Daniel Alejandro Fernandez Robles - A00354694
Camilo Enríquez Delgado - A00354532
Jesús Daniel Villota Villota - A00356255
Juan David Léctamo - A00354573

Functional requirements

Name	**FR#1**	**Add music libraries**
Summary	Add folders in which the user has stored their audio files. The paths of the directories that the user will be adding to the program will be saved in a serialized file to be loaded each time the application is opened.	
Input	A directory chosen by the user through the directory chooser	
Output	None	

Name	**FR#2**	**Play mp3 audio files**
Summary	The user will be able to listen to their songs while continuing to browse the application or through other programs unrelated to it and will see in real time the time that the song has covered	
Input	An int that represents the position of the requested song in the playlist	
Output	None	

Name	**FR#3**	**Remove directories from current libraries**
Summary	Allow to remove music directories from the current libraries so that they are no longer loaded at the start of the application. The music folder will not be removed if the songs in it are being played right now	
Input	The music folder to remove	
Output	None	

Name	**FR#4**	**Allow to sort the songs of the playlist by different criteria**
Summary	The program allows you to sort the playlist by criteria such as: title, name of the mp3 file, duration, name of the artist, album, genre and size of the mp3 file	
Input	The sorting criterion	
Output	The current playlist has been sorted according to the criterion	

