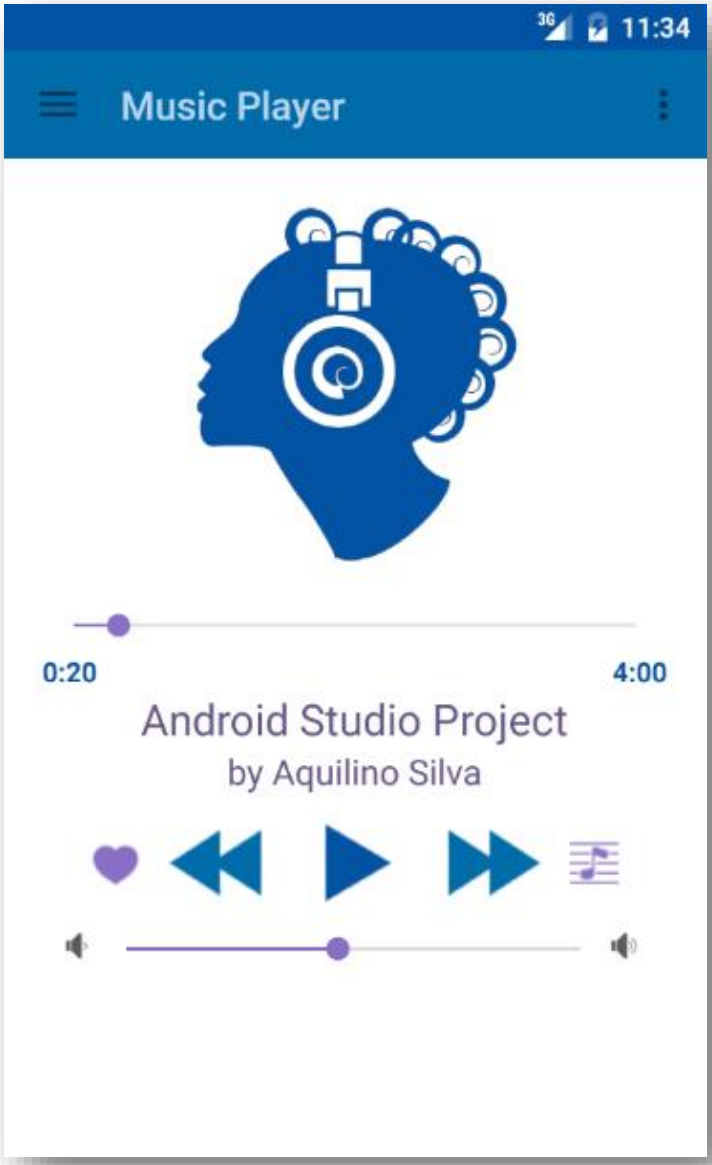

Music Player

ANDROID STUDIO PROJECT

Aquilino Silva | Software Structures for User Interface | May 18, 2016

Introduction

Regarding the course of Software Structures for User Interface, it was given to us the chance to explore the Android Studio by creating a project. The project was of our choice, given the options: either a Drone App – where an Android Interface would have to be created to control a Parrot AR Drone 2.0 – or a Media Player App. Despite the Drone App being an intriguing project, I decided to go forward with the Media Player App.

At the beginning, my solution was to create a Media Player App where would have three different activities in the same application, but then I decided to focused on just one activity and improve the best way I could.

How does the Interface works?

The interface of the Media Player Application was designed to be the most minimalistic and intuitive, without many effects and options, like the KISS design principle.

The application starts by displaying the Music Activity, showing the name of the song and the author – which consists of two *TextViews* -, the media controls and two other options (the "like" button and the playlist button) – which consists of five *ImageButton* -, two *Seekbar* – one for the song progress, and the other to control the volume -, and finally the album image.

To show the Application running, a video was recorded showing the Application at the Emulator (running on Android Nexus 5)

