**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 | |

|  |  |
| --- | --- |
| Name | R1. Load images from directories chosen by the user. |
| Summary | The program allows to load customized images from file system and draw them on the canvas. |
| Input | Directoy from file system |
| Output | The image is drawn on the canvas. |

|  |  |
| --- | --- |
| Name | R2. Save the changes made in the canvas |
| Summary | The actual state of the canvas is saved as an image in a chosen by the user directoy file. |
| Input | Directoy from file system |
| Output | The image is saved successfully in the chosen directory. |

|  |  |
| --- | --- |
| Name | R3. Allow to chose a color and shape to draw over the canvas. |
| Summary | The user can choose a color and shape from the displayed options and draw with them over the canvas. |
| Input | Wanted color and shape |
| Output | None |

|  |  |
| --- | --- |
| NAME | R4. Load random images |
| Summary | A randomly chosen image from a binary tree data structure is drawn on the canvas. |
| Input | None |
| Output | The image is shown on screen. |

|  |  |
| --- | --- |
| Name | R5. Navigate on a list of images |
| Summary | An image from a doubly linked list data structure can be chosen by the user and it's possible to visualize the next and the previous image of the actual one in the list. |
| Input | Name of the figure |
| Output | The figure is painted on the canvas |

**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 | :Song  -root  :MusicFolder  title = “title5”;  artist = “author2”;  album = “album3”  genre = "genre3";  size = 1.527223;  songPath = “music/bensound-badass.mp3”  fileName = “bensound-badass.mp3”  image = \*;  folder = new File("music");    -right  -left  :Song  :Song  title = “title6”;  artist = “author1”;  album = “album3”  genre = "genre2";  size = 1.481874;  songPath = “music/bensound-happyrock.mp3”  fileName = “bensound-happyrock.mp3”  image = \*;  title = “title1”;  artist = “author2”;  album = “album1”  genre = "genre3";  size = 1.467246;  songPath = “music/bensound- jazzyfrenchy.mp3”  fileName = “bensound- jazzyfrenchy.mp3”  image = \*;  -right  -right  :Song  :Song  title = “title7”;  artist = “author3”;  album = “album1”  genre = "genre2";  size = 1.542217;  songPath = “music/bensound- pianomoment.mp3”  fileName = “bensound- pianomoment.mp3”  image = \*;  title = “title2”;  artist = “author1”;  album = “album2”  genre = "genre3";  size = 4.349086;  songPath = “music/bensound- summer.mp3”  fileName = “bensound- summer.mp3”  image = \*;  -right  :Song  title = “title3”;  artist = “author3”;  album = “album2”  genre = "genre1";  size = 1.257691;  songPath = “music/bensound-clearday.mp3”  fileName = “bensound-clearday.mp3”  image = \*;  -right  :Song  title = “title4”;  artist = “author2”;  album = “album3”  genre = "genre1";  size = 2.050194;  songPath = “music/bensound-ukele.mp3”  fileName = “bensound-ukele.mp3”  image = \*; |
| setupStage1() | ImageOnListTest | A new instance of ImageOnList is created with a value of 10 |
| setupScenary2() | ImageOnListTest | A new list is created with the following objects:  Object 1:   * Value = 10 * next = Object 2 * Previous = null   Whetherject 2:   * value = 30 * next = Objeto3 * previous = objeto1   Object 3:   * value = 4 * next = Object 4 * previous = Object2   Bothct 4:   * value = 14 * next = null * previous = Object 3 |
| setupStage1() | ImageOnTreeTest | The root reference is initialized with a value of 50 |
| setupScenary2() | ImageOnTreeTest | The following objects are created:   * left1 as a new object of type ImageOnTree, with a value of 30 * left11 as a new object of type ImageOnTree, con valor de 20 * right12 as a new object of type ImageOnTree, con valor de 40 * right2 as a new object of type ImageOnTree, con valor de 60 * right22 as a new object of type ImageOnTree, con valor de 70   These references are added to the binary tree as it is shown in the following image:  [Tree distribution](https://correoicesiedu-my.sharepoint.com/:i:/g/personal/jesus_villota1_correo_icesi_edu_co/EcLxq79RlF1CqNYqmkIA2YgBEKIDgfbE0Y-JNahMdofOUg?e=i7G3Xz) |
| setupStage1() | ImageTest | A new object of type Image with value of 3 is instanced. |
| setupStage() | ListOfImages | Empty |
| setupScenary1() | ListOfImages | A new object of type ListOfImages with its first reference equals to null. |
| setupScenary2() | ListOfImages | A new object of type ListOfImages and 6 new objects are added to the list with where their correspondent values are the described down below:   1. 3 2. 6 3. 2 4. 5 5. 4 |
| setupScenary1 | TreeOfImages | Empty |
| setupScenary2 | TreeOfImages | An object of type TreeOfImages is initialized but with its < root > null attribute. |
| setupScenary3 | TreeOfImages | The relationship with the class to be tested is initialized.  7 nodes are added to the tree with the following values:  5, 2, 4, 1, 7, 6, 8 |

