UNIVERSITAT JAUME•I

APPLICATION FOR ATTENDEES OF A TALK SHOW EVENT

Carlos Hernández Prieto

Final Degree Work
Bachelor's Degree in

Video Game Design and Development

Universitat Jaume I

September 2021

Supervised by: José Vte. Martí Avilés

Acknowledgments

First, I want to thank my previous tutors for the patience they have shown and how understanding they have been with my situation in recent years, but especially José Vicente Martí Avilés for having supported me by always having kind words, understanding my problems, and having made a great effort to give me this opportunity. I would also like to thank my colleague José Antonio Gil Altaba for his patience in explaining all the concepts that I didn`t know or didn`t understand. Finally, I want to thank in a special way the patience of my wife and my daughters for the hours and hours spent in front of the computer without being able to enjoy their company.

Abstract

This document refers to the technical proposal for a final degree project in the Degree of Videogame Design and Development.

The work carried out during the development of the project aims to create a mobile application for the iOS platform, which allows those attending a talk show type event to be able to participate actively. This is intended to be achieved by creating an in-app gamification system that will offer a messaging wall, a program feed, and live quizzes.

Attendees within the application will have a username, email (access) and password, a user profile, an agenda, a list of sponsors, a list of guests and attendees, a wall where they can leave comments, a place in where the program team can share stories and a leaderboard with a ranking of the attendees which will count the participation to score points.

Contents

Index of tables

Index of figures

1
INTRODUCTION

The work is made up of five sections, covering the planning, the realization and the results obtained.

The first chapter describes the motivation that led to the realization of this project, the objectives, and the initial state. It talks about all the decisions that have been made before starting the project.

1.1. WORK MOTIVATION

The motivation behind this idea for the TFG arises from the need for large companies to create events in which to promote their brand. These events should provide gamification, enhance entertainment, and generate extra value for event attendees.

Participating actively in the development of apps for events gave me the idea of creating an application that can be used for certain television programs of the talk show type and in this way allow the public to participate actively during the same.

Thanks to the development of this technology, the following needs of the organizers of an event would be covered:

- Generate motivation in attendees to use the application whether you are going to attend the event or if you are a follower of the program, creating a community fan of the program that will receive news, prizes and raffles related to the programs and their content.
- Create a new television format that allows breaking the barrier of both television and new technologies, as well as participation within a television show.
- Animate events through a television medium using a gamification system that achieves the participation of the attendee, promoting both games and challenges and rewards the best in the ranking of each event.
- Capture the attention of attendees promoting interaction through the application to achieve rewards.
- Increase the active participation of the attendee to make them feel part of the show.

1.2. OBJECTIVES

- "To create an application from scratch in native development."
- "To create a user system with an agenda to plan the different shows"
- "To create a ranking system by points with the different users."
- "To manage a database hosted in the cloud that allows data to be stored and synchronized between users in real time."
- "To maintain a system that allows messages to be uploaded in real time to a messaging wall"

1.3. ENVIROMENT AND INITIAL STATE.

1.3.1. WORK ENVIROMENT

The entire project will be carried out individually, with the help of the tutor provided by the university and the collaboration of an iOS developer in the form of code review, resolution of doubts regarding developments and advice on good practices. These supports will allow to improve the content at a visual and experience level and the code in terms of optimization of resources and computational costs.

The work done locally from a MacBook pro laptop will be uploaded to a repository shared with both tutors, to allow review requests, in which they can give feedback on improvements to be made, malfunctions or errors made in the code.

1.3.2. INITIAL STATE

The project begins during the viewing of a television program in which the audience is allowed to interrupt and participate in the program.

When working in an events company, look at the possibility of making an application that allows the public to be part of a community linked to the program and that allows users who attend the program to participate directly through the application.

The database and the project application are made from scratch considering all the possibilities and details required for this specific case.

1.3.2.1. INTERNAL DECISIONS

1. The project must have a database that allows to store the data of the different users, quickly offer responses to the requests to the database and store the following data:
    - User information.
    - Program information.
    - Content of the publications.

2. This database must be hosted on an online server.
3. Users will have access to all the information pertinent to their profile and will be able to make modifications.
4. The information of the programs and the content of the publications must be inserted by the organizers.

1.3.2.2. EXTERNAL DECISIONS

1. In the future, the data will be accessible through Android development.
2. To carry out the project, a technical proposal, an analysis and design document, and a final report must be made.
3. All the documentation must be in English.
4. Within the application, the user's profile must be filled out by the user himself.

2

PLANNING AND RESOURCES EVALUATION

In this section, the planning and description of all tasks is shown, accompanied by the work time, including the Technical Proposal and its presentation. It also details the evaluation of the resources that are necessary for the correct development of the work, both human and team with an approximate cost.

2.1. PLANNING

Table 1 shows the list of main tasks for the project. This list shows the task to be done and the estimated duration of hours invested to get it completed.

Tasks of the work to be performed.	*Estimated duration. (hours)*
• *Develop an access system for users that allows them to create their password and access the application.*	40
• *Create a menu and a user interface.*	40
• *Generate a database project that allows users to access with their email and password.*	40
• *Create the different submenus and bind them to the database. Users, casting, wall messages, feed, etc.*	40
• *Implement a scoring system for the user's leader board.*	30
• *Create the different endpoints to be able to manage the message wall of the attendees and the ranking of scores.*	40
• *Make the technical proposal*	15
• *Perform final memory*	40
• *Prepare final presentation*	15
• **Total number of hours planned**	**300**

Table 1 - Task list and their duration

2.1.1. TECHNICAL PROPOSAL PLANNING

The final degree work proposal was made individually. With the intention of taking advantage of the knowledge acquired during the university and the short career as an iOS developer, the development of the technical proposal that was finally presented was studied. The breakdown of this process is divided into the following subtasks and their time cost.

Subtasks

- 10 Hours of realization.
- 2 Hours of layout.
- 3 Hours of transcription and correction into English.

2.2. RESOURCES EVALUATION

To achieve the results shown in this project, it would take around 270h assuming there will be only one worker. Below is a rough estimate of the costs (excluding VAT) and the earnings that person would have.

2.2.1. REQUIRED EQUIPMENT

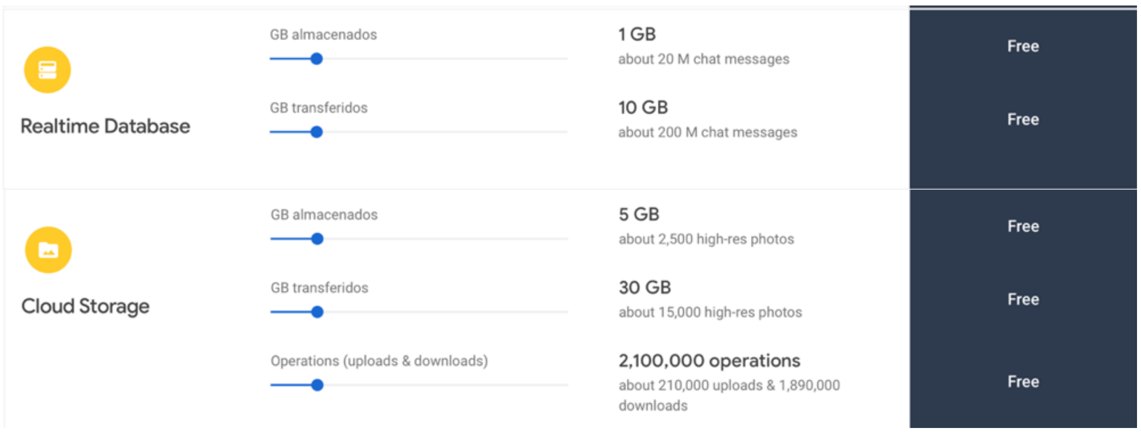
To develop the application, the following components and services are required:

- **A portable computer with an iOS operating system capable of running the xcode environment.** To develop the project, a MacBook Pro (15 inches, 2019) has been employed with the technical features shown in *Table 2.*

Components	Features
CPU	2,3 GHz Intel Core i9, 8 cores
Graphics card	Radeon Pro 560X 4 GB Intel UHD Graphics 630 1536 MB
Hard disk	500GB
RAM	32 GB 2400MHz DDR4

Table 2 - Technical features of the equipment employed.

- Internet connection.
- **Xcode 12.4,** it is possible to use previous versions.
- **Firebase,** online service for database development. The service is free if the margins detailed in *Table 3* are not exceeded.



Realtime Database	GB almacenados	1 GB about 20 M chat messages	Free
	GB transferidos	10 GB about 200 M chat messages	Free
Cloud Storage	GB almacenados	5 GB about 2,500 high-res photos	Free
	GB transferidos	30 GB about 15,000 high-res photos	Free
	Operations (uploads & downloads)	2,100,000 operations about 210,000 uploads & 1,890,000 downloads	Free

Table 3 - Economic Planning of the DB and Accommodation

2.2.2. WHY FIREBASE?

Firebase is a platform acquired by google in 2014 which offers a free service that allows users to create serverless applications, storing and synchronizing JSON data between database users in near real time with robust user-based security *(Moroney, L. & Anglin: 2017)*. It has several products that are very useful for the development of the application.

Firebase Realtime Database is a cloud-hosted NoSQL database that enables customers to store and synchronize data between users in real time. This tool allows:

- **Build serverless apps**: The real-time database links to a web or mobile SDK to enable developers to create applications without the need for servers.
- **Optimized for offline use**: If users go offline, the Realtime Database SDK uses the local device to publish and store changes, syncing this data automatically when the device is connected again.

Cloud Storage helps the customer to quickly and easily store and process user-generated content such as photos and videos. The characteristics of this tool are:

- Infrastructure designed to move easily from the prototype to the production stage.
- Automatically stops and resumes transfers when the application loses and regains mobile connectivity, saving users time and bandwidth.

Both Realtime Database and Cloud Storage integrate with Firebase Authentication to provide intuitive and easy authentication for developers.

This tool allows our app to work correctly even if the app is used by many users. The prices are free, the firebase service only has a cost when the need for more functions increases, which would be acceptable as the application would generate a lot of profit.

2.2.2.1. FIREBASE VS AWS

Along with firebase, AWS Amplify is another of the most powerful application development platforms. Backed by Amazon services, it uses a set of open-source java script libraries that facilitate web and mobile application development.

AWS Amplify benefits	*Google Firebase benefits*
Open source	Real time having a robust API
Local device datastore	Easy-to-use and fully integrated console
Supports SQL and No-SQL databases	Runs on Google's cloud
Supports GraphQL and REST API	Scalability
Scalability	Provides user side high security

Table 4 - AWS vs Firebase benefits

Among the listed benefits of firebase and AWS *Table 4*, the main features that led to the choice of firebase is that it is a real-time database. This feature is ideal for participating in a talk show event by posting messages on the wall or answering questions about gamification. On the other hand, although it is inconvenient to have to develop a No-SQL database, access to the data in these databases is much faster. The last point for the choice to have been firebase has been the high security that firebase offers on the user side.

2.2.3. HUMAN RESOURCES

The real cost of this project would incorporate a part of web development and services, to be able to manage and organize in the most optimal way all the details of the users, to be able to add new guests, manage the publications of the walls and control the gamification section (questions to attendees) more easily.

