POLITECNICO
MILANO 1863

Riccardo di Palma – Leonardo Giusti

DJParty Design Document

Design and Implementation of Mobile Applications 2023

INDEX

INTRODUCTION

1.1 Purpose and Document Structure

The main purpose of this document is to give a complete overview of the *DJParty* application, going through all the system's functionalities, architecture, user interface and external libraries. It is written for programmers who want to go in deep into the implementation and add new features and for users interested in understand and discover all the application's possibilities.

1.2 Application Scope and Overview

DJParty is a mobile application that allows users to have music for a party without the presence of a DJ. Interfacing with the Spotify API, the party host will connect the device to the audio system, and the party queue is organized by all the guests, which can add tracks directly from their applications. Periodically, users can vote to their favourite songs, and the order of the queue is updated with the most voted ones on the top.

1.3 Goals

The main *DJParty* functionalities are:

- **Authentication**: the user is allowed to create an account and authenticate to access the application functionalities.
- **Join/Create a Party:** the user can join a party as a guest or create one.
- **Add/Vote songs:** all the users are allowed to add songs to the party queue and to vote to their favorite songs in the voting phases.
- **Play songs:** the party host's system is allowed to play songs following the order of the queue.

- **Compete:** a ranking page shows all the people that joined the party. After each voting phase, the ranking is updated putting on top the users that added the most voted songs.
- **Playlist Creation:** after the ending of the party, each user can create a Spotify playlist with all the played songs inside.

All the functions will be explained in depth in **2.1**

1.4 Definitions and Abbreviations

- **ADMIN:** The party Host
- **API:** Application Programming Interface
- **DD:** Design Document
- **GUEST:** The user invited to a party
- **SDK:** Software Development Kit
- **UI:** User Interface
- **UX:** User Experience

FUNCTIONALITIES

2.1 Product Functions

2.1.1 Authentication

The Authentication Phase is implemented into *lib/page/auth folder*.
Three main sections are present here:

- Registration: a new user is allowed to create an account by inserting a valid e-mail, password and password confirmed. In order to confirm the account registration, an e-mail is sent to the user. After this procedure, the access to the application and the

account creation is completed, and the user will be automatically logged in during the following sessions.

- Google and Facebook authentication: in order to have a faster and user-friendly authentication procedure, signing in with Google or Facebook account is implemented. Using this approach, some of the user's information like name and profile picture are directly inserted into the *DJParty* account.

- Sign In: if the user logs out from the application, a sign in function is implemented into the *lib/page/auth/Login.dart* page. The account's e-mail and password can be inserted and the log in procedure is complete.

2.1.2 Create Party

The *lib/page/lobby/GenerateShare.dart* page deals with the creation of a party. Into a textField the Admin can insert the party title and can click on "Confirm" to create the party environment. The New party is identified by a random code with a mixture of 5 letters and numbers and a Qr Code.

2.1.3 Share Party

In the *lib/page/lobby/Home.dart* page of the application, the list of the user's parties is present. For each party, a Share function is implemented: by clicking the "Share" button, the user is allowed to send the party code and the respective Qr to other people.

2.1.4 Join Party

Once invited to a party, two different ways to join are implemented in *lib/page/lobby/InsertCode.dart* :

- Qr Scanning: the guest can scan the party QR directly inside the *DJParty* environment. After the scanning procedure, if it's successful, the user is redirect directly to the party page. If the Qr code is wrong a *'This code does not correspond to any party'* message is displayed.

- Inserting the Party Code: the guest can manually type the party code and access to the party page. If the code is wrong a *'This code does not correspond to any party'* message is displayed.

2.1.5 Add Songs to the Party Queue

Once inside the party page, the user (admin or guest) is allowed to add songs to the party queue. This can be done by interacting with the Spotify Web API. In the *lib/page/party/QueueSearch.dart* a textField is displayed for the searching procedure. Each time the user types a letter, an updated list on songs (cover image, title and artists) is displayed below the searching field. When the user finds the desired song, he can click on it and a dropdown menu appears with an "Add To Party Queue" button. Once clicked, the song is added to the party queue.

2.1.6 Voting Phase

One of the most interesting functions of the application is the voting phase. When the party is started, the users are allowed to vote periodically for their favorite tracks. The duration of the voting phase and the interval between the voting phases are set and can be modified by the admin only. At the end of the voting phase, the order of the party queue is updated with the most voted track on top, so that it will be the next song to be played, and the others ranked by the number of votes. Also the ranking page, i.e. the list of users in the party environment, is updated by the following criteria:

- the user who added the most voted track gain 1 point

- the users that voted the most voted track gain 1 point

2.1.7 Player

The admin's device is connected to the audio system and play the music. By interacting with the Spotify SDK, the application allows to have some player features. A distinguish between the admin's player page and the guests' player page is done:

- Admin Player: when a specific song is playing, the admin can play, pause, restart the song and skip forward. Is also present a progress bar that represent the current position of the song. Above the buttons section, the cover image of the current song is displayed. If the admin's application goes background, the music will automatically stop and when the admin goes back to the application, the next song in queue will start.
- Guest Player: the structure of the player for the guests is the same as the admin one, except for the button section, because the guests are not allowed to play, pause and skip songs.

2.1.8 Party Playlist

After a party is ended, the users are no more allowed to add songs, but they can obtain the playlist of the party, with all the songs played, directly in their Spotify application.

2.2 Design Constraints

The application requires a few permissions in order to properly satisfy alle the goals and functionalities mentioned above:

- **Registration**: the user must provide e-mail and password in order to create an account. If the user decides to log in with Google or Facebook, he has to give the right permission to the system in order to take the useful information to complete the access.

- **Admin**: for the correct behavior of the application, the admin of the party must have a Spotify Premium account and the Spotify app correctly installed in the device.

- **Guests**: the party guests must have a Spotify Account in order to let the system make the proper request to the Spotify Web Api.

- **Hardware**: DjParty has a few hardware limitations: the application is supported only on Android and iOs systems, and Internet connection is required.

2.3 Utils and Services

2.3.1 Utils

In the *lib/utils* folder are present some auxiliary functions that are useful for a better workflow in the system's implementation, like next screen functions, show snack bar or layout functions.

2.3.2 Services

In the *lib/services* folder are present some *.dart* files that contains functions for interaction with the Sign In Providers, the Firebase Firestore, the Spotify API and the Internet Provider.

- *lib/services/SignInProvider.dart* : in this page are present three main functions for the signing in process, which are the E-mail and Password, the Google and the Facebook sign in.

- *lib/services/FirebaseRequests.dart* : in this page there is a list of functions that allows the application to interact with the Firebase Firestore database.
  The main functions are divided by: "get" functions for extracting information from firestore, "check" functions for detecting if a party or a user actually exists, and some "add/remove" functions that allows to modify the content of the database by adding or removing data.

- *lib/services/SpotifyRequests.dart* : in this page are present the main functions for requesting to the Spotify API. The most

relevant ones are the creation of a Spotify playlist, the connection to the Spotify player and the request for the Spotify authentication token for validating all the requests.

- *lib/services/InternetProvider.dart* : in this page is present the function for detecting if the internet connection is present.

2.4 Functional Requirements

In this section are outlined the various functional requirements of the application divided by goals:

- **Authentication**
  1. The system must provide a form to let the user register.
  2. The system must provide a form to let the user log in.
  3. The system must notify a registering user if the inserted email is already assigned to a registered account.
  4. The system must notify a signing in user if the inserted password is wrong.
  5. The system must notify the user to insert a valid email if the pattern inserted isn't correct.
  6. The system must allow users to reset their password using their email.
  7. The system must securely store tokens used for authentication.
  8. The system must allow users to log out.

