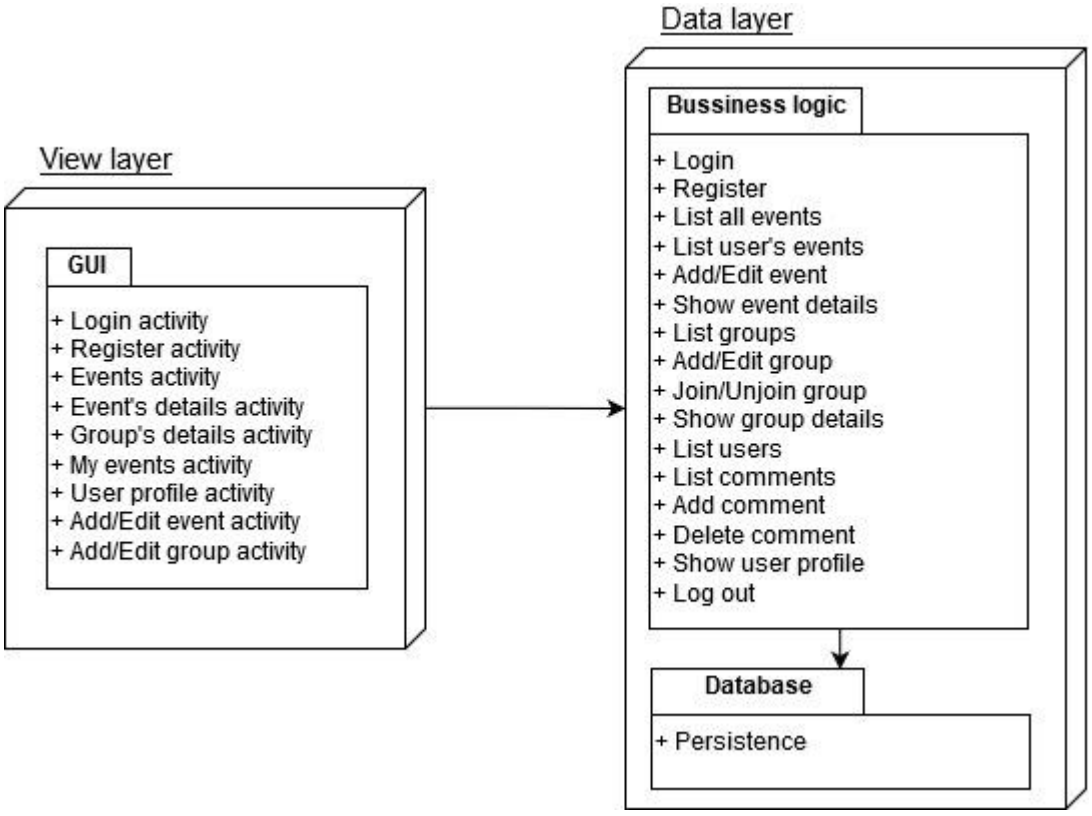


Figure 5: Components diagram

As it can be seen, in the view layer, it contains all the elements of the GUI[5], which are all the possible screen a user can access. Each of the activities, having its own design layout, may contain several basic elements (ViewText, EditText, ImageView, Button…) and more complex elements (RecyclerView, ViewPager…).

[2] UI (User interface) is the way through which the user communicates with the system

[3] REST (Representational State Transfer) is any interface between systems that uses HTTP to get data or generate operations on the data

[4] API (Application Programming Interface) is a set of subroutine definitions, communication protocols and tools for building software

[5] GUI (Graphical User Interface)

The activities can be composed also by fragments, which are independent modules of the screen, that are reusable in different activities and makes the modification of the different screen more efficient.

All the activity layers have assigned a Java class, which contains all the logical operations it is needed for the user to be able to interact with the system through the activities.

In the data layer, it is all the logical operations classes behind the graphical elements, that fills the data on the screen and provides the functionalities for the user to interact with the components of the system.

There are several different classes, each of them fulfilling a specific object:

- Model Java class: it is the class that defines the structure of a specific object, with all the required fields to be considered an entity of the application.
- Service Java class: in this class it is defined the different methods to provide specific functionalities of the application and the accesses to the database to obtain and save data.
- Adapter Java class: this class is normally used to fill the data of an item on the screen.

The database component provides a storage for the data of application objects and the image elements, with its corresponding methods to retrieve and save data, in a structured tree of JSON text objects or in a separate file storage.

The authentication process of the user in the system is also provided by the database component, including several methods to do the registration, such as Google, Facebook, Instagram…

CHAPTER 4: DESIGNING OF THE TECHNICAL SOLUTION

In this chapter it will be explained the approach of the solution and the description of the system implemented in Android. Firstly, an overview of the entire system and its different functionalities and then, a detailed description of each of the components of the application and how it has been implemented.

4.1 Introduction to the system

The application developed for this thesis is a social app which objective is to bring people together through meetings about cultural things. It allows the users to share events and gather a group of people to go together to these events, allowing them to meet new people that share same interests and hobbies, and expand their knowledge of culture.

The code is implemented in Java with Android Studio, using the Firebase[6] Realtime database[7] to store the data of the application. It is a NoSQL[8] database so there are not structures defined in the database for the objects, but it uses instead the Java defined object classes as the structure, stored in JSON[9] tree format.

The object classes used for this application are the following:

TABLE I
USERS
STORES THE INFORMATION OF THE USERS REGISTERED

Name	**Type**	**Nullable**	**Default**	**Unique**	**Description**
profileImage	String	Yes			Reference of the profile image stored
userName	String	No		Yes	Nickname in the application
name	String	No			User's real name
email	String	No		Yes	Email
phoneNumber	String	No			Phone number
age	Integer	Yes			Age

[6] Firebase is a Google platform for web and mobile applications development ubicated in the cloud.

[7] Realtime database is a NoSQL database of Firebase platform organized by JSON trees.

[8] NoSQL, also known as "No just SQL", is a non-relational structured data management system.

[9] JSON (JavaScript Object Notation) is a lightweight data-interchange format, in a human-readable text.

biography	String	Yes			Additional information about the interests and hobbies
friends	List<String>	Yes			Contains the emails of the user's friends
sharedEvents	List<String>	Yes			Contains the references of the events the user has created or has joined
joinedGroups	List<String>	Yes			Contains the references of the groups the user has joined

Table 54: Users

TABLE II EVENT STORES THE INFORMATION OF THE EVENTS					
Name	**Type**	**Nullable**	**Default**	**Unique**	**Description**
eventKey	String	No		Yes	Autogenerated unique ID
imageReference	String	No			Reference to the image of the event
title	String	No			Title of the event
date	Date	No			Date of the event
endDate	Date	Yes			Expiration date of the event
location	String	No			Location where the event is taking place
description	String	Yes			Additional information about the event
userCreator	String	No			Email of the user that created the event
eventGroups	List<String>	Yes			Contains the references of the groups created in the event

Table 55: Event

TABLE III GROUP STORES THE INFORMATION OF THE GROUPS					
Name	**Type**	**Nullable**	**Default**	**Unique**	**Description**
groupId	String	No		Yes	Autogenerated unique ID

eventId	String	No			Reference of the event where the group belongs
name	String	No			Name of the group
date	Date	No			Date of the group
location	String	No			Location where the group members will meet
description	String	Yes			Additional information about the group
userCreator	String	No			Email of the user that created the group
users	List<String>	Yes			Contains the emails of the users that has joined the group
maxUsers	Integer	No			The maximum number of users that can joined the group

Table 56: Group

TABLE IV
COMMENT
STORES THE INFORMATION OF THE COMMMENTS

Name	**Type**	**Nullable**	**Default**	**Unique**	**Description**
username	String	No		No	Username of the user
userEmail	String	No		No	Email of the user
comment	String	No			Content of the comment
reference	String	No			Reference of the object (group) where the comment belongs
postTime	Date	No			Time when the comment is posted
commentId	String	No		Yes	Autogenerated unique ID

Table 57: Comment

When the user launches the application for the first time, he will arrive to the login screen, not having an account yet, he will have to create a new one. Once created, the application will log the user automatically with the new account just created, taking him

to the main activity[10], the screen of the application where all active events are displayed and from where the user can access to the other functionalities implemented: create a new event, join groups of events already created or create a new group; check the events from which he is the creator or those events he has joined; access his profile, and from where he can edit or add information about himself; and, to log out.

If the user exits the application without logging out, he will be logged in automatically when the application is launched again.

4.2 Firebase setup

The database used for this project is the Firebase Realtime Database [14], a Firebase's service that provides an API that allows application data to be synchronized across clients and stored on Firebase's cloud. Client libraries[11] are provided to enable the integration with the application.

The Firebase platform provides also user authentication services [15], allowing the user to create and login with an account in the application. There are many authentication methods available, but in this application, it has been implemented the basic authentication email method.

A third storage service [16] is used, which allows the storing of any file generated in the application. For the moment, only pictures of the events and user profiles are saved.

The integration of the application with the different services of Firebase is done through Android Studio, as it provides built-in features to help the developer to connect the application with the platform, step by step.

On the platform side, it is necessary to create a new project for the application, and then access to the Console of Firebase, from where it is possible to check all the tools provided by the platform.

[10] An Activity is a component of the application which is generally a screen with a user interface and provides a specific functionality of the system.

[11] A library, in programming languages, is a collection of code previously written and compiled, that provides one or many functionalities.

To set up the authentication service, it is necessary to enable the desired methods in the authentication section of the platform, otherwise, they will not work when calling the methods from the application.

The database can be used right after connecting it with the application, allowing anyone to read and write on it. The increase the level of security of the accesses, it is necessary to change the rules of the database, allowing anyone to read, but only authenticated users can write on it. It is also possible to add other rules on the data, such as allowing the access to a specific tree only for a specific group of users or declaring indexes on the trees to improve querying efficiency. In this application, since users are queried by their email, and the events and groups by their identifier, it has been declared three indexes, one for each of the mentioned field.

After everything is set up and added the corresponding libraries in the Android Studio project, the application is ready to start using the services.

4.3 Login

4.3.1 Functionality

It's the launching activity and the screen in the *Figure 6* that will be shown to the user when he launches the application.

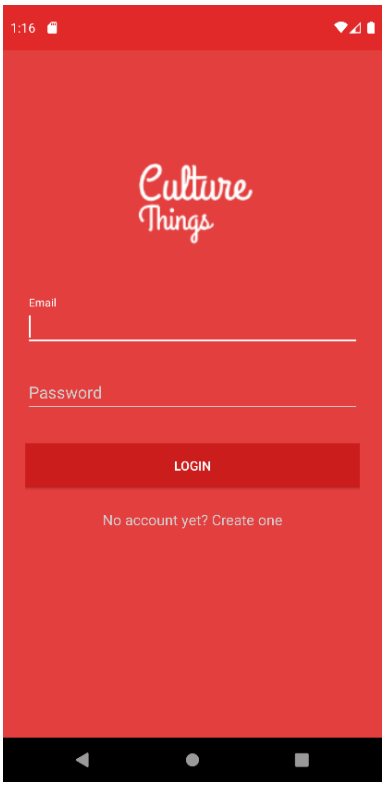


Figure 6: Login activity

In this Activity, the user can login with the registered account or access to the signing up screen, by clicking on the link below the “Login” button, to create a new account. Once the user has successfully logged in with a valid account, *Figure 7* will be shown with the text “Authenticating” while the application is loading all the events; when it is done it will take the user to the screen with all the events.