The project developed for this work does not incorporate the development of services or web. In this project, only the development costs of the database in firebase are taken into account, which are shown in *Table 5* and the application development cost that is explained in *Table 6*. This means a lower cost, but it translates into a smaller system with the limitations of managing only from the application itself.

The project is developed only for iOS. This implies that carrying out the development for Android would duplicate the economic planning section shown in that table.

Construction of the DB	Duration	Price	Total
Creation of DB		100 €/Data tree	500 €
Maintenance	2 h/month	70 €	35 €
Modifications	35 €/hour		According to stipulation made for the proposed modification.

Table 5 - Economic Planning of the database

Maintenance would take approximately two hours a month to avoid or solve possible problems that may arise. Regarding the growth of the database, the planning will be balanced if necessary due to an increase that requires future modifications.

Generation and approximate annual cost.	Price by unit	Units	Total
Develop	300€ / functionality	12	3600€
Design	150€ / view	12	1800€
Design modifications	50€ - 100€ / section	To be evaluated by the technical team	50 - 100€
Maintenance	500€ / Year		500€
Final app price			5400€
Maintenance and extras			500€ – 600€

Table 6 - Economic Planning of the app

2.2.4. FINAL PRICE AND SALARY

The final price to develop the project with the indicated functionality would be around 7139 euros, VAT included. The calculation of the final prices is shown in *Table 7*.

The approximate cost of hours to complete the development of the project will be 270 hours, which is equivalent to 34 days of work. The daily cost would be 209,97 euros, which is equivalent to 26,44 € / hour. This price falls into the price range per hour for freelancers and development companies, according to a study published by *Luis Picurelli, CEO of Yeeply*.

Subsequent maintenance, modification, and extension work on the application would provide a later amount of variable revenue that has not been considered in the initial planning since it cannot be quantified with complete certainty.

Final Price of the DB	500€
Final Price of the APP	5400€
Combined Price.	5900€
VAT (21%)	1239€
Final Price.	**7139€**

Table 7 - Final economic planning

3

SYSTEM ANALYSIS AND DESIGN

This section presents the analysis of the system through functional and non-functional requirements and the design of the system through diagrams, flowcharts, and database tables. Requirement's analysis and job architecture topics are also addressed.

3.1. REQUIREMENTS ANALYSIS

The analysis is going to focus first on describing and analysing the different functional requirements that form the basis of the application and then on exposing the non-functional requirements of the application.

3.1.1. FUNCTIONAL REQUIREMENTS.

The mobile application will be developed with the premise of being able to provide users with the following actions:

- The system will allow creating users.
- The system will allow users to be deleted, both by the user who owns the account and by the administrator.
- The system will allow users to be listed.
- The system will allow users to view their profiles.
- The system will allow users to view the details of other users.
- The system will allow users to post comments.
- The system will allow users to have favourite users.
- The system will allow users to view the organization's notifications.

The following describes the input and output of the different requirements and a brief description of your task.

Input: Create user.
Output: Add a new field with the user's data in the users table in the DB.
The application sends a username and password to the database to register a new user in the application.

Input: Delete user.
Output: Deletes the field from the users table in the DB.
The application sends a username to the database requesting to be removed and all stored fields that are linked to the user are removed.

Input: List users.
Output: The database returns a list based on the input condition.
The application requests users, a cast member or an assistant and the database return a list according to the request made.

Input: View user profile.
Output: The database returns a profile based on the input condition.
The application requests the profile of a user, and the database searches and returns the profile selected by the user.

Input: Add/delete user to favourite list.
Output: The database a (Badal & Luis, 2020)dd or delete user from favourite user list.
The application sends a user to be added or removed from the requesting user's favourites list. The database adds or removes the received user from the requesting user's favourites list.

Input: View user’s details.
Output: The database shows the details of a user.
The application requests the details of a specific user and the database searches and displays the details of the selected user.

Entry: Post comment.
Output: The database records a comment from a user on a wall.
The application requests the database to create a new comment for user "x" in place of "y".

Input: View publications.
Output: The database shows the posts of a wall.
The application requests the publications of a specific wall and the database searches and returns the publications.

3.1.2. NON-FUNCTIONAL REQUIREMENTS.

The development of the mobile application seeks to offer users the following requirements that, although not essential, will offer a better user experience:

- Being an efficient system that manages and controls many simultaneous users.
- Being a system that allows data and information to be stored and made available in real time, keeping them updated.
- To be an easy-to-use system in which users do not find it difficult to access internal sections.
- Being a system with a simple and secure registration, Firebase offers an authentication system that allows registration, manages accesses, and achieves greater security and protection of data.
- Being a system with a certain degree of storage in the cloud, offering a storage system in the cloud, where the application files can be saved.

3.2. SYSTEM DESIGN

Next, a series of diagrams are exposed to facilitate the understanding of the project and provide a better vision of the system design. These diagrams show the actions that users can perform in the application, as the different classes are intertwined with each other and the flows of the most prominent interactions.

3.2.1. USE CASE DIAGRAMS

In each of these sections a diagram and a brief description of the elements involved are shown. The diagrams represent each of the four actors that can intervene in the application, as well as their possible actions.

3.2.1.1. USER

The user will be able to carry out different types of interactions depending on their condition in the application. However, all users share the initial screen. This use case is shown in *Figure 1.*

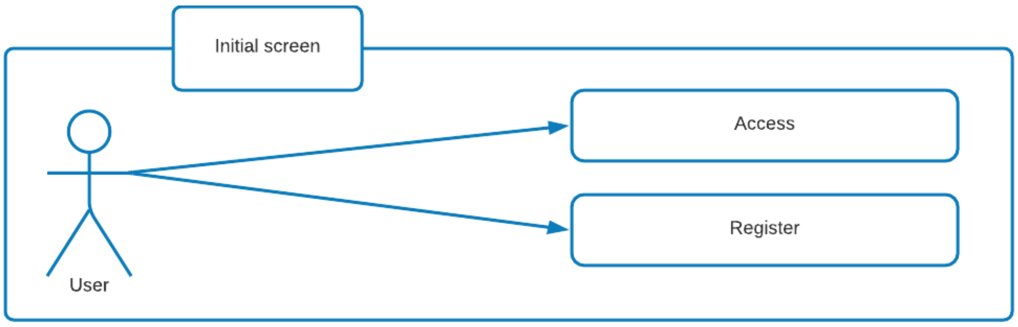


Figure 1 - Initial screen for all the users

INTERACTIONS

- **Access**: The user must be registered in the system by email, username, and password. If the user already has an account, he can access through the initial access system. If the user does not have an account, the user has an automatic registration that will allow him to create an account.

- **Register**: The user must register in the system giving a series of data (username, email (access) and password. Once the registration is completed, the system will send the user to the initial view where they can access the application.

The user within the application is divided into three ranges of access to content: basic user, assistant user, team user or staff member and administrator user. Users who are not attendees will have the functionalities shown in *Figure 2.*

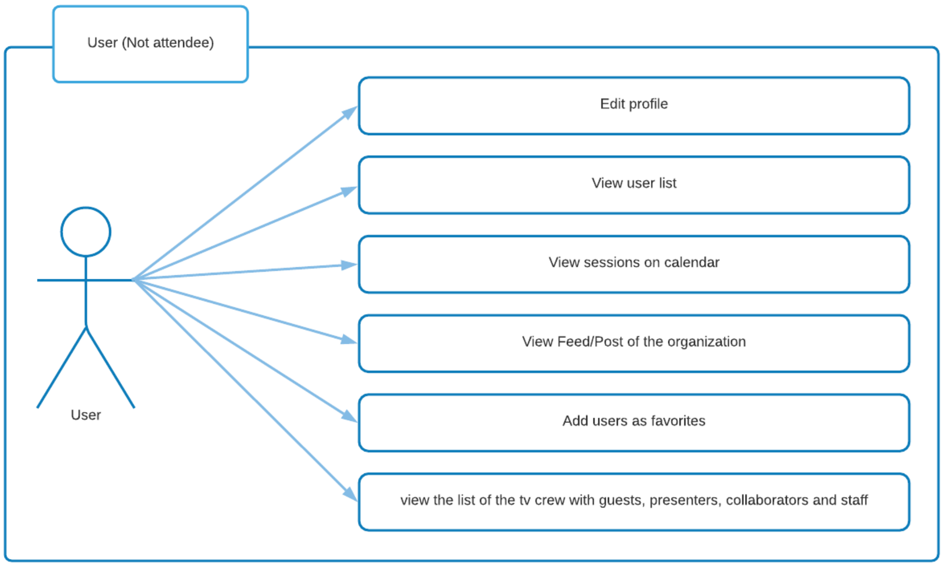

User (Not attendee)
Edit profile
View user list
View sessions on calendar
View Feed/Post of the organization
Add users as favorites
view the list of the tv crew with guests, presenters, collaborators and staff
User

Figure 2 - User interactions (not attendee).

INQUIRIES

- **View list of users:** The user will be able to consult a list with all the users who are registered in the application.

- **View session agenda:** The user will be able to see all the sessions available to him in the agenda.

- **View Organization Feed / Post:** The user will be able to view all the organization posts that have been created.

- **See the "TV CREW" list:** The server will show the user a list of "TV CREW" with guests, presenters, collaborators, and staff.

INTERACTIONS

- **Edit the profile**: The user will be able to edit its profile information. To change the username, the system will validate that there is no other username equal, while to change the email, the system will verify the email account by sending an email.

- **Add to favourites**: The user will be able to add other users as favourites.

Users who are attendees have the same level as basic users but will be able to perform more actions related to the show they attend, as shown in *Figure 3*. These actions are a complement to the interactions they already had as users of the application.

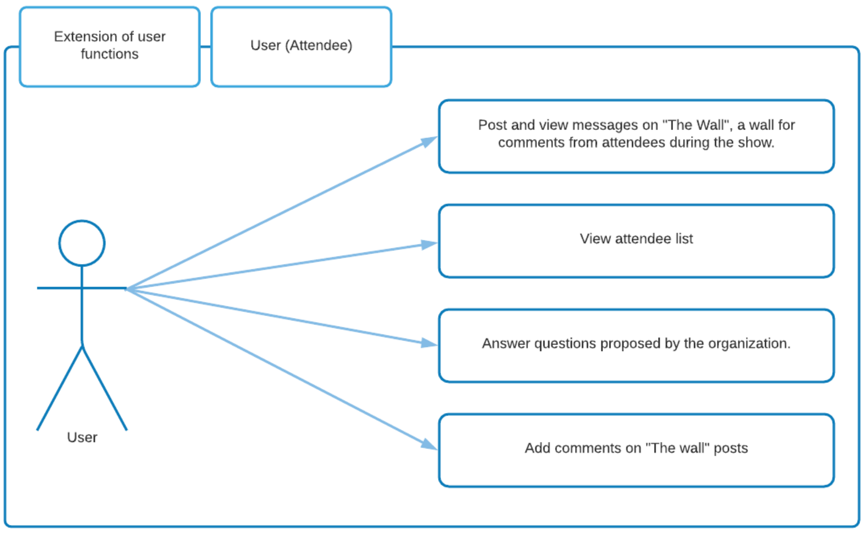

Extension of user functions
User (Attendee)
Post and view messages on "The Wall", a wall for comments from attendees during the show.
View attendee list
Answer questions proposed by the organization.
Add comments on "The wall" posts
User

Figure 3 - User interactions (Attendee)

INQUIRIES

- **View messages**: The attendees will be able to consult the messages on the comment wall.

