DIMA Project report: MBox

Simone Abelli, Stefano Azzone

May 6, 2022

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide more technical and detailed information about the mobile application developed. The Design Document is a guide for the programmer that will manage the future development of the codebase for the application in all its functions. The document will explain and motivate all the architectural choices by providing a description of the components and their interaction. We will also enforce the quality of the product through a set of design characteristics. Finally we describe the implementation, integration and test planning. The topics touched by this document are:

- high level architecture
- main components, screens and widgets
- runtime behavior
- design patterns
- more details on user interface
- requirements and their mapping on the architecture
- implementation and test planning

1.2 Scope

MBox is an application for mobile devices that allows users to access all the music present on their devices without having to worry about manually managing the metadata. The target user has basic knowledge of the functions of the mobile device (e.g. is able to interact with the filesystem). Since the application is focused on users who want to listen to music, which is usually done with a portable device, we will focus smartphones maily (the application also works on tablets). For the development of this applications we have decided to use flutter since it's a framework thought for multiplatform deployment.

1.3 Glossary

Queue: a queue is a data structure containing a list of tracks which are to be played according to a FIFO policy.

Metadata: metadata are information of a track regarding the track itself (e.g. track title, artist, album cover . . .)

2 Description and requirements

2.1 Product description

MBox has all the functionalities of a music player:

- Detect music present in the Music folder
- Allow management of playlists
- Organize music by artist, album and playlist
- Allow to arrange tracks in a playback queue
- Play tracks (in random order if desired)
- Visualize metadata related to tracks

In addition to this, MBox allows a more advanced and automated metadata management, and the access to other music information:

- Automatically set missing metadata (Title, Artist, Album, Cover, Lyrics, Track number, Artist image, ...)
- Manually edit metadata
- Visualize information about artists like a brief description, albums and other songs
- Search on the internet for other songs and listen to them

2.2 Assumptions

- When downloading metadata or looking for other music, internet connection is available
- The user is able to move their music to the Music folder
- The tracks are present on Spotify to download metadata
- When the user looks for a track not present on their device, it must be present on YouTube
- The tracks are in mp3 format (otherwise metadata management is harder)
- The access to the filesystem is granted
- The device on which the application is used has some means to play music

2.3 Functional requirements

1. The application can access the filesystem and fetch songs from the music folder
2. The application can play the selected tracks
3. The application allows the user to add or remove tracks from the playback queue
4. The application allows the user to pause and resume playback
5. The application allows the user to skip tracks in the queue
6. The application should save the queue state to resume playback if the application is closed
7. The application allows the user to add or remove playlists
8. The application allows the user to add or remove tracks from playlists
9. The application organizes the tracks by artist and album
10. The application allows the user to view lyrics of currently playing track
11. The application allows the user to edit metadata of the tracks
12. The application automatically adds missing metadata using an external service (in our case Spotify); already present metadata is kept
13. The application can show information about the artists of the tracks present on the device (such as all their albums and tracks)
14. The application allows the user to search for tracks not present on their device, and play them through an external service (in our case YouTube)
15. It should be possible to use the application without an internet connection, obviously with limited functionalities (no external song search, metadata editing ...)

2.4 Non functional requirements

1. The application should feel snappy and responsive
2. The application layout should be intuitive to use
3. The application should be reliable

3 Architectural Design

3.1 Architectural Style and Patterns

The application is logically subdivided in a frontend and a backend. The frontend presents the information to the user and allows them to interact with the backend. The backend contains and manages all the data and interfaces with the external services.