Same screen but with text “Loading events” is shown when the user launches the application again when has already logged in.

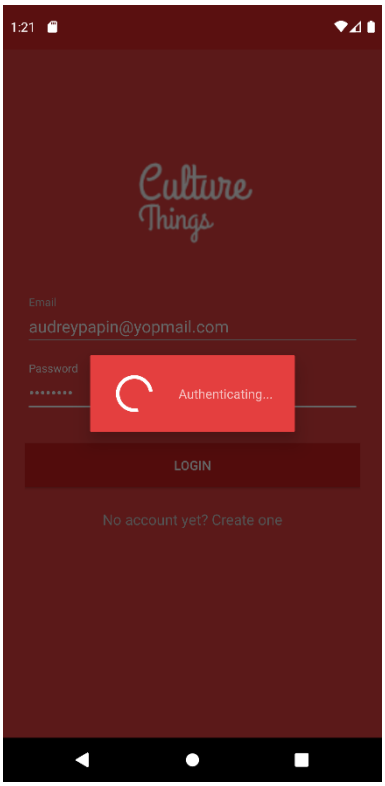


Figure 7: Login popup

4.3.2 Implementation in Android Studio

In the design of the Activity layout it has been used four basic elements, shown in *Figure 8*:

- An ImageView element for the logo of the application.
- Two EditText elements for the email and password inputs.
- A Button element to send the request to login.
- A TextView element for the link to go to the sign-up screen.

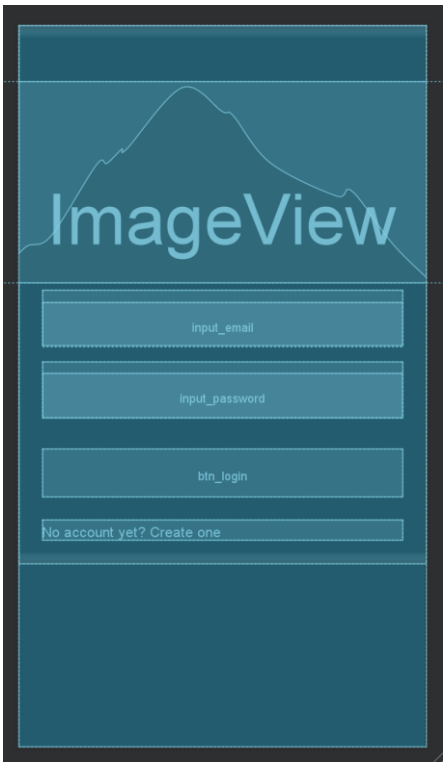


Figure 8: Login layout elements

To use the element from a layout in a Java class it is necessary to set the activity's content view first with the layout where the element is located, then bind it to a variable using the "findViewById" method passing the id of the element as the argument. In this specific activity, the library Butterknife[12] has been used to bind the elements.

To implement the login, the library Firebase-auth provided by Firebase has been used, which provides the functionalities to authenticate a user in the application.

When the user types his email and password and clicks on "Login" button, the method "signInWithEmailAndPassword" provided by the library is called, passing the email and password as arguments, a request is sent to the database to attempt to login and a listener is necessary to know the result of the request, so an "OnCompleteListener" is added in the same calling. If the task is successful, an instance of the logged user is created, and a popup, using the ProgressDialog class, is displayed while the events are loaded from the database.

[12] Butterknife is a library that simplifies the usual way of view bindings in a more elegant and clear code.

To query the database and get the events, it is necessary to use the methods of the library “firebase-database” provided by Firebase, as it is not allowed to use standard SQL queries. For example, to get all the events using SQL is:

*“SELECT * FROM event”*

And using the “firebase-database” library is:

“FirebaseDatabase.getReference(“event”).addChildEventListener(…)”

As it can be seen, there are two methods appended one after another. Because the database data are organized by JSON trees it necessary to get the reference of the node of events elements, and then, add a listener to get all the elements in the node.

There are different listeners to get the elements from a node, differing in how the elements are returned and the life cycle of the calling:

- ValueEventListener: returns the node key and its value, which are all the child nodes in a single object, i.e., a list of objects. To work with the objects, it is necessary to get the children by calling the method “getChildren”. Keeps listening for new values in the node and returns the entire node every time the node value is modified.

- ListenerForSingleValueEvent: same behaviour as ValueEventListener but it is only executed once and does not listen for future values and modifications.

- ChildEventListener: returns one by one the children of the node, i.e., a single object. Keeps listening for events in the node (“childAdded”, “childChanged”, “childRemoved”, “childMoved”), but returns only the child node.

To get the events it has been used a ChildEventListener, it is more efficient than a ValueEventListener, and keeps the list of events updated every time a user adds a new event in the application.

Once all the events have been loaded correctly, the user will be taken to the activity with all the events listed.

4.4 Sign-up

4.4.1 Functionality

This is the Activity where the users can create an account in the application.

When the user clicks on the sign-up link in the login screen, he will be taken to this screen, shown in *Figure 9*.

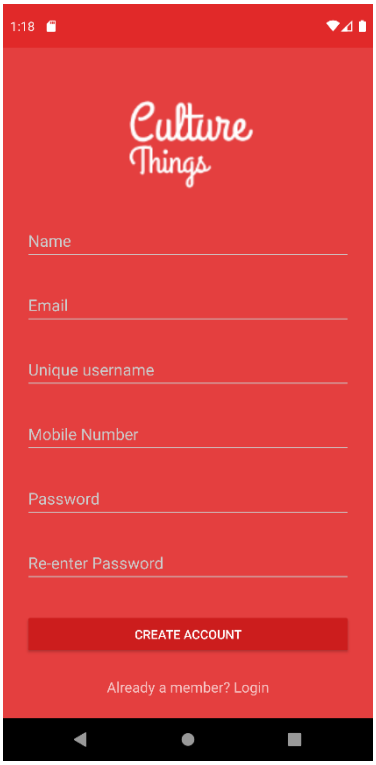


Figure 9: Sign-up activity

To create the account, the user must fill all the fields and, if any field is wrong, and error message will be displayed on the right side of the field (Figure 5). Both email and username name fields are unique for each user, so the signing up process will fail if the user introduces already existing ones.

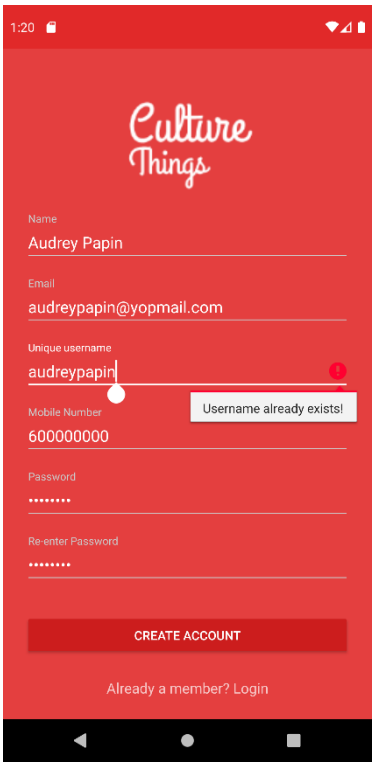


Figure 10: Sign-up form

When the user clicks on the button, a popup will be shown while the account is being created and, if the process is successful, the user object will be saved in the database and he will be taken back to the login activity, logging in automatically with the account just created.

The user can return to the login activity without creating a new account by clicking on the text at the bottom of the screen, below the button.

4.4.2 Implementation in Android Studio

In the design of this Activity layout (*Figure 11*), it has been used the same type of elements than in the login layout (ImageView, TextView, EditText and Button), but in this case it has been used a "ScrollLayout[13]" instead of a "LinearLayout[14]" to the enable the scrolling when all the elements does not fit in the screen.

[13] ScrollView is a view group that allows the view placed within to be vertically scrolled.

[14] LinearLayout is a layout that arranges the views placed within either vertically or horizontally.

Figure 11: Sign-up layout

When the user clicks on "Create account" button, some validations are made to values of the fields (fields are not empty, passwords matches, unique username…), and only when there is not any error, a request is sent.

To validate that the username introduced by the user has not been already taken by another user, it is necessary to query the database

The sign-up functionality is also provided by the library "firebase-auth" of Firebase, calling the method "createUserWithEmailAndPassword" and passing only the email and password as arguments, the service will attempt to create a new user account, checking that the user's email does not already exists in the database.

Declaring a listener to get the result, if the task is successful, the user's information will be inserted in the database using the next method:

"FirebaseDatabase.getReference("users").push().setValue(user)"

First, it is necessary to get the reference to the users' node first, then the "push()" method adds a child node with an autogenerated random key value, which is the reference of the child node, and finally, the user object value is set to the node.

The user will the taken back to the login screen and will start to login automatically with the account just created, loading all the events and proceeding to the events' activity when it's done.

4.5 List of events

4.5.1 Functionality

This is the main Activity of the application (*Figure 12*), it is where all the active events are displayed, listed one by one and ordered by the date of the event. Each element is clickable and takes the user the detail screen of the event clicked, and for each of them, the following information is displayed:

- Image of the event
- Title of the event
- Location where the event is taking place
- The date on which the event should occur
- The username and profile picture of the user that has created the event

The user can slide down from the top of the list to refresh it with the new events of modifications made to the existing events.

There are also some other functionalities that are only accessible from this screen:

- Create a new event, by clicking on the floating button at the bottom.
- Access the user's profile, by clicking on the additional options in the application bar and choosing the option "Profile".
- Access to the activity with the list of events that the user is owner or is participating in a group of the event.
- Log out, by choosing the option in the application bar.

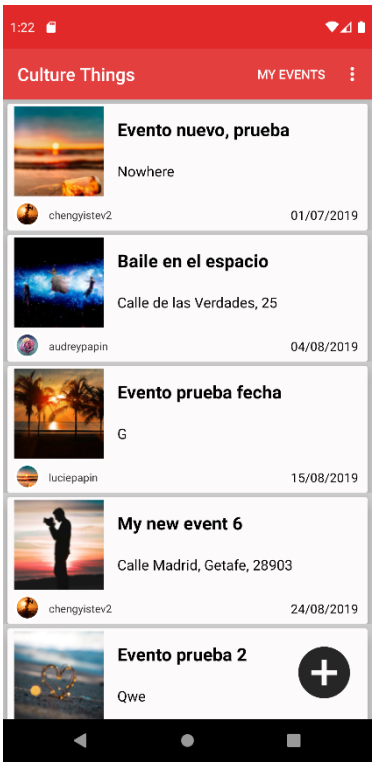


Figure 12: List of events, main activity

4.5.2 Implementation in Android Studio

To design this activity, it has been used a ConstraintLayout[15] that allows to position the elements in function of other elements in the layout, and not just linearly one after another, as in the LinearLayout. Thanks to this, it is possible to position the button floating over the fragment below it, staying always in the same place.