- **View attendee list**: The attendee will be able to view all the users who will attend the session.

INTERACTIONS

- **Publish messages**: Then user of type attendee can post messages on the comment wall for attendees to the event.

- **Answer questions**: The assistant user will be able to answer the various questions that are proposed from the organization.

- **Add comments:** The assistant user will be able to add comments to the posts on the wall.

The program will have a type for user to be able to carry out the organization functions of the different sections of the application. This user will be able to perform the functions shown in *Figure 4*.

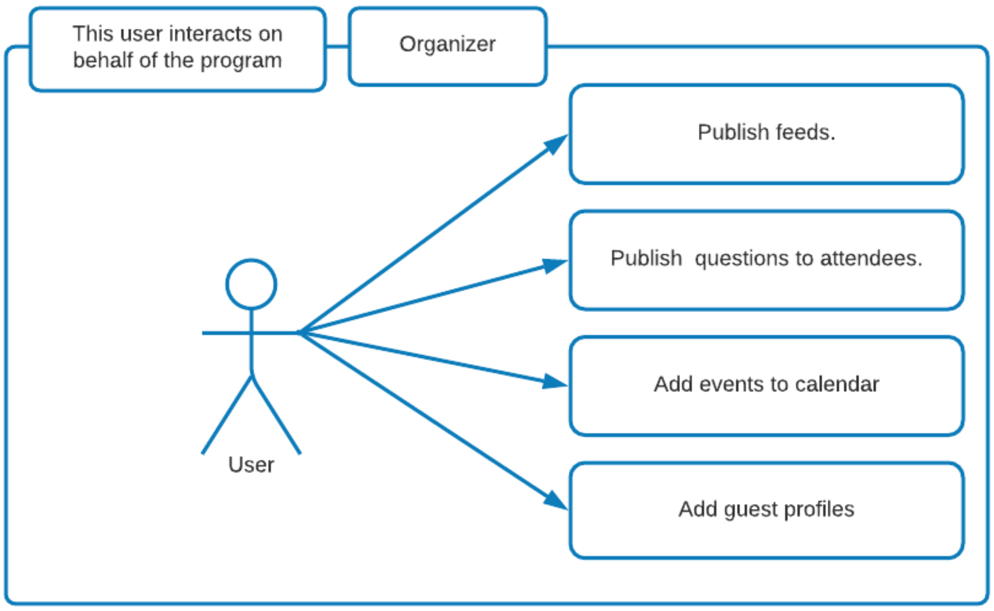

This user interacts on behalf of the program
Organizer
Publish feeds.
Publish questions to attendees.
Add events to calendar
Add guest profiles
User

Figure 4 - Organizer interactions

INQUIRIES

- **View the options for add or delete events, feeds, questions, and guests**: Organizer type users can see, within certain sections, special functionalities that belong to their type to manage the organization of the event.

INTERACTIONS

- **Add or delete feeds**: Organizers can add or remove text posts or also include images to provide information about the session or globally about the program. This information will be displayed in the program's feed.

- **Add or delete questions**: Organizers can add or remove questions for a specific event. Later they can see the answers and give a score to each attendee according to the answer offered.

- **Add or delete events:** Organizers can add or remove events to the calendar so that other users can see it.

- **Add or delete guest profiles:** Organizers can add, modify, or delete guest profiles with their permission.

The user with more access is the administrator type user, it can be seen in *Figure 5*. This user can change the range to users to be basic users (assistant or not), organizer or administrator, and delete any user from the application. With these features, the administrator will be able to create new organizers who can oversee uploading content to the program's feed, or selecting the questions to upload in each show, among other things, they can also change users to the administrator type.

The administrator must bear the risks of changing to a role user.

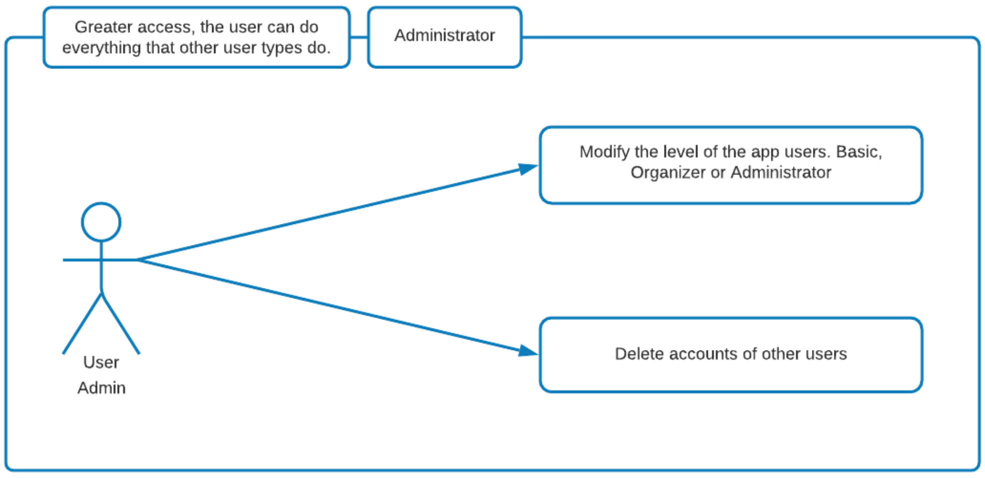

Greater access, the user can do everything that other user types do.
Administrator
Modify the level of the app users. Basic, Organizer or Administrator
Delete accounts of other users
User Admin

Figure 5 - Administrator interactions

INQUIRIES

- **See the type of user**: Administrators will be able to check the type of each user and see special functions.

INTERACTIONS

- **Delete users**: Administrators will be able to delete any user by accessing the user details.

- **Change type of users**: Administrators will be able to change the type of each user.

3.2.2. CLASS DIAGRAMS

In this section, the methods and attributes are omitted, showing only the connection between the different classes / views of the application. The intention of this section is to show the internal connection that the application has, this is shown in *Figure 6.*

Purple lines indicate navigation from the main menu. As you can see, users can access the login, user profile, calendar, user list, cast list, attendee list, and news message list.

To access the details of a user, users must access through the list of users or attendees in case the user attends the show of the program. On the other hand, when a user wants to access the detail of a crew, the user will have to search the crew list or through the calendar, enter the session in which the crew that the user is looking for participates and enter on the detail of the crew.

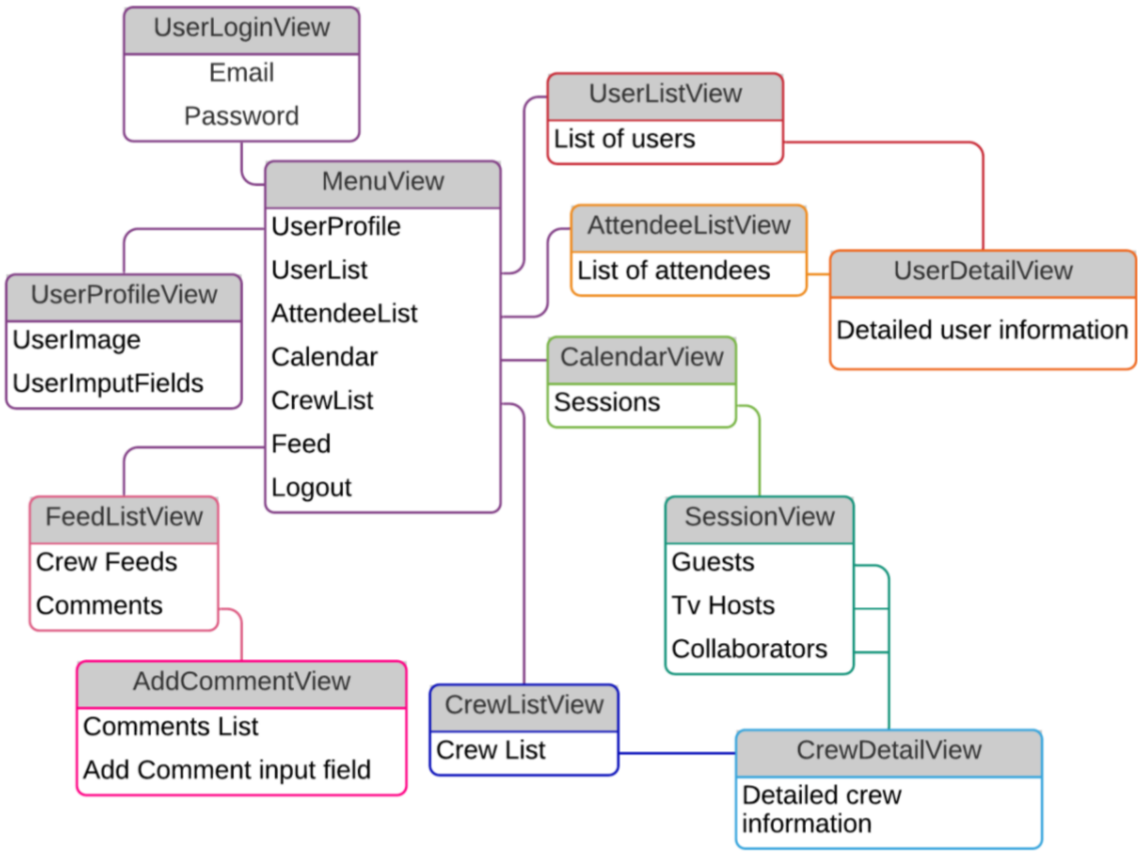


Figure 6 – Internal app connection

3.2.3. FLOWCHARTS

The flowcharts that are outlined are intended to observe the behaviours of the system with the user. First, it will be focused on the course that the application follows when a user registers. Then the access and the main actions available to the user are shown.

3.2.3.1. REGISTRATION

When the application is executed, a username and password are requested to be able to access. In this situation there is the possibility of making two options, registering, or accessing the system. If the user's email is not in the database, the user must register before being able to access the system. To register, the user must enter a username that cannot match another user, an email that should not be registered in the database and a personal password. With this data, the system will verify the email account by means of a verification message that is sent to the user when the registration is completed.

The system will allow the user to insert only the username if the inserted name is already in the database. This has been done with the intention of making the search for a username more pleasant and that this new name does not matches that of other users.

The system provides three paths when a user is in the registration section of the application, this is shown in the flow diagram of *Figure 7*.

- Valid username, email, and password → Access to the application menu.
- Incorrect username, email, and password → Enter new data or exit.
- Wrong username, correct email, and password → Enter new username or exit.

