



# **Physical Activity Motivator**

by Daniel Muñoz Pérez

---

Final Degree Project

Bachelor's Degree in

Video Game Design and Development

Universitat Jaume I

May 2020, Castellón

## Acknowledgments

Thanks to my parents for investing their life in me and trusting and encouraging me to continue studying. Thanks to my friends, for supporting me these months, to all those who gave me different ideas, those who have dedicated part of their time to help me to solve some of the problems that have been appearing, and, of course, the knowledge that the teachers taught me throughout the degree.

Without all of you, this would not be possible.



## Abstract

This document explains how this project about an motivating of physical activity application was developed, which is consists in make the user to move within the campus and could use it as a guide through it.



## Contents

<b>1. Introduction .....</b>	<b>07</b>
1.1 Motivation of work.....	07
1.2 Objectives.....	07
1.3 Context and initial state.....	08
<b>2. Planning and resources evaluation.....</b>	<b>09</b>
2.1 Planning.....	09
2.1 Resource Evaluation.....	13
<b>3. System Analysis and Design.....</b>	<b>15</b>
3.1 Requeriment Analysis.....	16
3.2 System Architecture.....	16
3.3 Interface Design.....	17
<b>4. Work development and Results.....</b>	<b>20</b>
4.1 Work development.....	20
4.1 Results.....	31
<b>5. Conclusion and Future Work.....</b>	<b>32</b>
5.1 Conclusions.....	32
5.2 Future Work.....	32
5.3 Link to repository and app.....	32
<b>6. Bibliography.....</b>	<b>33</b>



## 1. Introduction

This section will provide a compilation of what will be done during the development this project. The motivations that propitiated this project will be exposed and the objectives set out in the beginning, which will contribute to better understand the evolution of the project.

### 1.1 Motivation of work

Having to use a game engine that I already know, such as Unity3D<sup>1</sup>, for the development of a mobile application, the purpose of this and also, the study and implementation of plug-ins and tools that I had not used previously is what called my attention to make this project.

### 1.2 Objectives

The main objective of this Final Degree Project is to encourage physical activity at the university campus and, at the same time, serving as UJI's guide / tour for new/ ERASMUS students.

Also, the application must be capable of tracing the player's location the player using the mobile service location and save the user's data (such the login info, points of interest visited, etc) in a database using Google's<sup>2</sup> Firebase API.

---

<sup>1</sup> Unity3D is a game engine, it refers to a series of programming routines that allow the design, creation and operation of a video game

<sup>2</sup> The application programming interface (API) is a set of subroutines, functions, and procedures that a certain library provides for use by other software.



### 1.3 Context and initial state

At the beginning of the year, after finishing the first semester exams, the work schedule passed without too many setbacks and I was investigating the different known options that I had at my disposal to start the project. But in mid-February, my father was hospitalized due to covid and two weeks later he died, which was a serious blow and I was away from this project for almost a month, this because my mind was elsewhere and I had to assume certain responsibilities and do the necessary paperwork in these situations.

In addition, various problems arose with the tools I was using in the project which slowed me down a lot because I had to search for other alternatives, but these problems will be explained later.

Summarizing, the project really started at the end of March, which left me with much less time than expected and forcing me to make several changes in the initial schedule.

## 2. Planning and resources evaluation

This part of the report will explain how the project was initially planned. In addition, it will briefly discuss the costs that it would have had if it were a commercial project.

### 2.1 Planning

As has been said before, the initial planning (see Figure 2.1) has been significantly modified, both in hours and in content, since the research for information required more time than expected, as well as the implementation of the used tools.

Fortunately, when re-planning, based on my experience in other projects, tasks usually take longer than expected and unforeseen problems may arise, so I was more flexible when setting objectives.

Tasks	Planification	Total time (h)
<b>Research</b>		<b>30 h</b>
	Learn about ArcGIS Map SDK.	15 h
	Study if SmartCampus can be used in Unity and how.	10 h
	Short investigation about Vuforia functionalities.	5 h
<b>Programing Work</b>		<b>230 h</b>
	Implementation of ArcGIS Map.	35 h
	Create the ranking system.	25 h
	Implementation of SmartCampus.	25 h
	Implementation of Vuforia.	20 h
	General code work	125 h
<b>Testing</b>		<b>15 h</b>
<b>Documents</b>		<b>25 h</b>
	Final Degree Presentation	5 h
	Final Degree Work Report	20 h
<b>Total:</b>		<b>300 h</b>

**Figure 2.1** The initial planning of the project.

The final planning can be seen in the following table (see Figure 2.2). As can be noticed, there is a planification phase in each task, this is made in order to reduce improvisation as much as possible and advancing the project in the most orderly way possible.

	TASK	SUB TASK	DURATION (H)
<b>1</b>	<b>Research</b>		<b>55</b>
	1.0	Geoposition Tools Investigation	30
	1.1	Database Tools Research	20
	1.2	Planification	5
<b>2</b>	<b>Programming work</b>		<b>195</b>
	2.0	Implement the geoposition system.	30
	2.1	Create and implement the database in Unity.	60
	2.2	Implement the navigation system.	10
	2.3	Augmented Reality Plugin implementation.	15
	2.4	Generic code work (bug solutions, make the UI, etc)	75
	2.6	Planification	5
<b>3</b>	<b>Testing</b>		<b>15</b>
	3.0	Test the database system in the Android build	5
	3.1	Test the geoposition system in the Android build	5
	3.2	Test all together in the Android build	5
<b>4</b>	<b>Documents</b>		<b>35</b>
	4.0	Make the Final Degree Project Memory	20
	4.1	Make the Final Degree Project Presentation	15
		<b>Total:</b>	<b>300</b>

Figure 2.2

The final time distribution of the project.

