**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 | |

|  |  |
| --- | --- |
| Nombre: | R1. Cargar imágenes de directorios escogidos por el usuario |
| Descripción: | El programa permite cargar imágenes personalizadas en el canvas |
| Entradas: |  |
| Salidas: |  |

|  |  |
| --- | --- |
| Nombre: | R2. Guardar los cambios del canvas como imagen en un directorio escogido por el usuario |
| Descripción: | Se toma un pantallazo de lo hecho en el canvas, y se le pide al usuario que escoja el lugar donde desea guardar la imagen |
| Entradas: | La ruta especificada por el usuario |
| Salidas: | Se guarda la imagen en un formato elegido por el usuario |

|  |  |
| --- | --- |
| Nombre: | R3. Permitir elegir diferentes colores y formas de lápiz para pintar en la pantalla |
| Descripción: | El usuario puede elegir en una paleta de colores el color deseado para dibujar en la pantalla |
| Entradas: | Color deseado |
| Salidas: |  |

|  |  |
| --- | --- |
| Nombre: | R4. Cargar figuras random |
| Descripción: | Se escoge de un árbol binario una figura al azar para ser añadida al canvas |
| Entradas: |  |
| Salidas: | La figura es añadida al canvas |

|  |  |
| --- | --- |
| Nombre: | R5. Buscar figura |
| Descripción: | Se carga una figura especificada por el usuario |
| Entradas: | Nombre de la figura |
| Salidas: | La figura se pinta en el 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 | Se instancia un nuevo objeto ImageOnList con valor de 10 |
| setupScenary2() | ImageOnListTest | se crea una nueva lista de la siguiente manera.  Objeto1:   * next = Objeto2   Objeto2:   * value = 30 * next = Objeto3 * previous = objeto1   Objeto3:   * value = 4 * next = Objeto4 * previous = objeto2   Objeto4:   * value = 14 * next = null * previous = objeto2 |
| setupStage1() | ImageOnTreeTest | Se instancia una raíz ImageOnTree con valor de 50 |
| setupScenary2() | ImageOnTreeTest | Se instancian los siguientes objetos:   * left1 como un nuevo Objeto tipo ImageOnTree, con valor de 30 * left11 como un nuevo Objeto tipo ImageOnTree, con valor de 20 * right12 como un nuevo Objeto tipo ImageOnTree, con valor de 40 * right2 como un nuevo Objeto tipo ImageOnTree, con valor de 60 * right22 como un nuevo Objeto tipo ImageOnTree, con valor de 70   Se agregan a el árbol binario de la siguiente manera:  left1 a la izquierda del root, right2 a la derecha del root, right22 a la derecha de right2, left11 a la izquierda de left1, y right12 a la derecha de left1. |
| setupStage1() | ImageTest | Se instancia un nuevo objeto tipo Image con valor de 3. |
| setupStage() | ListOfImages |  |
| setupScenary1() | ListOfImages | Se instancia un nuevo objeto tipo ListOfImages |
| setupScenary2() | ListOfImages | El mismo que el setupScenary1().  Se agregan 6 nuevos objetos a la lista con los siguientes valores y orden:   1. 0 2. 3 3. 6 4. 2 5. 5 6. 4 |
| setupScenary1 | TreeOfImages | Vacío |
| setupScenary2 | TreeOfImages | Se inicializa un objeto del tipo TreeOfImages pero con su atributo <root> nulo. |
| setupScenary3 | TreeOfImages | Se inicializa la relación con la clase a probar.  Se añaden 7 nodos al árbol con los siguientes valores:  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: Se comprueba que el siguiente es el que debería ser. | | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** | |
| ImageOnList | getNext() | setupStage2() | ImageOnList second = new ImageOnList(30);  ImageOnList third = new ImageOnList(4);  ImageOnList fourth = new ImageOnList(14); | El siguiente de la referencia “imageOnList” es igual a second,  El siguiente de second es igual third,  el siguiente de third es fourth. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Se comprueba que se puso correctamente el objeto siguiente en la lista. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | setNext() | setupStage2() |  | Vacío  Si se puso correctamente el objeto siguiente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Se comprueba que el anterior en la lista es el que debería ser. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | getPrevious() | setupStage2() | None | Vacío  Si obtuvo correctamente el objeto anterior. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Se comprueba que se puso el anterior correctamente. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | setPrevious() | setupStage2() | None | Vacío  Si se puso correctamente el objeto anterior. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Se comprueba que el estado de los objetos sea el indicado. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | isSelected() | setupStage2() | None | Vacío  Si los estados de cada objeto son los correspondientes. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Se comprueba que se cambia correctamente el estado de los objetos. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnList | getPrevious() | setupStage2() | None | Vacío  Si se puso correctamente el objeto siguiente |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que la raíz no es nula después de agregar el primer objeto. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | ImageOnTree(int value) | setupStage2() | value = 50 | vacío  Si la raíz es diferente de null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar si se obtienen correctamente los objetos de la izquierda. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | getLeft() | setupStage2() | None | vacío  Si se retornan los objetos correspondientes y nulo en los casos que aplica. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar si se pone de manera correcta un objeto a la izquierda. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | setLeft(imageOnTree toAdd) | setupStage2() | Objeto tipo ImageOnTree tmp, con valor de 35  Objeto tipo ImageOnTree tmp2,con valor de 55 | vacío  Si se agregaron correctamente los objetos a la izquierda |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar si se obtiene el objeto correspondiente a la derecha. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | getRight() | setupStage2() | None | vacío  Si se obtuvieron los objetos correspondientes y null cuando aplica. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se agregó correctamente un objeto a la derecha. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ImageOnTree | setRight(imageOnTree toAdd) | setupStage2() | Objeto tipo ImageOnTree tmp, con valor de 55  Objeto tipo ImageOnTree tmp2, con valor de 45 | vacío  Si se agregaron correctamente los objetos correspondientes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se obtiene el valor correspondiente al del objeto instanciado. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| Image | getValue() | setupStage1() | None | vacío  Si el valor del objeto obtenido es igual al valor que se asignó. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se cambió correctamente el valor a un objeto Image ya creado. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| Image | setValue(Int value) | setupStage1() | value = 50 | vacío  Si se puso correctamente el valor que se le pasó por parámetro. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se convirtió a String el valor del objeto tipo Image. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| Image | toString() | setupStage1() | None | vacío  Si las cadenas de texto del objeto instanciado son igual a la que debería ser al convertirlo a String. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Prueba que el objeto siguiente fue seleccionado. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | selectNext() | setupStage2() | None | vacío  Si el estado del objeto siguiente del primero está seleccionado. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Prueba que se crea correctamente un nuevo objeto tipo ListOfImages. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | ListOfImages() | setupStage() | None | Vacío  Si la lista es diferente de null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se agregaron los objetos en el orden especificado. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | addNode() | setupStage2() | Los del setupStage2() | Vacío  Si los nodos que se agregaron no son nulos y si los objetos se agregaron el orden especificado |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que el primer objeto es igual al primero agregado y es diferente de null. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | getFirst() | setupStage2() | None | vacío  Si el valor del primer objeto es el correspondiente y es diferente de null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se puso como primer elemento un nuevo elemento. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | setFirst(ImageOnList toAdd) | setupStage2() | Un objeto tipo ImageOnList temp con valor de 21 | Vacío  si el primero es diferente de null y el first es igual al que acaba de ser colocado como nuevo first. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que el tamaño de la ListOfImages es la correcta. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | size() | setupStage2() | None | Vacío  Si el tamaño de la lista es igual al valor correspondiente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar se seleccionó el objeto previo correctamente. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | selectPrevious() | setupStage2() | None | Vacío  Si se seleccionó correctamente el objeto . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se obtiene correctamente el último elemento de la lista. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | getLastNode() | setupStage2() | None | Vacío  Si objeto tomado es igual al que está de último. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que el último elemento de la lista está seleccionado. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| ListOfImages | lastSelected() | setupStage2() | None | Vacío  Si el estado del último objeto corresponde con el que debería tener |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que el constructor de la clase TreeOfImages crea una instancia correctamente. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | TreeOfImages | setupScenary1 | None | Una instancia de la clase TreeOfImages es creada correctamente. Su atributo (root) es nulo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que se añada un nodo correctamente al árbol de imágenes. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | addNode(value:int) | setupScenary3 | * value = 10 | Se añade un nodo al árbol de tipo ImageOnTree cuyo attributo (value) es igual a 10. |
| TreeOfImages | addNode(n:ImageOnTree, current: ImageOnTree) | setupScenary3 | ImageOnTree n:   * value = 9 | Se añade el nodo de entrada, tomando como nodo de inicio la raíz del árbol que, en el escenario actual, es aquel cuyo atributo <value> es 5. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que cuando se selecciona un nodo aleatorio, este pertenece al árbol. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | selectRandomNode | setupScenary3 | None | Se busca el valor del nodo resultante dentro del árbol y se verifica que el nodo efectivamente pertenece al árbol. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que casos extremos no suceden al seleccionar un nodo aleatorio muchas veces del árbol. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | selectRandomNode2 | setupScenary3 | 2000 nodos aleatorios seleccionados del árbol. | Se comprueba que los casos extremos evaluados (1. que después de 2000 selecciones, un nodo no haya sido seleccionado nunca y 2. que un solo nodo haya sido seleccionado las 2000 veces. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que el método size retorna el peso actual del árbol. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | size | setupScenary2 | None | El método retorna 0 ya que el árbol está vacío. |
| TreeOfImages | size | setupScenary3 | None | El método retorna 7 ya que esa es la cantidad de nodos que hay en el árbol. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective: Probar que el método getRoot devuelve la referencia correcta. | | | | |
| Class | **Method** | **Scenario** | **Input** | **Result** |
| TreeOfImages | getRoot | setupScenary2 | None | El método retorna null. |
| TreeOfImages | getRoot | setupScenary3 | None | El método retorna el nodo del árbol actual cuyo attributo <value> es igual a 5. |