ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

Mobile 2020-2021

FRAMEWORK AND DEVELOPMENT

Name: Mobile programming
Unit: APP development frameworks
Professor: Rubén Blanco
Course: 2020/2021
Author: Alexandre Calabuig Langa

INDEX

1.- FRAMEWORKS

1.1 Introduction

Frameworks are software structured to support application developers in creating new programs for one or more platforms on which the created application will run.
This type of software offers a multitude of tools to facilitate the creation of applications. On many occasions, companies that own operating systems offer these native tools so that developers can create more software for this platform, although there are also frameworks in which it is possible to develop for several platforms at the same time, although these may not have the depth of a framework especially dedicated to a particular operating system.

1.2 Types

As we will see later in more depth and comparing them with each other, there are different types of frameworks:

- There are frameworks dedicated to the creation of multimedia content such as videos and photos, although in this kind of programs it is difficult to find that you have to write any line of code and it is more focused on a more artistic profile.
- Dedicated frameworks are those which are made to work for a specific platform and these are usually more specialized than frameworks that cover more systems. They offer tools to squeeze the maximum out of the platform where the application is going to be directed.

Ilustración 1: Android Studio logo, one of the native frameworks

- On the other hand, there are frameworks which allow the applications developed in them to be executed in different operating systems, although sometimes these may include less depth at the code level.
- It also happens that these hybrid frameworks can be used on various platforms because those platforms can share a way of running certain programs, such as HTML or JavaScript, which are a standard for the creation of web pages, which can be visited and created through any modern operating system today.

Visual Studio Code

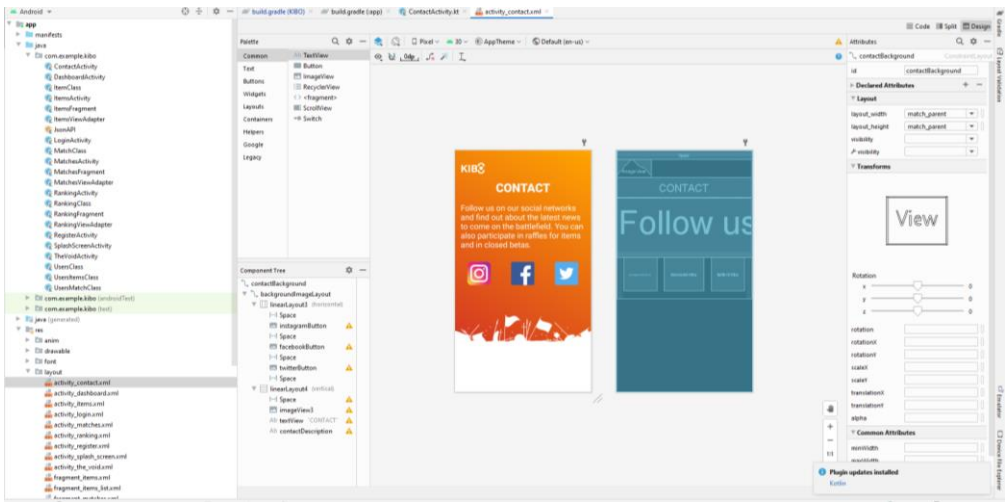
Ilustración 2: Visual Studio Code logo, used among many things, to develop web pages.

2.- ANDROID STUDIO AND XCODE

In this document we will focus mainly on the two main frameworks for creating mobile applications: Android Studio for creating applications on Android devices, and XCode, for developing applications that can be opened on Apple iOS devices.