The layout is composed by a Fragment[16], an ImageView as the floating button and, also the options in the application bar. In the *Figure 13* it can be seen the Fragment of the list of events and the floating image to add new events, and the curly lines on the screen shows that the element is constraint to the sides of the screen. Inside the fragment there is only a RecyclerView[17] to list the events. A fragment is used instead of adding it directly in the activity because it is easily reusable by several activities and it makes it easier and more efficient to modify it afterwards.

[15] ConstraintLayout is a layout which allows positioning and sizing in a flexible way.

[16] A Fragment represents a behaviour or a portion of user interface in an Activity. Multiple fragment can be combined in a single activity.

[17] RecyclerView is a view group that displays a view repeatedly, in a way that the views are reused to generate only those shown in the screen.

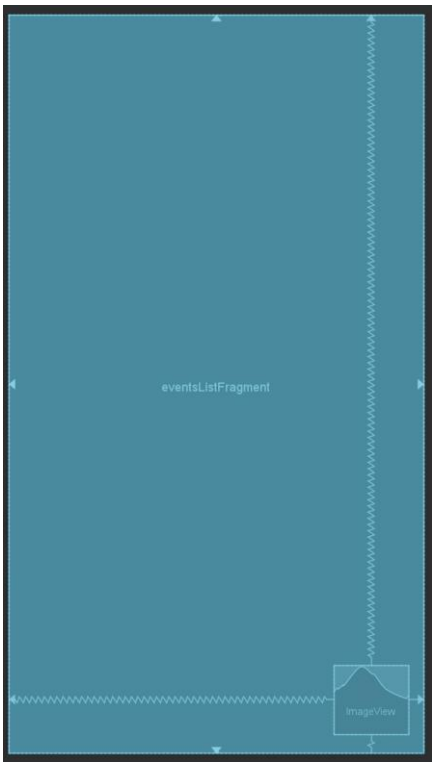


Figure 13: Events activity's layout

RecyclerView is considered the successor of ListView[18] and GridView[19], as it allows to create both latter view layouts with improved performance and additional features. Using it alongside the CardView[20] to define the appearance of the elements in the list, it is very easy to create custom dynamic lists.

To create the item layout using CardView, it is necessary to do it in a new independent separate layout, which contains only the design of the item view. For the events, the layout of the item is shown in *Figure 14*. Once the design is done, it is necessary to populate them with data upon listing them, and to do so it is necessary to use the Adapter[21] class, which binds the components in the item layout allowing to work with them.

[18] ListView is a view group that displays a vertically scrollable collection of views, where each view is positioned immediately below the previous view in the list.

[19] GridView is a view group that displays items in two-dimensional scrolling grid.

[20] CardView is a frame layout with a rounded corner background and shadow.

[21] Adapters provide a binding from an app-specific data set to views that are displayed within a RecyclerView.

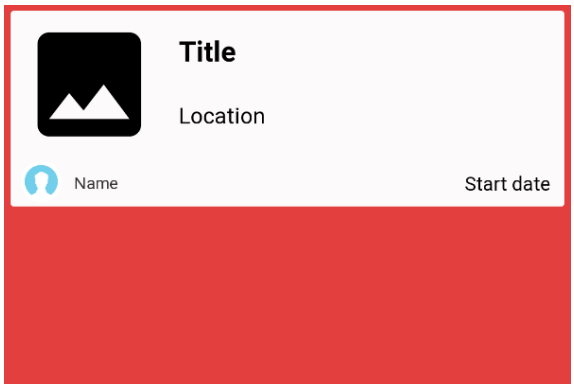


Figure 14: Event item layout

The texts components are set in the same way than previous activities, but for setting the images, it is necessary to download them from the Firebase storage[22]. Using the Firebase's library "firebase-storage" methods, it is possible to download the image in a similar way than querying the database:

- First, it is necessary to get an instance of the storage and the reference to it.
- Then call the "child" method, passing as argument the path where the image is stored and the identifier of the image.
- Finally, add a listener to get the result. If the image is found, an URI[23] is returned and it is used to download the image from the storage.

The image can be set to the view element using standard libraries, but it is not efficient and too costly, downloading unnecessary big images. So, the Glide[24] library is used, allowing the application to compress the images, keeping similar quality, and speeding up the setting of all the images of the list. It also provides resizing and cropping methods, to adapt the image to the layout needs, such as "centerCrop", which fits the image to fill the space. Taking an image like *Figure 15*, since it is larger than the space set for it, the image will be cropped as shown.

[22] Firebase Storage provides a service that allows applications to safely upload and download any type of file.

[23] URI (Uniform Resource Identifier) is a standard for identifying document using short string of number, letters and symbols.

[24] Glide is an Image Loader library.

Figure 15: Image cropping example

The behaviour of the item when it is clicked is defined in the activity, implementing the interface declared in the Adapter, a different action can be set for each activity. In this case, the user will be taken to a new screen with the details of the event and extra information about it, including the list of groups that have been created in the event.

Lastly, to add the options in the application bar, an independent layout is needed. *Figure 16* is the layout of the options in the application bar, the elements can be chosen to be always shown, when there is space or never. In this case, "My events" is always shown and the others, inside the black box below the bar, are only visible when the user clicks on the dots' icon.

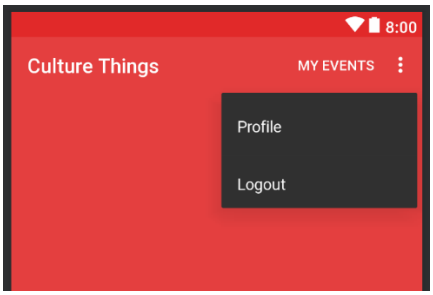


Figure 16: Application bar menu layout

4.6 Add event

4.6.1 Functionality

It is the activity where the user can create a new event in the application. It is accessible by clicking the floating button or choosing the option in the application bar in

the activity with the list of events. When the user arrives to the screen, all the fields are empty, with hints about what should go in each field, as shown in *Figure 17*.

To add an event successfully, the user must fill the mandatory fields, which are the title (What is it?), the date (When is it?), the place (Where is it?) and the image. When the user clicks on the image icon, he will be taken to an external gallery application to choose an image, and then he will be returned, and the image chosen will be loaded on the screen. Upon clicking on "ADD EVENT", if the fields are correct, the event will be created and added to the list of events and will take the user back to the previous screen, otherwise, error messages will be shown on the field that is incorrect.

Figure 17: Add event activity

4.6.2 Implementation in Android Studio

The layout of this activity is composed by some basic elements already explained in the previous sections.

Upon creating the view, all the elements are bind to variables, so it makes it possible to work with them. On the image element it is set an action that upon clicking, it is ask to the system to open a gallery application to choose an image. Whether the user chooses or not an image, he will be taken back to this activity with a result code, if the result code is

"OK", it means the user has chosen an image and a URI to the image is also returned. Using this identifier, the image is set to the ImageView of the screen.

When the user clicks to button to add the event, the application validates that all the required fields are filled and then, proceeds to save the event object in the database. For this, the image must be stored in the Firebase storage, so a random id is generated to identify the image and set to the corresponding field in the event object, and a reference with this identifier is created in the storage. The image is compressed before uploading it, so it is lighter to load afterwards. Once the image is uploaded correctly, the user is taken back to the previous activity, returning all the data of the event just created, and will add the event object in the database, under the events tree. The identifier of the event that has been just created will be added to the list of events in the user object that has created it.

4.7 Event's details

4.7.1 Functionality

This is the screen where the user will be taken to when he clicks on an item of the list of events. It is displayed the end date of the event and the description, which are two optional fields, and the user can search the location of the event, by clicking on the map icon on the right side of the box, asking the user to choose, if there are more than one, a map application installed. An example is shown in the *Figure 18*, where all the fields are informed. When the description text is very large, as in the example, a maximum of four lines will be shown, and a scroll bar will be shown next to it, indicating the user that he can scroll down to see the rest.

Figure 18: Event's details

The groups of the event are listed below the event information, only the actives ones are displayed, and they are ordered by the date. Each of the items shows the name of the group, the date and the place the meeting is taking place, the number of users that has already joined the group, the total members allowed in the group and a button to join or unjoin the group. The button description changes according to the next conditions:

- The user has not joined the group: “JOIN”
- The user has joined the group: “UNJOIN”
- The group is full, and the user did not join the group before: “FULL” and the button is disabled.
- If the user is the creator of group, an additional “OWNER” text is displayed over the description in the button.

Any user can add a new group in the event, by clicking on the “plus” icon next to the image, or by choosing the option the application bar.

The event information can also be edited, by choosing the option in the application bar, but it is only displayed to the user that is the creator of the event.

4.7.2 Implementation in Android Studio

The activity is composed by several elements, as shown in the *Figure 19*:

- An ImageView to show the picture of the event.
- An ImageView that acts as a button to add groups.
- A fragment that contains the details of the event.
- A fragment that contains the RecyclerView to list the groups.

Additionally, there is also the layout for the options in the application bar, which does not differ from the one used in the previous activity but on the actions. In this case, the options available are:

- Add group: same functionality than the button, takes the user to another screen to add a new group to the event.
- Edit event: available only if the user is the creator of the event. Takes the user to a screen where the event data can be edited.

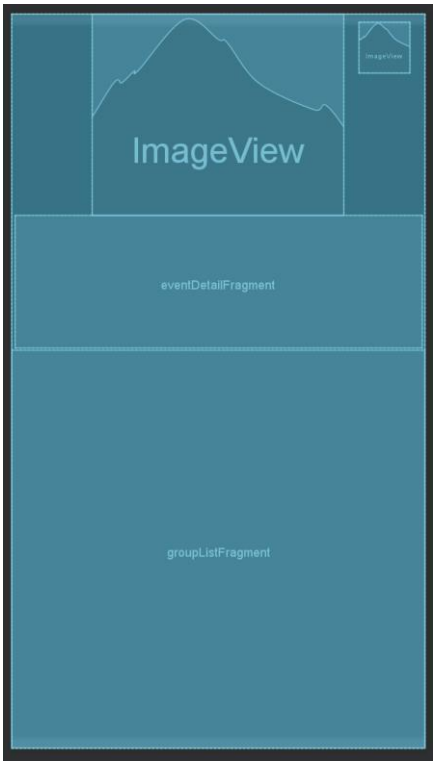


Figure 19: Event's details layout

When clicking an event from the list, the user is taken to this activity, passing the index of the item as argument, which is used to know which item has been clicked and get it from the list of events.