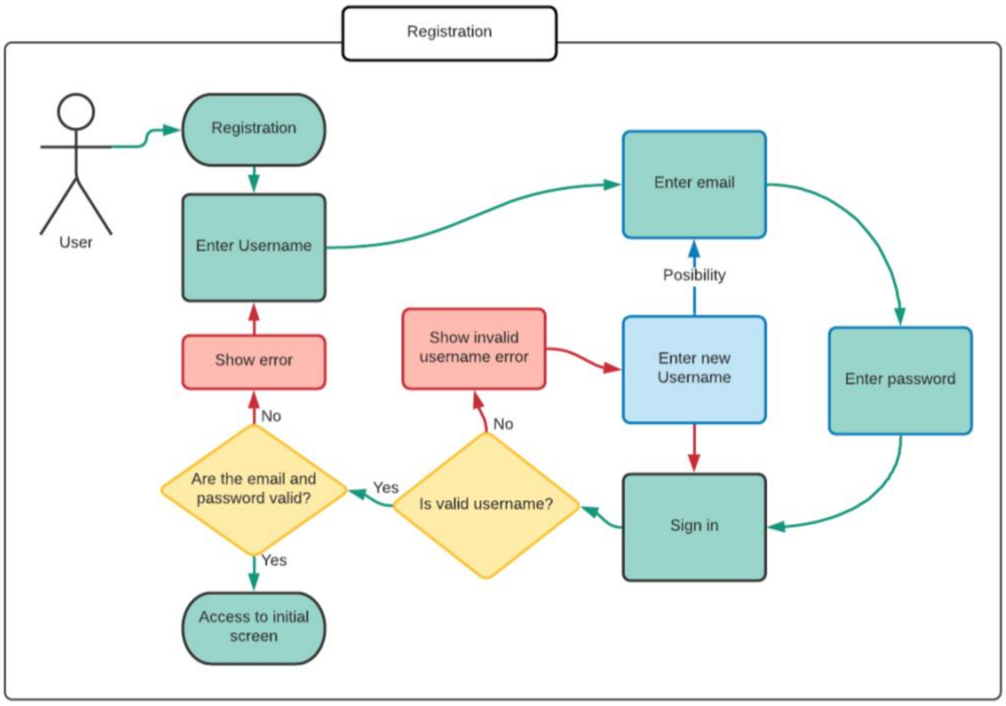


Figure 7 - User registration

3.2.3.2. USER LOGIN

For access, as shown in *Figure 8*, the user must enter the email and password that was entered at the time of registering in the application. One of the proposed future modifications is that the application allows the login with the username or email indistinctly.

On this screen the system generates two possible results:

- Valid email and password information → Access to the application menu.
- Incorrect email and password data → Show error and allow user to start over.

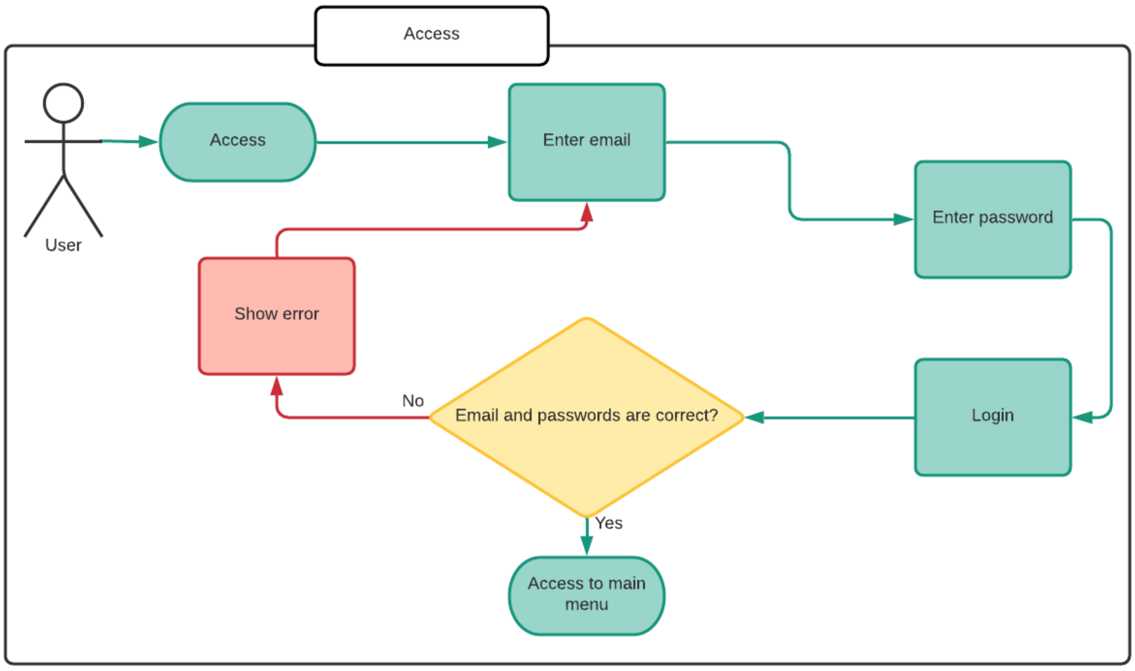


Figure 8 - Access to the application

3.2.4. INTERACTION DIAGRAMS

In these diagrams, several use case scenarios are recreated, but this time focusing on showing the interaction between the user, a set of objects (system and database) that cooperate with each other for the correct operation of the application.

3.2.4.1. LOGIN

This is the user's first interaction with the system. *Figure 9* shows, as mentioned above, that the user must fill in the email and password fields. When the Login button is pressed, the system sends the data written by the user to the database and checks if the information is correct.

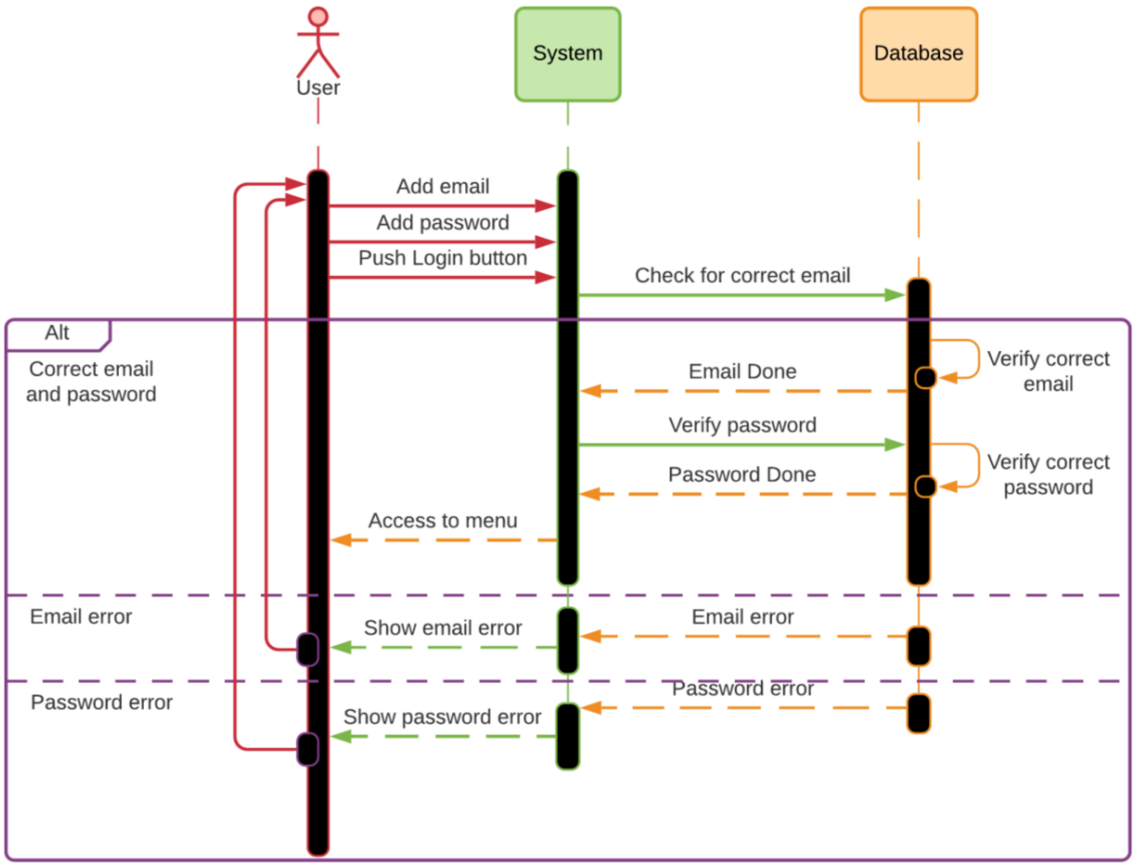


Figure 9 - Access to the application

- If **there is incorrect data**: The database communicates with the system to issue an error message, indicating the reason for the error (email or password).
- If **there is correct data**: The database communicates to the system that the process has been carried out correctly and the system allows the user to access the main menu.

3.2.4.2. USER REGISTRATION

Figure 10 shows the registration process for a new user. To register, a user who does not have an account must press the button to create an account in the application. The system shows three fields to complete (username, email, and password). This information is mandatory and necessary to register the user.

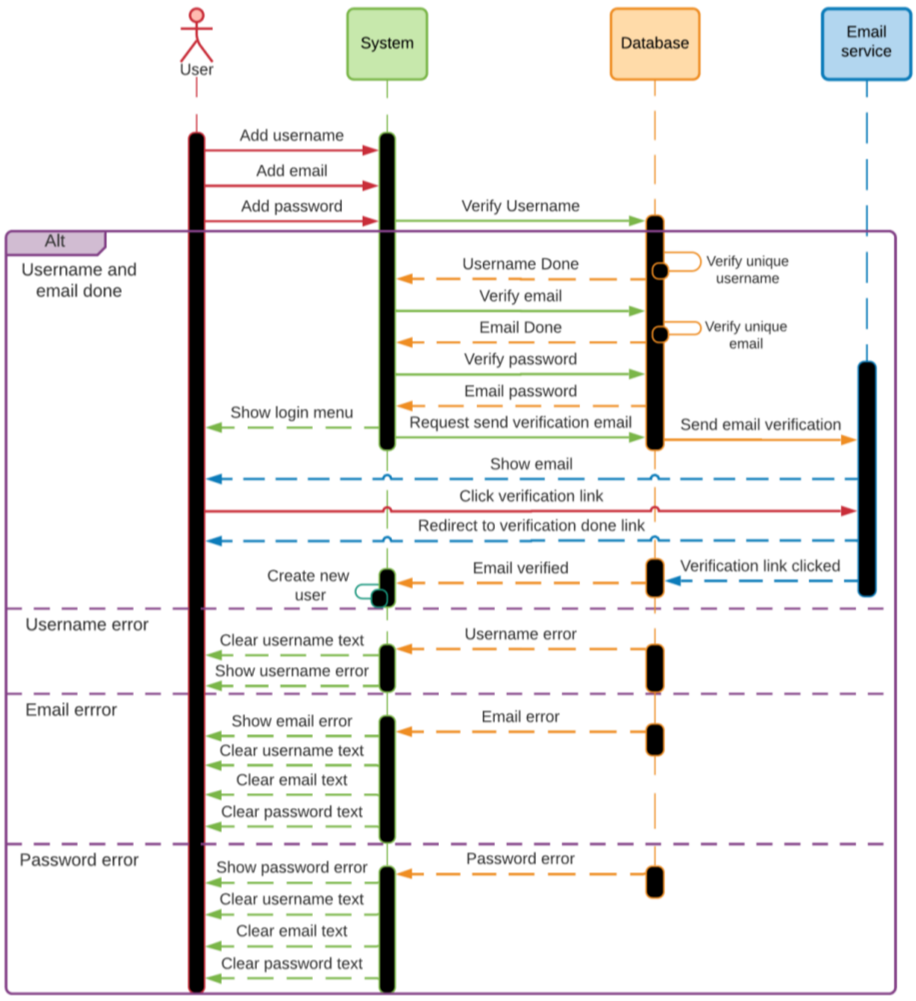


Figure 10 - User registration

When the user inserts the data and clicks the "*Log in*" button, the system verifies the data of the fields. This causes three possible options:

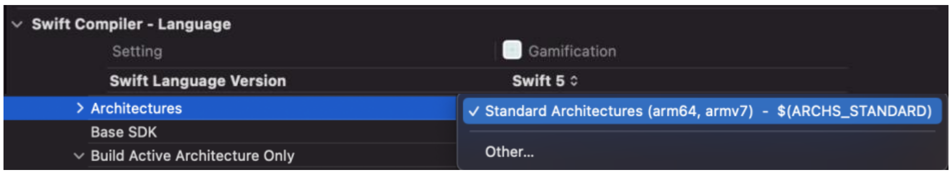
- **Incorrect data (email or password)**: The system informs the user that an error has occurred in one of the fields (email or password), all the fields are reset, and the flow returns to its starting point.
- **Incorrect data (username)**: The system informs the user that an error has occurred in the username field, the username field is restarted, and the flow returns to its starting point but showing the email and password fields full.
- **Correct data**: The system sends an email with a verification link to the user's account, when the user enters the link, the user's account is activated and can be accessed through the access screen.