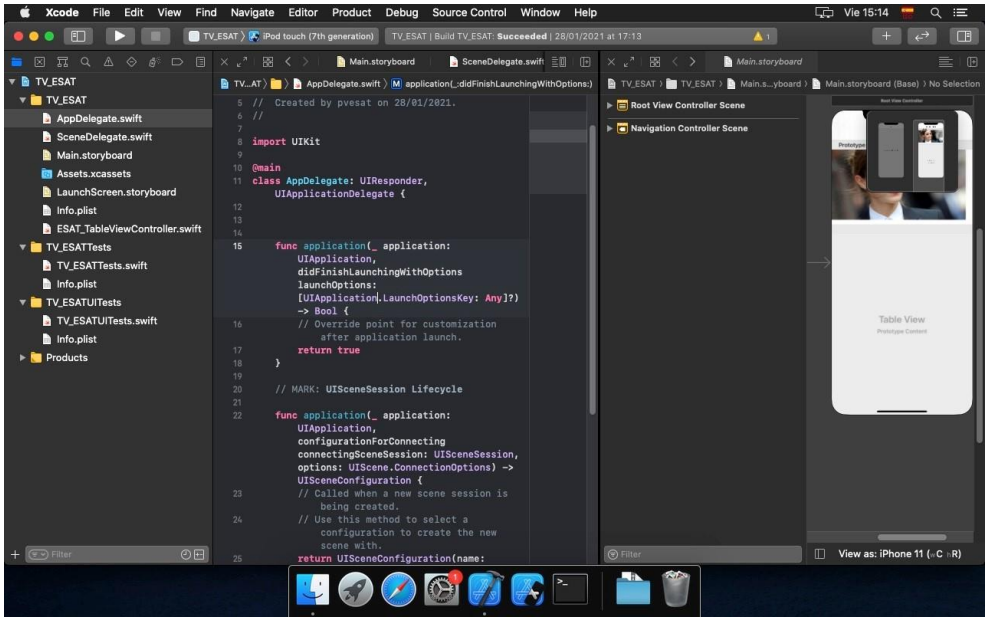
2.1 APP

The two frameworks are created so that the applications developed in them are displayed on mobile devices such as phones and tablets, so at the level of visual style (UI) may be similar, although at the time of development there are differences between the two.



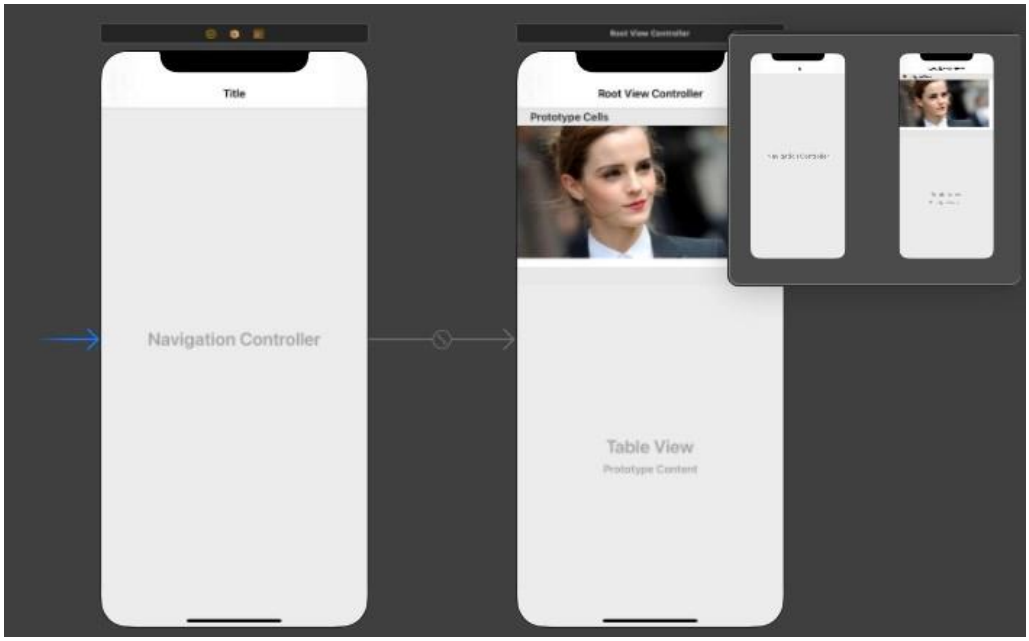
The image above shows the Android Studio interface. On the left side we can see the hierarchy of folders that contains the project.
In all this hierarchy you can modify both the code and the interface. Everything is separated into folders in which you will have to put all the desired resources and classes created.
Just to the right of it, and depending on the type of tab, there are the elements that exist in the interface of that particular file.