Upon entering the screen, it will get the event item from the corresponding list of events (all events or user's events) and set the data in all the fields, if there any empty

field (optional field), the element will be hidden. The action of the button when it is clicked is also set, to take the user to a screen to create a new group.

Each event object has a list that contains the identifiers of the groups that belongs to it and, if it is not empty, they will be loaded. To get the group objects, it will be necessary to query the Group tree, but instead of querying them one by one, using the list of groups identifiers from the event object, all the groups will be gotten using the event's identifier, as group objects contains the id of the event it belongs to. The query is at follows:

ref.orderByChild("eventId").equalTo(event.getEventKey())

Ref is the reference to the Group tree and, combining "orderByChild" and "equalTo", it gets all the groups whose "eventId" is equal to the identifier of event passed. And, as always, a listener is needed to get the result of the query.

The group item layout is shown in *Figure* 20, and to display them in the list, it is needed an adapter for setting the information of each of them, for this, the previously retrieved list of groups is set to the adapter. The adapter sets the data of the group item and for the button, as stated before, both text and action are different according to the next conditions:

- If the user is not a member of the group and clicks on the button, the user's email will be added to the list of users of the group object, the group identifier will be added to the list of groups of the user object and, the event identifier is also added to the list of events of the user object if it does not already contain it.
- If the user is a member of the group and clicks on the button, user's email will be removed from the list of users of the group object, the group identifier will be removed from the list of group of the user object and, if the user is not the creator of the event, the event identifier will be removed from the list of events of the user object.

  If the user attempts to leave the group when it is less than 24 hours to the meeting time, a warning popup will be displayed, asking the user to confirm that he wants to leave.

To commit the changes of the modifications to the database, it is necessary to add the object again to the database, but when it detects that there is already an object with the same identifier, it will substitute the old object with the new one.

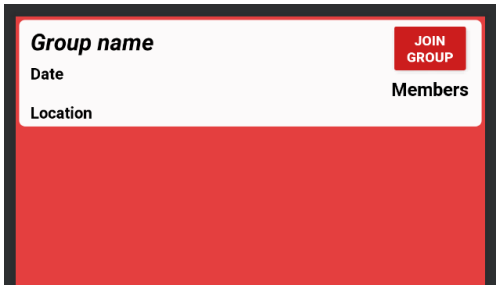


Figure 20: Group item layout

Like the event items, when a group item is clicked, the user is taken to a screen with the details of the group, passing the group identifier as argument.

4.8 Edit event

4.8.1 Functionality

This activity is like the activity to add a new event, but it is accessed from the event detail's activity, choosing the edit option in the application bar, and it is displayed to the user with filled with the data of the event from where it was accessed.

The user can edit the information of the event from this screen, it has the same mandatory fields than when adding a new event. If the use clicks on "SAVE EVENT" button, the event will be updated with the new values and will go back to the previous screen, showing the information of the updated event.

4.8.2 Implementation in Android Studio

To implement this activity, the layout of adding a new event has been reused, but instead of an empty layout, the data of the event is set upon entering the screen, using the event identifier passed as argument and getting the event object from the database.

4.9 Add group

4.9.1 Functionality

In the activity of the *Figure 21*, the user can create a new group in the event. It works similarly than adding an event, with some mandatory fields and optional ones. When the user has filled the form with the required information and clicks on "ADD GROUP", a new group item will be added to the list of groups of the event and it returns the user to the event detail's activity, adding the group created in the shown list.

When a user creates a group, he will be automatically included in the group, as the objective of creating a new group is to gather people to go on the date and the place that the user is interested on, although he can still leave the group afterwards.

Currently, it is not possible to delete a group, but it is possible to hide it from other users by setting the date before the current date, as it will be counted as an expired group.

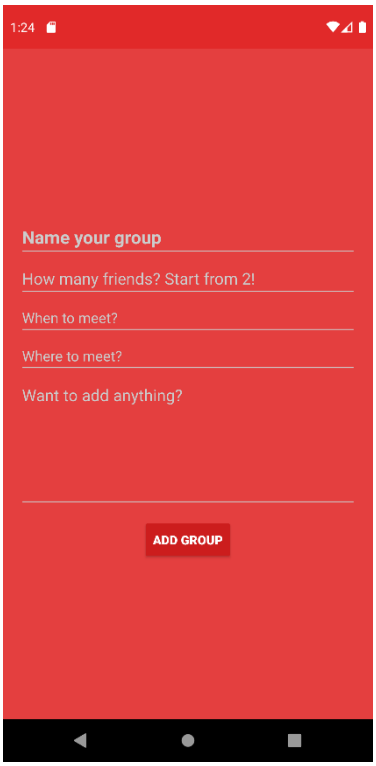


Figure 21: Add group activity

4.9.2 Implementation in Android Studio

The layout of this activity is simple, with several EditText elements and a Button. Upon filling the form and clicking the button, there will be a validator that checks that all the required fields are filled correctly, then, it will return the values filled and will be set to a new group object, which is appended to the Group tree in the database.

4.10 Group detail

4.10.1 Functionality

This activity is accessed by clicking on a group item, from the list of groups, and it will contain the information of the group clicked. It is composed by two different screens, displayed as tabs, where the user can switch between them by scrolling horizontally or clicking on the desired tab.

The first screen, shown in *Figure 22*, it is where the details of the group and the list of the users that has joined the group is displayed. For each of the user items, it will be shown only its profile picture and the username in the application, which is unique for each user, so it gives a simple view of who are the members. If the user clicks on a user item, he will be taken the profile of the user.

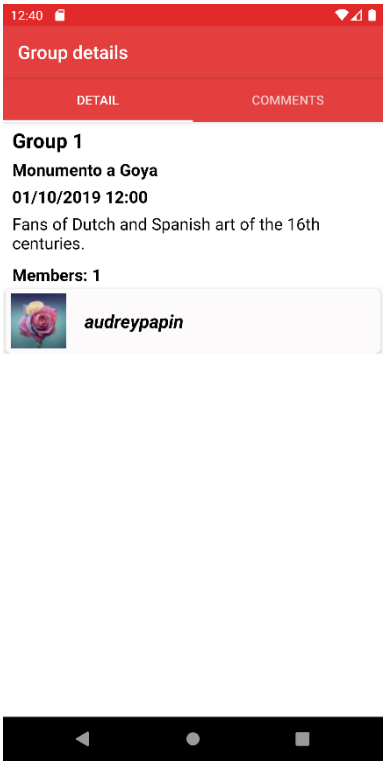


Figure 22: Group detail activity, first screen

The second screen, shown in *Figure 23*, displays all the comments in the group that the users has left, and any user can comment in the group, using the text box at the bottom of the screen, but empty messages are not allowed.

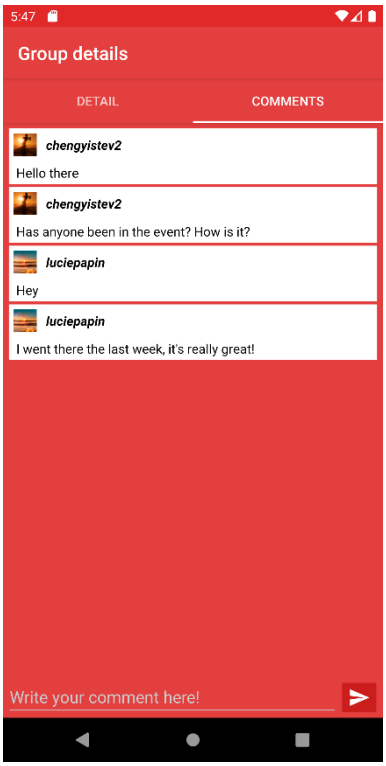


Figure 23: Group detail activity, second screen

The comments are ordered in descending order by the date it was created, being the oldest one the one at the top of the list. The comments are updated when a user sends a new one, being added at the bottom of the list and shown automatically, without the need the reload the screen. The users can delete their comments, by clicking on the item, and a popup will be shown to confirm the action (*Figure 24*), but if the user clicks on a comment that is not his, nothing will happen.

There is not a limit set on the length of the message, but when writing it in the text box, if is too long, a maximum of 4 lines will be seen and the user will have to scroll vertically to see the rest of the message.

The idea of having a comments section is to allow the users to communicate between them, giving some detail and share their personal experiences, so it can help other users to decide whether they want join or not group to assist to the event.

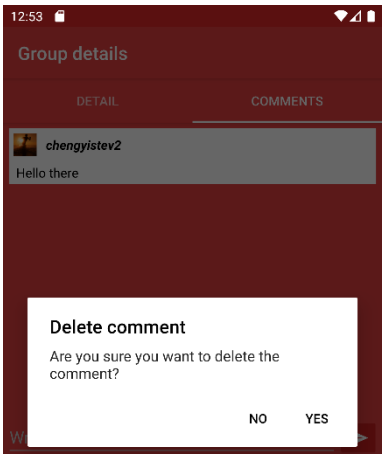


Figure 24: Delete comment popup

4.10.2 Implementation in Android Studio

In the implementation of this activity it has been used two new elements:

- A TabLayout, to show the tabs available in the activity.
- A Viewpager, that allows to flip left and right through different pages.

The layout of the activity is shown in *Figure 25*, where it only contains the elements mentioned before, but not the elements that will be shown in this screen. This screen is of general purpose, being reusable in any other activity, the view elements composing the activity will be set afterwards.

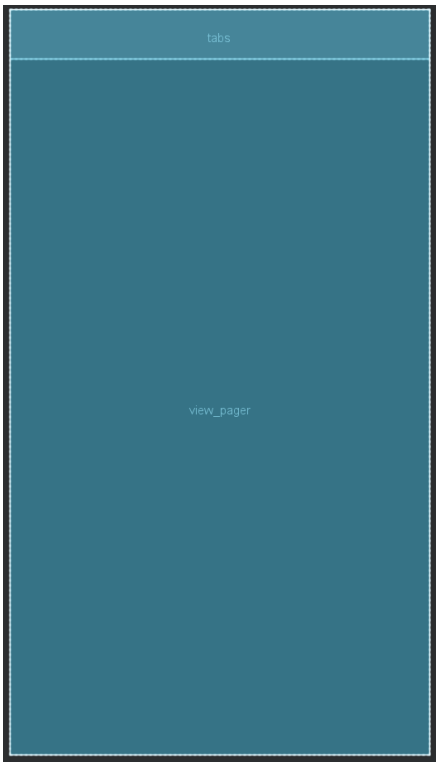


Figure 25: Group details tab activity

The two view elements that will be set are two Fragments, one containing the elements to show the details of the group and its list of users, and the other one containing the list of comments.

