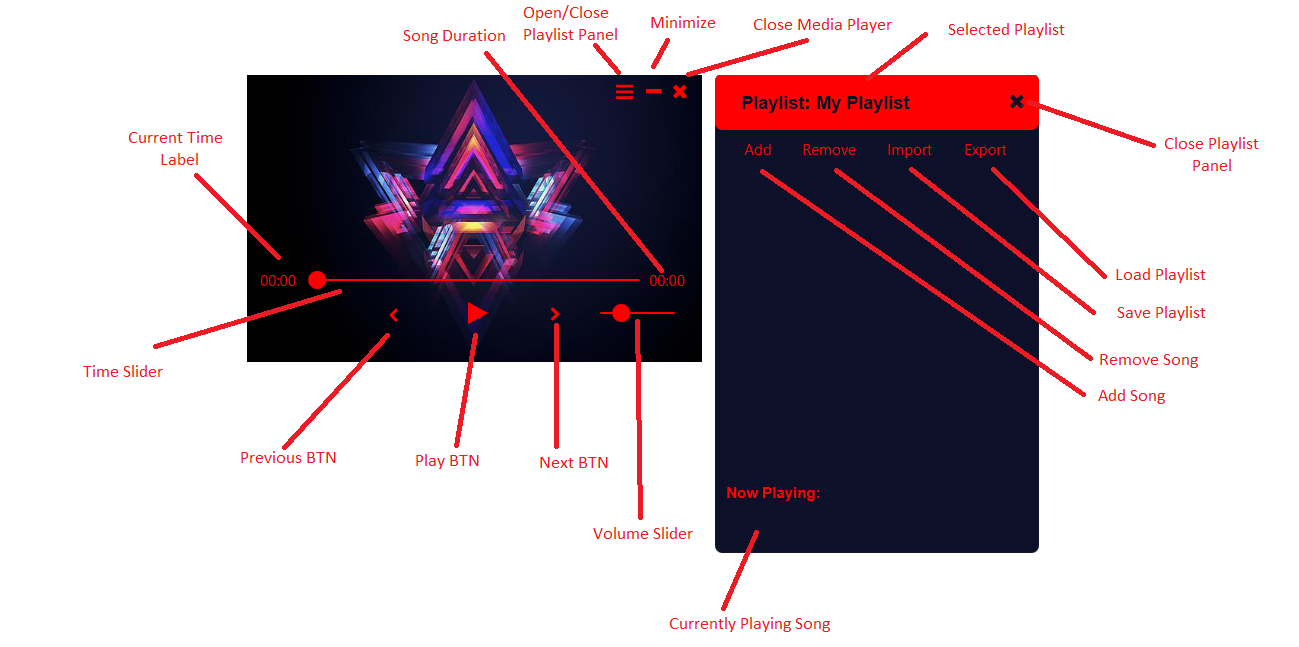
**MUSIC PLAYER USING JAVAFX**

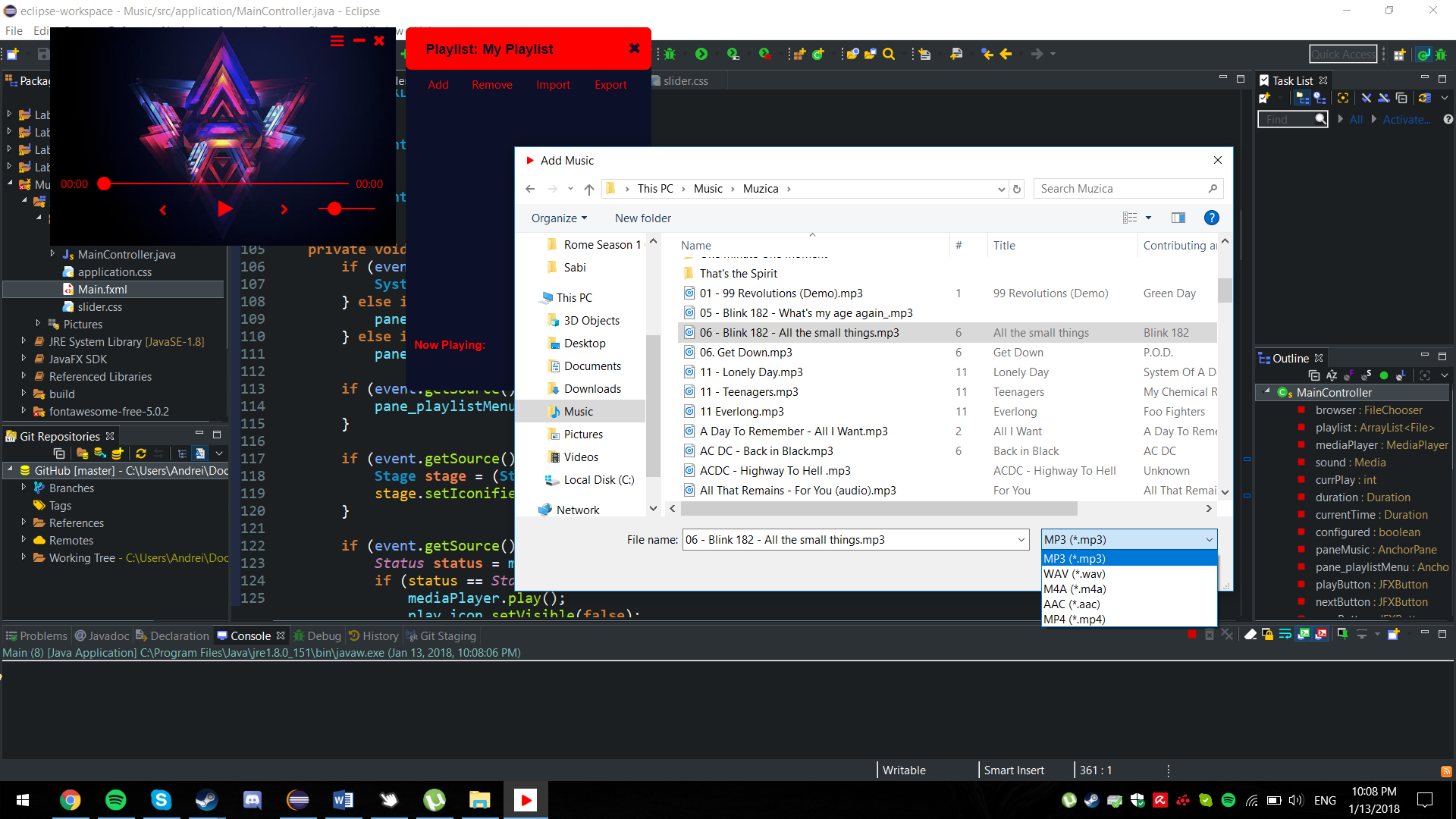
**Introduction**

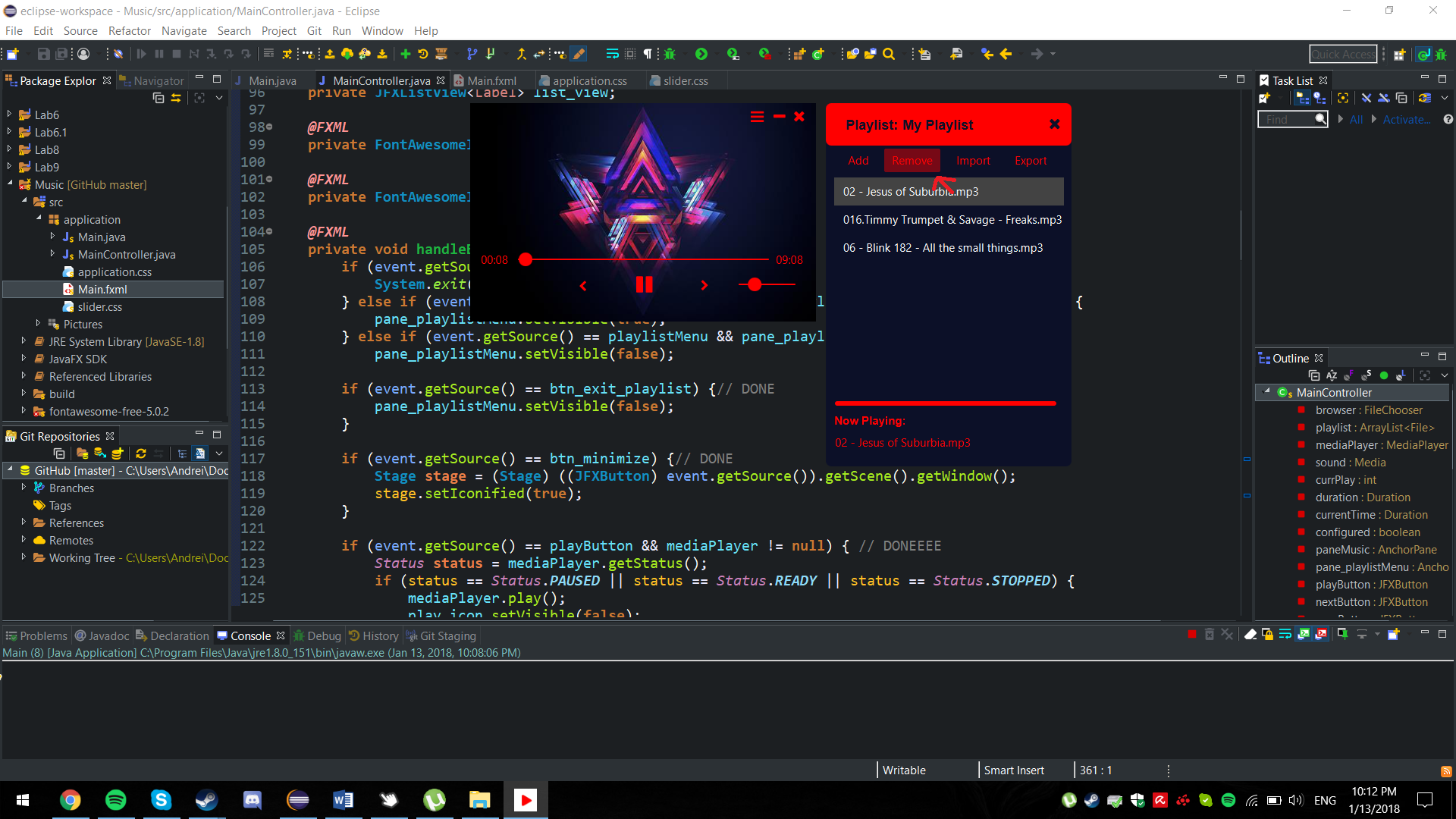
My project consisted in the development of a music player made in Java programing language with elements of JavaFX and some additional libraries (those mostly being used for GUI). The music player is able to add audio files(.MP3, .MP4, .FLAC, .WAV, .M4A, .AAC) to a list-view (that will be the graphical representation of the playlist), to remove them, to export playlists (create text files in which the path of the files is saved) and import playlists.