- **Lobby**
  1. The system must allow users to create a new party.
  2. The system must allow users to join new parties by inserting the right code.
  3. The system must notify the user if the party code is wrong.
  4. The system must allow user to join parties with the Qr scanner.
  5. The system must notify the user if the Qr-code is wrong.
  6. The Home Page must display all the user's parties, not started, started or ended.

7. All the items of the party list in the Home page must be expandable, in order to access the exit,share and join buttons.
8. The system must allow users to update their personal information into the editProfile page.

- **Party**
    1. The system must allow users to search for a song.
    2. The system must allow users to add a song to the party queue.
    3. The system must switch between voting and non-voting phases.
    4. During a voting phase, the system must allow users to vote for their favourite songs.
    5. The system must update the order of the party queue.
    6. The system must show the ranking of the party.
    7. The system must update the ranking order according to the scores of the users.
    8. The system must show the cover image of the currently playing song in the Player section.
    9. The system must show the progress bar that identifies the duration and position of the currently playing song.

2.5 Assumptions and Dependencies

In this section we present a list of assumptions that has to be taken in order to let the system work as expected:

1. Users have a valid e-mail address or a Google/Facebook account with which they can register and access to their area.
2. The external dependencies and services which allows the system to work properly are reliable and updated regularly.
3. The internet connection works properly and without failures.
4. Users are willing to give us access permission to their Spotify Accounts in order to interact with the Spotify API and the Spotify Player SDK.
5. The users will contribute valuable content to the app and not spam.
6. Users have a basic knowledge of the English language.

2.6 Performances

In this section we provide a set of graphics obtained through the Flutter-DevTools that show the actual performance of the application client:

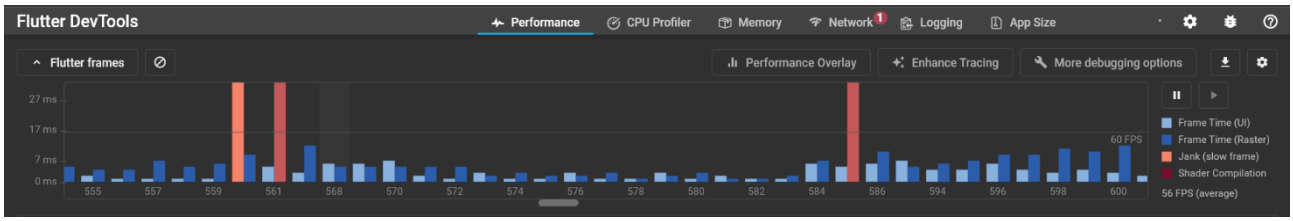


Figure 1 -Flutter FrameChart

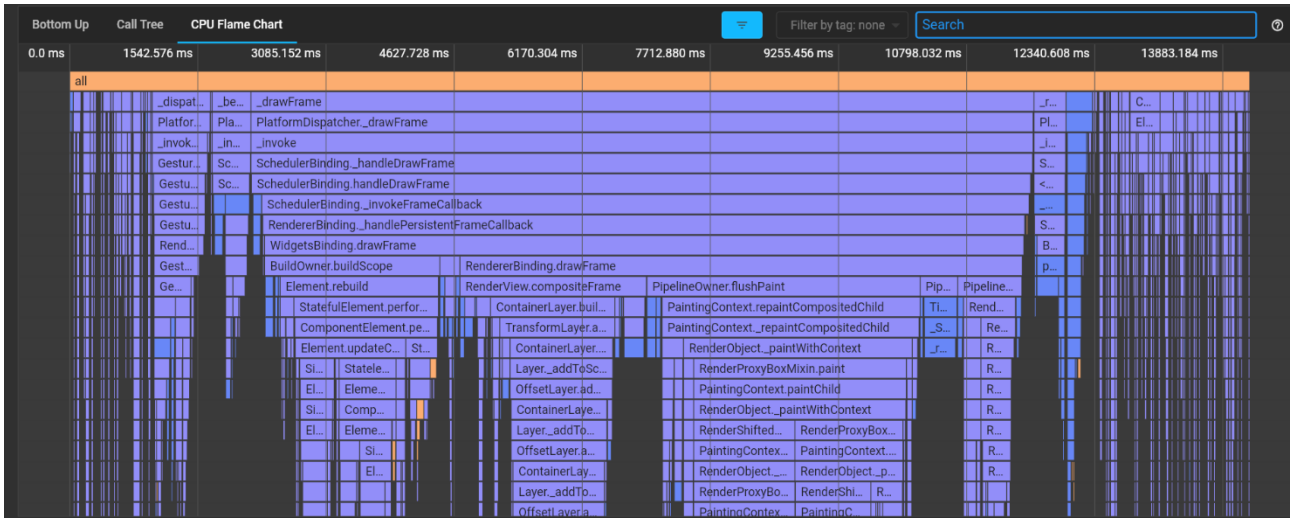


Figure 2 - analysis of the CPU samples for the selected frame event

2.7 System Attributes

2.7.1 Reliability

The application should be available 24/7 and the Admins should release updated version of the application periodically.

2.7.2 Availability

The system has to be available 99.99% of the time and depends on Firebase and Spotify, which can be considered as very reliable.

2.7.3 Security

All the users' sensitive information, i.e. the e-mail addresses and the profile images, has to be stored securely, in order to avoid external or not authorized users to read the data.

2.7.4 Maintainability

In order to guarantee a good maintainability of the application, the code will be simplified as much as possible, with sections properly commented into the English language.

ARCHITECTURE

DjParty employes the client-server architecture. The client-side was developed using Flutter, and the server-side through Firebase.

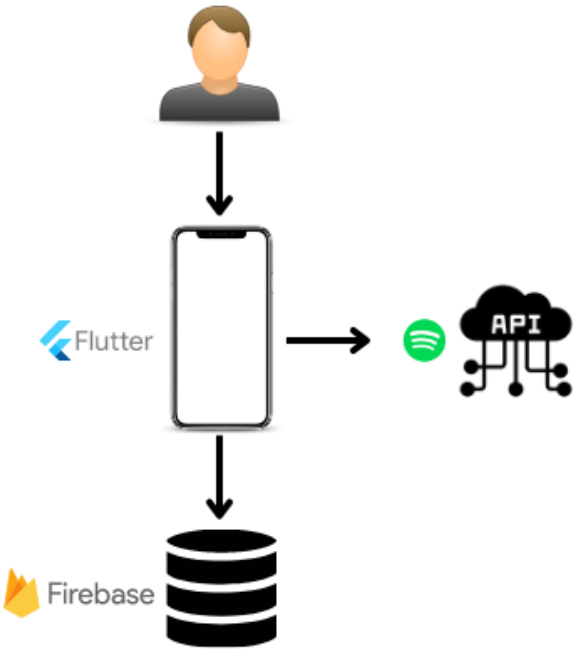


Figure 3 - DjParty architecture

3.1 Client-side Application

Flutter is a UI toolkit developed by Google, that lets developers create beautiful and fast UI's with relative ease. It's meant to be cross platform, which means that it enables developers to make production level Android and iOS apps with a single codebase. Differently from other cross-platform frameworks, Flutter

avoids performance problems caused by the need for a JavaScript bridge by using a compiled programming language, namely Dart. Dart is compiled “ahead of time” (AOT) into native code for multiple platforms. This allows Flutter to communicate with the platform without going through a JavaScript bridge that does a context switch. It also compiles to native code which in turn improves app startup times. In Flutter, it is all about Widgets, which are the elements that affect and control the view and interface to an app.

3.2 Server-Side Application

The server-side of the application is entirely hosted by Google Firebase, which is a Backend-as-a-Service (BaaS) that relies on the more complex and flexible Google Cloud Platform (GCP) cloud provider. The backend of the app is completely serverless. Serverless is a cloud-native development model that allows developers to build and run applications without having to manage servers. There are still servers in serverless, but they are abstracted away from app development. This gives a great advantage in terms of ease and speed of development, and perfectly suits our use-case, as we are more interested in these factors than squeezing all possible performance and customizing our backend code, which does not provide much value to the end user, especially during the earlier days of the app. The server-side functionalities used by the app can be divided into the database, storage, and cloud functions, which are all implemented through services provided by Firebase and can be accessed and modified through the built-in console or the cli tools.