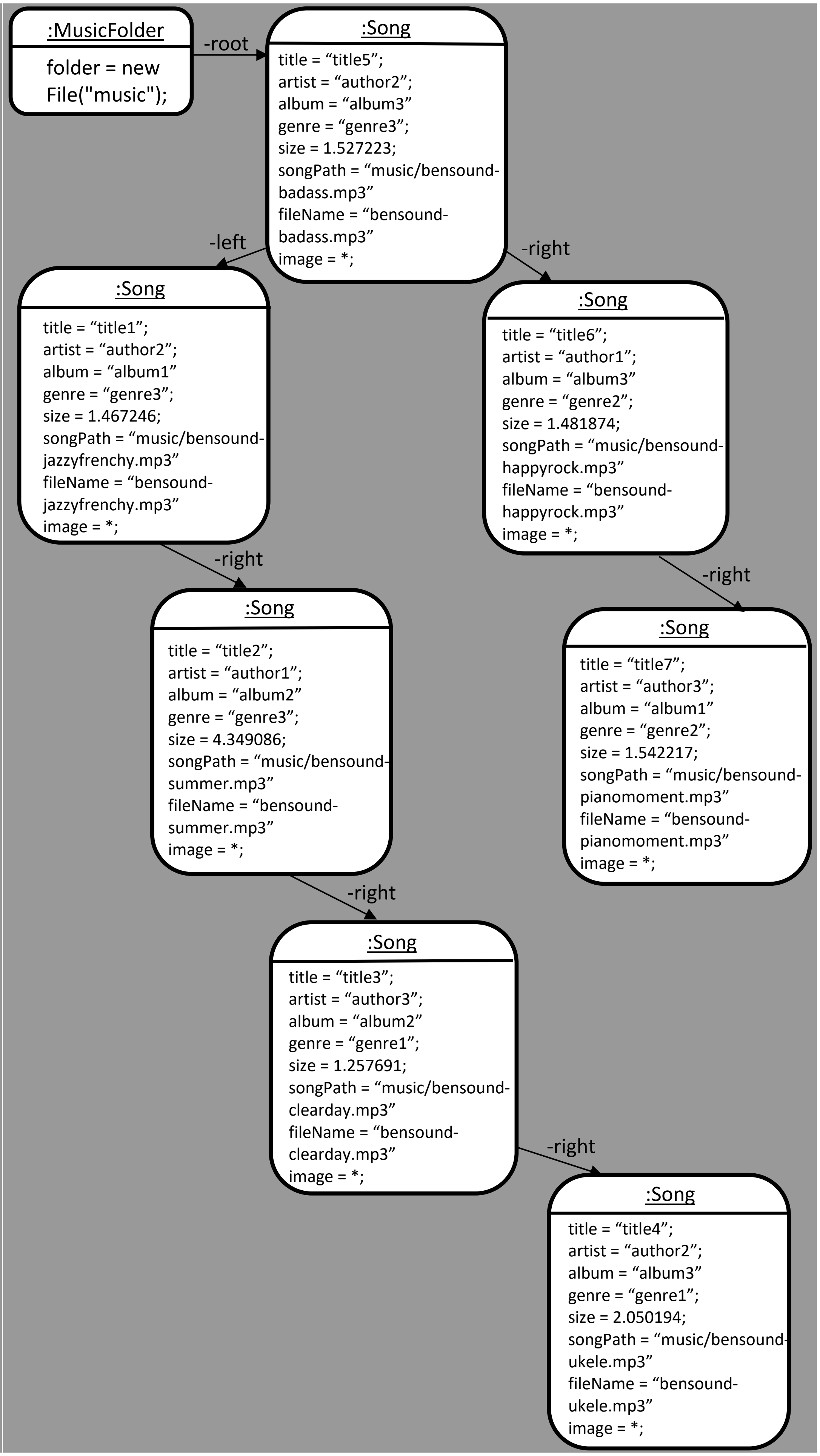
Non-functional requirements

ID	**Description**
NFR#1	Create the graphic user interface using JavaFX
NFR#2	Save information from music folders so that the program is persistent

Unitary Tests Design

(Scenarios Setting)

Name	Class	Scenario
setupScenario1	Song	Empty
setupScenario1	MusicFolder	Empty

setupScenario2	MusicFolder	

Unitary Tests Design

(Tests Development)

Test Objective: This test verifies that a Song is created successfully when a valid path and audio format are delivered as parameters in the constructor.

Class	Method	Scenario	Input	Result
Song	Song	setupScenario1	new File(“music”+File.separator+”bensound-happyrock.mp3”)	The song was created successfully.

Test Objective: This test verifies that a Song is not created successfully when an invalid path is delivered as parameter in the constructor.

Class	Method	Scenario	Input	Result
Song	Song	setupScenario1	new File("idonotexist.mp3")	The song wasn’t created successfully due to "idonotexist.mp3" is not a valid path.

Test Objective: This test verifies that a Song is not created successfully when a valid path but an invalid audio format are delivered as parameters in the constructor.

Class	Method	Scenario	Input	Result
Song	Song	setupScenario1	new File("data"+File.separator+"testfile.txt")	The song wasn’t created successfully due to testfile.txt is not a valid audio format.

Test Objective: This test verifies that a MusicFolder is created successfully when a non-existent folder path is given as parameter in the constructor.

Class	Method	Scenario	Input	Result
MusicFolder	MusicFolder	setupScenario1	new File("idonotexist")	The music folder wasn’t created successfully due to "idonotexist" is a folder that doesn’t exist.

Test Objective: This test verifies that a MusicFolder is created successfully when a valid folder path with mp3 files is given as parameter in the constructor.

Class	Method	Scenario	Input	Result
MusicFolder	MusicFolder	setupScenario1	new File(“music”)	The music folder was created successfully.

Test Objective: This test verifies that a MusicFolder is not created successfully when a valid folder path without mp3 files is given as parameter in the constructor.

Class	Method	Scenario	Input	Result
MusicFolder	MusicFolder	setupScenario1	new File("test"+File.separator+"model")	The music folder wasn’t created successfully due to test->model is a folder that doesn’t have mp3 files inside.

Test Objective: This test verifies if the method inorder() from the Song BST returns a sorted list of songs.

Class	Method	Scenario	Input	Result
MusicFolder	inorder()	setupScenario2	None	The returned list of songs is in order.

Test Objective: This test verifies if the method sortSongsByTitle() returns a list of songs sorted by title.

Class	Method	Scenario	Input	Result
MusicFolder	sortSongsByTitle()	setupScenario2	None	The list is sorted by title.

Test Objective: This test verifies if the method sortSongsByAlbum() returns a list of songs sorted sorted by album.

Class	Method	Scenario	Input	Result
MusicFolder	sortSongsByAlbum()	setupScenario2	None	The list is sorted by album.

Test Objective: This test verifies if the method sortSongsBySize() returns a list of songs sorted by size.

Class	Method	Scenario	Input	Result
MusicFolder	sortSongsBySize()	setupScenario2	None	The list is sorted by size.

Test Objective: This test verifies if the method sortSongsByGenre() returns a list of songs sorted by genre.

Class	Method	Scenario	Input	Result
MusicFolder	sortSongsByGenre()	setupScenario2	None	The list is sorted by genre.

Test Objective: This test verifies if the method sortSongsByArtist() returns a list of songs sorted by artist.

Class	Method	Scenario	Input	Result
MusicFolder	sortSongsByArtist()	setupScenario2	None	The list is sorted by artist.