**How to use the application**

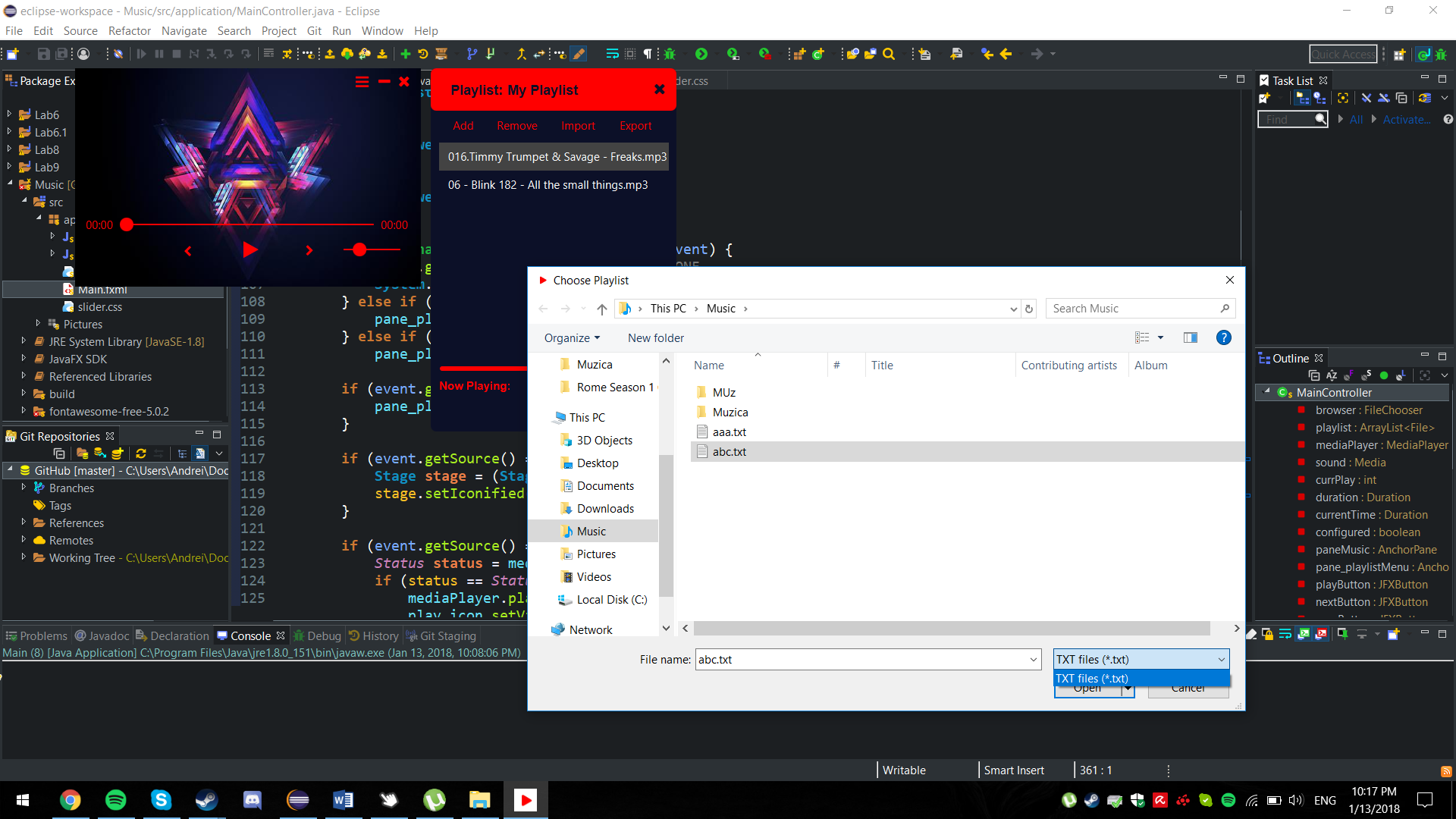


**To add** a song to the current playlist press “Add” button on the playlist panel. A file chooser window will appear. On the right select the type of audio file that you want to add. Then go to the path were your music file is.