In the end, the difference in hours between the initial and final planification for each task is not too much, it is only significantly greater (about 25 hours) in the research phase since it took much longer than expected. The time increase in this task, in combination with the extra time needed to carry out this report, caused the time dedicated to the programming part to be reduced, but due to my experience in this field is , it was not a big deal.

Down below you can see a Gantt chart (see Figure 2.3) that i have made to visually represent the distribution of tasks over the weeks. As you can see there are several tasks that overlap, this is because that week I was working on different tasks at the same time, fact which has intensified in recent weeks.

TASK	SUB TASK	DURATION (h)	FEBRUARY				MARCH						
			WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8			
<b>1 Research</b>	1.0 Geoposition Tools Investigation	30											
	1.1 Database Tools Research	20											
	1.2 Planification	5											
<b>2 Programming work</b>	2.0 Implement the geoposition system.	30											
	2.1 Create and implement the database in Unity.	60											
	2.2 Implement the navigation system.	10											
	2.3 Augmented Reality Plugin implementation.	15											
	2.4 Generic code work (bug solutions, make the UI, etc) Planification	75											
	2.6	5											
<b>3 Testing</b>	3.0 Test the database system in the Android build	5											
	3.1 Test the geoposition system in the Android build	5											
	3.2 Test all together in the Android build	5											
<b>4 Documents</b>	4.0 Make the Final Degree Project Memory	20											
	4.1 Make the Final Degree Project Presentation	15											
<b>Total:</b>		<b>300</b>											

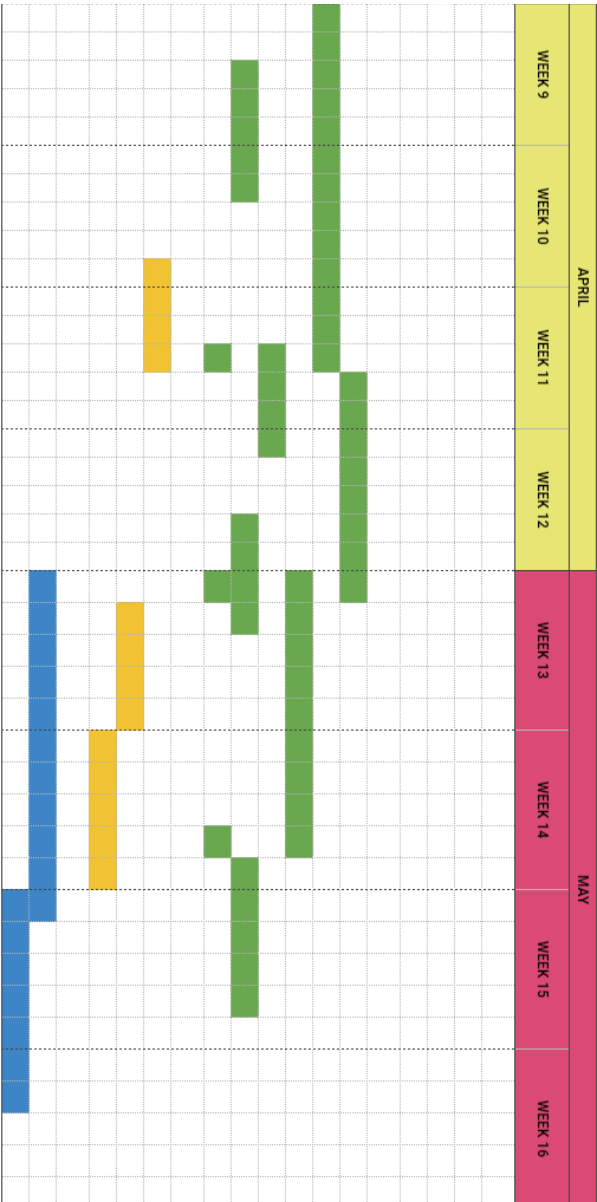


Figure 2.3 Gantt diagram of the project.

## 2.1 Resource Evaluation

If this project had been a paid one, assuming a 16'25€ per hour , the average salary of a junior programmer (31.200€/year), the cost of the project would be, more or less, 4.874€ (300x 16'25).



## 3. System Analysis and Design

This chapter will review all the topics related to the analysis, design and architecture of the system

### 3.1 Requirement Analysis

The requirements can be divided in two types: the functionals and the non-functionals. The first one refers to all those functions of the system described as a set of inputs, outputs and its behaviour, meanwhile the second type cover the conditions of the project design and its implementation.

#### 3.1.1 Functional Requirements

- If the app user do not have an account, given an username, email and password, create the account.
- If the app user do have a created account, given an email and password, log in the database.
- Obtain de mobile device location to place the player in the map.
- Given a location, calculate the route from the player to the POI<sup>3</sup> and show it in the minimap or AR.<sup>4</sup>

#### 3.1.2 Non-Functional Requirements

- The app must be easy and intuitive to use.
- The project must work in most of Android phones.
- The code must be understable and readable in order to be easily scalable (include new functionalities).

---

<sup>3</sup> POI (Point of Interest)referes to a specific location that may have something interesting.

<sup>4</sup> Augmented Reality (AR):Technologies that allow a user to visualize part of the real world through a technological device with graphic information added by it.