Separated in two, it shows a list of elements that can be included and on the other hand, the elements that are included.
In the rest of the image, the current state of that part of the interface is displayed and the characteristics of each element if it has been selected.
Android Studio uses a broad style so that you can understand where you are at any given moment.



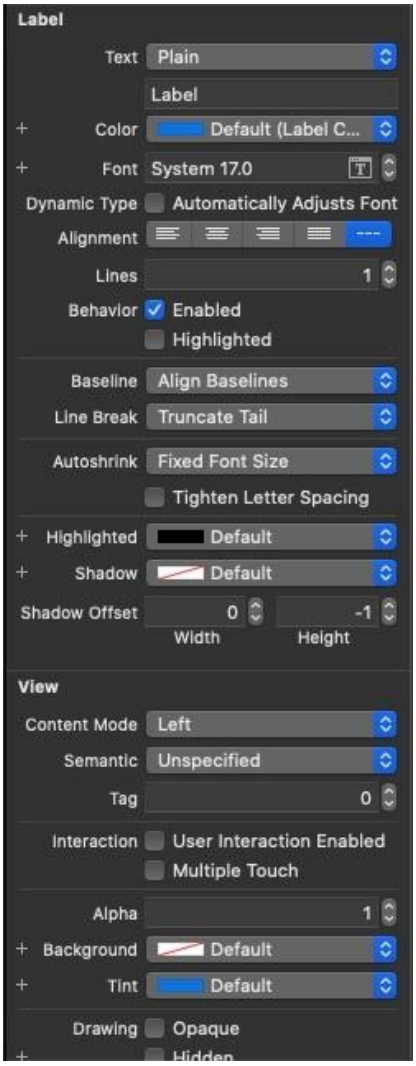
In XCode, as you can see above, it has a similar style in terms of the distribution of the parts of the program.
It is somewhat less intuitive to move between the hierarchy of folders, being able to have it less orderly at first, but as in Android Studio, you can create folders to suit the needs of the developer.
It is more compact with respect to Android Studio and the flow between the interface and the code is good only if you have a large screen, because it takes away a lot of space between one and the other.
To create elements and move through the interface, it is a bit more clumsy compared to its Android competitor.

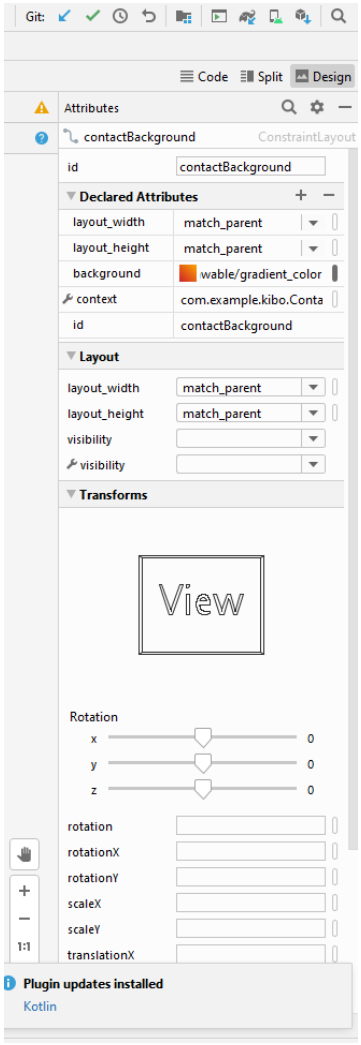
As for the differences within the creation of visual elements, in XCode, unlike Android Studio, you can move through the mouse, the element that will be the entrance to the application, and not only that, but you can select by clicking and dragging, other scenes that will be accessed depending on which button is pressed or not.
This fast way of creating connections between levels does not have Android, since in Android Studio you can only see one interface at a time, and the connections between screens must be done through code in their respective classes.