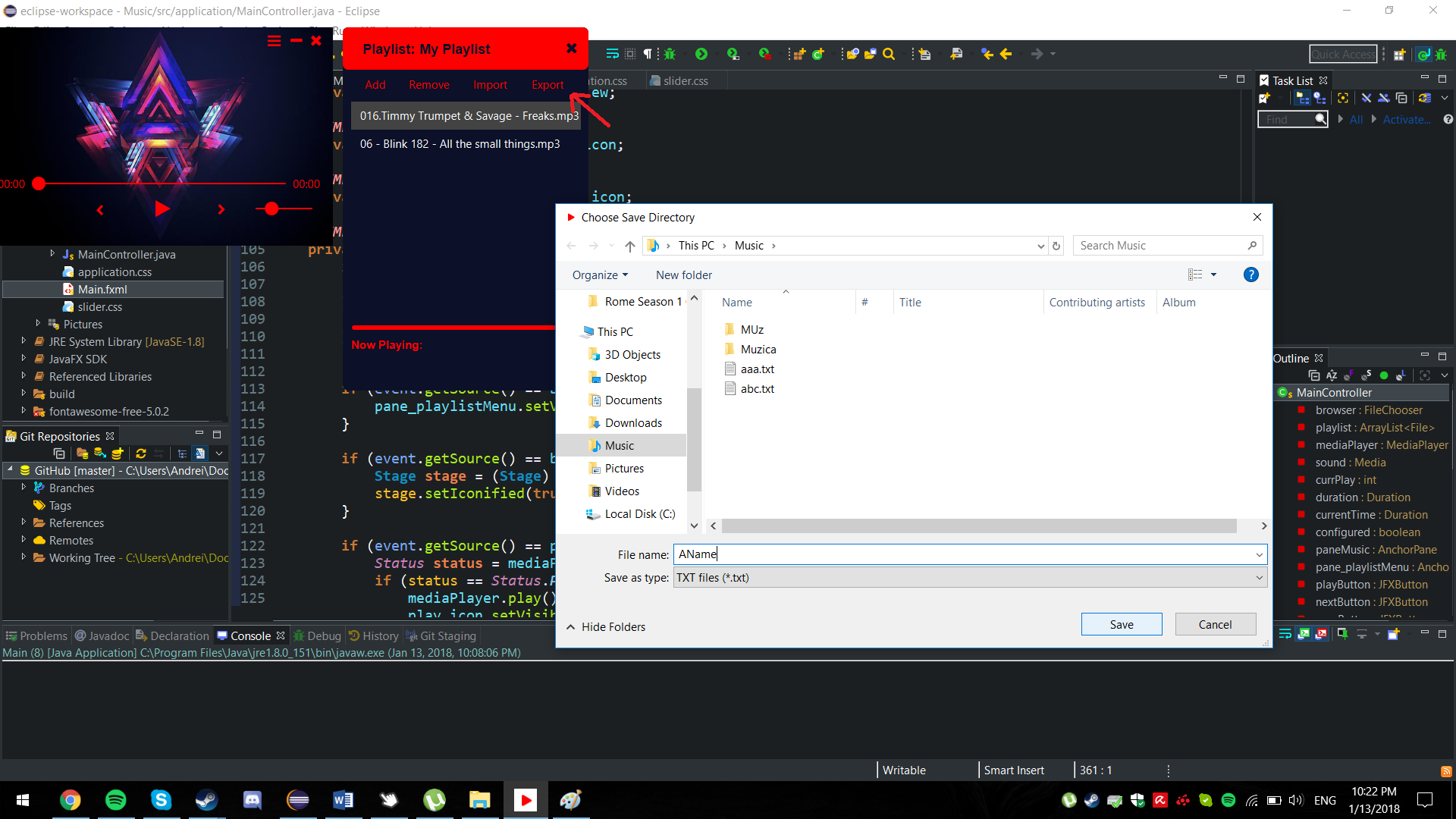


**To remove** a song from the current playlist, select a song from the playlist panel and click the “Remove” button.

**To import** a saved playlist, use the “Import” button and again using the file chooser select the playlist that you want to add. (



**To save/export** a playlist select the “Export” button and in the file chooser select a name for the playlist and a location.



**One limit use case** is the case of the time slider. It works normal when the current time of the song is not changed. Using the slider too many times causes it to desynchronize from the actual song time. This is apparently due to a bug in the “seek()” function that is used to locate the time of the song when it is manually changed. After some research on the subject I found out that this bug appears because of the bitrate of the audio file.

**Design and Classes**

This application was designed using JavaFX and also by adding the JFeonix library for the buttons and sliders and Font Awesome Icons library which in turn was used for the icons in the GUI.