3.2.1 Firestore Database

Cloud Firestore is a NoSQL document database that simplifies storing, syncing, and querying data for mobile and web apps at global scale. Its client libraries provide live synchronization and offline support, while its security features and integrations with the Firebase and Google Cloud platforms accelerate building truly serverless apps.

In our model we organized the data into two main collections: users and parties.

The users collection contains all users’ data, with a subcollection named “party”, where are listed all the parties created or joined by each user. The parties collection lists all the parties that are present in the application’s database.

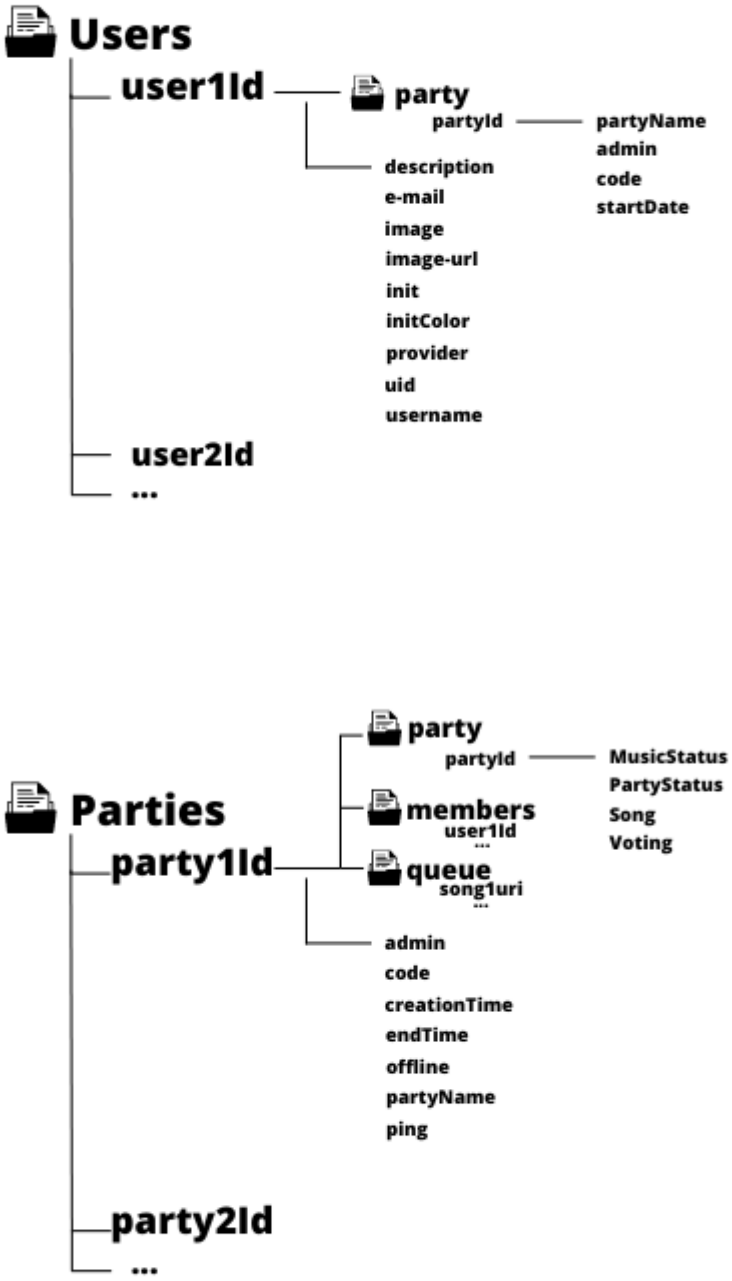


Figure 4 - Database structure

USER INTERFACE

DJParty is implemented in Flutter, which comes with the great benefit of being cross-platform. Flutter provides the material design package that provides developers with pre-built UI components.

Below we provide the mockups and screenshots for the mobile app application that will be used by our users.

4.1 Logo

For the logo design, we opted for an intuitive and minimalistic one. We inserted the Spotify logo in order to identify the strong relationship between our application and the streaming service. We then decided to insert some headphones around the Spotify logo for identify the djing function of the system.

Figure 5 - Logo

4.2 Look and Feel

4.2.1 Theme

The theme of the application is overall minimalistic as the logo, not redundant and with the aim to provide an enjoyable experience to the user, also on the graphical appearance. It is a modern design inspired by Spotify look and Feel.

The dominant colors are the ones corresponding to the Spotify Color Pallette, in order to be coherent with the scopes of the application. In this way a user

who employes the Spotify application regularly can find the application workflow and structure more intuitive.