3.3. SYSTEM ARCHITECTURE

This article describes the necessary hardware and software requirements, a brief explanation of what service Firebase uses and the benefits of using that service.

3.3.1. HARDWARE AND SOFTWARE REQUIREMENTS

Figure 11 shows the compilation language used (Swift 5) and the architecture (standard - arm64, armv7)

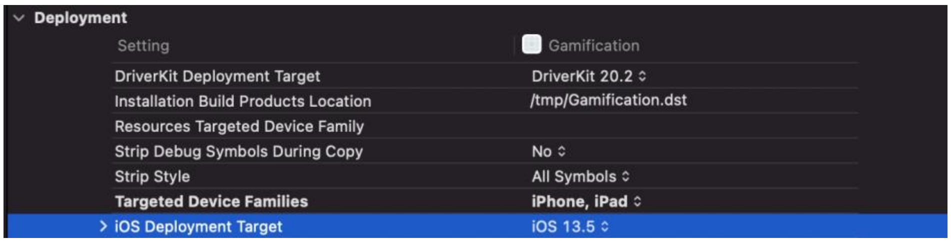


```
⌄ Swift Compiler - Language
      Setting                                  Gamification
      Swift Language Version                   Swift 5 ◇
    › Architectures                            ✓ Standard Architectures (arm64, armv7) - $(ARCHS_STANDARD)
      Base SDK
    ⌄ Build Active Architecture Only           Other...
```

Figure 11 – Architecture and swift version

To use this application, a mobile phone is needed with an internet connection and an operating system iOS 13.5 or higher as shown in *Figure 12*. This decision has been made because iOS versions lower than 13.5 imposes high restrictions on the use of libraries and look at the market shares of the different versions.

Version 13 has a 5.06% market share, while previous versions have 4.49%. Being the remaining 90.46% share for iOS 14, thus obtaining a total market share of 95.52%.



```
⌄ Deployment
      Setting                                  Gamification
      DriverKit Deployment Target              DriverKit 20.2 ◇
      Installation Build Products Location     /tmp/Gamification.dst
      Resources Targeted Device Family
      Strip Debug Symbols During Copy          No ◇
      Strip Style                              All Symbols ◇
      Targeted Device Families                 iPhone, iPad ◇
    › iOS Deployment Target                    iOS 13.5 ◇
```

Figure 12 - Deployment target

On the other hand, among the points to consider for the development of this project is the internet connection. According to a study *(Laura, 2019)*, in our country there are around 46.4 million people and almost 43 million already have access to the internet. If this data is converted to a percentage, 92.67% is obtained.

One of the most difficult decisions has been to develop in the iOS environment. According to a study *(Fernández, R.: 2021)*, in Spain there are 54 million mobile lines, Apple being the second most used operating system, with a market share of 12.7%. Occupying the first place is Android with 87.1% and leaving 0.2% for Windows. Even knowing this statistic, it has been decided to implement the application in the most favourable environment, this is due to the work experience achieved, but continuing with the development in the future implies that one of the first steps to follow is to release the application for Android and thus achieve a market share of 99.8%.

When the above data is gathered, an approximation of the number of people who could use this application. If there are 54 million mobile lines in Spain, 6,858,000 have an iOS operating system. However, only 95.52% have an operating system equal to or greater than iOS13.5, the final figure being 6,550,761 potential users.

3.3.2. MODEL VIEW PRESENTER + CLEAN ARQUITECTURE

For the development of the application, it has been chosen to use the Model View Presenter (from now MVP) architecture in the user interface layer and to follow a clean architecture to request external services. This implemented methodology can be seen in *Figure 13. MVP + Clean Architecture*

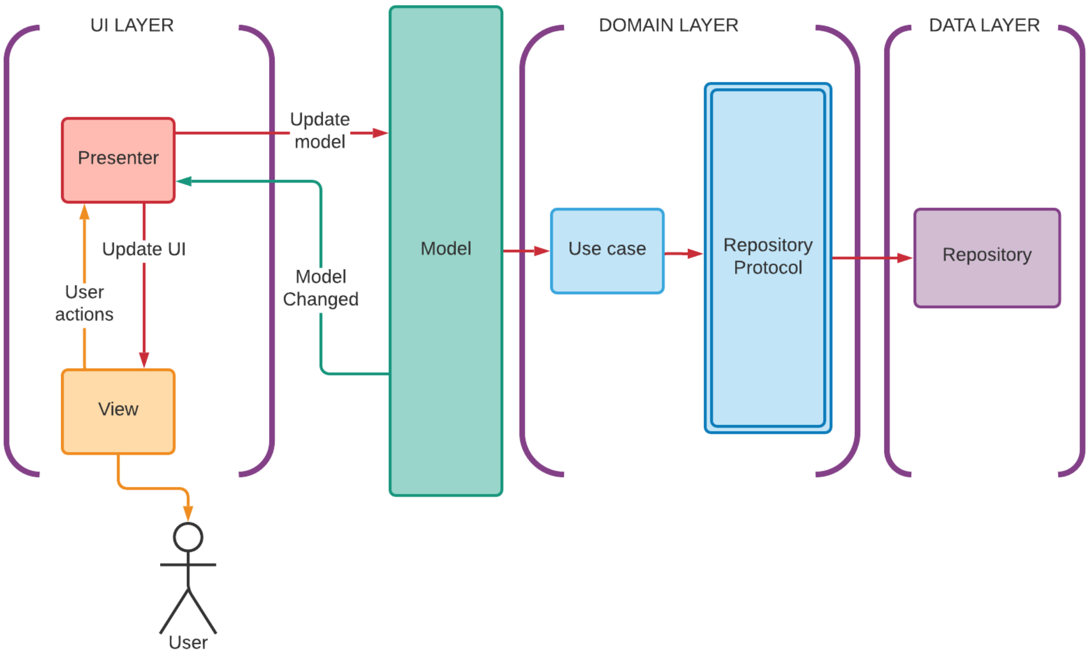


Figure 13. MVP + Clean Architecture

The MVP architecture is commonly implemented for user interfaces, allowing to perform unit tests of the graphical interface.

The model is distributed in three basic components:

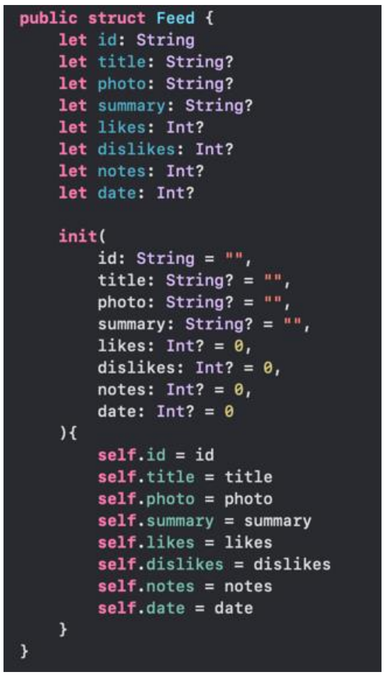
- Model: Allows entry to the domain layer. It is the input to the user interface layer. The model only communicates with the presenter. It has the logic to acquire and persist data from external services.
- Presenter: It oversees carrying the logical load of each view to show. It acts as an intermediate point between model and view. Also update the model if necessary. The presenter does not communicate directly with the view, it implements an interface.

- View: It oversees displaying the data and receiving user interactions to send them to the presenter.

This design pattern comes from the Model View Controller (MVC). MVP improves presentation logic concerns through great control of view-presenter interaction *(Carrera, J. G.: 2014).*

In addition to the MVP pattern, the so-called clean architecture is implemented. Although the term "*clean architecture*" really refers to the organization of the project. This aims to be more understandable, with a decoupled code, robust, scalable, adaptable to changes and sustainable *(Martin, R. C.: 2012).* To achieve these purposes, a series of components are established that perform important functions in the code.

- **Entities:** These contain the rules that are critical for development. They belong to the business so it will always meet that requirement. These rules are created as structures within the project and are unknown to the other layers. These are completely independent of other classes. The *Figure 14* is an example of an entity, that could be that of feedback that has a unique and mandatory identifier, which may have a title and must be of the text type, which will have likes and dislikes, or it may not have any.

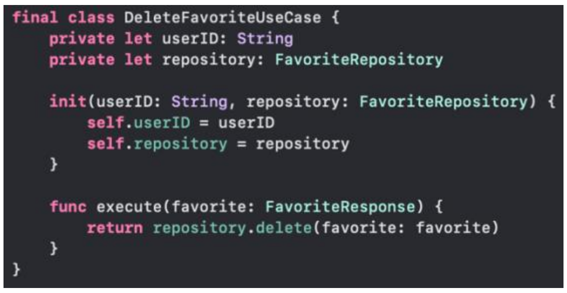


```
public struct Feed {
    let id: String
    let title: String?
    let photo: String?
    let summary: String?
    let likes: Int?
    let dislikes: Int?
    let notes: Int?
    let date: Int?

    init(
        id: String = "",
        title: String? = "",
        photo: String? = "",
        summary: String? = "",
        likes: Int? = 0,
        dislikes: Int? = 0,
        notes: Int? = 0,
        date: Int? = 0
    ){
        self.id = id
        self.title = title
        self.photo = photo
        self.summary = summary
        self.likes = likes
        self.dislikes = dislikes
        self.notes = notes
        self.date = date
    }
}
```

Figure 14. Feed entity.

- **Use case:** These are independent components that perform a specific logic in the project. Use cases interact with entities, but do not reach other layers in the project. Example of this component is a use case that removes a user from the user's favorites list, *Figure 15*.



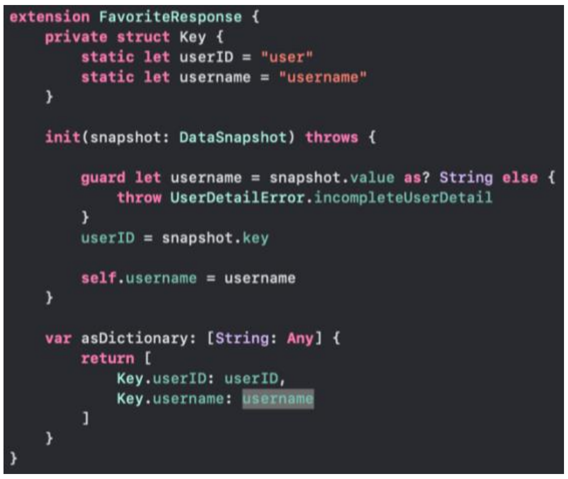
```
final class DeleteFavoriteUseCase {
    private let userID: String
    private let repository: FavoriteRepository

    init(userID: String, repository: FavoriteRepository) {
        self.userID = userID
        self.repository = repository
    }

    func execute(favorite: FavoriteResponse) {
        return repository.delete(favorite: favorite)
    }
}
```

Figure 15. Delete Favorite user use case.