### 3.2 System architecture

Due to the plugins and assets used in the app, a phone with the following characteristics is required:

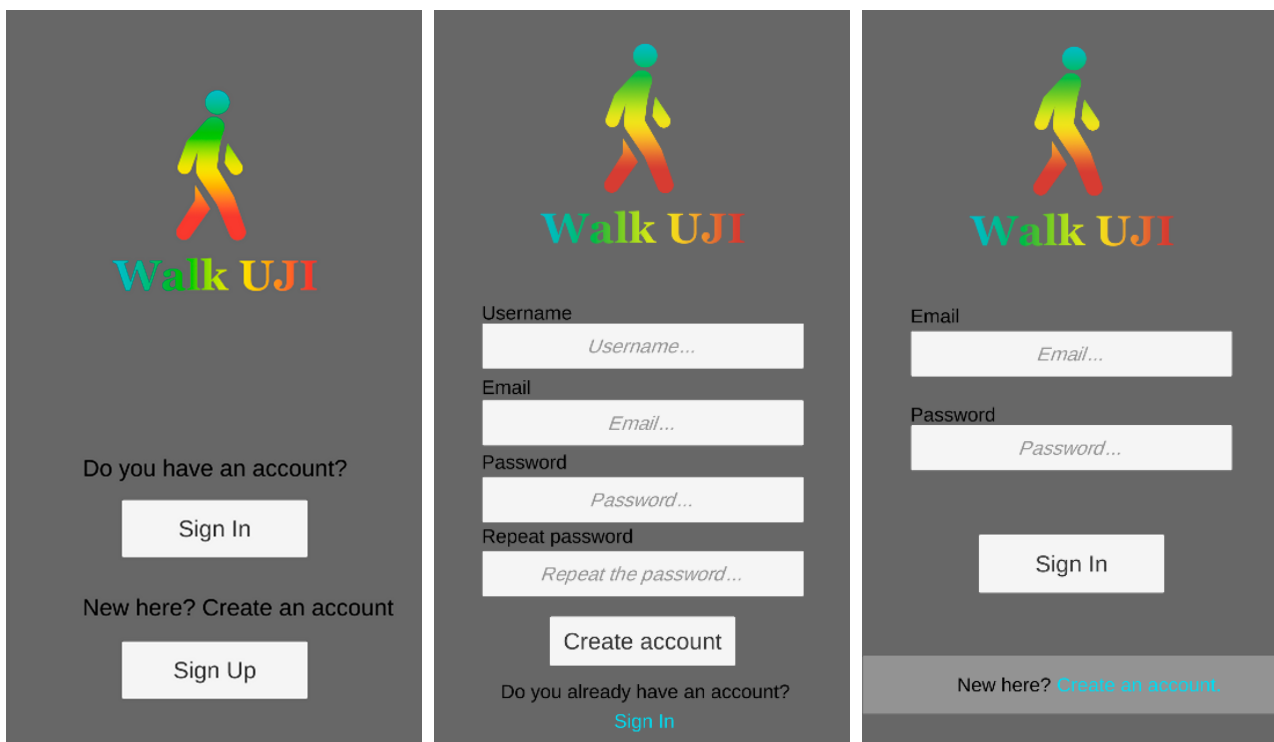
- The processor should use 64-bits architecture.
- 4GB of RAM, which should not be a problem if we take in to account that the standard is 6GB. The more RAM the better.
- The device must have, at least, Android Nougat (version 7.0). Most of Android phones nowadays have Android 9 or 10 (latest is version 11).

In the development of this project, the phone's characteristics which were used to debug are:

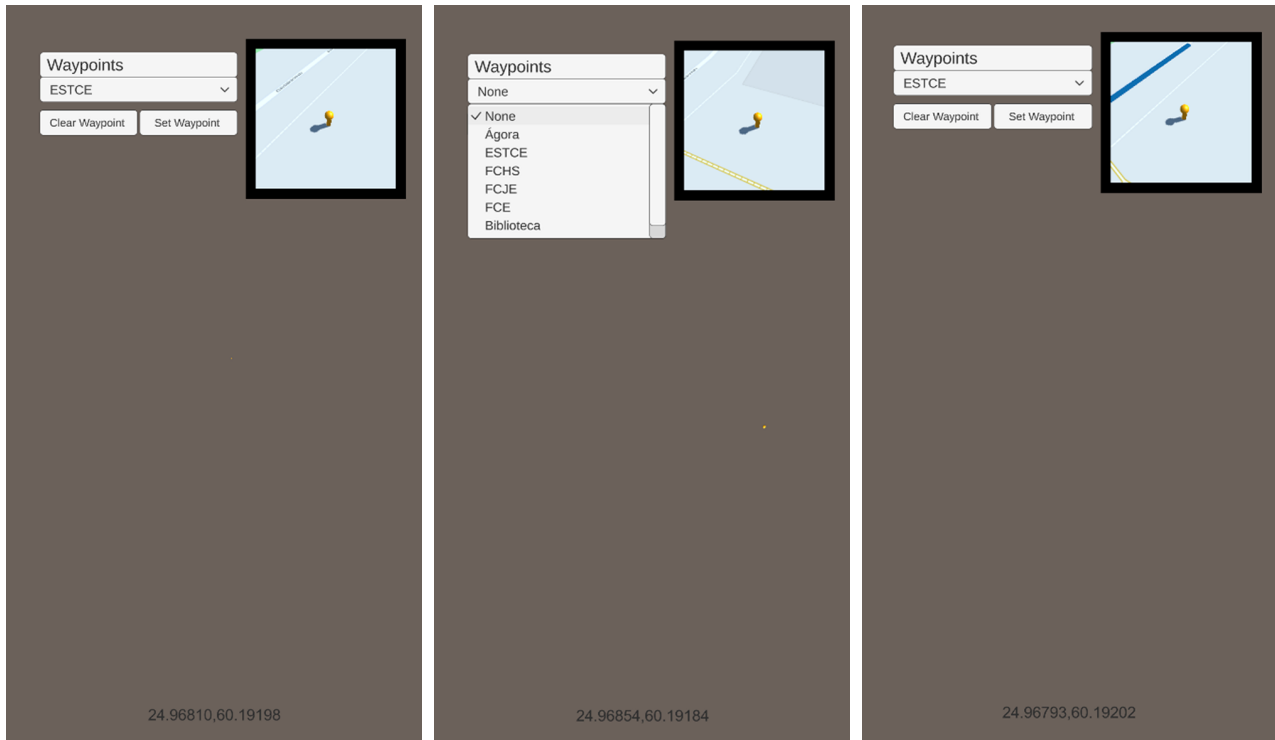
- Processor: Snapdragon 855+
- RAM: 8GB
- Android version: Android 11

