# 3 Design

The third chapter contains the description of the design I created for my application. First of all, I will write about the design strategy I decided to choose for this project. Secondly, I will explain all of the design architectures and design patterns I intended to implement in my application, along with the motives for my decisions. Thirdly, the UML diagrams of my application's design will also be presented, along with their explanations. Moreover, the chapter also contains the most important changes I had to make in the design part of the project since the progress report. Finally, the design of one of the most important aspects of my application – the user interface – will also be described so as to give the reader of this document an idea as to what my application was intended to look like.

## 3.1 Design Strategy

Similarly to any other full-scale software project, it is important to specify who its target will be and what would they expect of the application. As I chose to write a music player, the answer to the first part of the question seems relatively simple: its user would potentially be anybody who wishes to listen to the music in FLAC format. It was much more difficult to identify what would the users expect of my application.

Firstly, I believe they anticipate, since it is foremost a FLAC music player, that it would play their FLAC tracks flawlessly, which is quite understandable. Secondly, they would expect the user interface of the application to be so intuitive that they would have no problem with trying to get used to the player. Stating what a template of such a user interface looks like was, in my opinion, quite simple. Comparing different kinds of music players, such as the aforementioned andLess or Meridian Media Player Revolute for Android, or WinAmp for Windows, or even car music players allowed me to extract the feature common for each of them: the buttons used to handle playing of a song.

## 3.2 Overall Architecture

Choosing a design architecture for my application was one of the most difficult decisions I encountered while working on this project. The first idea which came to my mind was to simply create my own architecture, used only for this program. I felt, however, that this would entail some unwanted problems. First of all, I was worried my coding would become very chaotic and could thus easily lead to creating spaghetti code. Moreover, without a clear idea as to what I would like my architecture to look like at the very beginning of my project, it would change in ways I did not anticipate, which would make my code unnecessarily complex even more. What is more, such architecture would then be very difficult to explain to others who could possibly want to understand my application and how it works. It would also be more difficult for me to actually come back to my program and pick up coding where I had left it. Thus, I had to consider another option for my application's architecture.
I then immediately thought of the MVC architecture – Model-View-Controller. It is one of the most widely used architectural patterns, and its main feature and advantage is the fact that it divides all the code in a project into 3 parts: the Model, which in this case will contain all the data models, which represent the songs; the View, which deals with the graphical user interface of my application; the Controller which lets the other two communicate and handles all the events in the application.