- **Adapters:** These components transform the information that enters and leaves to decouple the external elements of our code. Create an abstraction layer over the use cases. They are also on the interface side, transforming the data and information of the views to the understandable by the use cases. As an example, an extension is shown that converts elements from an external type to an internal one of the project, *Figure 16*.



```swift
extension FavoriteResponse {
    private struct Key {
        static let userID = "user"
        static let username = "username"
    }

    init(snapshot: DataSnapshot) throws {

        guard let username = snapshot.value as? String else {
            throw UserDetailError.incompleteUserDetail
        }
        userID = snapshot.key

        self.username = username
    }

    var asDictionary: [String: Any] {
        return [
            Key.userID: userID,
            Key.username: username
        ]
    }
}
```

Figure 16. Adapter to convert favorite item from firebase to a favorite entity.

3.3.3. FIREBASE

Firebase is a BaaS (Backend as a Service) that provides customers with various services when creating a mobile application, such as user creation and authentication, database, storage, analytics, error management and notifications...

There are several services, which considerably reduce the development time of the application thanks to their unification. The difference between traditional versus BaaS can be seen at the *Figure 17*.

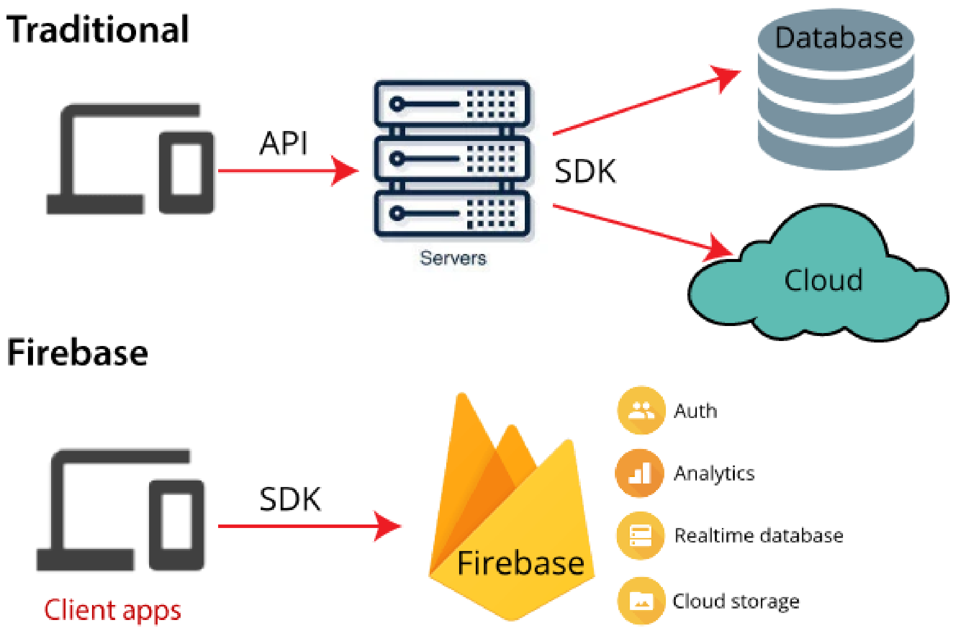


Figure 17 - Traditional vs firebase model.

3.3.4. ADVANTAGES AND DISADVANTAGES OF A BACKEND PLATFORM AS A SERVICE

BaaS provides many benefits, but the most important ones for the application to be developed will be detailed below:

- **Agile and fast provisioning:** It is not necessary to hire servers, databases or install software.
- **Faster development:** Eliminates the need to program all layers and unifies the backend into one, reducing time considerably.
- **Security:** Provides tools to keep user`s private information safe.

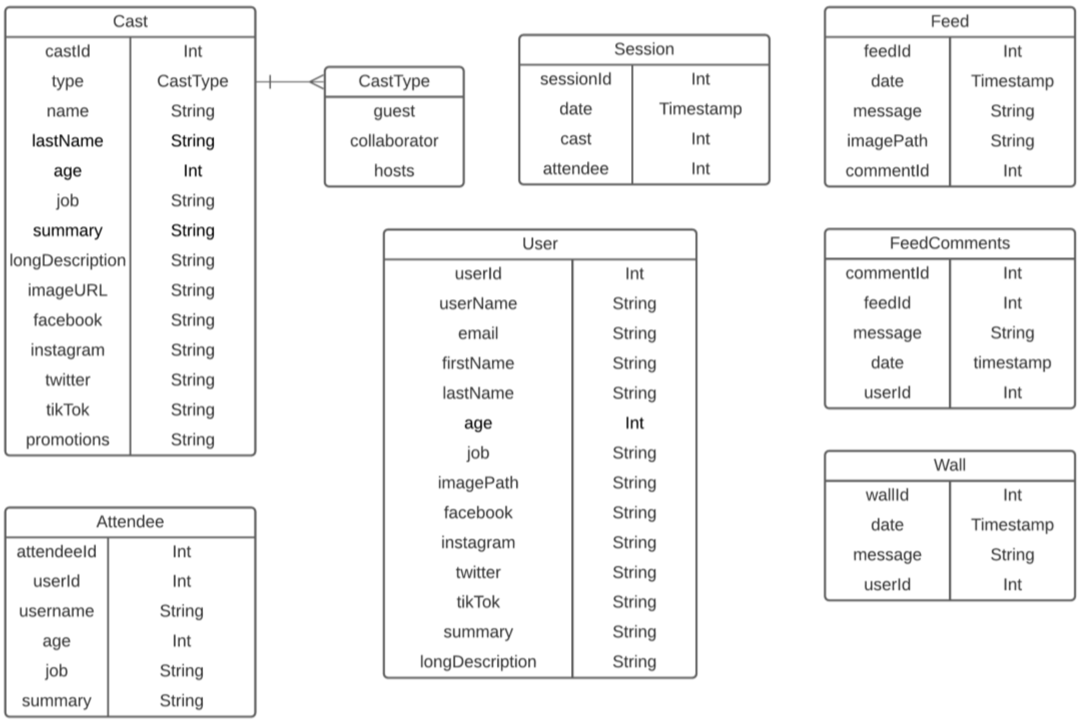
However, there are also disadvantages that come from using a backend as a service:

- **Less flexibility** compared to custom programming.
- A lower level of customization compared to a custom backend.
- **Dependence** on a provider.

As can be seen, the disadvantages of this service are insignificant compared to the advantages it brings to the project.

3.4. DATABASE DESIGN THROUGH FIREBASE

Figure 18 shows the database containing the following tables: cast, user, attendee, session, source, news, feed comments, and wall.



Cast	
castId	Int
type	CastType
name	String
lastName	String
age	Int
job	String
summary	String
longDescription	String
imageURL	String
facebook	String
instagram	String
twitter	String
tikTok	String
promotions	String

CastType
guest
collaborator
hosts

Session	
sessionId	Int
date	Timestamp
cast	Int
attendee	Int

Feed	
feedId	Int
date	Timestamp
message	String
imagePath	String
commentId	Int

User	
userId	Int
userName	String
email	String
firstName	String
lastName	String
age	Int
job	String
imagePath	String
facebook	String
instagram	String
twitter	String
tikTok	String
summary	String
longDescription	String

FeedComments	
commentId	Int
feedId	Int
message	String
date	timestamp
userId	Int

Attendee	
attendeeId	Int
userId	Int
username	String
age	Int
job	String
summary	String

Wall	
wallId	Int
date	Timestamp
message	String
userId	Int

Figure 18 - Database table layout

3.4.1.1. CAST

This table stores the cast data, such as social networks, a summary of their work, what they do, age, etc.

3.4.1.2. USER

Base user of the application, here you will find all the information of the users: name, email, surname, social networks, work, age, description, type, and summary.

3.4.1.3. ATTENDEE

Assistant user, only those who are in a session on the same day of the session will be assistant users. This table will be updated daily to accommodate only attendees. The information that is stored is more limited: name, age, job, and summary.

3.4.1.4. SESSION

Table that allows linking the sessions with the cast and the attending users.

3.4.1.5. FEED

Table of messages from the organizers, which are composed of an image, text.

3.4.1.6. FEED COMMENTS

Table that stores the comments of the feed, the id of the user who makes it and the date.

3.4.1.7. WALL

The wall is a storage place for the messages of the users attending the day of the event. It will provide a kind of unique chat in which to comment on anything.

3.5. INTERFACE DESIGN

This section shows the entire interface, with the different screens of the application and their respective explanation.

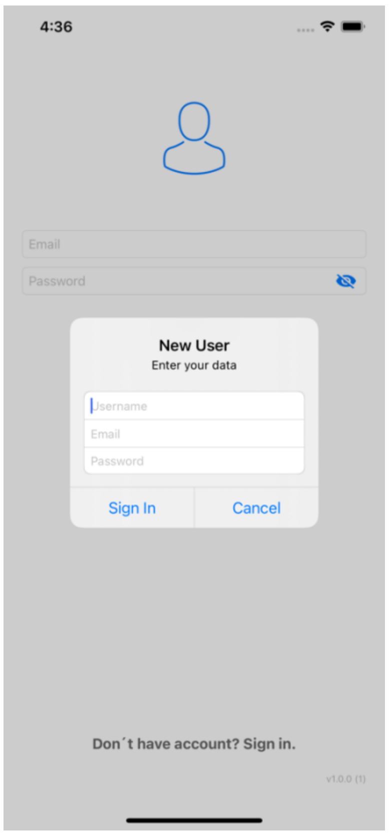


Figure 19. First registration layout

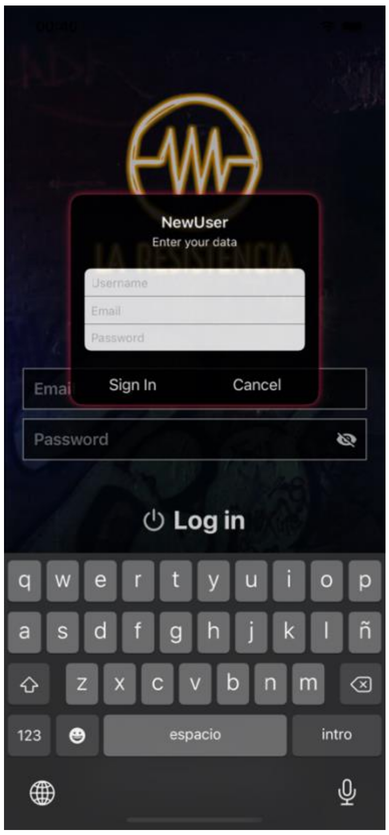
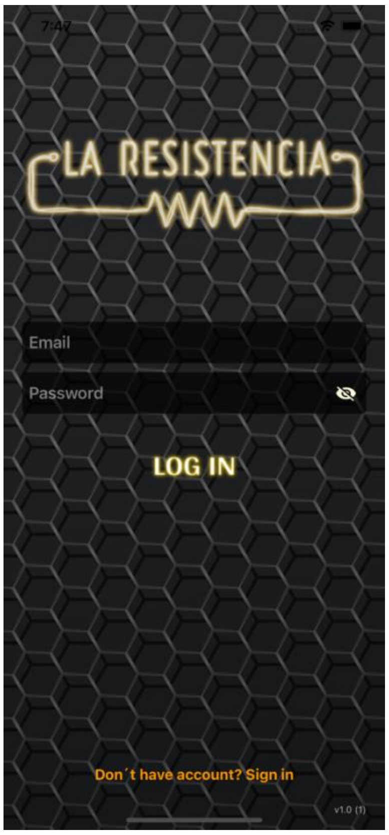