Video Link: https://vimeo.com/166991548

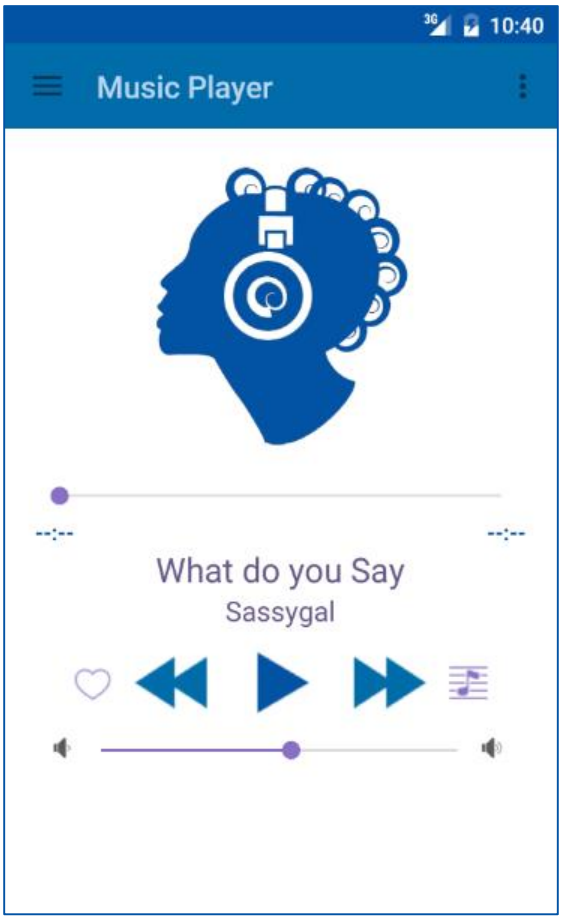

Figure 1: Music Activity, when the application starts

Play/Pause Button

When the Play *ImageButton* is clicked, it prepares the music to play and subsequently changes to the Pause *ImageButton*. When the music is done preparing, it starts playing and the progress bar (the *Seekbar* beneath the album image) updates accordingly the current and total length of the song, as also the *TextView* of both current and total time.

Figure 2: Music Activity, when the Play *ImageButton* is clicked

Like Button

The Like *ImageButton* – which can be found to the left of the previous *ImageButton* - was made so that the user could select the songs that he most loves. By selecting it, the *ImageButton* changes to other icon where gives feedback to the user that the music selected is now a "favourite" one.

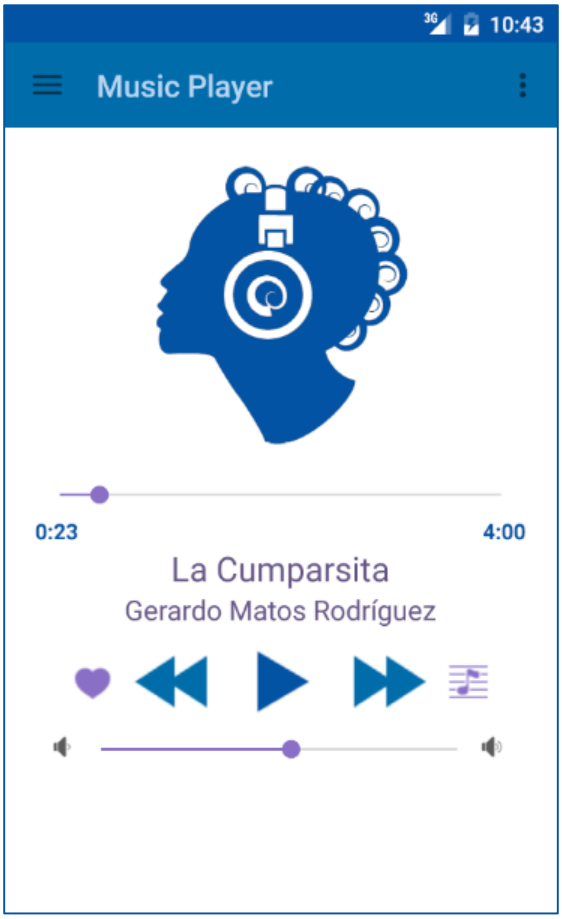

Figure 3: Music Activity with the Like *ImageButton* selected, providing feedback to the user of its action

Next/Previous Button

In a way to control the Media Player, more precisely the song the user wants to hear, it was added two *ImageButton* – the Previous and Next – where the design was based on the iOS 9 Music Player.

Case the user clicks on the Next button, it goes to the song that comes after the current one. If the Media Player is found playing when the user clicks on the Next button, the Music Player selects the next song and starts playing automatically, otherwise it only updates the *TextView* of both song and author name, and reset the progress.