### 3.3 Interface Design

The current interface is subject to changes, these photos show how the app look during its development. Some menus and options to add are missing, but hope it helps to understand the idea.

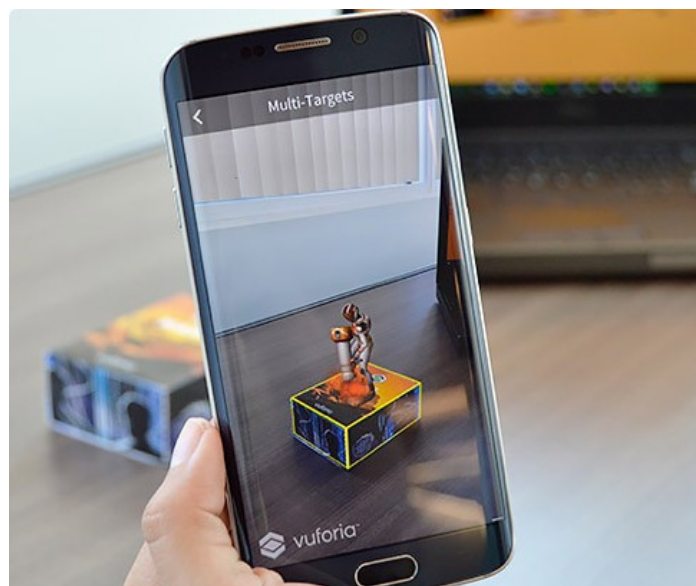


Start screen interface



Main screen interface

The background colour will be the Vuforia camera.  
(example down below)





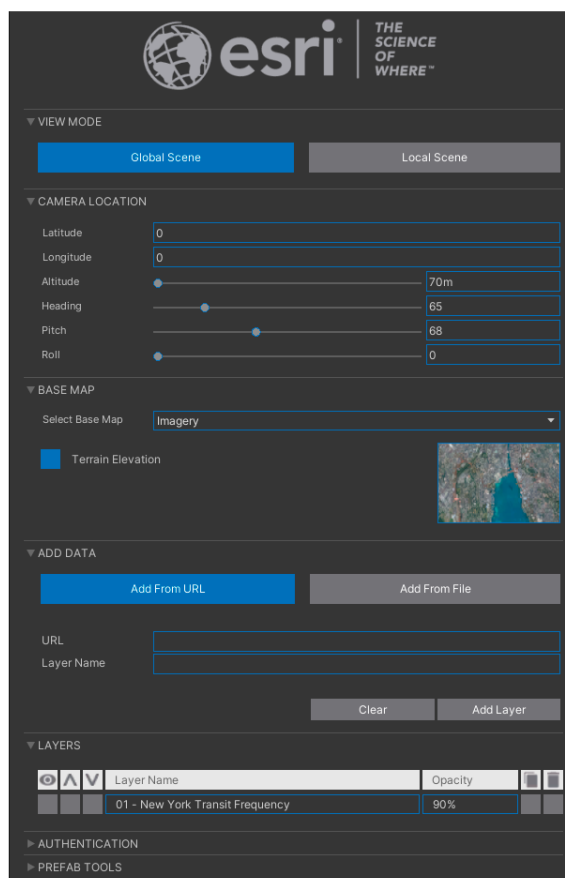
## 4. Work development and Results

This section of the report explains the path followed to develop this project, as well as the reasons why the different decisions were taken.

### 4.1 Work development

By the nature of my project, an application for Android mobile devices developed with Unity3D, my first task was to make a research for the possible tools to use in order to accomplish the requirements.

The first task I wanted to deal was how to calculate the player's position in the world and display it in the app, so I started by researching the Unity plugin ArcGIS Maps, as it was the first geopositioning tool that my tutor suggested me.



At first it seemed correct for the uses it was intended to give it, allowing the player to be placed in specific world coordinates, put markers, building for Android, etc. So, once I had read the tutorial-type documentation that ArcGIS provides to users, I decided that would be much better to implement it (see Figure 4.1) in Unity and make several tests on the PC.

**Figure 4.1** ArcGIS's inspector UI.

These tests were going relatively well (with the difficulties that involve the use of a new tool), the basics ideas planned for the project was achieved, such as the minimap and one POI (just for testing) and that worked fine in PC, but when the project was build and ran it in the phone, nothing was rendered,<sup>6</sup> just a black screen was showed. That was really confusing, a solution needed to be found, but because of the state of the tool (beta)<sup>5</sup> there was not much info about that issue, but after about a week of searching and testing, some useful information was found. The ArcGIS documentation explained<sup>7</sup> that it requires a specific type of rendering pipeline<sup>8</sup>, HDRP<sup>8</sup> or URP<sup>9</sup>. Due to my personal lack of experience making Android apps using Unity, I made the mistake of create the project with HDRP, which is not supported for Android builds, so the whole project was remade using URP, but it ended like the las try, a black screen. Almost two weeks had passed and I did not have anything, non a working geoposition system neither solutions. Then the setback previously mentioned in chapter 1.3 kept me away from the project for nearly a month. When I came back to work, disappointed by the ArcGIS fail, I started looking for an alternative for the geoposition system, as well as the research about tools avaiables for the database part.