4.2.2 Initial Sketches

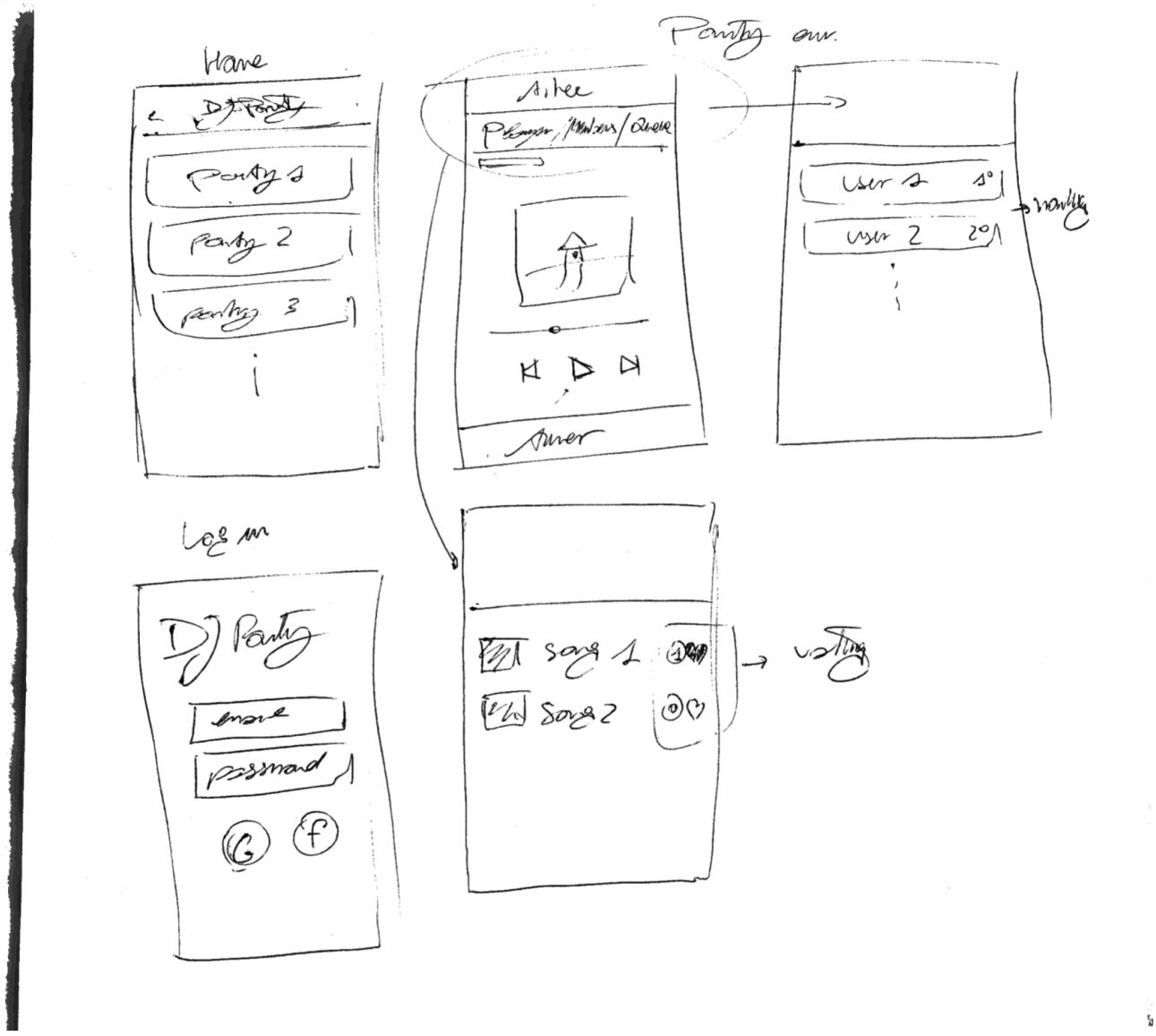


Figure 6 - Initial sketches

4.3 Pages

4.3.1 Authentication

The Login page is the first page that appears. The page is showing all the different sign in options, with an “don’t have an account? Sign up” button at the

bottom that leads to the SignUp page. After the two textFields in the Login page, a “Forgot Password?” button is displayed, that leads to the ResetPassword page, where a textField is displayed to type the e-mail address to which the system will send the e-mail for the restoring of the password. In the Sign Up page, the user can register to the application. Three textFields are displayed: e-mail, new password and confirm new password. Below the textFields a “Have already an account? Log in” button is displayed, which leads to the login page.

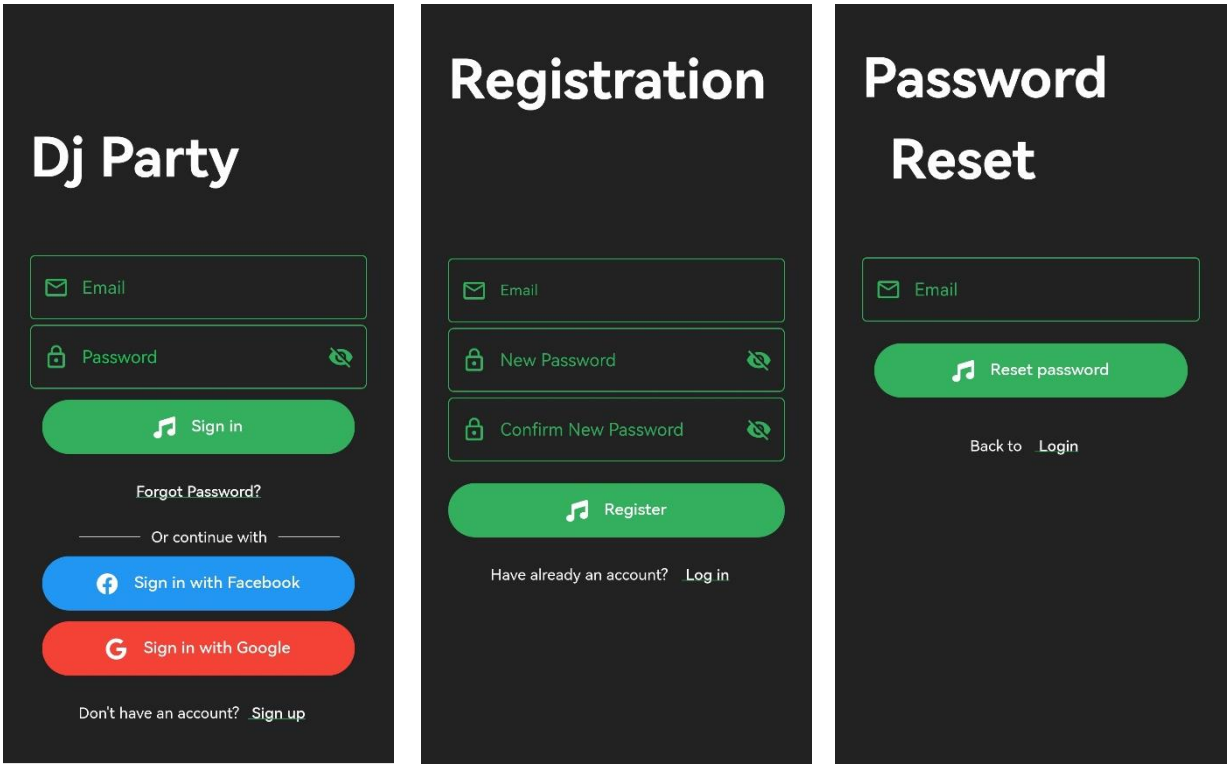


Figure 7 - Login Sign Up and Reset Password

4.3.2 Lobby

4.3.2.1 Home

The Home Page presents a scrollable list of expanded cards representing all the user’s parties, joined and created. When a card is closed, only the party name, the party date and the Guest/Admin icon are displayed. With the expanded card, the party code + 3 buttons are added to the layout:

- **Exit:** by clicking the button, the user will be deleted from the members’ list and the party’s card will be no longer displayed in the list.

- **Share:** this button allows the user to share the party infos, i.e. the partyCode and the Qr, through the main sharing platforms/e-mail/social networks.
- **Join:** this button allows the user to access to the party page and its functionalities.

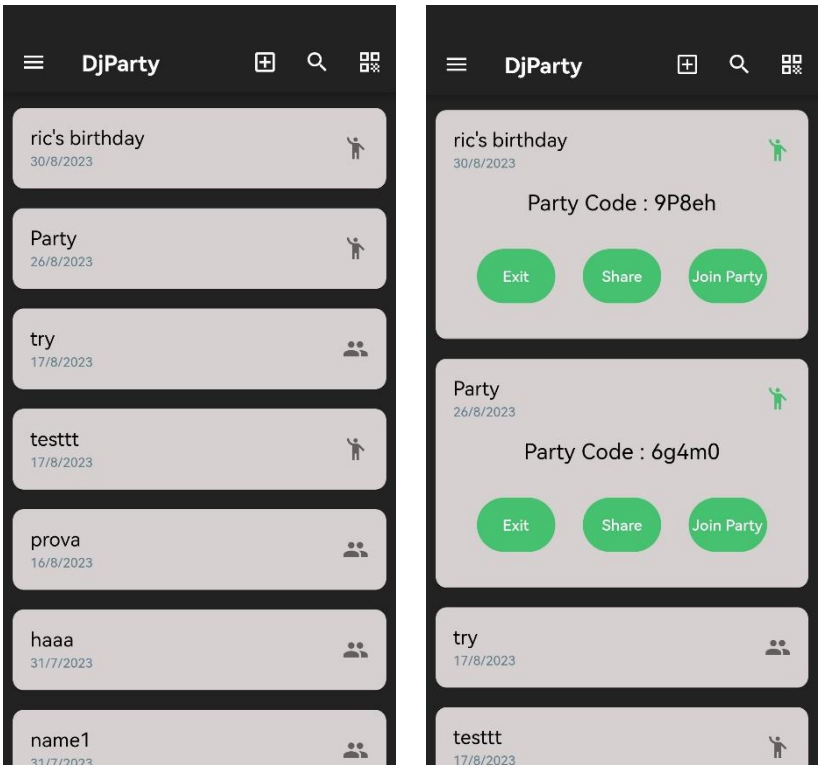


Figure 8 - Home Page with closed and expanded cards

4.3.2.2 Profile

The Profile Page can be accessed by the menu icon present in the Home page. The layout consists of the User's image and name on the top, followed by a sequence of some buttons:

- **Profile**: leads to the editing page of the user's profile.
- **Home**: leads to the home page.
- **Logout**: disconnect the user, who has to log in again if he wants to access the application again.