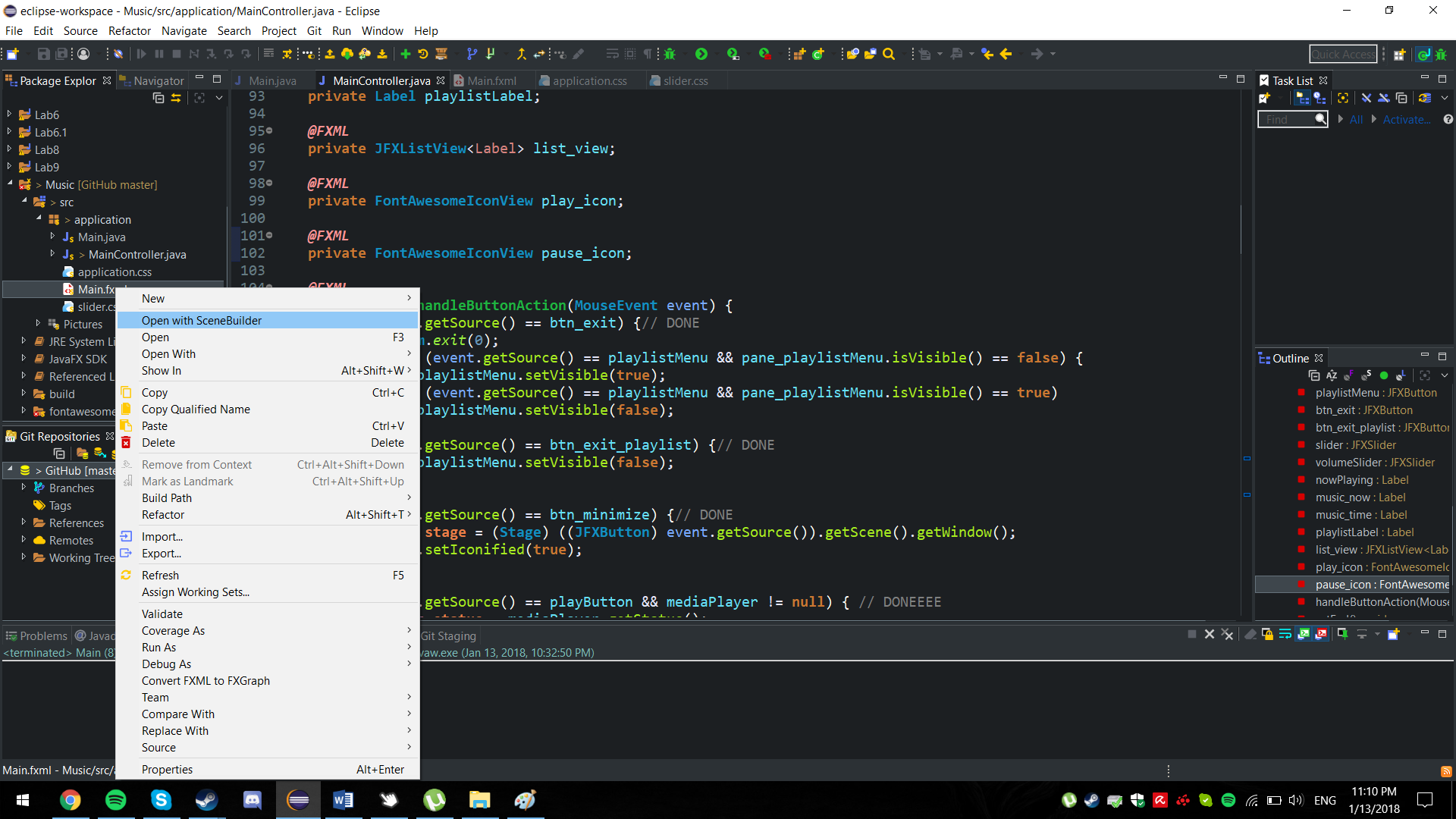
Using JavaFX became much simpler once I discovered an official tool from Oracle called JavaFX Scene Builder 2.0.