In the image above, through the blue arrow marked, it is indicated from which screen you want to initialize the application, and from there, select where you want to continue.

Another key point to show is the difference between the two ways of displaying the data of an element in the interface. In this case, of a Label. The left side shows how it looks in XCode, with few separations, which can be confusing when looking for certain variables to change. In contrast, Android Studio has a better separation and the sections are excellently marked, being able to collapse and expand them to see only the desired features.





So, for a developer, being able to modify or change the desired data is much easier and visual in the Android framework.

To place the elements on the map, they are controlled through constraints, which are more visible and easier to understand in Android Studio, since it is more visual and simpler than with respect to XCode, although with XCode you see the constraints created in a more organized way on the left side of the hierarchy of elements in the scene.

2.2 Code

Curiously, the fight between Android and iOS has made the two programming languages used very similar in their use, even though they have different names.
On the Apple side, its programming language is Swift, and on the Android side, Kotlin is used.

	SWIFT	KOTLIN
Variables, Constants and NULL	var, let, nil	var, val, null
String interpolation	var x = “dog” print(“\(x)”)	var x = “dog” print(“${x}”)
Functions	func start()-> Bool	func start(): Boolean
Self object reference	self.x = x	this.x = x
Arrays	[“a”,”b”]	arrayOf(“a”,”b”)
Dictionaries	[“a”:5]	mapOf(“a” to 5)
Class contructor	init	constructor

As the main syntaxes when programming, the ones shown in the table are usually the most used and the ones that tend to cause the most confusion for developers when switching from one platform to another.

Some key developmental differences to keep in mind include:

- There is not a data class in Swift.
- Delegated Classes and properties are missing in Swift.
- Kotlin does not have struct or passing data by value.
- The two implement different memory management systems. Kotlin uses Garbage Collection and Swift uses Automatic Reference Counting, so Swift is considered to manage memory more efficiently.

Although it has not been mentioned, Android also allows its development using Java, but little by little it is being recommended that it be programmed in Kotlin as it is totally focused on mobile development, unlike Java, which has other types of functionalities and another way of working the code.

On a general level, as both languages are created for mobile programming in their respective software, both have great performance and it is only in how the frameworks can handle the data that you can tell which language is more comfortable to work in.

3.- NATIVE AND HYBRID FRAMEWORKS

As briefly discussed in point 1, there are several types of frameworks, some called "Native" which only focus on one type of operating system, and hybrid frameworks, which can be developed to be executed for several types of software.

The main differences between the two are as follows:

- Native frameworks are focused on the development of applications for only one platform. Mostly this type of frameworks are directed towards mobile platforms such as Android and iOS. This type of frameworks are specialized by having only one type of platform to work on.

- There are native frameworks that only work on the operating system on which it is being developed, as is the case of XCode, which can only run on iOS, so, in order to be able to program on this type of operating system, the developer must have software that is the same as the target development.

- Android Studio on the other hand, although its final software will run on Android devices, it can be used on Windows and Linux systems.

- Hybrid frameworks have the ability to develop for different platforms. An example could be Unity, which is a video game creation framework that can run on iOS and Windows. Unity can create applications for Windows, Android, iOS and other platforms. Although it is a very complete program, it never reaches the complexity at the code level that native frameworks do. Native frameworks can access the features of the software itself in a deeper way than hybrids.

4.- BENEFITS OF ANDROID STUDIO AND XCODE

Both frameworks have their advantages and disadvantages, but we will focus on the advantages of developing applications in each.

4.1 Android Studio

- In order to use this software it is not necessary to purchase a machine with the same type of software as the one to be developed, making this more attainable for more developers.

- The ability to design the UI is very simple and intuitive.

- The IDE is very prepared for Android development with continuous updates because the owner of the software is the company that owns Android, so it is the Android framework that is less obsolete.