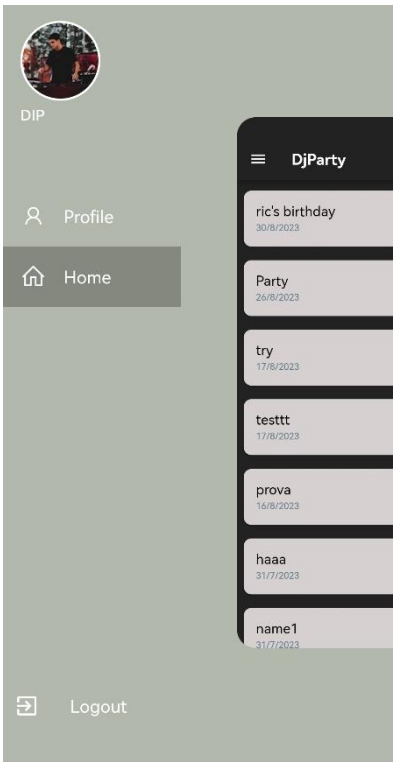


Figure 9 - Profile Page

4.3.2.3 Edit Profile

In this page, the user can edit the profile details as image, name and a bio. By clicking the “save” button, all the changes will be updated.

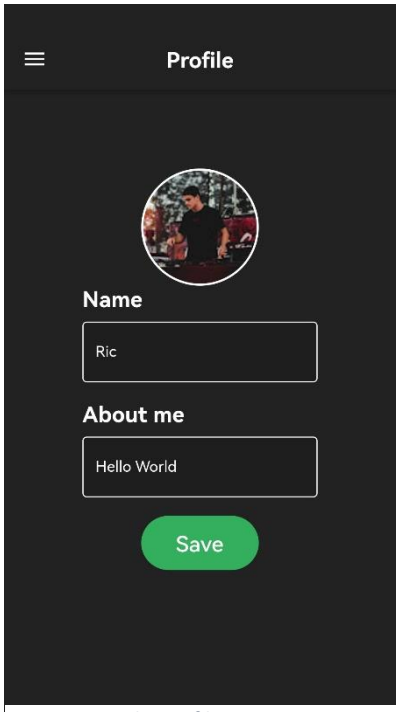


Figure 10 - Edit Profile Page

4.3.2.4 Create Party

In this page, the user can create a new party. A textField is displayed, where the user can write the party title, with a “confirm” button to confirm the creation of the new party environment.

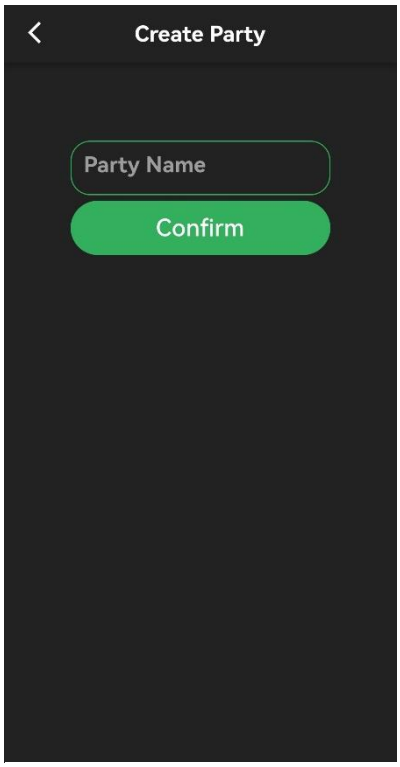


Figure 11 - Create Party Page

4.3.2.4 Join Party

From the AppBar of the Home page, the user can join a new party in two ways:

- By clicking the search icon, the InsertCode page is displayed. The layout is the same as the CreateParty page, with a textField, in which the user can insert the party code, and a “Join” button that adds the joined party in the user Home page’s list.
- By clicking the QrCode icon, a page with the QrScanner is displayed. If the user scan the Qr of an existing party, his Home page’s list is updated with the joined one.

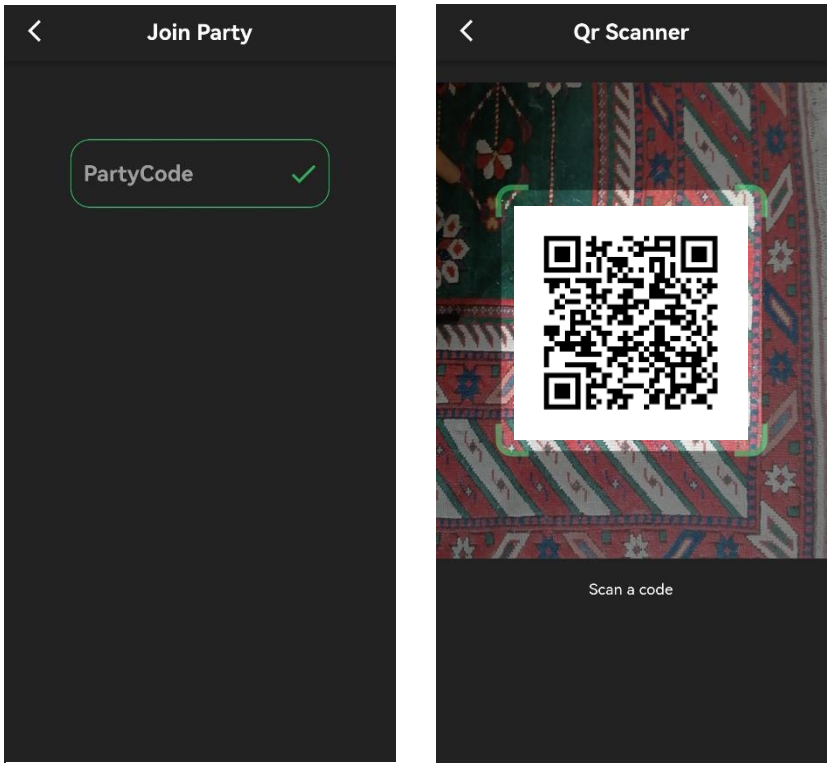


Figure 12 - Join Party Pages

4.3.3 Admin Party

In this section the admin's party environment will be explained. In the AppBar are present the title of the party in the center, the settings icon on the right, which allows the user to set the time intervals of voting and non-voting phases, and the back icon to go back to the home page. Once the party is started, the admin is not allowed to exit the party, so if he clicks the back arrow, a disclaimer will be displayed for letting the admin decide if he wants to end the party or if he wants to come back to the party env. Below the AppBar, a TabBar View is present to let the users navigate through the pages: Ranking, Player and Queue/Search.

In the bottom, below the Tab section, two different layouts are displayed depending on the partyStatus:

- Party Not Started: only a "Party status: not started" text is displayed.

- Party Started: a linear bar identifying the progress of the current song is displayed, following by a clock timer that explain the remaining time for the switching between voting/non voting phases.
- Party Ended: only a “Party status: ended” text is displayed.

4.3.3.1 Ranking

The Ranking page can be reached by putting the tab cursor onto the “Users” section. The page displays a scrolling list of all the members of the party, ordered by the scoring rule explained in **2.1.6**

Below the list, the “Start the Party” button is displayed. By clicking that button, the admin can start the party.

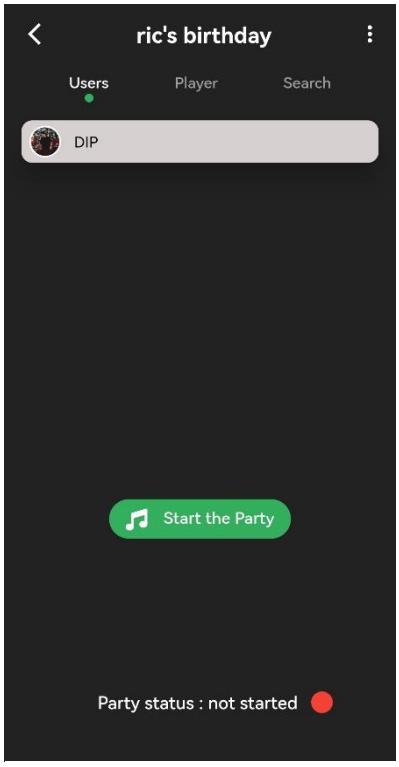


Figure 13 - Admin Ranking Page

4.3.3.2 Player

The Player page can be reached by putting the tab cursor onto the “Player” section.