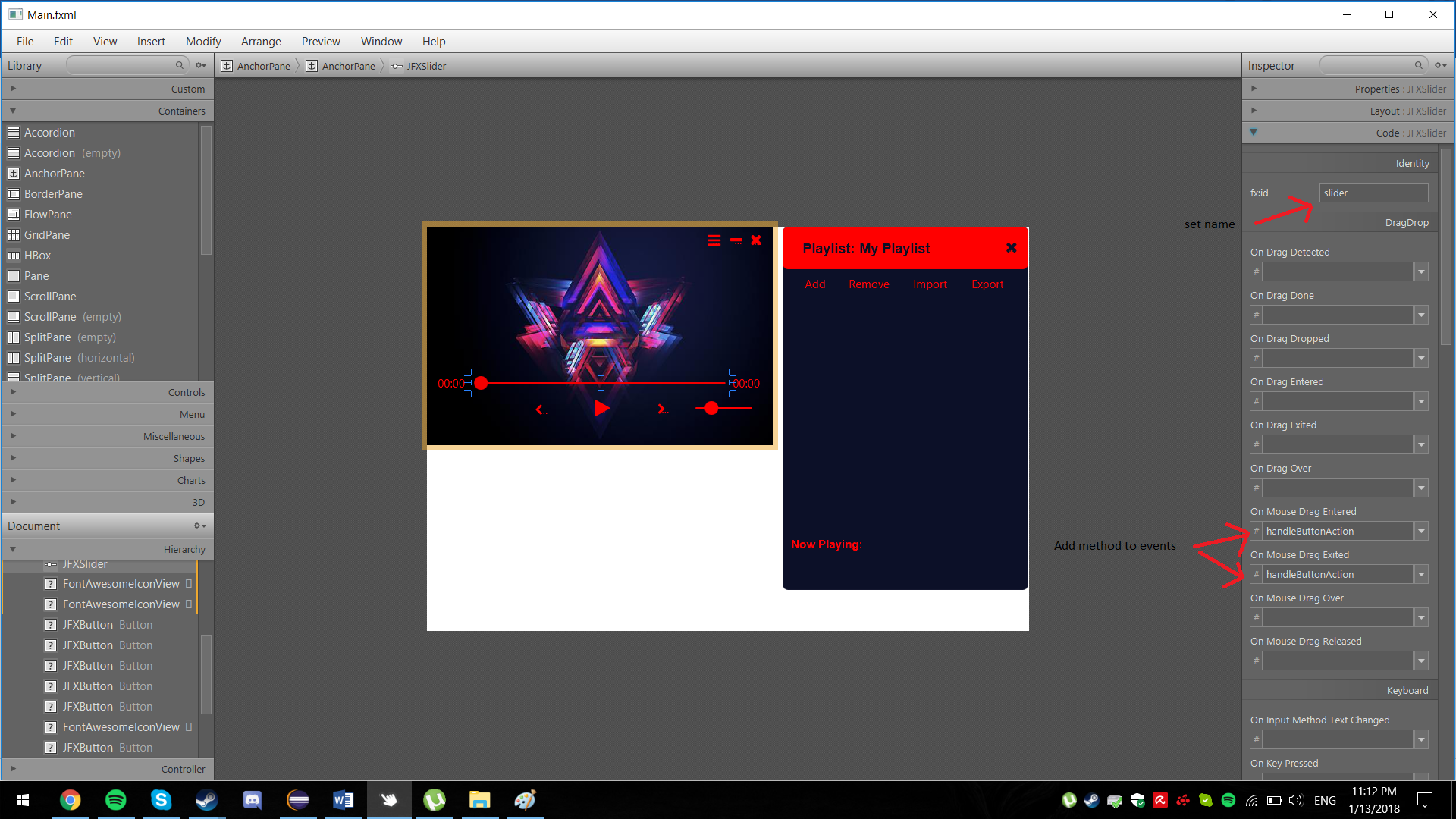
There are 2 Classes in my application, an FXML file for JavaFX and 2 CSS files which are used with the FXML file to create the GUI.

The way JavaFX work is that you have a main class (Main in our case) that configures the Stage (like JFrame). Inside this class you first have to add with the FXMLLoader the “root class” which will set the stage and configure it (for example set the background transparent if you have more than 2 windows like in our case).