Figure 20. Final registration layout

Registration view

For the user registration it has been obtained by a subview that requests the necessary data. This subview is superimposed on the previous one. In the first design, *Figure 19*, no special colors were set for this view.
The final design, *Figure 20*, shows various enhancements such as a translucent background, text colors, and the application of a shadow effect around the view.
This screen shows the different fields that the user must complete (Username, email, password)



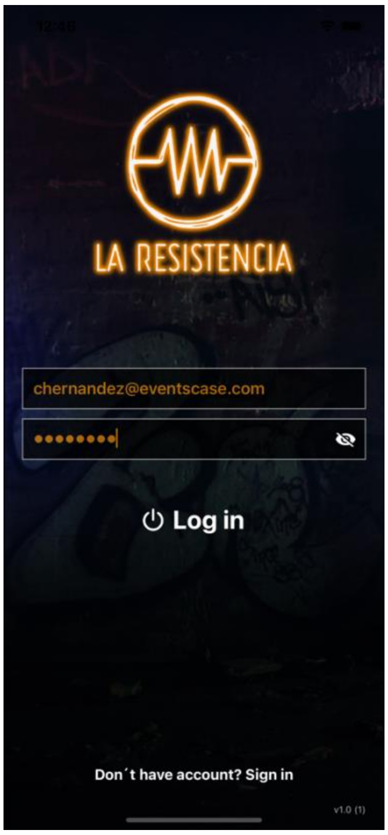


Figure 21. Initial login view layout

Figure 22. Final login view layout

Initial view

Figure 21 shows the initial design, *Figure 22* shows the final design. The redesign of the application to have a more current appearance is noticeable between the two figures.
This view displays the fields so that users can log into the application and access the main menu.
It has two bars where you enter your email and password.
The password bar shows the hidden text, making it visible by pressing the button on the right.
To access you must click on the central login button.
The user will be able to access the registration menu through the text at the bottom.
The version of the application is displayed at the bottom right.

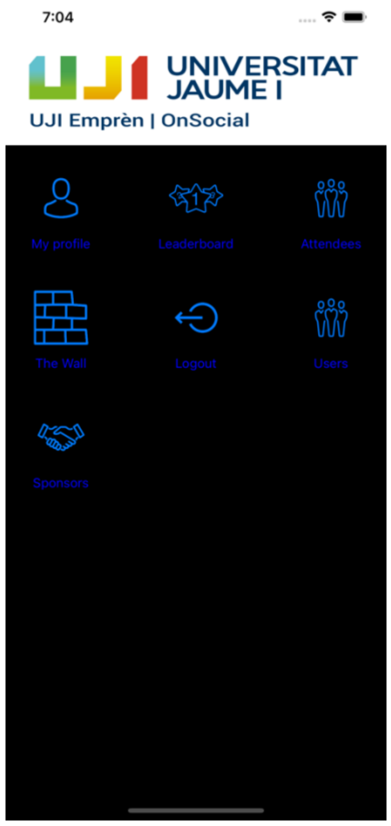


Figure 23. Initial menu view layout

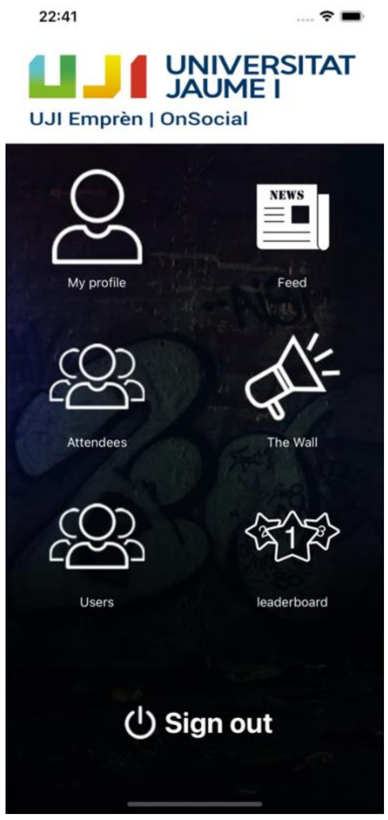
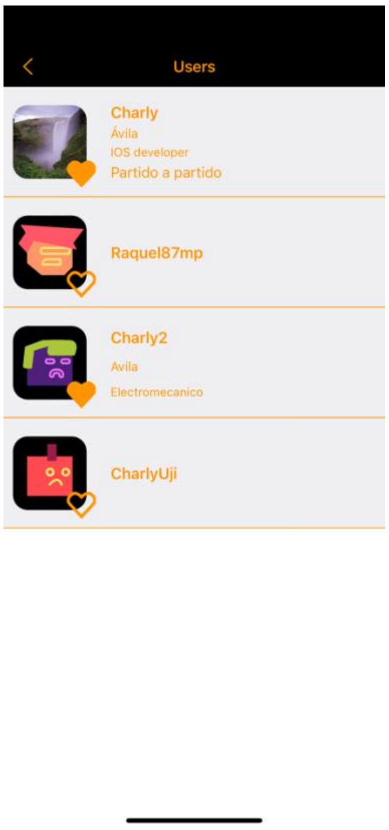


Figure 24. Final menu view layout

Menu view

The main menu is an icon-based view of the different sections. The first layout, *Figure 23*, offered smaller icons in rows of three items, but in the final layout, *Figure 24*, a large bottom button is implemented to allow logout and a series of icons in rows of two items.
As you can see from the screenshots, a banner can be entered at the top and applied at the bottom together or separately from the top.



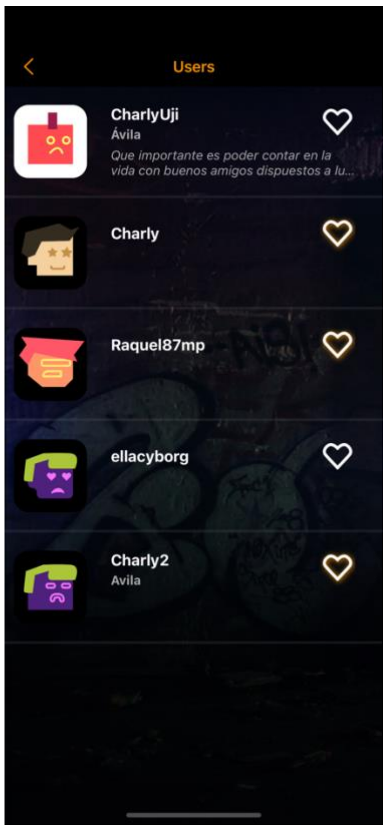

Users
CharlyUji
Ávila
Que importante es poder contar en la vida con buenos amigos dispuestos a lu...
Charly
Raquel87mp
ellacyborg
Charly2
Avila

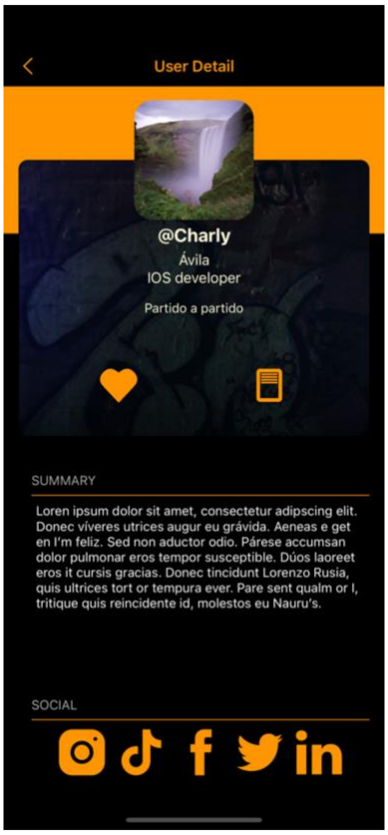
Figure 25. Initial user list view layout

Figure 26. Final user list view layout

User list view

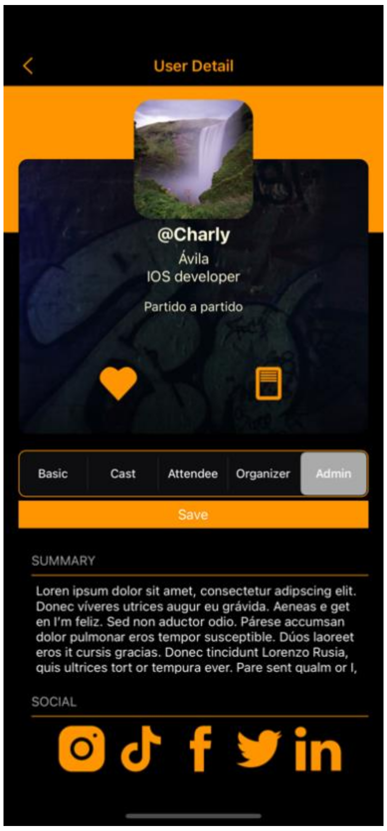
The upper figures are two different designs of the view of the list of users of the application. In the initial design, *Figure 25*, the background of the table was opaque, the texts were less highlighted according to their category and the favorite button was part of the user image. In the final design, *Figure 26*, changes are made so that the list is more integrated and with a more professional design.

From this view the user can access the different details of the users, as well as add or remove users from their favorites. When the leaderboard is implemented, it will be possible to see through an icon that users have obtained a trophy in gamification games.



Loren ipsum dolor sit amet, consectetur adipscing elit. Donec víveres utrices augur eu grávida. Aeneas e get en I'm feliz. Sed non aductor odio. Párese accumsan dolor pulmonar eros tempor susceptible. Dúos laoreet eros it cursis gracias. Donec tincidunt Lorenzo Rusia, quis ultrices tort or tempura ever. Pare sent qualm or I, tritique quis reincidente id, molestos eu Nauru's.