---

<sup>5</sup> Render: Process of drawing a scene on the computer screen.

<sup>6</sup> Beta: First full version of a computer program or other product, which may be unstable but useful for it to be considered a technical preview.

<sup>7</sup> Render Pipeline: Conceptual model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D screen.

<sup>8</sup> HDRP(High Definiton Render Pipeline): To create applications to a high graphical standard.

<sup>9</sup> URP(Universal Render Pipeline): To create optimized graphics applications.

For the database, Firebase was, a platform developed by Google for creating mobile and web applications which provides some services (see Figure 4.2), among them, database functions and authentication systems, just what was needed. Once the documentation was read, these services had to be implemented, little by little, one after the other. Luckily, in their website and YouTube's channel they provide tutorials for each service they offer for each platform (Android, Web, Unity, etc). So these tutorials were followed and the basic Log-In system with e-mail and password were implemented, wich is fine but the login wanted for the project requiered a username when the player signs up for the app, but they do not offer that type of register system (see Figure 4.4), so ended up leaving that log-in method, (just e-mail and password) but asking for a username anyway at the moment of register and saving it as a object of the custom class "Player"(see Figure 4.3) and then upload it to the database. So, the next step was clear, have a database to store the data.

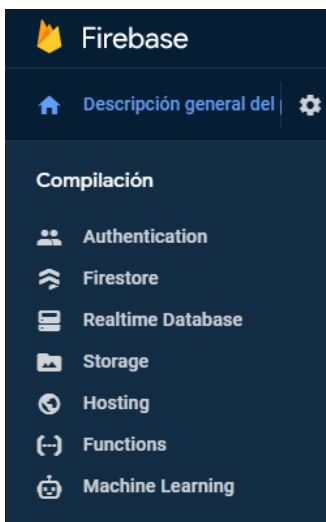


Figure 4.2 Firebase services available

```
public class Player : IComparable<Player>
{
    public string Username;
    public string Password;
    public string Email;
    public int Steps;

    0 referencias
    public Player() { }
    1 referencia
    public Player(string username, string email, string password )
    {
        this.Username = username;
        this.Email = email;
        this.Password = password;
        this.Steps = 0;
    }
    0 referencias
    public void SetDistance(int distance) { this.Steps = distance; }
    0 referencias
    public int GetDistance() { return this.Steps; }

    0 referencias
    public int CompareTo(Player other)
    { // A null value means that this object is greater.
        if (other == null)
            return 1;
        else
            return this.Steps.CompareTo(other.Steps);
    }
}
```

Figure 4.3 Player Class

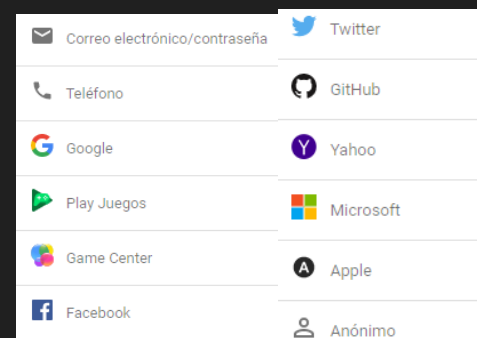
As mentioned above, they also offer database functions, specifically the Realtime Database was an interesting option because it is simple to use, mainly due to the programming methods are understandable and the data is stored in JSON<sup>10</sup> format, which are relatively easy to use and it is easy to modify the sample codes they provide to suit the data structure we are looking for. But despite how good it sounds, much of the documentation was out of date, which slowed down the development of this task. The main problem was that the way to set the database URL (see Figure 4.5) had changed, in previous versions it was specified within the code itself, while now the API takes the URL from a file that Firebase generates called "google-services", file that my project did have, but the URL had errata (see Figure 4.6). Realizing that was the problem took more than it should, but once found it, was easy to solve.



**Figure 4.5** Realtime Database

```
"project_info": {
  "project_number": "522333759810",|
  "project_id": "walk-uji",
  "storage_bucket": "walk-uji.appspot.com"
```

```
"project_info": {
  "project_number": "522333759810",
  "firebase_url": "https://walk-uji-default-rtdb.europe-west1.firebaseio.com",
  "project_id": "walk-uji",
  "storage_bucket": "walk-uji.appspot.com"
```



**Figure 4.4**  
Log-In methods

**Figure 4.6** UP: the google-services file my project had. DOWN: How it should be.

<sup>10</sup> JavaScript Object Notation (JSON): Simple text format for data exchange.

<sup>11</sup> Uniform Resource Locator (URL), colloquially termed a web address.



With the bases of the authentication and data storage system established, was time to set in motion the implementation of geoposition functionalities. During the research done after the interruption previously mentioned, a very promising tool was found, "Mapbox", an SDK<sup>12</sup> for Unity that offered the same services as ArcGIS but with the difference that it was not in beta, It had much more information online, more varied tutorial videos, and even examples focused on videogames, thing that ArcGIS didn't have.