To set the view elements, it is necessary to use an Adapter, extending FragmentPagerAdapter class, which returns the fragments that will be set in each of the pages. In previous activities, the fragments were included in the layout of the activity, making it easier to initialize its values and communication between fragment and activity were direct, but now, for every fragment, it is necessary to initialize them individually upon creating a new instance of them, and the communication must be done through arguments.

When creating the new instances, the clicked group identifier is passed as argument. For the first fragment, this will be used to get the group object from the database, and set the information about the group; and the list of users in the group object, which contains their emails, is used to get the user object from the database to set the list of users. The layout of the user item is shown in *Figure 26*, and it is necessary to get the profile image identifier and the username to set each of the them.

For the second fragment, the group identifier will be used to get all the comments of the group from the Comment tree in the database. It has been decided to put the

comments in a separate tree to make the loading of group objects faster, as the list of comments can get really large, otherwise, they will be also loaded when listing the groups in the event's detail but it is not used. The layout of the comment item is shown in *Figure 27*.

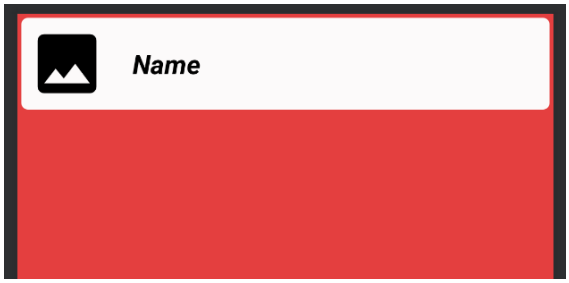


Figure 26: User item layout

Figure 27: Comment item layout

To update the list of comments automatically it is necessary to use a ChildEventListener on the tree of comments of the group, so when an object is added in the tree, the listener will get it and add it to the list of comments.

4.11 User's profile

4.11.1 Functionality

This activity can be accessed from the main activity, by choosing the option in the application bar, or when the user clicks on a user item in the list of users.

An example of the activity when the user accesses from the main activity is shown in *Figure 28*. It contains the profile picture of the user, the name, the username, the email, the age, the phone and a biography section where the user can put a little introduction of himself. The profile can be edited by clicking on the pencil icon on the top right corner, but it's only shown to the owner of the profile.

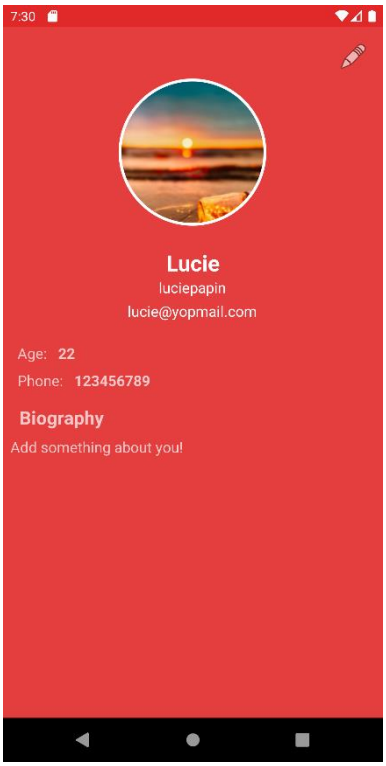


Figure 28: Logged user's profile

If the user edits the profile, he will be taken to another activity, shown in *Figure 29*, that looks identical to the design, but some elements are changed to be editable. The username and the email of the user cannot be changed. To change the profile picture, the user must click on the picture, and he will be asked to choose a picture from the phone's gallery. To save the changes, the user must click on the icon again, which has changed to a saving icon.

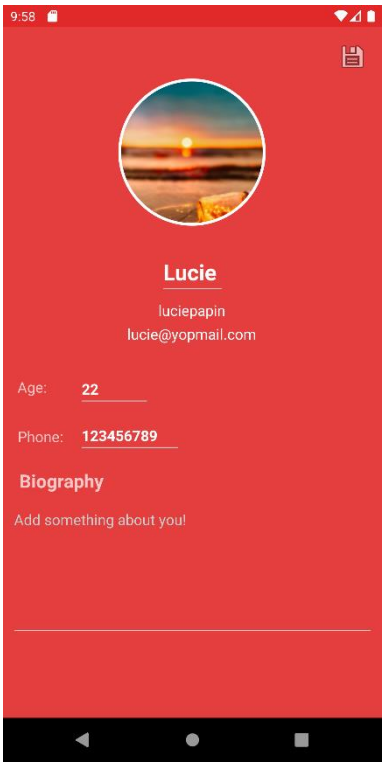


Figure 29: Edit profile activity

If the user access to it by clicking an item in the list of users, and it's not his own profile, a similar screen is shown but with some fewer elements. The pencil icon will not be shown, as stated before, and the email of the user will not be displayed either, as shown in the *Figure 30*.

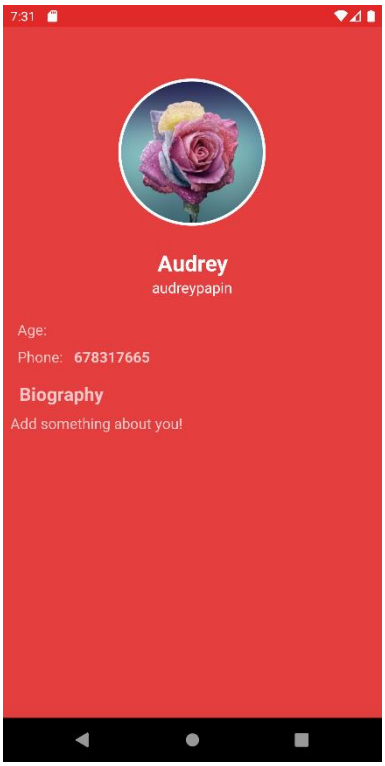


Figure 30: Another user's profile

4.11.2 Implementation in Android Studio

The layout of this activity is simple, shown in *Figure 31*, but for the profile picture, it has been used a CircleImageView, a special image element which allows the images to be shown in a circular shape.

When entering the activity, using the user's email, it is necessary to get the user's object from the database. First, it is necessary to get a reference to the User tree, and then, using the query "*userReference.orderByChild("email").equalTo(userEmail)*" and adding a listener, it will return the object if it is found in the database. In this case, it has been used a ListenerForSingleValueEvent, as it is not necessary to keep listening for changes or modifications.

While setting the values, it is checked if the logged user is the owner of the profile, in the negative case, the email text view and the icon image view will be hidden. The logged user can be gotten from the instance that was created when the user logged in the application.

If the user chooses to edit the profile, he is taken to another activity. The process to set the values in the screen is the same than for the profile activity.

When the user clicks on the saving icon, a popup will be shown while the fields are validated. If the user has changed the profile picture, the new picture will be uploaded to the storage and, upon returning the previous activity, the new data will be set into the fields of the screen, and the user object will be update in the database.

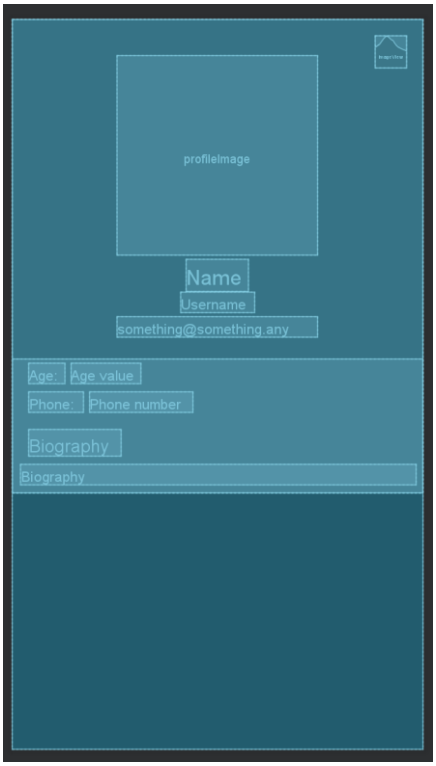


Figure 31: Profile activity layout

CHAPTER 5: TEST AND RESULTS

In this chapter it is shown the evaluation of the application, by using the results of the survey done by some users, after testing the application.

The outcome of the survey will be also very useful to know what aspect of the application needs to be improved and what are the aspects that user cares the most.

5.1 Evaluation and survey

To evaluate the application developed, it has been given to some users to test it, and they have been asked to do a short survey about the functionalities, the usefulness and aesthetic aspects of the application. There are also some questions about their interests, to see what profile of users would use the application.

The questions of the survey are shown in the next figures:

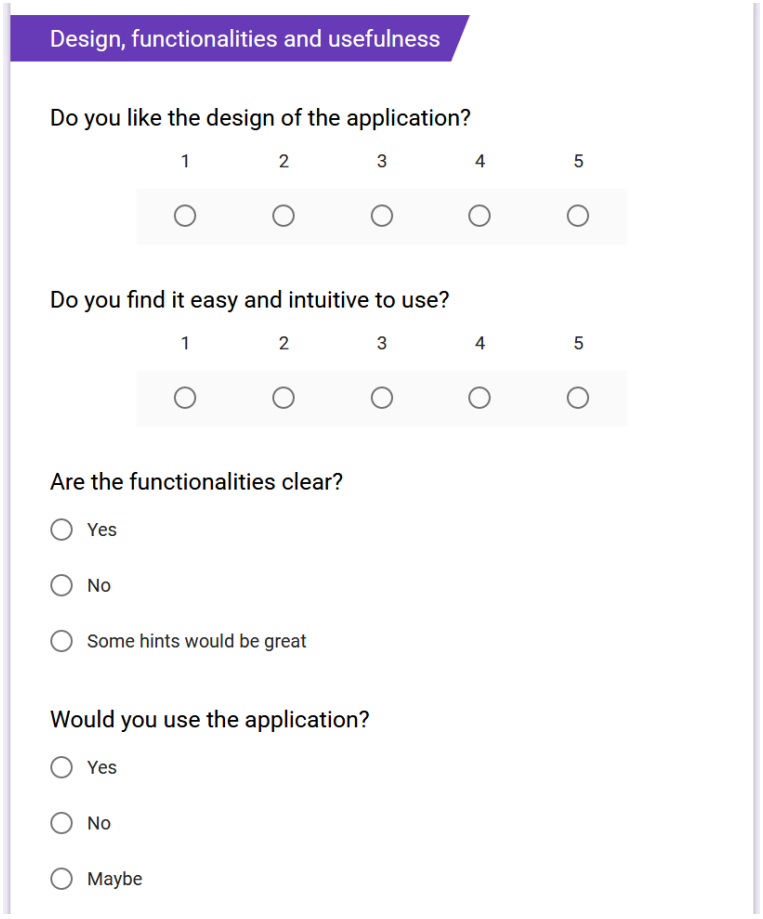


Figure 32: Survey, part 1

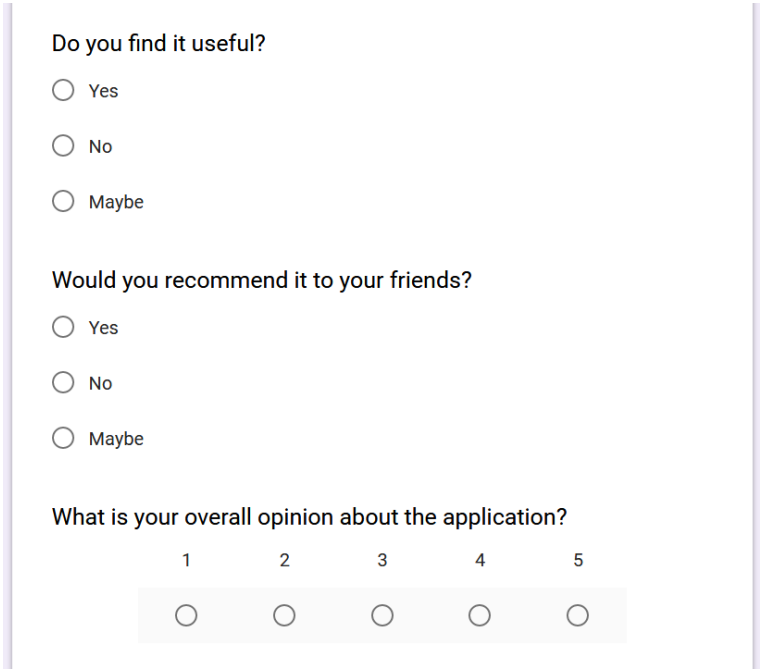


Figure 33: Survey, part 2

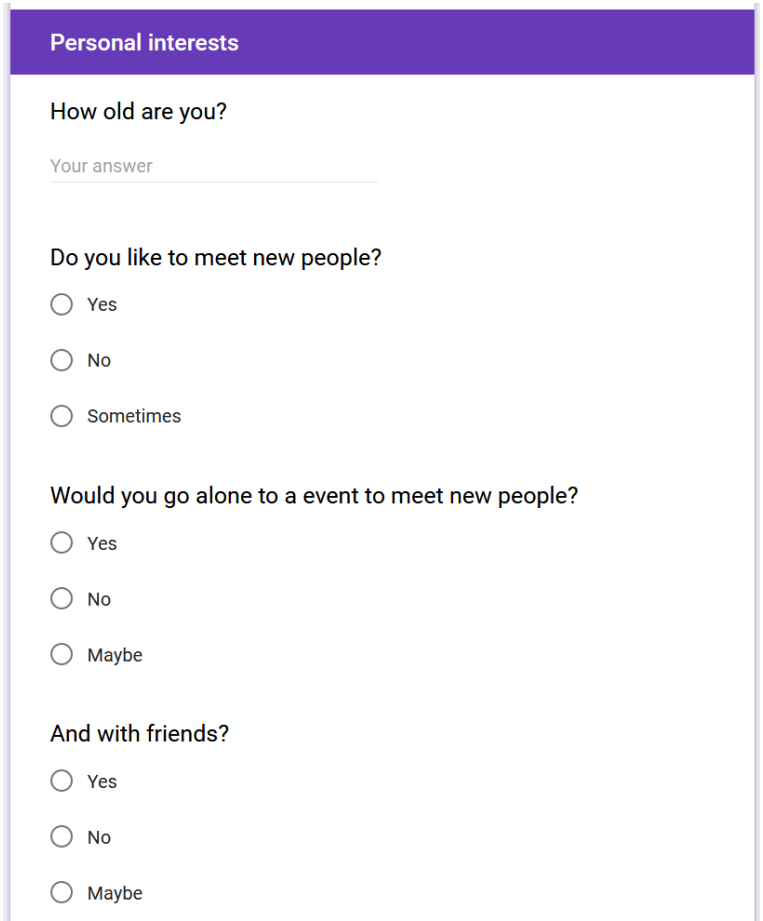


Figure 34: Survey, part 3

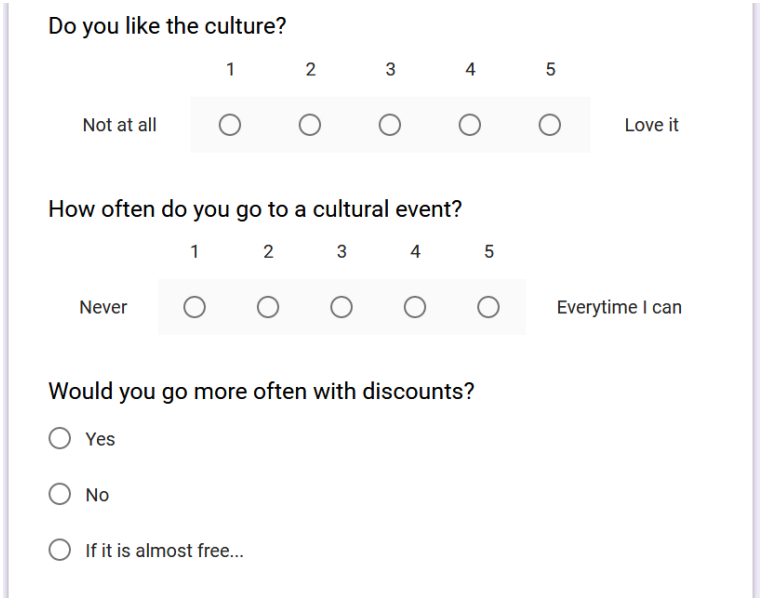


Figure 35: Survey, part 4

5.2 Results

From the results obtained, it needs to be highlighted that all the users find the application useful, even if some of them are not really interested in the culture and do not usually go to cultural events.

The design of the application needs to be improved, as more than half of the users find it normal (*Figure 36*), and some hints would be helpful for the users to understand how the application works and all its functionalities (*Figure 37).*

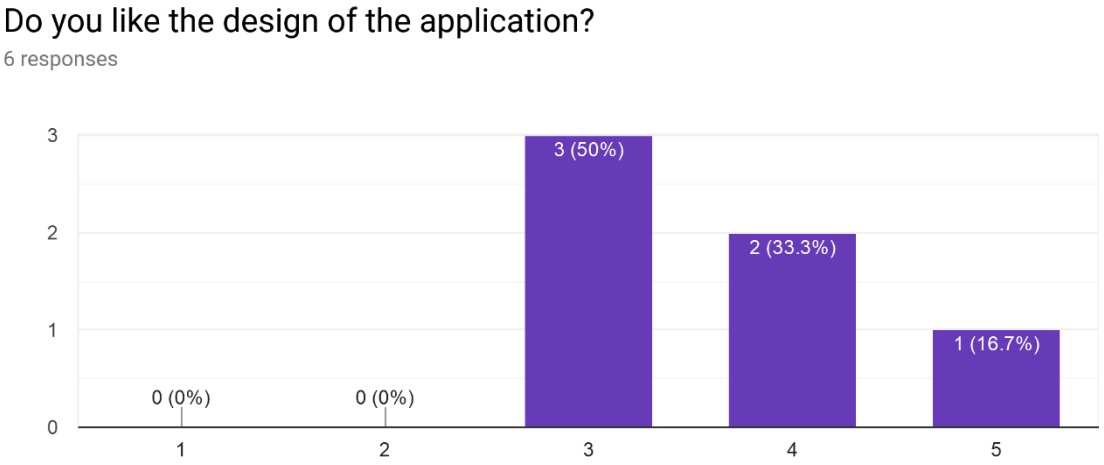
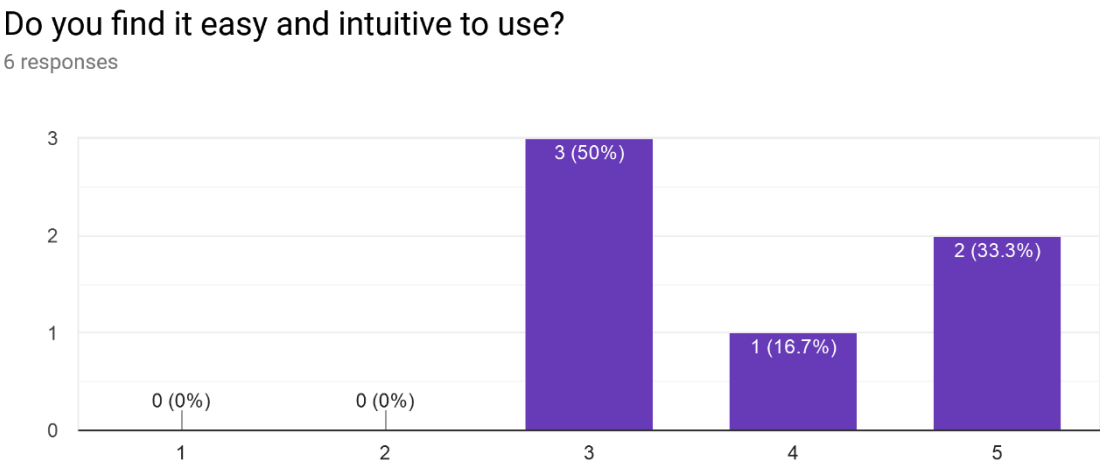


Figure 36: Answer, design



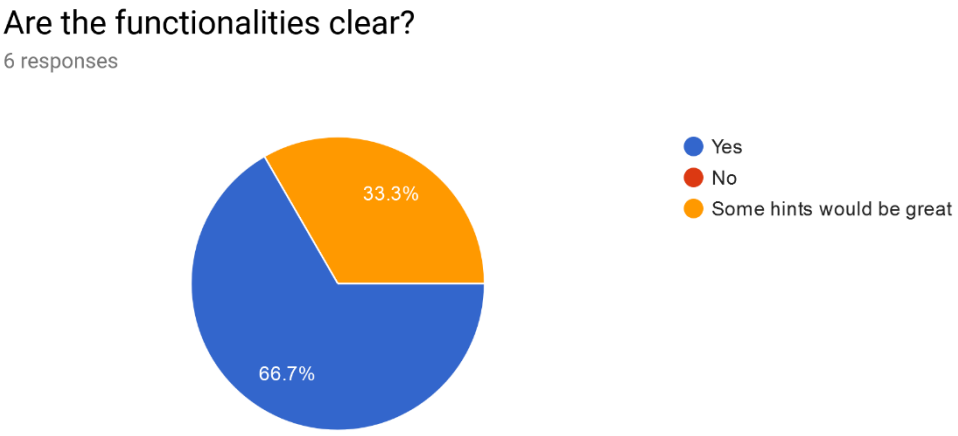


Figure 37: Answer, ease to use

Even if most of the user find the application useful, and that they would share it with their friends (*Figure 38*), many of them are not sure to use the application (*Figure 39*). The reason is that, if we look on the answer of their personal interests, most of them prefers to go to the events with friends, and not alone (Figure 40). So, at the time of doing the survey, they did not have friends using the application yet, and they chose to answer "maybe".