The page’s layout is different depending on the party status:

- Party not started: the logo app is displayed with a “Songs in the Queue will be reproduces when the party will start” text below.
- Party started: a player layout is displayed with the cover image of the playing song on top, followed by skip backward, play/pause and skip forward buttons.
- Party Ended: in the player page are displayed the party results, i.e. the duration of the party, the user that held the first place in the ranking and the most voted song.

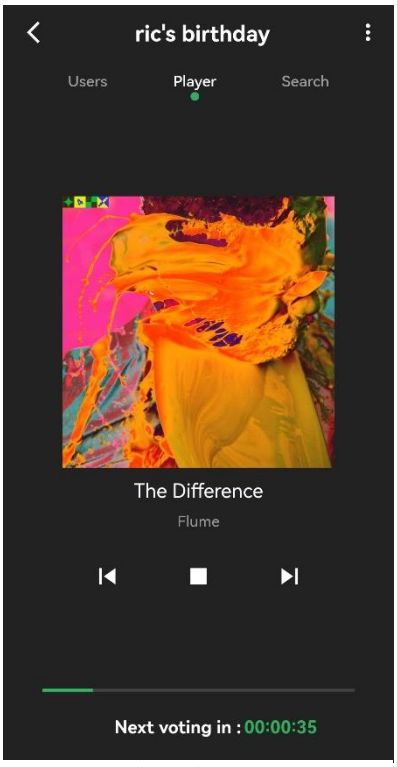


Figure 14 - Admin Player Page

4.3.3.3 Queue/Search

The Queue/Search page can be reached by putting the tab cursor onto the “Search” section.

The page’s layout consists of a textField with a search icon, followed by the list of songs that are present into the party queue. Once the textField is tapped, the user can type to search for a song to add and the queue list is substituted by a new list of songs that corresponds to the typing results of the textField.

During the voting phase, near every song in the queue a heart icon appears, and the user can tap and vote for the song. The queue will be updated accordingly to the votes.

Once the party is ended, an elevated button is displayed. By clicking that, a Spotify playlist will be automatically created into the user's Spotify account. Below the button, all the songs that are reproduced in the party before the end are displayed in the right order.

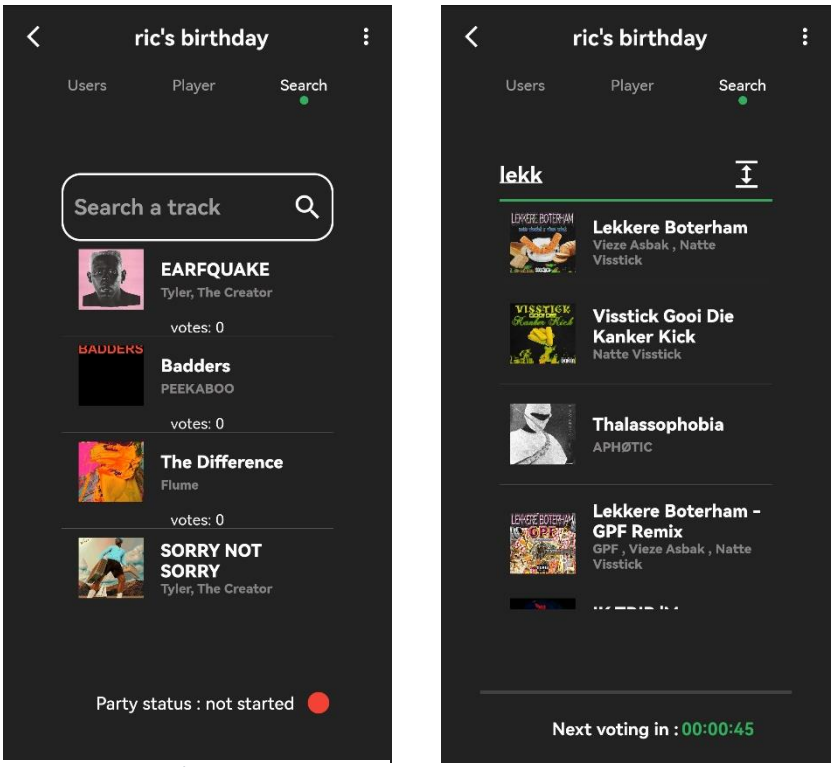


Figure 15 - Queue/Search Page

4.3.4 Guest Party

In this section the guest's party environment will be explained. In the AppBar are present the title of the party in the center, the menu buttons icon on the right, which allows the user to share the party, and the back icon to go back to the home page. Below the AppBar, a TabBar View is present to let the users navigate through the pages: Ranking, Player and Queue/Search.

In the bottom, below the Tab section, three different layouts are displayed depending on the party status:

- Party Not Started: only a "Party status: not started" text is displayed.

- Party Started: a linear bar identifying the progress of the current song is displayed, following by a clock timer that explain the remaining time for the switching between voting/non voting phases.
- Party Ended: only a “Party status: ended” text is displayed.

4.3.4.1 Ranking

The Ranking page can be reached by putting the tab cursor onto the “Users” section. The page displays a scrolling list of all the members of the party, ordered by the scoring rule explained in **2.1.6**

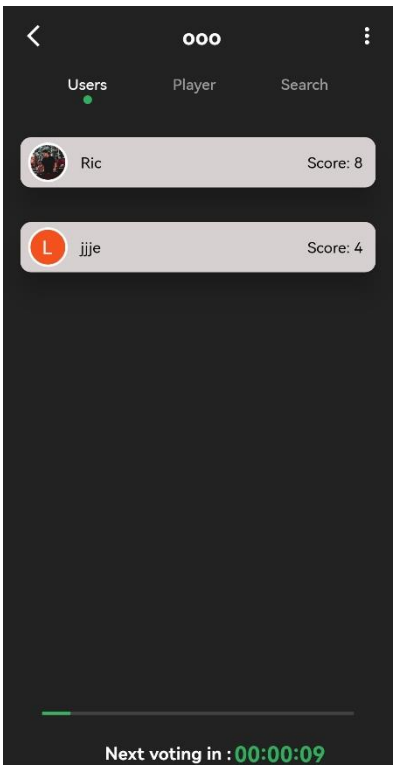


Figure 16 - Guest Ranking Page

4.3.4.2 Player

The Player page can be reached by putting the tab cursor onto the “Player” section.

The page’s layout is different depending on the party status:

- Party not started: the logo app is displayed with a “Songs in the Queue will be reproduced when the party will start” text below.
- Party started: a player layout is displayed with the cover image of the playing song.

- Party Ended: in the player page are displayed the party results, i.e. the duration of the party, the user that held the first place in the ranking and the most voted song.