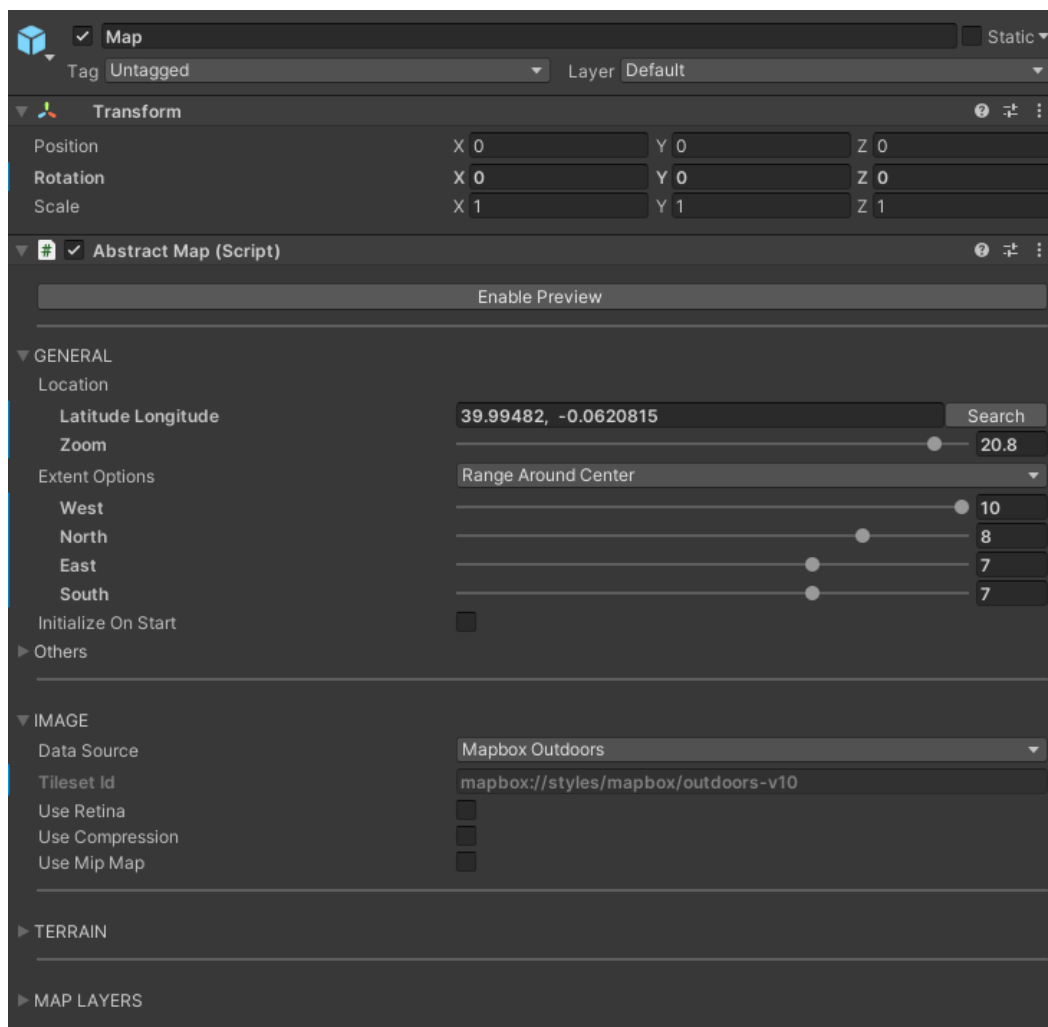
This time I was more careful when implementing the plug-in, I made sure to configure my project correctly, since, from past experiences, combining different add-ons may cause errors and specifically the combination of Mapbox with Firebase, according to the developers experiences read during the research in different forums, they used to cause conflicts between them, for example, one of these errors occurred if augmented reality is used, which is just one of the functionalities that the project will have later on.



---

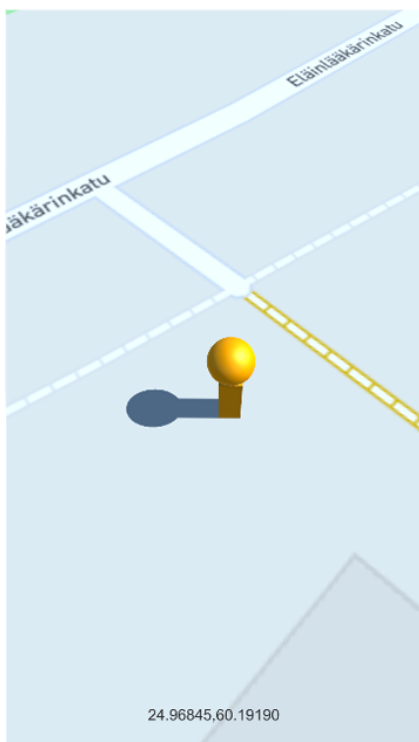
<sup>12</sup>Software development (SDK): Collection of software development tools in one installable package

This SDK, unlike ArcGIS, does not have a user interface (UI) like the one we saw previously in figure 4.1, although it allows the modification of certain code parameters within the Unity inspector (see Figure 4.7), while in the ArcGIS UI it was barely necessary to touch the code, with this plug-in it has been necessary to modify some things.