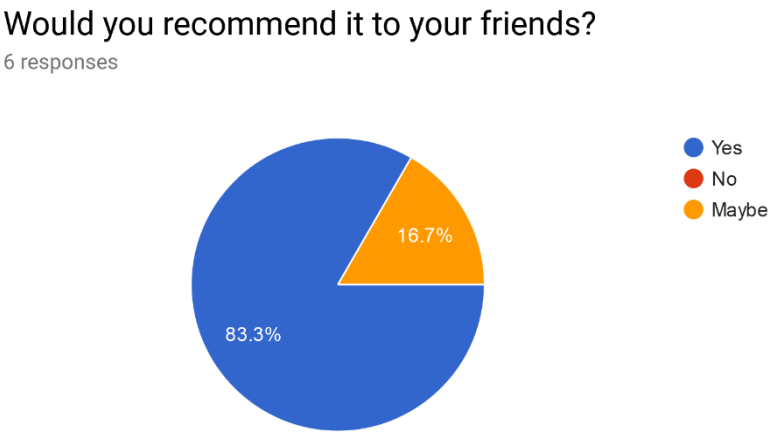


Figure 38: Answer, friend recommendation

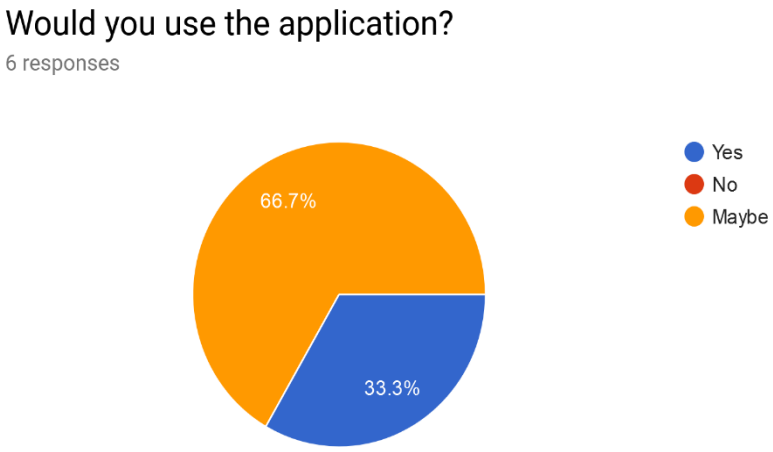


Figure 39: Answer, to use the application

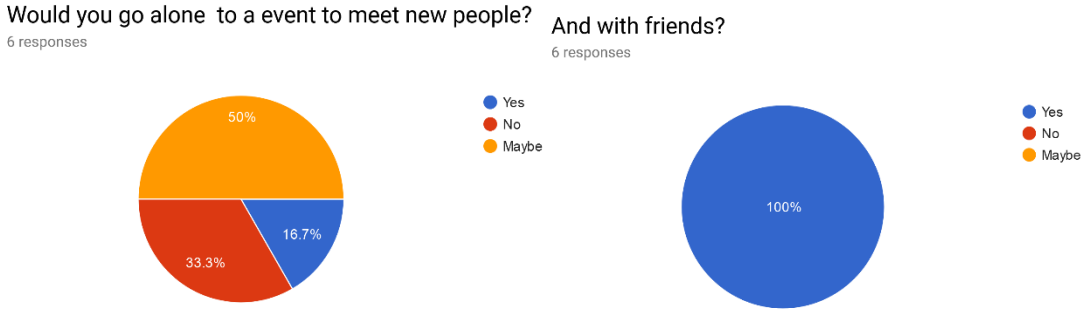


Figure 40: Answer, assistance accompanied or alone

In general, the people interviewed like the culture (Figure 41), some more than the others, but none hated it, but on the contrary, they do not go often to cultural events (Figure 42). Even if there would be promotions on the prices, just half the users would go more often (Figure 43).

Do you like the culture?

6 responses

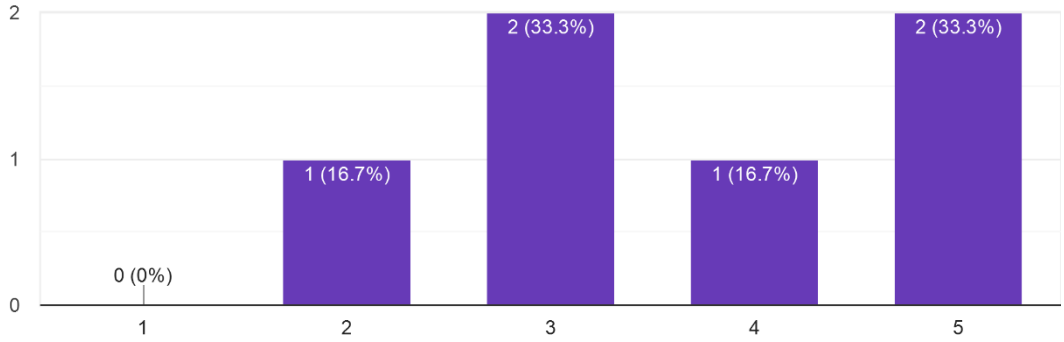


Figure 41: Answer, love for the culture

How often do you go to a cultural event?

6 responses

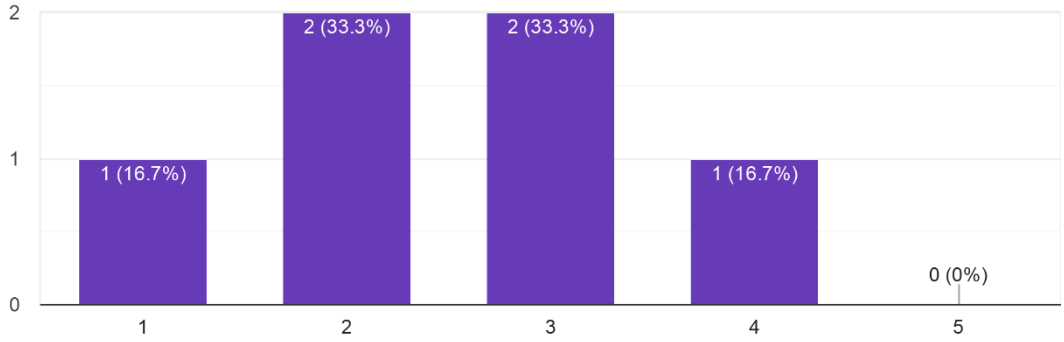


Figure 42: Answer, frequency of assistance to cultural events

Would you go more often with discounts?

6 responses

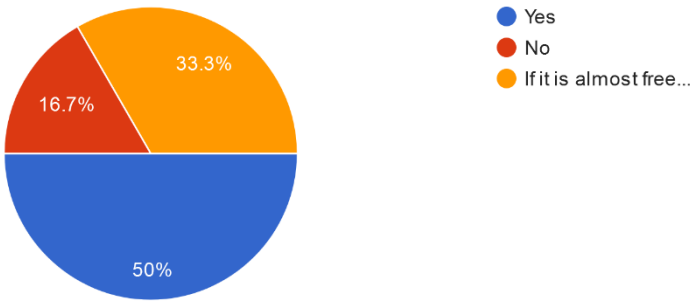


Figure 43: Answer, how about discounts?

The overall opinion about the application is good, the people liked it and found it useful, but there is still a lot of work to make the application more attractive to the users and make them the want to use it and share it will all their friends.

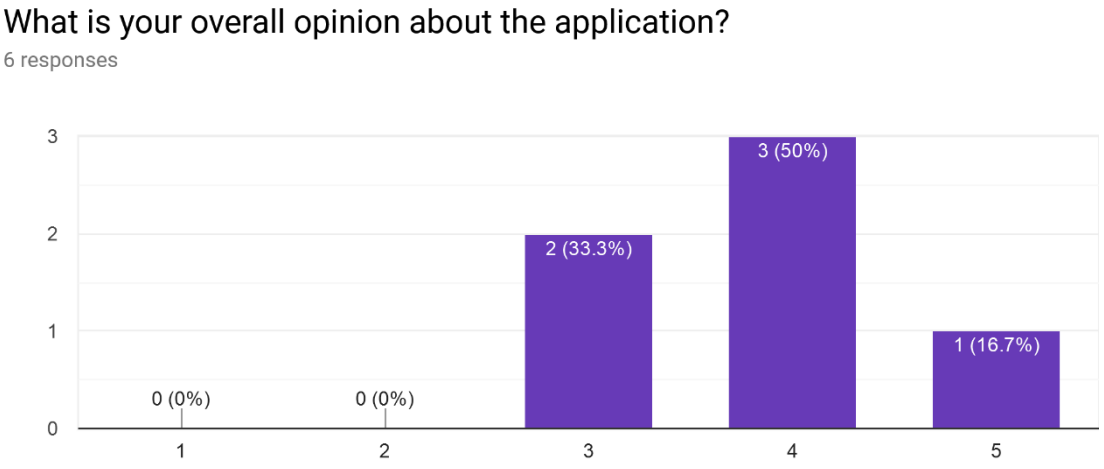


Figure 44: Answer, overall opinion

CHAPTER 6: PROJECT MANAGEMENT

In this chapter, it will be exposed the planning done to work on the project, and an estimation of the budget needed.

6.1 Planning

It has been used the GanttProject tool to make a Gantt diagram (*Figure 45*) from the planning of the project.

The planning has been done without distinguishing between weekdays and weekend, as it is a personal project. And the average working time per day is 2.5 hours, as the working time on weekdays was shorter than on weekends.