- When it comes to publishing the application, it is simpler than in other software, in addition to the fact that everything possible is provided for the successful and error-free publication of the APP.

- It is developer friendly, as plugins and updates are easily installable and there is plenty of documentation for optimal development.

- It has a complete integrated application testing system, so you can check if everything is working correctly.

- It is programmed in the optimal language designed by Android for application development.

- The code with which these applications are developed is open source, so everyone can see how it is designed and written inside.

4.2 XCode

- It has less fragmentation than Android. iOS, having a closed software system only on Apple devices, development in XCode is easier to improve the perfection of the software.

- To publish the application on iOS, there is an exhaustive validation process, but it allows you to publish the application with the highest possible quality.

- It has a complete integrated application testing system, so you can check if everything is working correctly.

- It is programmed in the optimal language designed by Android for application development.

- The code with which these applications are developed is open source, so everyone can see how it is designed and written inside.

- As you type, you may see compiler errors or issues and also a message detailing helpful information.

5.- THE ANDROID STUDIO

There are two points for which the development of the application in Android Studio has been chosen. The free cost of the tool and not having to have a dedicated device for it, has been one of the main reasons for this, although there are also others.

5.1 Technical features

- Besides the fact that there is a portable version that you can take to other computers to work on other sites and continue where you left off, Android Studio offers you the possibility to connect to a Git repository to work in a team and save the progress in the cloud. Android Studio

itself offers an "ignore" file which allows you to upload files to the repository without excess and unnecessary files, so version control is faster.

- The interface design is, to my taste, much better than other frameworks, so this application has been chosen to be able to create the design of the APP screens in a faster and clearer way.

- The ability to be able to program in two programming languages depending on the developer's taste and skills, being able to do it in Java or Kotlin, and having fully detailed documentation of the two types in the official Android pages.

- Ease of integration of plugins and new versions of the target SDK. With a few clicks, new types of devices can be integrated.

5.2 Code maintenance and updates

- Being an official Android software, it is the first software that will receive the latest changes, improvements and new features on the market. So, if you want the application to be fully updated on multiple devices, using this framework is the best option.

- If you need to use old code in a new version, the framework will notify you of all the changes and tell you how to update the code base in a few simple steps.

- Being connected to Git services, version control is easy and you can always revert to previous versions if there have been errors in the introduction of new features.

6.- CODE AND INTERFACE REVIEW

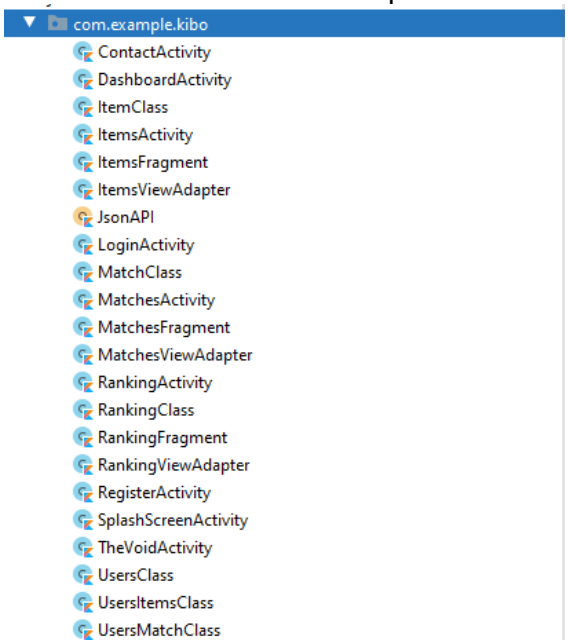
6.1 Code

From the very beginning of the work planning, it was taken into account that the application should contain Activities, Data Class, Objects and the use of Fragments and View Adapter, as well as being able to connect to the Internet to collect data through JSON.

First we made a scheme of what classes should exist in the application and with what type of variables, then we made the JSON and the necessary tests to verify that the data collection was done correctly, and finally, the design of all the Activities that were in the application.