Figure 27. User detail view without type functionality

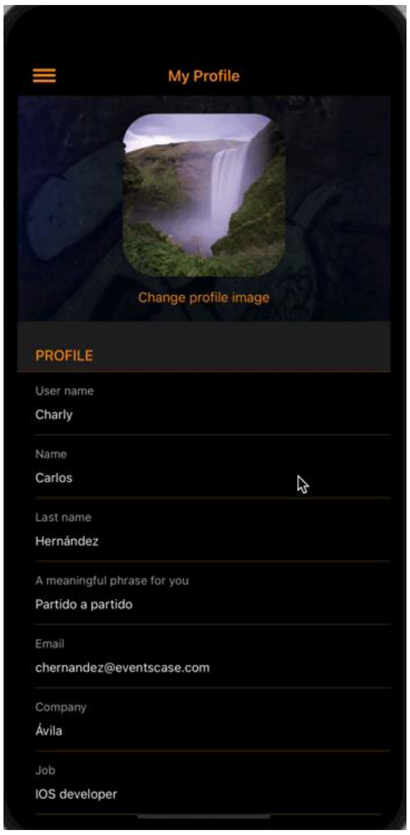


Loren ipsum dolor sit amet, consectetur adipscing elit. Donec víveres utrices augur eu grávida. Aeneas e get en I'm feliz. Sed non aductor odio. Párese accumsan dolor pulmonar eros tempor susceptible. Dúos laoreet eros it cursis gracias. Donec tincidunt Lorenzo Rusia, quis ultrices tort or tempura ever. Pare sent qualm or I,

Figure 28. User detail view with type functionality

User detail view:

In this view the user can learn more data about a particular user, *Figure 27*. Access to the view is through the list of users. In the view there is the "favorite" button to mark the user as a favorite, the "note" button to add a note about it or the buttons to access the user's social networks. In addition, if an administrator type user accesses, an extra functionality is shown that allows modifying the type of user and thus being able to add more administrators, organizers or even change a user as an assistant manually, this can be seen in *Figure 28.*



PROFILE

User name
Charly

Name
Carlos

Last name
Hernández

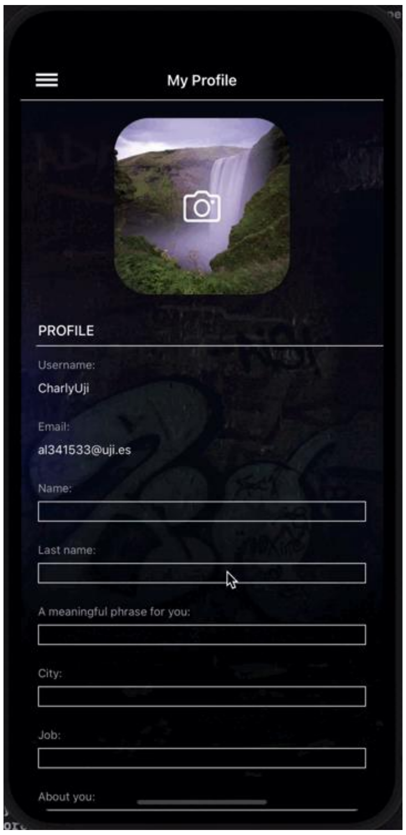
A meaningful phrase for you
Partido a partido

Email
chernandez@eventscase.com

Company
Ávila

Job
IOS developer

Figure 29. First user profile layout



PROFILE

Username:
CharlyUji

Email:
al341533@uji.es

Name:

Last name:

A meaningful phrase for you:

City:

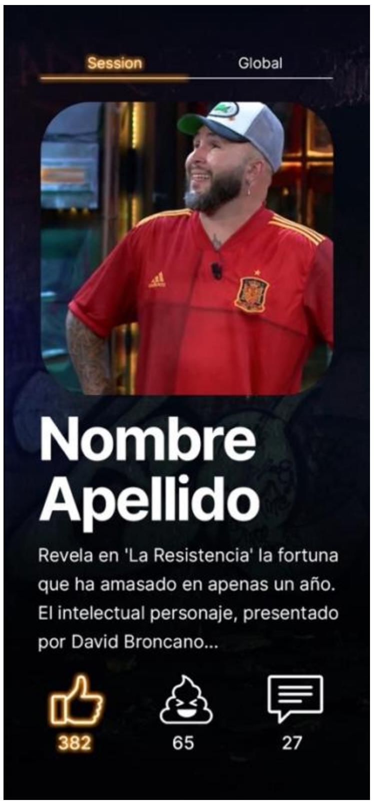
Job:

About you:

Figure 30. Final user profile layout

User Profile:

One of the most important views is that of the user profile, the initial design can be seen in *Error! Reference source not found.* and the final design can be seen in *Figure 30*. This view allows users to change and review their data, although there is non-modifiable data such as email or username, or change the profile photo. These data will be visible in the user's details except for the name, surname and email to preserve the user's confidentiality.



Revela en 'La Resistencia' la fortuna que ha amasado en apenas un año. El intelectual personaje, presentado por David Broncano...

Figure 31. Feed view layout

Feed view:

In the *Figure 31*, the final design of the feed view is shown. This view shows the posts that organizers will create to provide information to users. The view is divided into two subviews that contain the posts that refer to the current session and the posts that refer to global events. In this view you can see an image, a title or first and last name, and a summary of the news that is referenced. Users can click the Like or Dislike button or comment on the post.

4 WORK DEVELOPMENT AND RESULTS

Next, the most relevant aspects of the project will be explained, the problems encountered when carrying out each section, the objectives that have been achieved during the project and the results of carrying out the work.

4.1. WORK DEVELOPMENT

4.1.1. OBJECTIVES ACHIEVED

Develop an access system for users that allows them to create their password, access the application and retrieve the password in their email.

The result of this section has been much higher than initially expected. At this point, not only a system has been implemented that allows the user to access the application with a password and email. Rather, it provides several substantial improvements to that section.

The first, verification by sending email to the user. The user accesses the email received and clicking on the link ensures that he is the owner of that email, avoiding duplicate email accounts.

The second, an own and unique username for each user, providing a user identifier that allows it to be recognized by other users and offers the possibility of improving the login in the future by changing email for username.

The third improvement consists of an improvement when the application is accessed again once you have been logged in. With this improvement, it is possible to access the menu without having to enter the username and password each time it is opened. Finally, users will have a unique avatar as soon as the account is created. This avatar is linked to their username, the user can change it from the profile at any time and is achieved through an automatic online avatar generator.

Service web API to generate a random avatar image for users https://api.minimalavatars.com/avatar/<NombreDeUsuario>/png

Create a menu and a user interface.

This point has been achieved and its behavior has also been improved with that initially proposed. A more intuitive app has been created for users, with a more refined aesthetic and that in general terms offers a better user experience. Banners have been implemented that allow advertising to be inserted within the application and a loading screen that can be customized with the desired image. The appearance of the application itself has been refined twice because the first design seemed rough and outdated, giving way to a much more current design in tune with the design of the television show itself.

Generate a firebase project that allows users to access with their email and password.

This development has been exceeded meeting expectations. Users have a registration area in which their data is processed safely. The necessary characteristics have been fulfilled so that users can collect global information from the database, obtain the details of each user and an online storage space has been created for the images of the user's avatars.

Create the different submenus and bind them to the database. Users, wall messages, feed, etc.

In this case, the problems found within the different user implementations have resulted in the development for casting, questions, or leaderboard not being finalized. On the other hand, if the development of users and favorites has been completed, having managed to create their firebase tables. As an improvement on this point, several new tables have been created. These tables are unique usernames linked to user identification and stored images of each user.

Implement a scoring system for the user's leaderboard.

This point has remained without being able to implement since the valuation of hours was too optimistic. The lack of knowledge of the time cost of generating certain views of the application has led to a delay in developments.

Create the different endpoints to be able to manage the message wall of the attendees and the ranking of scores.

In this section, the message storage system has been completely generated in the feed, but it has not been possible to cover the development of punctuation and questions to users.

Make the technical proposal

The technical proposal has been generated successfully, complying with what is established within the university guidelines.

Perform final memory

The final report has been successfully generated and completely revised.

Prepare final presentation

This item has been successfully completed within the correct time frame.

4.1.2. PROBLEMS FOUND

Several problems have been encountered in the development of this final degree project. First, the development has been done with the knowledge of a junior developer in the field of programming in Swift. The ignorance of certain functionalities of the native classes of the language and that making views that seemed simply took longer when the desired implementation was not achieved has caused a delay in the entire project.

Another problem encountered that has delayed development the most has been the search to implement a correct model and a clean architecture.

This development would have been much faster using only an MVP architecture without more, but it has been desired to apply a clean architecture based on use cases, entities, and adapters to improve the requests to the database, as well as avoid the existence of parts of the code dependents. Over time, this implementation offers greater testing ease and allows modifications to be less expensive, but initially it has caused poor implementation of entities to generate extra debugging work and in which several days of work have been lost due to misuse of entities within classes.

Finally, the development of the views in the Xcode environment, although more visual, is much more complex than in the Android Studio environment, since "constraints" must always be applied with respect to the other elements or the views. These constraints are difficult to handle and not placing them correctly causes display failures or compilation errors. During the development of this project, much has been learned how to hold the elements in different ways using these constraints, making the work of generating a view less expensive.

4.1.3. RELEVANT ASPECTS

The development of the application has followed the model-view-presenter (MVP) pattern complemented by the clean architecture based on repositories.

Following this pattern has led to the development being initially more expensive in terms of time, because in addition to the views necessary to implement this pattern, use cases must be generated that use adapters to feed from external repositories and entities as information containers. All this architecture avoids that a dependency is created with any service external to the development and thus it is easier to modify the conditions of use or change the service according to the needs.

4.2. RESULTS

After completing the development of the Project, an application has been generated in which the users of the television show can meet, get to know each other, and receive feedback from the program.

5 CONCLUSIONS AND FUTURE WORK

5.1. CONCLUSIONS

The conclusion obtained from this development is that, although I was working as a developer for iOS when I started to develop the project, I did not have enough notions to assess the temporary costs of carrying out its execution. A developer who enters a project and works making modifications does not learn or learns very little about the costs of generating from scratch, the views, and the implementations of the different classes.

Thanks to the development of the project, we have learned much better how to deal with requests for external services, how to execute the implementations of use cases that perform a single function in the code and to better understand the different internal classes and their multiple properties that facilitate the development of the final application.

5.2. FUTURE WORK

One of the first jobs to be done in the future is to generate a version for Android devices. This version will be made in Kotlin to take full advantage of both the language similarities with Swift and the new features that Android offers to developers and that are only accessible for this language.

On the other hand, another great advance would be to offer a Backend service for the application since in this way the application could provide a much greater service in addition to allowing better management of all the administration procedures of the displayed elements.

Of course, the gamification, leaderboard, casting and agenda sections would have to be finished.

In turn, if the program is persuaded to incorporate this application into its services, it would be necessary to add different sections such as access to its sales website.

As interesting features, a video stream system could be introduced that would allow users to see the program in real time and be able to participate in it. It would also be interesting to be able to group people by interests or allow users to know how many they are favourites or an internal messaging system.

6

CITED WORKS

Moroney, L., Moroney, & Anglin. (2017). Definitive Guide to Firebase (pp. 51-71). Apress.

https://www.yeeply.com/blog/cuanto-cuesta-crear-una-app/

Fernández, R. (2021, April 29). Statista. Retrieved from https://es.statista.com/estadisticas/473759/tasa-penetracion-sistema-operativo-smartphone-espana/

Laura. (2019, 21 May). Orange.es. Retrieved from https://blog.orange.es/noticias/acceso-internet-mundo

Carrera Guanoluisa, J. G. (2014). Análisis comparativo de la productividad entre los patrones de dieño Modelo Vista Controlador (MVC) y Modelo Vista Presentador (MVP) aplicado al desarrollo del Sistema Nómina de Empleados y Rol de Pagos de la Distribuidora Soria CA (Bachelor's thesis).

Martin, R. C. (2012). Código limpio. Anaya Multimedia.