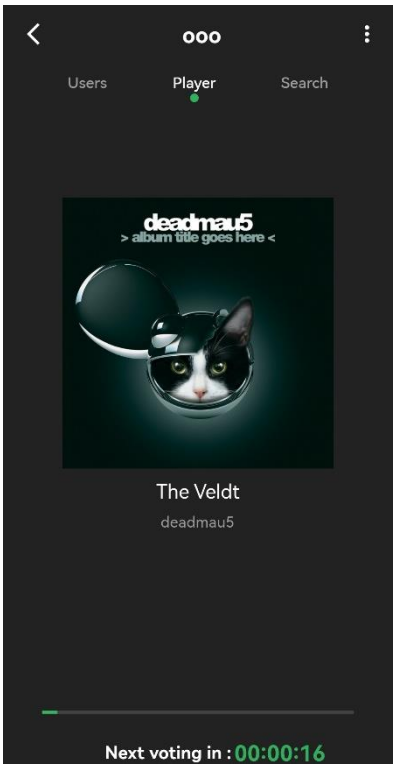


Figure 17 - Guest Player Page

4.3.4.3 Queue/Search

The Queue/Search page can be reached by putting the tab cursor onto the "Search" section.

The page's layout consists of a textField with a search icon, followed by the list of songs that are present into the party queue. Once the textField is tapped, the user can type to search for a song to add and the queue list is substituted by a new list of songs that corresponds to the typing results of the textField.

During the voting phase, near every song in the queue a heart icon appears, and the user can tap and vote for the song. The queue will be updated accordingly to the votes.

Once the party is ended, an elevated button is displayed. By clicking that, a Spotify playlist will be automatically created into the user's Spotify account. Below the button, all the songs that are reproduced in the party before the end are displayed in the right order.

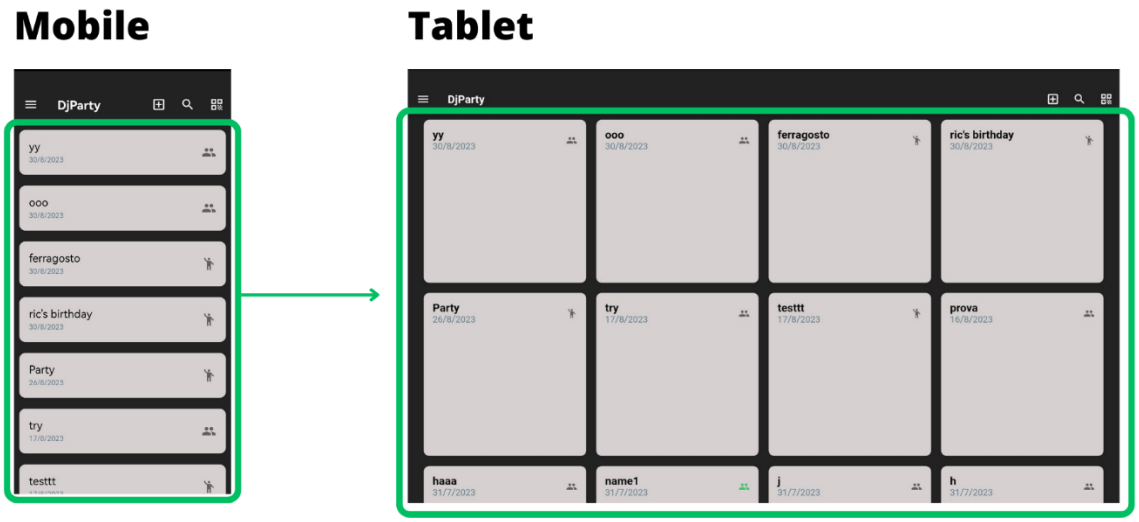
The admin and gest queue/search pages have the same structure, shown in Figure 13.

4.3.5 Tablet Layout

For a better look and feel and user experience, the application has two different layouts, depending on the user's device: mobile or tablet.

The main differences are:

- The ListViews in the Home Page and the Ranking page are substituted by GridViews in order to be more suitable for a large tablet's screen.
- In the party environment, the tabBar is vertically placed in the left section of the screen and the words that identifies the different sections of the tabView are substituted by icons.



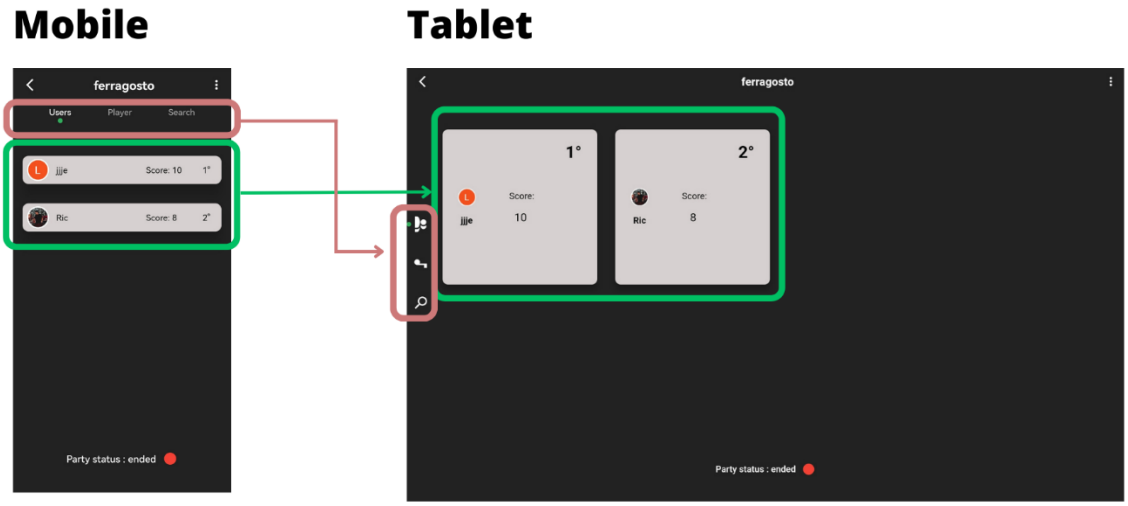


Figure 18 - Mobile and tablet Layouts

EXTERNAL LIBRARIES

5.1 APIs

The following are the external APIs that we used to complete our backend by getting additional data, in some cases essential to provide the main functionalities of the app:

- **Spotify:** *DJParty* relies on the Spotify API and related SDK for all the functions that deals with the music reproduction and the songs' information.
- **Facebook Log-In:** Lets users log in with Facebook, from which we retrieve their relevant data.
- **Google Log-In:** Lets users log in with Google.

5.1 Packages

The application takes advantage of a list of packages offered by both the flutter team and third-party developers. In the following section we list some of the most relevant plugins that allowed us to save time and code while delivering a lot of functionalities and a strong user experience:

- **Firebase plugins**: used to access the various services, provided by firebase, used in the application. We included firebase_core, firebase_auth, and cloud_firestore.
- **Authentication**: flutter_facebook_auth and google_sign_in.
- **Spotify SDK**: package used for an easier interaction with the Spotify SDK.
- **Animation**: packages used for animation purposes like for eample the linear timer that represents the progress of the currently playing song.
- **Testing**: useful mocking external libraries and internal entities like mockito, firebase_auth_mocks, google_sign_in_mocks and fake_firebase_firestore.

IMPLEMENTATION, INTEGRATION AND TESTING

6.1 Implementation

DjParty is a complex system composed of different layers, modules and submodules that interact and communicate among each other. Before starting with the implementation, the VSC tool must be correctly set up. Then, the database must be designed and implemented. After the database the serverless functions can be created, thus completing the backend. Once these components are created, it's time to start developing the mobile application: first, the UI is implemented using mock data to understand the flow of the user experience inside the application. Then the blocs that we employ to retrieve data and talk with the server should be created.

6.2 Integration

The integration of the system will be based on its architectural structure. The components that need to be integrated are:

- Firebase Authentication: It has the objective of performing the authentication process, including the management of authentication tokens and the encryption of users' information.
- Cloud Firestore: Its purpose is to act as a middleware between the clients and the database.

- Mobile Application: Its purpose is to act as a front-end application to let users access the service.
- External APIs: They are needed to provide additional data and features to the mobile application.

6.3 Testing

To implement the system, we decided to test the application mainly through debugging on both emulators and real devices, but we decided also to perform some automated tests on the application's main components.