The main objective before starting the application, was to show the list of games of a user, with the ability to see the items in possession and to read the descriptions that these could have. Practically everything was carried out with the exception of an extension of the Matches section, where the original idea was to be able to click on each game and

be able to see more detailed information. The latter was finally discarded because it would complicate the type of code too much, taking into account that all the JSON have been created by hand and more would have to be created to be able to relate some tables with others.
This was not done and it was decided to improve the visual aspect.



Here we show the total number of classes and activities created as well as a sample of the code created to manage the ranking of player scores, collecting data from several tables and merging them into a new one.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_ranking)
    setBackgroundColor()

    for(j in 0..JsonAPI.matchesArrayAPI.size - 1){
        for(i in 0..(JsonAPI.matchesArrayAPI[j].usersMatch.size - 1))
        {
            // Find in RANKING CLASS if exist and ID that matches the user of the current match
            val found = rankingArray.find { it.id == JsonAPI.matchesArrayAPI[j].usersMatch[i].user}

            // If it is not null, to that found that it returns, we add the points to the ones it already has
            if(found != null){
                found.points = found.points?.plus(JsonAPI.matchesArrayAPI[j].usersMatch[i].kills!!)
            }
            else{
                // If it is null, it means that this player is not inserted in the ranking, so in the array of users we find the name of that player
                val f = JsonAPI.userArrayAPI.find { it.id ==  JsonAPI.matchesArrayAPI[j].usersMatch[i].user}

                // f the find exists, it will have found the match player with the user list
                if(f != null){
                    rankingArray.add(RankingClass(f.id, f.nickname, JsonAPI.matchesArrayAPI[j].usersMatch[i].kills
                        )
                    )
                }
            }
        }
    }