TABLE
PROJECT PLANNING

Name	**Begin date**	**End date**	**Duration (Days)**
Researching of the culture interest in Spain	31/05/2019	04/06/2019	3
Studying of mobile phones	04/06/2019	07/06/2019	4
Studying of mobile operating systems	07/06/2019	10/06/2019	2
Studying of Android platform	10/06/2019	24/06/2019	11
Studying of Firebase platform	24/06/2019	27/06/2019	4
Installing and configuring Android Studio	27/06/2019	01/07/2019	3
Configuring the database	01/07/2019	04/07/2019	4
Designing of the application	04/07/2019	10/07/2019	5
Development of main functionalities	04/07/2019	05/08/2019	23
Development of additional features	31/07/2019	12/08/2019	9
Testing	01/08/2019	16/08/2019	12
Report	12/08/2019	23/09/2019	31
Bachelor Thesis, total time	**31/05/2019**	**23/09/2019**	**104**

Table 58: Project planning

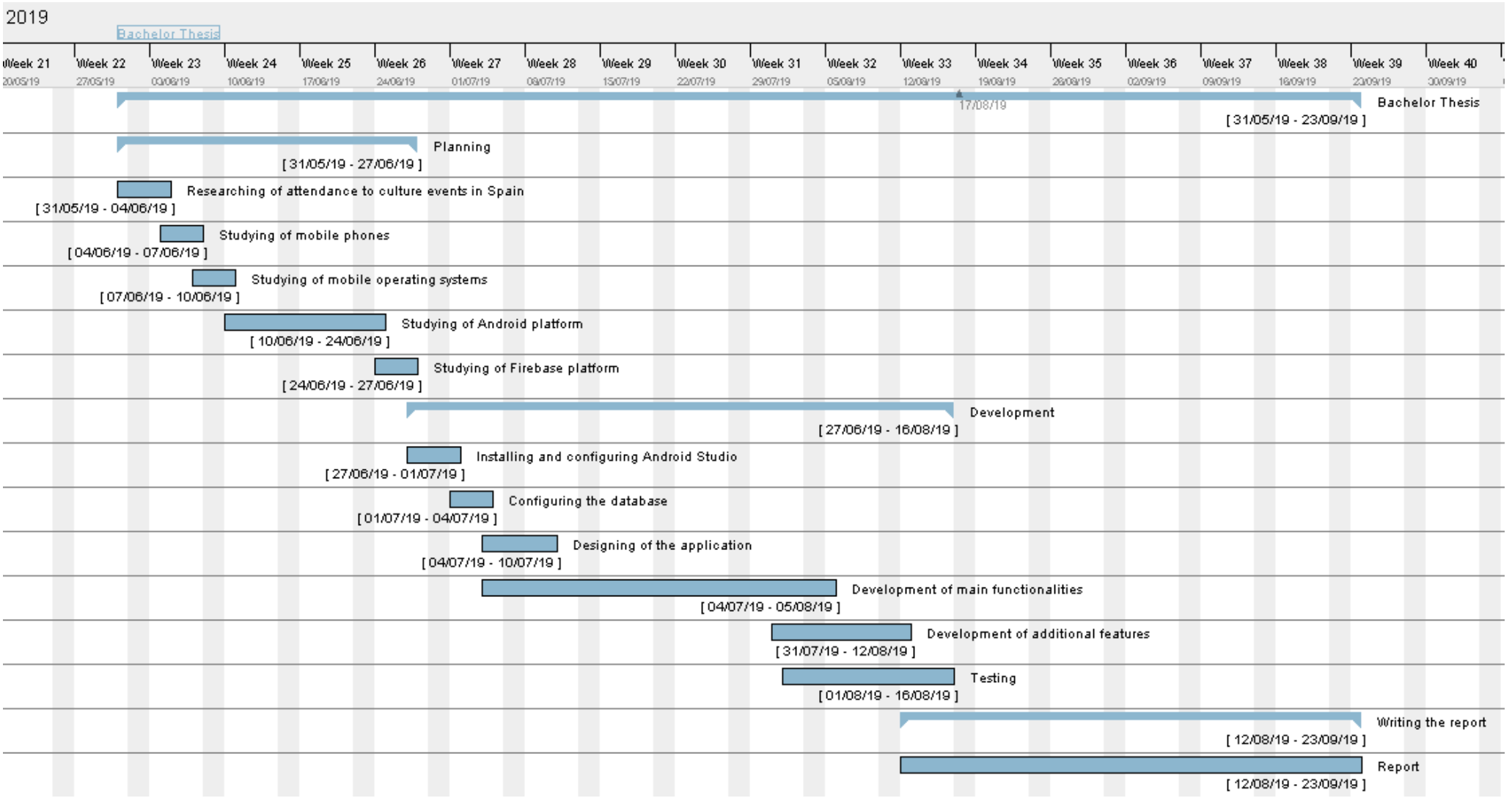


Figure 45: Gantt diagram

6.2 Budget

6.2.1 Human resources cost

The project has been developed by a junior Java engineer. The total cost is shown in the next table:

Name	Category	Salary/Hour	Hours	Total cost
-	Analyst	30 EUR/h	30	900 EUR
Chengyi Fu	Junior engineer	15 EUR/h	240	3900 EUR

Table 59: Human resources cost

6.2.2 Resources and material cost

To do the project, some hardware and software resources were needed, being all the software free of charge for the developer. The list of resources is shown below:

Hardware

- Desktop : 1000 EUR
- Monitor: 200 EUR
- Mouse: 40 EUR
- Mechanical keyboard: 150 EUR

Software

- Android Studio: Free
- Firebase: Free (Basic plan)
- Android SDK (Software Development Kit): Free
- JDK (Java Development Kit): Free
- GanttProject: Free
- Microsoft Office: Free (Student license)

In the next table it will be shown the total material cost:

Product	Price	Amortization period	Usage period	Total cost
Desktop	1000 EUR	60 months	3 months	50 EUR

Monitor	200 EUR	60 months	3 months	10 EUR
Mouse	50	60 months	3 months	2.5 EUR
Mechanical keyboard	150 EUR	60 months	3 months	7.5 EUR
Total				**70 EUR**

Table 60: Material resources cost

6.2.3 Total budget

The project total cost, adding the indirect cost (20%) and the VAT (21%), is shown in the next table:

Resource	**Cost**
Human	4800 EUR
Material	70 EUR
Indirect costs (20%)	974 EUR
Subtotal	5844 EUR
VAT (21%)	1227.24 EUR
Total	**7071.24 EUR**

Table 61: Project budget

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

In this chapter, it this exposed the conclusion reached once the project has been finished, and some ideas about the future improvements that can be made to the application developed.

7.1 Conclusions

The project has reached a first ending, and it can be concluded that the objectives exposed in the first section of this document, has been achieved: it has been acquired the knowledge to develop an Android application, connected to a remote database, using Java programming language, and it has been developed an application to connect people through cultural meetings.

It has been designed and developed an application that, based on the results obtained in the evaluation section, and among all, useful, which means that, with the enough work and dedication to it, it could totally achieve the goal it was designed for. Also, based on those results, it has been seen that there are still many aspects to improve to be able offer the user a great experience using the application. Slightly more attractive design, hints to help the user to understand the application, and above all, new features that improves the interaction among the users and a friend system, to encourage the users to use and share the application with their friends.

The hardest part of the project was to understand the structure of an Android application, how it works, and the relation and communication between the elements, as it was the first experience with mobile application programming. Using a NoSQL database for the first time was also confusing, as there are not defined entity relations and the queries are totally different.

To conclude, it has been satisfactory to do the project, as the objectives proposed has been achieved, but moreover, because the knowledges acquired by doing it will be very useful in future jobs and projects.

7.2 Future work

The application is in its very earliest version, having implemented only basic functionalities that allows it to fulfil the object of the thesis, but many extra functionalities

can be added, and improvements are needed to increase the overall satisfaction of the user, as seen in the evaluation.

In the next list, it will be exposed some of the functionalities that would increase the value of the application:

- Friends: the possibility to add other user as friends, allowing them to invite a friend to participate in an event.
- Share events: allow the user to create a link that references the event and share it in external applications, the user that accesses the link will need to download the application first and use it to see the event. This can bring many potential users.
- In-App tickets buying: there are free cultural events or expositions, but many of them are paid, so including a ticket buying system will facilitate the user to acquire tickets and, to have a system of promotions for the users that buys the ticket using the application, such as coupons or group discounts.
- Professional accounts: alongside the tickets system, it will need companies that are interested to support the application and use it as an alternative way to sell and share their events. These accounts will have special privileges over normal users, and they will be able to create promotions and coupons on the tickets.
- Likes: add a system of likes on the events for the users to vote. Additionally, another page of events is added but ordered by the popularity of the events, so the users can know what are the events that people likes most.
- Private messaging: add a chatting system, so the users are be able to communicate privately inside the application.
- Notifications: a service to notify the user when someone has joined the group they have joined, and when it gets full. More notifications in the future, such as a new message or a new friend request.

BIBLIOGRAPHY

[1] "Ayuda Ley Protección Datos," [Online]. Available: https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/. [Accessed 23 September 2019].

[2] "PowerData," [Online]. Available: https://www.powerdata.es/gdpr-proteccion-datos. [Accessed 23 September 2019].

[3] J. Hur, "Bebussinessed," [Online]. Available: https://bebusinessed.com/history/history-cell-phones/. [Accessed 20 September 2019].

[4] "Statcounter," August 2019. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed 20 September 2019].

[5] T. G. Portillo, "Gran Pausa," 2 September 2015. [Online]. Available: http://granpausa.com/2015/09/02/las-consecuencias-del-21-de-iva-cultural/. [Accessed 20 09 2019].

[6] ICC consultors, 1 February 2013. [Online]. Available: http://www.sgae.es/recursos/boletines/2013/InfoSGAE_numero5/Dictamen_Impacto_real_del_aumento_del_IVA_en_el_sector_de_las_Artes_Escenicas.pdf. [Accessed 20 09 2019].

[7] Á. García, "El Gobierno rebaja el IVA del arte," El País, Madrid, 2014.

[8] ADEPI, "Asociación para el Desarrollo de la Propiedad Intelectual," 28 June 2017. [Online]. Available: http://adepi.net/2017/06/28/iva-los-espectaculos-culturales-vivo-baja-al-10/. [Accessed 20 September 2019].

[9] E. País, "El IVA del cine baja del 21% al 10% con la aprobación definitiva de los Presupuestos," El País, Madrid, 2018.

[10] S. G. T. División de Estadística y Estudios, “Ministerio de Cultura y Deporte,” [Online]. Available: https://www.culturaydeporte.gob.es/servicios-al-ciudadano/estadisticas/cultura/mc/naec/portada.html. [Accessed 20 September 2019].

[11] “Wikipedia,” [Online]. Available: https://es.wikipedia.org/wiki/Crisis_econ%C3%B3mica_espa%C3%B1ola_(2008-2014). [Accessed 20 September 2019].

[12] M. J. P. V. y. S. S. F. Juan Prieto Rodríguez, “Observatorio Social de "la Caixa",” January 2018. [Online]. Available: https://observatoriosociallacaixa.org/indicador/-/asset_publisher/ATai9MyKZiYq/content/el-consumo-cultural_cuestion-de-gusto-o-de-precio. [Accessed 20 September 2019].

[13] I. Carballo, “Android Application Architecture,” Labs Ribot, 12 2015. [Online]. Available: https://labs.ribot.co.uk/android-application-architecture-8b6e34acda65. [Accessed 20 September 2019].

[14] Google Developers, “Firebase Realtime Database,” Firebase, [Online]. Available: https://firebase.google.com/docs/database/. [Accessed 25 June 2019].

[15] Google Developers, “Firebase authentication,” Firebase, [Online]. Available: https://firebase.google.com/docs/auth/. [Accessed 24 June 2019].

[16] Google Developers, “Cloud Storage,” Firebase, [Online]. Available: https://firebase.google.com/docs/storage/. [Accessed 26 June 2019].