3.1.1 Frontend

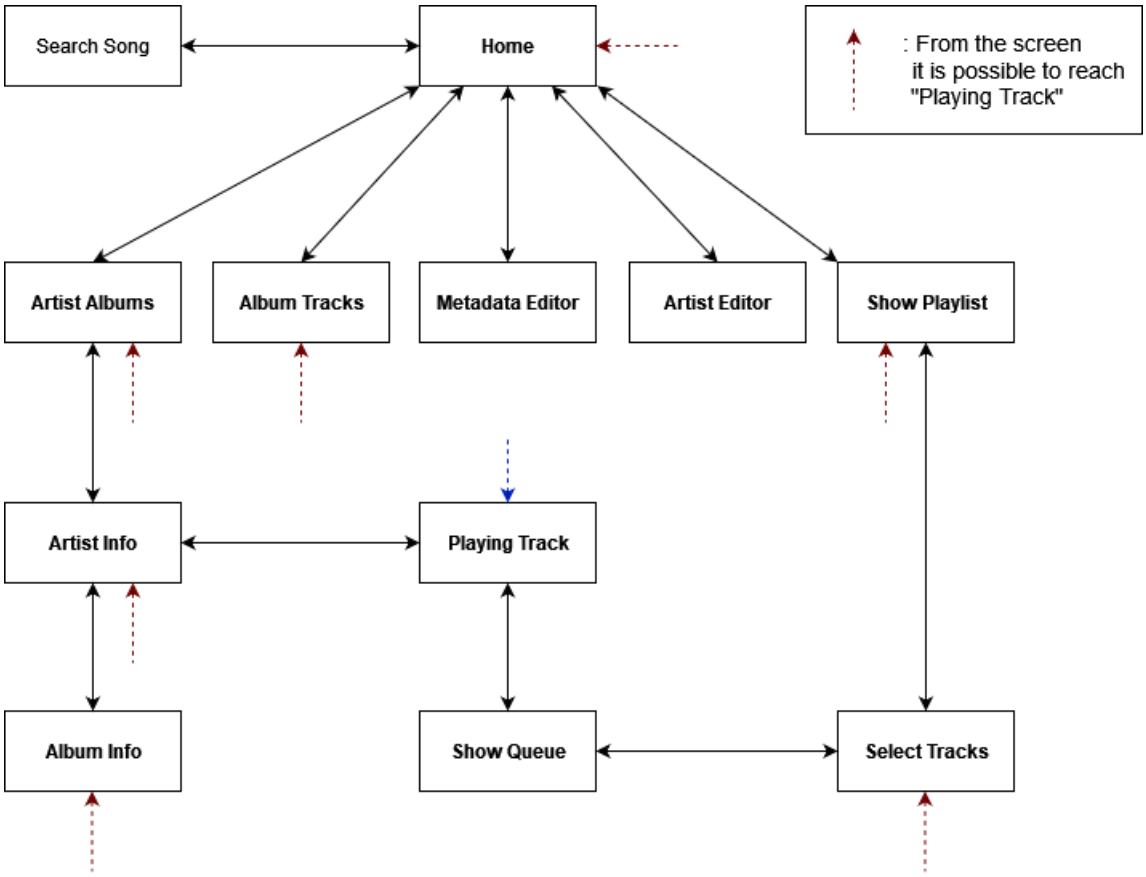

Figure 1: Frontend system architecture

The frontend is composed by the screens of the application. Once opened MBox shows the **Home** screen. The Home screen is composed by 4 tabs:

- *Tracks*: a list of the tracks in alphabetical order; by selecting one we switch to the **PlayingTrack** screen, and start playing that track.

- *Albums*: a list of the albums in alphabetical order; by selecting one we switch to the **AlbumTracks** screen, where we can select a track to play.
- *Artists*: a list of the artists in alphabetical order; by selecting one we switch to the **AlbumArtists** screen, where we can select an album to show.
- *Playlists*: a list of the playlists in alphabetical order; by selecting one we switch to the **ShowPlaylist** screen, from which we can start the playback; in the Playlists tab it is also possible to add a new playlist or remove an existing one.

By keeping a track pressed it is possible to open a drop down menu with a few entries:

- *Edit metadata*: switch to the **MetadataEditor** screen to modify the metadata of the track
- *Add to queue*
- *Add to playlist*

By keeping an artist pressed it is possible to switch to the **ArtistEditor** screen in order to change its image.

By touching the magnifying glass in the top right corner of the application, it is possible to reach the **SearchSong** screen that allows the user to look for songs that are not present on their device, and play them through an external service (in our case YouTube).

From the **PlayingTrack** screen it is possible to:

- *pause* and *resume* playback
- *skip* forward and backward in the queue
- *change playback position* of the currently playing track
- view the *album cover* and the *lyrics* of the currently playing song
- check the *artist information* by switching to the **ArtistInfo** screen: this screen shows a brief description of the artist (from Wikipedia) and a list of their albums; by selecting one of these, the user is brought to the **AlbumInfo** screen, from which they can view all the tracks and play them using YouTube
- *show the queue* by switching to the **ShowQueue** screen.

The **ShowQueue** screen shows the current track queue; the tracks are ordered according to their position in the queue, where the currently playing track has index 0, the past tracks have negative index, and the future tracks have positive index. From here it is possible to add new tracks to the queue by means of

the **SelectTracks** screen. Almost every screen has a **PlayBar** that allows to pause and resume playback, and shows currently playing track info and artwork. By pressing it, the user is brought back to the **PlayingTrack** screen.