6.3.1 Unit tests

In the *test/unit_test/* folder are present two dart pages in which we implemented some unit tests for the interaction of the system with the Spotify API and with Firebase.

6.3.2 Widget tests

In the *test/widget_test/* folder are present dome dart pages that performs widget testing.

In widget testing, the testing library wraps the widget that is under analysis in order to provide a BuildContext to it. Then, the test environment performs a sequence of interactions (taps, scrolls, etc.) and scans the screen in order to verify that the widget's UI appears and works as expected.

6.3.3 Use cases

In this subsection are listed the main activities performed to debug the application.

REGISTER

ACTOR	USER
INPUT CONDITION	USER HAS DJPARTY ON HIS DEVICE AND NOT REGISTERED
EVENT FLOW	1. OPEN THE APP 2. CLICK ON SIGN UP BUTTON 3. INSERT E-MAIL 4. INSERT PASSWORD 5. CONFIRM PASSWORD
OUTPUT CONDITION	THE SYSTEM LOGS THE USER INTO THE SERVICE
EXCEPTION	1. E-MAIL PATTERN NOT RESPECTED 2. PASSWORD TOO SHORT (MIN 8 CHARACTERS) 3. PASSWORD CONFIRMATION NOT THE SAME AS THE INITIAL PASSWORD 4. THE USER ALREADY EXISTS

LOGIN

ACTOR	USER
INPUT CONDITION	USER HAS DJPARTY ON HIS DEVICE AND IT IS REGISTERED BUT NOT AUTHENTICATED
EVENT FLOW	1. OPEN THE APP 2. INSERT E-MAIL 3. INSERT PASSWORD 4. CLICK ON LOGIN BUTTON
OUTPUT CONDITION	THE SYSTEM LOGS THE USER INTO THE SERVICE
EXCEPTION	1. E-MAIL PATTERN NOT RESPECTED 2. WRONG PASSWORD 3. THE USER DOESN'T EXISTS

FORGOT PASSWORD

ACTOR	USER
INPUT CONDITION	USER HAS DJPARTY ON HIS DEVICE AND IT IS REGISTERED BUT DOESN'T REMEMBER THE PASSWORD
EVENT FLOW	1.OPEN THE APP 2.CLICK ON FORGOT PASSWORD BUTTON 3.INSERT E-MAIL ON THE FORGOT PASSWORD PAGE 4.THE USER RECEIVE THE E-MAIL 5.THE USER RESTORES THE PASSWORD
OUTPUT CONDITION	THE SYSTEM SENDS THE E-MAIL TO THE USER
EXCEPTION	1.E-MAIL PATTERN NOT RESPECTED 2.THE USER DOESN'T EXISTS

GOOGLE LOGIN

ACTOR	USER
INPUT CONDITION	USER HAS DJPARTY ON HIS DEVICE AND A GOOGLE ACCOUNT
EVENT FLOW	1.OPEN THE APP 2.CLICK ON LOGIN WITH GOOGLE BUTTON 3.THE USER ALLOW THE USE OF GOOGLE
OUTPUT CONDITION	THE SYSTEM LOGS THE USER INTO THE SERVICE
EXCEPTION	1.GOOGLE DISABLED 2.THE USER EJECT TO ALLOW THE USE OF GOOGLE

FACEBOOK LOGIN	
ACTOR	USER
INPUT CONDITION	USER HAS DJPARTY ON HIS DEVICE AND A FACEBOOK ACCOUNT
EVENT FLOW	1.OPEN THE APP 2.CLICK ON LOGIN WITH FACEBOOK BUTTON 3.THE USER ALLOW THE USE OF FACEBOOK
OUTPUT CONDITION	THE SYSTEM LOGS THE USER INTO THE SERVICE
EXCEPTION	1.FACEBOOK DISABLED 2.THE USER EJECT TO ALLOW THE USE OF FACEBOOK

JOIN PARTY WITH CODE	
ACTOR	USER
INPUT CONDITION	USER IS LOGGED, HIS E-MAIL IS VERIFIED AND HE HAS THE HOME DISPLAYED
EVENT FLOW	1.CLICK ON THE “SEARCH” ICON IN THE APPBAR 2.TYPE THE PARTY CODE 3.CLICK ON “DONE” ICON
OUTPUT CONDITION	THE SYSTEM ADDS THE USER TO THE PARTY ENVIRONMENT AND ADDS THE PARTY TO THE HOME PAGE
EXCEPTION	1.THE CODE IS WRONG AND DOESN’T CORRESPOND TO ANY PARTY

JOIN PARTY WITH QR

ACTOR	USER
INPUT CONDITION	USER IS LOGGED, HIS E-MAIL IS VERIFIED AND HE HAS THE HOME DISPLAYED
EVENT FLOW	1. CLICK ON THE "QR" ICON IN THE APPBAR 2. LET THE SYSTEM ACCESS TO THE CAMERA 3. SCAN THE QR
OUTPUT CONDITION	THE SYSTEM ADDS THE USER TO THE PARTY ENVIRONMENT AND ADDS THE PARTY TO THE HOME PAGE
EXCEPTION	1. THE QR-CODE IS WRONG AND DOESN'T CORRESPOND TO ANY PARTY 2. THE USER DOESN'T ALLOW THE CAMERA

CREATE PARTY

ACTOR	USER
INPUT CONDITION	USER IS LOGGED, HIS E-MAIL IS VERIFIED AND HE HAS THE HOME DISPLAYED
EVENT FLOW	1. CLICK ON THE "PLUS" ICON IN THE APPBAR 2. TYPE THE PARTY TITLE 3. CLICK ON CONFIRM BUTTON
OUTPUT CONDITION	THE SYSTEM CREATE THE NEW PARTY ENVIRONMENT AND ADDS IT TO THE HOME PAGE
EXCEPTION	1. THE USER DID NOT CHOOSE A TITLE

ADD SONG TO QUEUE	
ACTOR	USER
INPUT CONDITION	USER IS LOGGED, HIS E-MAIL IS VERIFIED AND HE IS INTO A PARTY ENVIRONMENT
EVENT FLOW	1.GOES TO THE "QUEUE" SECTION OF THE TAB VIEW 2.TYPE THE NAME OF A SONG 3.CLICK ON THE SONG 4.ADD THE SONG TO THE QUEUE WITH THE DROPDOWN MENU
OUTPUT CONDITION	THE SYSTEM ADDS THE SONG TO THE PARTY QUEUE
EXCEPTION	

VOTE FOR A SONG	
ACTOR	USER
INPUT CONDITION	USER IS LOGGED, HIS E-MAIL IS VERIFIED AND HE IS INTO A PARTY ENVIRONMENT. THE PARTY IS STARTED AND A VOTING PHASE IS PRESENT
EVENT FLOW	1.GOES TO THE "QUEUE" SECTION OF THE TAB VIEW 2.VOTE FOR HIS FAVOURITE SONG IN THE QUEUE
OUTPUT CONDITION	THE SYSTEM UPDATES THE QUEUE ACCORDING TO THE VOTING
EXCEPTION	

IMPROVEMENTS AND ADDITIONAL FEATURES

Following we report some additional ideas that might be implemented in the future in the *DJParty* app:

- Extend the possibility for the music player function like dragging the progress bar to go to a specific point of the playing song.
- Sharing the ranking position on social media.
- Extend the possibility to use the application not only for parties but for other types of entertainment like twitch streams.
- Possibility to search for a user and send invitation directly into the application's environment.

REFERENCES

This section includes the documents and the main sources referenced during the writing of this document and useful for the development of the application:

- https://flutter.dev/docs
- https://developer.android.com/
- https://pub.dev/
- https://firebase.google.com/docs
- https://firebase.flutter.dev/docs
- https://developer.spotify.com/
- https://developers.facebook.com/docs/facebook-login/