    var f: Fragment = RankingFragment(rankingArray)
    rankingArray.sortByDescending { it.points }
    supportFragmentManager.beginTransaction().add(R.id.containerRank,f).commit()
```

Due to the intense work, the development of the application was done quickly, finishing two weeks earlier than initially planned.
For future improvements, all the code has been documented and both classes and functions have the appropriate names for future understanding.

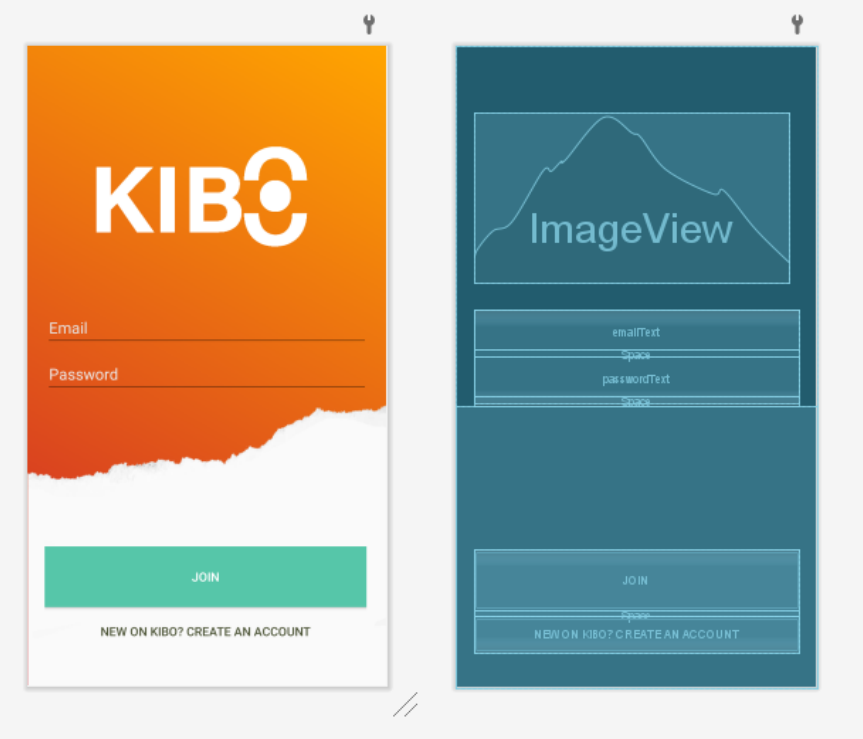
6.2 Interface

In order to make the most visible part of the application, it took less time than expected due to the speed of visualization of the objects to be displayed on the screen.
The graphical part was finished before the code part, in order to be able to help me graphically on the changes that I inserted through the code in Kotlin.

Animations and permanent color changes of the wallpaper were made through a simple button, being able to choose the color to the user's taste.

The planning to be able to make the desired icons and images, was somewhat longer than expected, since we wanted all the elements to be displayed correctly and with the same visual style.

A lot of time was also invested in the placement of the graphic elements per screen so that they would look good on different devices. Android, having a huge number of different devices and sizes, makes the development on the visual side a little more laborious than usual.



Being able to visualize in two different modes at the same time, allowing to see how many layers are superimposed on others, is really useful to know if we are doing a correct job of positioning and visualization for the final result.

It was expected that the final visual quality would have a more optimal result than what was obtained. The cause of this is the knowledge in the design of icons and various elements, in addition to the placement of the same and that it could be seen correctly on all Android devices.

Despite finishing the work ahead of schedule, we could have insisted a few more days to improve some graphical elements.

6.3 How Android Studio helped me

As it has been said in other points (in point 5, for example) about Android Studio and its strengths, the amount of official documentation and the speed of the framework, have helped me to make a correct and functional work, being able to be executed in several Android devices without any problem.

7.- HOW XCODE COULD IT HAVE HELPED ME

In order to find out how using XCode could have helped me, I have been able to program in this software and this could have helped me in the following points:

- Knowing another framework for another platform means that the knowledge I could have gained by doing the work on this platform would have been very similar.

- The similarity between the way of working between one and the other would have helped me in the future to change framework and programming in a more comfortable way.

- Being able to work with XCode could have been helpful when designing the interface of my application. The types of formats and resolutions for Apple devices are well defined and re-gulated, so I would know with full certainty if the visual elements would have looked good on the actual devices.

- The fact that a developer at Apple is better paid than at Android could have helped me in the future when it came to finding a job ahead of other programmers.

- By knowing first hand, although not much, the XCode software, I can open the doors of an important company in the future having touched two types of framework for the same market.

8.- DEVELOPMENT PROCESS AND FUTURE

Looking back and with some time after finishing the development, some planning and development processes could have been improved, although, from my point of view, everything has been executed in an optimal way with respect to the time we had at that moment.
Before starting the development of the project, small exercises were carried out on features that were to be introduced in the final project. Therefore, I came quite prepared for the planning of the lessons and activities.

As points to improve for future developments, if I know that I have to introduce certain features to the project, better to test them in other projects separate from this one and not use it as a testing ground, since, when inserting new classes or elements, these can affect the development of the application. In this case, I have made the APP by myself, but in larger projects and with more people, I could not afford to use the official project to test code that may then be left in the final version by mistake and this affects the rest of the colleagues.

With all that I have learned during these months of development and research to make this application, I look forward to the future with more information to be able to apply for various jobs.

Knowing various types of frameworks and being able to create applications with them is a good starting point to expand knowledge for the future.

9.- BIBLIOGRAPHY

The images shown in the document are taken by me, except for the Android Studio (Google) and Visual Studio Code (Microsoft) logos.

- https://tipos.com.mx/tipos-de-framework
- https://www.offerzen.com/blog/which-mobile-app-framework-should-you-use
- https://medium.com/better-programming/swift-vs-kotlin-the-similarities-and-differences-you-should-know-b2f1be201888
- https://www.mobileappdaily.com/2018/07/25/swift-vs-kotlin-debate
- https://willowtreeapps.com/ideas/swift-and-kotlin-the-subtle-differences
- https://www.androidauthority.com/developing-for-android-vs-ios-697304/
- https://medium.com/@developer45/5-advantages-of-android-app-development-f06c5a9283d1