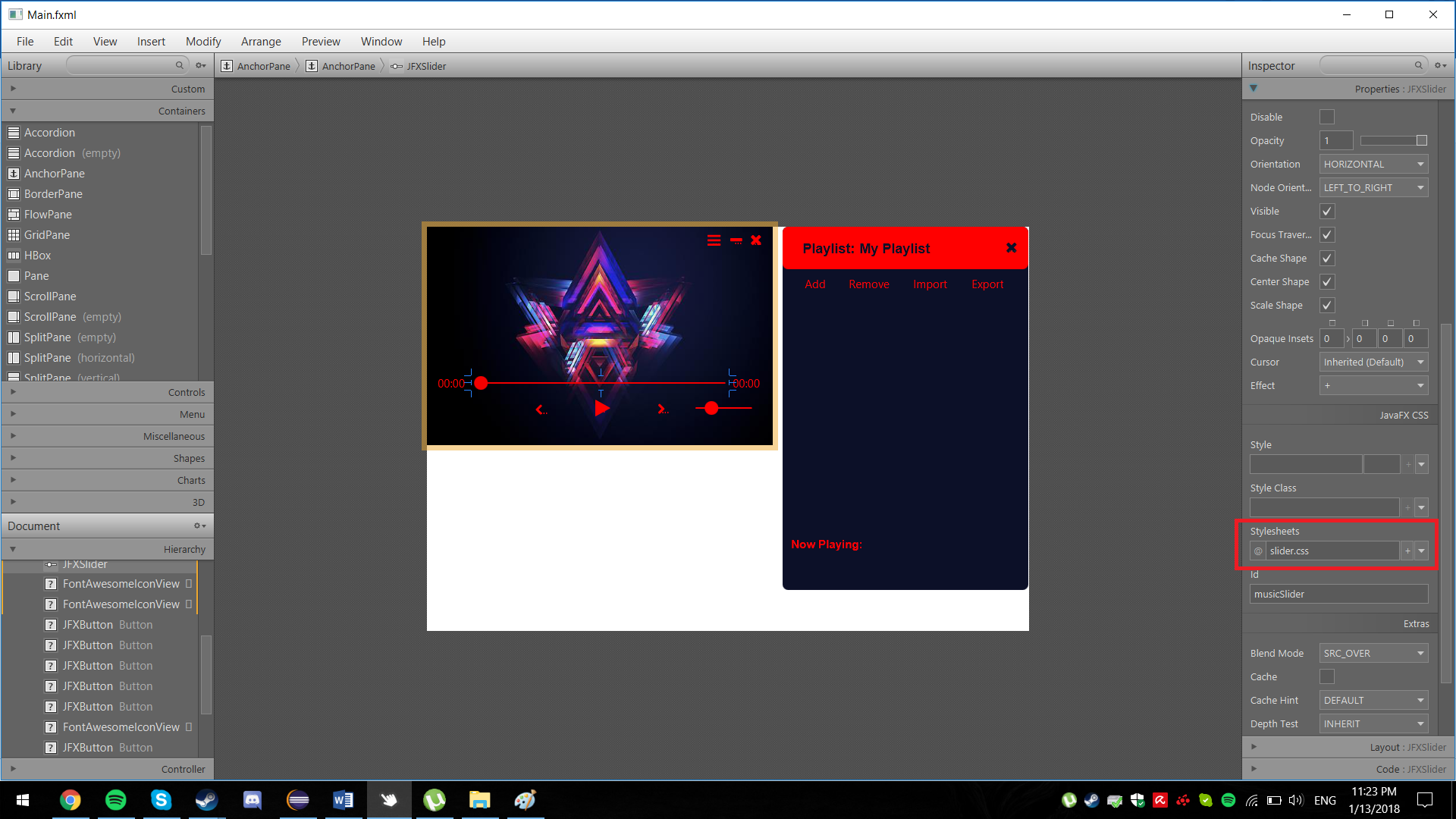
Now we get to the class that defines functionality for every part of the stage. This is called a controller (MainController in our case). Here I declared all the objects using @FXML before each in order to be searched in the FXML file to see if they exist. Then I defined a handleButtonAction(MouseEvent event) method which defines the functionality for each @FXML object defined. We need to also add the @FXML in order to be found in the FXML file and Scene Builder.

Then we get to the FXML file. Here we declare and adjust all the objects in the Scene. It can be hardcoded or (like I did) use a program called Scene Builder 2.0. How it works is that you right click the FXML file in your project and click “Open in Scene Builder”. Here you can add button, sliders, etc. In the “Code” section you need to set an fx:id to each object that you’ll use, with these names you’ll declare the objects in the FXML file. After that you’ll add methods that you defined on events that are present here.





And finally using JavaFX allows us to make use of CSS to further help our Front-End development. Creating an FXML file also creates a CSS file. In this file you can easily change colors or hide parts of an object. In order for the object to have the proprieties defined in the CSS file you need to add the file to the Scene Builder Proprieties tab under the “stylesheets” tag.



**Future Developments**

* Change the color pallets of the GUI
* Add a music visualizer
* Somehow fix the seek function (maybe recreate one)
* Fix a minor bug (by double clicking on the empty cells of the list-view you can reset the song)