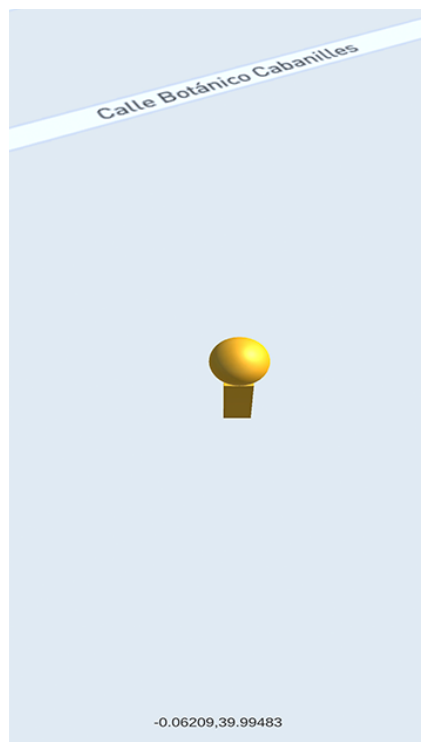


**Figure 4.7** Mapbox UI.

The first task was to create a test scene to check that the map rendering was performed correctly both on the PC and on the phone. It was nice news to see that in spite of the map was not showing at my location (see Figure 4.8), the mere fact that the map was displayed on the phone was already a major advance. With these bases in place, the next step was to make the map get loaded from the location of the mobile (see Figure 4.9), a task that was relatively simple since this plugin offered a tool for it. That worked correctly, the project already had the geopositioning and rendering system of the mobile map fully functional, and after this, as the objective is to use it as a minimap, this just needed to be modified so that it is displayed within a box on the screen and not occupying its entirety (see Figure 4.10), although this task was simple and quick to make.



**Figure 4.8** Map on phone.  
(random location)



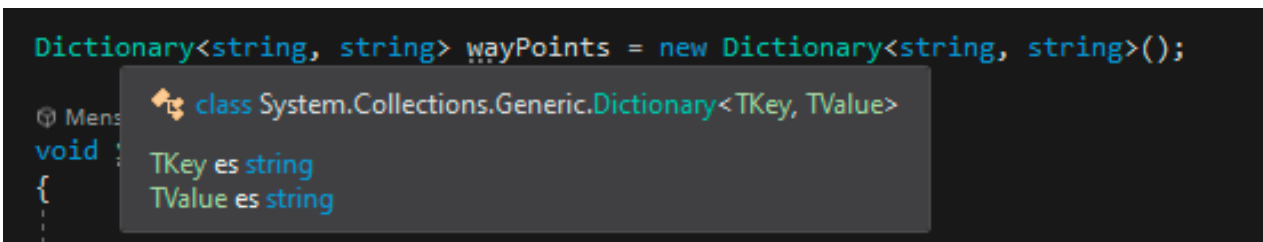
**Figure 4.9** Map with  
my current location.



**Figure 4.10** Minimap  
on phone.

The next task was to make the route drawing system, for which a dictionary was created whose, for each element, the key is the name of the place and the value its coordinates (see Figure 4.11). Initially, the data of the different faculties, the Agora and the Library were entered in this dictionary (see Figure 4.11) and later, from the tool that Mapbox offers, which, given a latitude and longitude, places a marker in those coordinates in the map and calculates the possible route. With this, the predefined type of route are calculated to be done by car, so that had to be changed to walking. Luckily this was very simple, just a quick modification in the code and its done (see Figure 4.13).

```
Dictionary<string, string> waypoints = new Dictionary<string, string>();
```



**Figure 4.11** Dictionary creation.

```
wayPoints.Add("Ágora" , "39.99462748472183 , -0.06899925332420609");
wayPoints.Add("ESTCE" , "39.99292500863973 , -0.06714578790633553");
wayPoints.Add("FCHS" , "39.995069680811625 , -0.06998945459106296");
wayPoints.Add("FCJE" , "39.99427946078642 , -0.0661726370694918");
wayPoints.Add("FCE" , "39.99436656101847 , -0.06612569518093699");
wayPoints.Add("Biblioteca" , "39.99408375855287 , -0.06936995897610927");
wayPoints.Add("Instalaciones Deportivas" , "39.995125473921874 , -0.07170665776320667");
```

**Figure 4.12** Adding the data.

```
for (int i = 0; i < count; i++)
{
    wp[i] = _waypoints[i].GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale);
}

var _directionResource = new DirectionResource(wp, RoutingProfile.Walking); //Type of route
_directionResource.Steps = true;
_directions.Query(_directionResource, HandleDirectionsResponse);
```

**Figure 4.13** Code modification for the correct route calculation.

With this task done, was time to get to work in the step counter task. The initial intention was to use the geopositioning system itself, but after several hours of searching, the conclusion was that that was a dead end road. So, the task was approached in the way in which this functionality could work in the largest number of possible devices. The easiest way to make it work is using the pedometer some phones have, but due to the reason mentioned before, this was not a viable option. After a quick research, it was concluded that the best option was to use the accelerometer, since all or almost all mobile phones nowadays incorporate it in their hardware. Was hard to find information about how to use the phone's accelerometer from Unity, but ended up finding an explanation in one of the many posts visited during the research at Unity Forum. Unfortunately I did not save the link to the forum, but in that was the whole explanation to the code used. Down below there is a photo of the main part of mentioned code(see Figure 4.14).

```

void FixedUpdate()
{ // filter input.acceleration using Lerp
  curAcc = Mathf.Lerp(curAcc, Input.acceleration.magnitude, Time.deltaTime * fHigh);
  avgAcc = Mathf.Lerp(avgAcc, Input.acceleration.magnitude, Time.deltaTime * fLow);
  float delta = curAcc - avgAcc; // gets the acceleration pulses
  if (!stateH)
  { // if state == low...
    if (delta > hiLim)
    { // only goes high if input > hiLim
      stateH = true;
      steps++; // count step when comp goes high
      acc.text = "steps:" + steps; //Act text
    }
  }
  else
  { // only goes low if input < loLim
    if (delta < loLim) { stateH = false; }
  }
}

```

**Figure 4.14** Main code from the StepCounter.cs

At this point, all the basic functionalities were implemented, except Vuforia, so it was time to get it done. Due to my previous personal experience with this plugin in the subject VJ1234 - Tècniques d'Interacció Avançada, this task did not cost much effort, just had to take care with the conflicts this plugin may create, but fortunately did not happen. With the AR camera in the scene, a placeholder QR code was used to test it. Everything worked fine, so it was time to continue and jump to the next task, the player ranking.