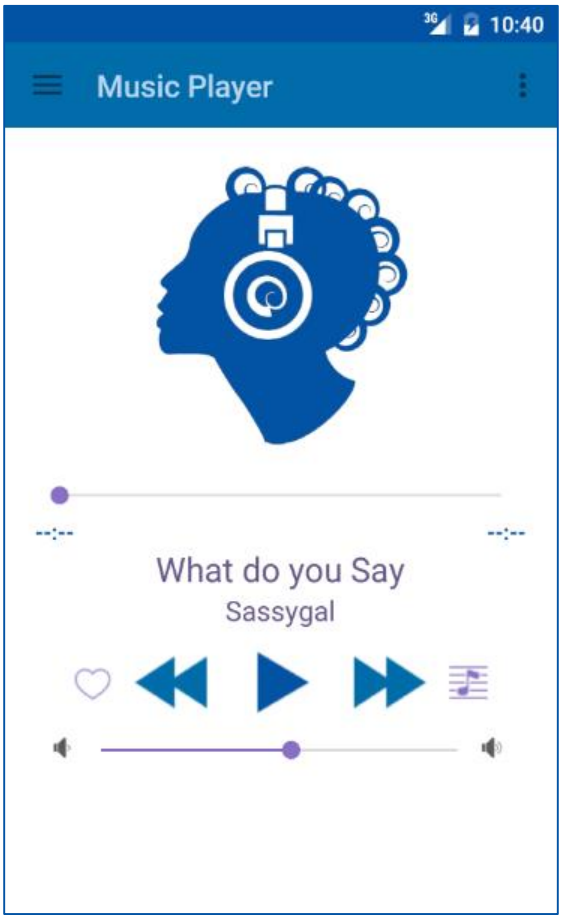

Figure 4: Music Activity showing updated *TextView* and Progress Bar, when Next button was clicked

Proximity Sensor

The Proximity Sensor was used on the Application as a failsafe for the user, in case the Android Device would be brought close to the ear. As the Android Device is brought near the ear, the Application automatically reduces the song volume to a predefined value. When the user decides to put away the Android Device from the ear, the Application automatically sets the volume back to its previous state.

Since it is not possible to replicate the use of sensors through the Emulator, the Application was recorded running into the Android Nexus S, showing off the Application itself and the use of sensors and its responsiveness.

Video Link: https://vimeo.com/166995995

Figure 5: Using the proximity sensor to change the Volume

PlayList Button

The PlayList *ImageButton* can be found next to the Next *ImageButton*, and its goal was to show the songs that can be found at the Application. Unfortunately, it was not yet implemented. Therefore, it ends up showing a *Toast* with the information "Feature not available".

Figure 6: PlayList Button was clicked, and showed a Toast

Media Player Drawer

By pressing the icon found at the top left corner (or simply by Right-Swiping), it opens a Drawer with different items. Each item takes to a new Activity (Music, Video or Radio Activity). To return to the Main Activity (Music Activity), the Back button simply needs to be pressed (may be physical or digital, depending on the Android Device).

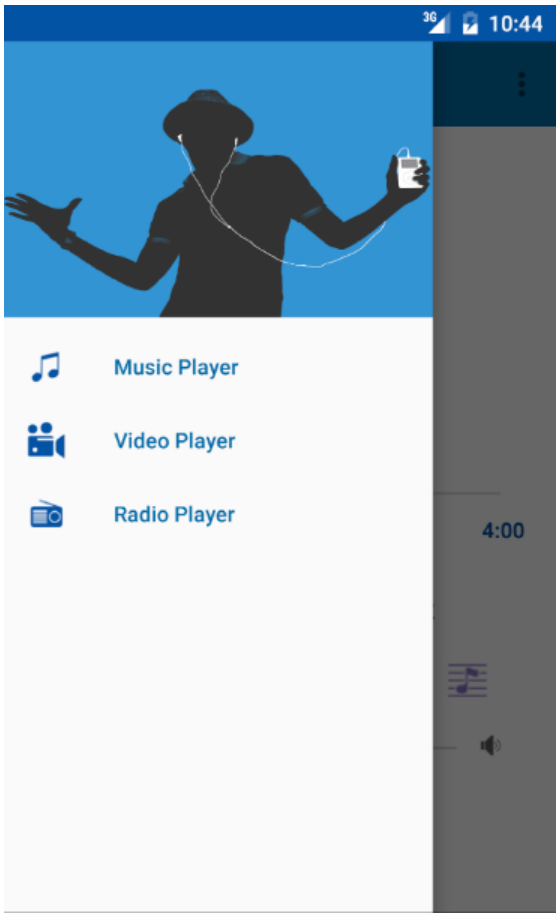

Figure 7: Drawer with different Items

Choosing the Color Palette

So that the colors could be wisely chosen, it was used the Coolors website, that generates a palette of colours. The main color of the Application is the blue color, while some features can be found in purple.

Figure 8: Coolors Palette