**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. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Objective: It is verified that the next is the one that should be. | | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** | |
| ImageOnList | getNext() | setupStage2() | ImageOnList second = new ImageOnList(30);  ImageOnList third = new ImageOnList(4);  ImageOnList fourth = new ImageOnList(14); | The following of the reference "imageOnList" is equal to second,  Next second is equal third,  Next third is fourth. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective Test: It is verified that the following object was correctly placed in the list. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | setNext() | setupStage2() |  | Empty  If the following object was correctly placed. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: It is verified that the previous one in the list is the one that should be. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | getPrevious() | setupStage2() | None | Empty  If you got the previous object correctly. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective Test: It is verified that the previous one was correctly set. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | setPrevious() | setupStage2() | None | Empty  If you put the previous object correctly. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective Test — Check that the status of the objects is indicated. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | isSelected() | setupStage2() | None | Empty  If the states of each object are the corresponding ones. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective Test — verifies that the status of objects is correctly changed. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | getPrevious() | setupStage2() | None | Empty  If you put the following object correctly |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the root is not NULL after adding the first object. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | ImageOnTree(int value) | setupStage2() | value = 50 | Empty  If the root is different from null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Test whether the objects on the left are correctly obtained. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | getLeft() | setupStage2() | None | Empty  If the corresponding objects are returned and Null in the cases that apply. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective Test: Test If you put an object correctly on the left. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | setLeft(imageOnTree toAdd) | setupStage2() | ImageOnTree TMP type Object, valued at 35  ImageOnTree TMP2 type Object, valued at 55 | Empty  If objects were successfully added to the left |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Test if the corresponding object is obtained on the right. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | getRight method() | setupStage2() | None | Empty  If the corresponding objects were obtained and null when applied. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that an object was successfully added to the right. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | setRight(imageOnTree toAdd) | setupStage2() | ImageOnTree TMP type Object, valued at 55  ImageOnTree TMP2 type Object, valued at 45 | Empty  If the corresponding objects were successfully added |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the value corresponding to the instantiated object is obtained. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| Image | getValue() | setupStage1() | None | Empty  If the value of the obtained object is equal to the value that was assigned. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the value was successfully changed to an already created Image object. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| Image | setValue(Int value) | setupStage1() | value = 50 | Empty  If you put the value, you passed by parameter correctly. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the value of the Image-type object was converted to String. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| Image | toString() | setupStage1() | None | Empty  If the text strings of the instantiated object are equal to what it should be when you convert it to String. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proves that the next object Fuand selected. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | selectNext() | setupStage2() | None | Empty  If the state of the next object of the first is selected. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proves that a new ListOfImages type object is successfully created. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | ListOfImages() | setupStage() | None | Empty  If the list is different from null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the objects were added in the specified order. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | addNode() | setupStage2() | Those of the setupStage2 () | Empty  If the nodes that were added are not null and if the objects were added the specified order |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the first object is equal to the first aggregate and is different from null. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | getFirst() | setupStage2() | None | Empty  If the value of the first object is the corresponding one and is different from null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that a new element was put as the first element. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | setFirst(ImageOnList toAdd) | setupStage2() | An object type ImageOnList temp with a value of 21 | Empty  If the first is different from null and first is the same as the one that was just placed as new first. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Prove that the size of the ListOfImages is correct. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | size() | setupStage2() | None | Empty  If the size of the list is equal to the corresponding value. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Test the previous object was selected to run. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | selectPrevious() | setupStage2() | None | Empty  If the object was successfully selected . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the last item in the list is successfully obtained. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | getLastNode() | setupStage2() | None | Empty  If object taken is the same as last. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the last item in the list is selected. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | lastSelected() | setupStage2() | None | Empty  If the state of the last object corresponds to the one that should have |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the constructor of the TreeOfImages class creates asuccessful instance. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | TreeOfImages | setupScenary1 | None | An instance of the TreeOfImages class is created correctly. Its (root) attribute is null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that a node is successfully added to the image tree. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | addNode(value:int) | setupScenary3 | * value = 10 | A node is added to the ImageOnTree type tree whose Attributo (value) is equal to 10. |
| TreeOfImages | addNode(n:ImageOnTree, current: ImageOnTree) | setupScenary3 | ImageOnTree n:   * value = 9 | Add the input node, taking as the start node the root of the tree that, in the current scenario, is the one whose attribute < value > is 5. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that when a random node is selected, it belongs to the tree. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | selectRandomNode | setupScenary3 | None | It looks for the value of the resulting node within the tree and verifies that the node actually belongs to the tree. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that extreme cases do not happen when you select a random node many times from the tree. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | selectRandomNode2 | setupScenary3 | 2000 random nodes selected from the tree. | It is verified that the extreme cases evaluated (1. That after 2000 selections, a node has never been selected and 2. That a single node has been selected the 2000 times. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the size method returns the current weight of the tree. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | size | setupScenary2 | None | The method returns 0 since the tree is empty. |
| TreeOfImages | size | setupScenary3 | None | The method returns 7 since that is the number of nodes in the tree. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objective test: Proving that the getRoot method returns the correct reference. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | getRoot | setupScenary2 | None | The method returns null. |
| TreeOfImages | getRoot | setupScenary3 | None | The method returns the current tree node whose Attributo < value > is equal to 5. |