This task was left for the last because, as it uses the Firebase Realtime Database plugin, it was supposed to be relatively easy due to one of the previous works done (i.e. storage system). Indeed it was, the "most complex" part was to create the current display list due to the way you have to work with them. An example view can be seen in Figure 4.15.

A brief explanation how the ranking is filled would be: First, an empty list of the class "Player" needs to be created, then we fill that list using a call to the database (see Figure 4.16), pass that list to the "RankingManager.cs", sort the list using the function '.Sort()' (this can be done due to the implementation of the interface IComparable) and then, filling the ListView, task that is very simple (see Figure 4.17)

PLAYER RANKING	
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.
100.	DaniPK7777: 13.000 steps.

Figure 4.15 Ranking example.

```
public void GetPlayers()
{
    if (user != null)
    {
        FirebaseDatabase.DefaultInstance.GetReference("").GetValueAsync().ContinueWith(task
        {
            if (task.IsCanceled) { }
            if (task.IsFaulted) { Debug.LogWarning("The database is Empty"); }
            if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                string playerData = snapshot.GetRawJsonValue();
                foreach (var child in snapshot.Children)
                {
                    string t = child.GetRawJsonValue();
                    Player extractedData = JsonUtility.FromJson<Player>(t);

                    databasePlayers.Add(extractedData);
                    print("Added: "+ extractedData.Username);
                }
            }
        });
    }
}
```

Figure 4.16 Call to the database.

```
players.Sort(); //Ordena de menor a mayor
players.Reverse(); //Lo invertimos para que este de mayor a menor

//Añado al ranking cada usuario
for (int i = 0; i < players.Count; i++)
{
    GameObject go = Instantiate(listElement_Prefab);

    if (i + 1 == 1) go.transform.GetChild(0).GetChild(0).GetComponent<Image>().color = first;
    else if (i + 1 == 2) go.transform.GetChild(0).GetChild(0).GetComponent<Image>().color = second;
    else if (i + 1 == 3) go.transform.GetChild(0).GetChild(0).GetComponent<Image>().color = third;
    else go.transform.GetChild(0).GetChild(0).GetComponent<Image>().color = rest;

    go.transform.GetChild(1).GetComponent<TextMeshProUGUI>().text = (i + 1).ToString(); //Pos in the rank

    string name = players[i].Username;
    if (players[i].Username.Length > 10) { name = players[i].Username.Substring(0, 10);}

    go.transform.GetChild(3).GetComponent<TextMeshProUGUI>().text = name + ": " + players[i].Steps + " steps."; //Name and steps

    go.transform.SetParent(transform);
}
```

Figure 4.17 Filling the ListView.

## 4.2 Results

The current status of the project is satisfactory, there are many things that may be done, but things works well. This was a kind of challenge since I did not have too much prior knowledge about the bases of the project (geolocation and database management from Unity), but all possible dedication has been put into it. The results obtained to are the following:

- Authentication system.
- Database system storage.
- Geopositioning system.
- Updated map rendering according to user location.
- Display the POIs in the map.
- Route planner.
- Step counter.
- Ranking system.
- QR code reader.
- AR camera.



## 5. Conclusions and future work

### 5.1 Conclusions

Without any doubt, this is the most complex project I have done throughout this years. Almost all the tools used were new to me, I had previous experience using Unity both in class and in my private life, but I had never used/ needed to use geolocation or work with a database. I learned a lot during the development of the project, to understand other people's code, theoretical knowledge especially organization and work under pressure, forcefully learn to be flexible with time distribution.

### 5.2 Future Work

Obviously the project is expandable, both in code optimization and in features, but there are certain deadlines and implementing all the desired features is almost impossible. In the future I would like to keep using Unity in my projects, but I also want to learn other game engines like Unreal and work in a team for a big game project.

### 5.3 Link to repository and app

Link to the project: [DaniPK7/WalkUJI: Final Degree Project](#)

Link to the APK: [DaniPK7/WalkUJI: APK](#)

## 6. Bibliography

- [ 1 ] ArcGIS Unity SDK Documentation                      Accessed: 06/20/2021  
[ArcGIS Maps SDK for Unity | ArcGIS for Developers](#)
- [ 2 ] Mapbox Unity SDK Documentation                      Accessed: 06/20/2021  
[Maps SDK for Unity: 3D worlds, AR, & POIs | Mapbox](#)
- [ 3 ] Conflict between Mapbox & Firebase                      Accessed: 06/20/2021  
[Solution of plugin conflict between Mapbox and Google Firebase in Unity/Android Build - YouTube](#)
- [ 4 ] Firebase for Unity Documentation                      Accessed: 06/20/2021  
[Agrega Firebase a tu proyecto de Unity](#)
- [ 5 ] Firebase Authentication for Unity                      Accessed: 06/20/2021  
[Primeros pasos con Firebase Authentication en Unity](#)
- [ 6 ] Firebase Realtime Database for Unity                      Accessed: 06/20/2021  
[Cómo comenzar con Firebase Realtime Database para Unity](#)
- [ 7 ] Vuforia for Unity Documentation                      Accessed: 06/20/2021  
[Getting Started with Vuforia Engine in Unity | VuforiaLibrary](#)
- [ 8 ] Class Step Counter Documentation                      Accessed: 06/20/2021  
[Class StepCounter | Input System | 1.0.2 \(unity3d.com\)](#)
- [ 9 ] Scrollable List View Tutorial                      Accessed: 06/20/2021  
[Unity3D: How to make a scrollable List view/Table view. - YouTube](#)