Requerimientos funcionales:

R1: Registrar a un nuevo jugador, el cual tiene nombre, apodo y género. Este usuario queda registrado dentro del sistema por lo cual después no deberá de volver a ingresar todos sus datos, el apodo debe de ser único, no está permitido que haya dos apodos iguales.

R2: Girar la ruleta, el usuario tendrá un botón en el cual al hacer clic la ruleta comience a girar y de acuerdo con esto saldrá una canción y seguido de esto, la canción se quitará de la ruleta.

R3: Seleccionar canciones, el jugador tendrá la opción de configurar la ruleta. Este puede configurarlo de acuerdo con que las canciones salgan en modo aleatorio, salgan de forma ordenada alfabéticamente.

R4: Desbloquear canciones, para desbloquear canciones hay un modo de juego dentro de la aplicación en donde al superar cada nivel recibe puntos los cuales se van acumulando, cada canción tiene una cantidad de puntos determinado, necesarios para poder desbloquearlos, cuando la persona obtiene la cantidad de puntos necesarios y desbloquea una canción, esta se añade a la lista disponible para usar en la ruleta.

R5: Ordenar Usuarios, el sistema estará en la capacidad de mostrar los usuarios, y cuando estos estén mostrados sea de forma ordenada de acuerdo con el nombre ingresado.

R6: Buscar Usuarios, se va a poder buscar a una persona ingresada dentro del sistema principalmente para que la persona pueda ver la cantidad de puntos que lleva acumulados mientras está jugando, debido a que cada persona tiene un apodo distinto, la búsqueda se llevara a cabo de acuerdo con el apodo.

R7: Acumular puntos, cada usuario va a tener una cierta cantidad de puntos inicial (20) tan solo por registrarse en el juego, después de esto podrá ir consiguiendo más a medida en que vaya jugando, por lo cual se llevará un registro de cada punto para después mostrar la totalidad actual cuando el usuario desee verlos.

R8: Verificar si el usuario existe, el sistema debe de verificar que un usuario si existe antes de intentar jugar o buscarlo, dado el caso que no exista se le notificara al usuario por medio de un mensaje en pantalla.

R9: Agregar canción, cuando el usuario consigue una cierta cantidad de puntos, es capaz de agregar una nueva canción para que esta aparezca en la ruleta y queda guardada dentro de la lista interna del sistema para las próximas veces que se juegue.

R10: Mostrar minijuego, el cual consiste en juego de adivinanzas, si el jugador acierta, gana un total de 10 puntos, si pierde se cierra el minijuego y debe de volver a intentarlo en el próximo.

R12: Parar canción, cuando la persona haya adivinado la canción puede parar la canción y responder cual es la canción que estaba sonando.

R13: Buscar canción, La persona puede buscar la canción que escucho o la que quiere colocar dentro del juego.

R14: Ordenar canción, la persona puede seleccionar este modo de reproducir las canciones, por ende, deben de estar ordenadas alfabéticamente cuando el usuario seleccione esta opción.

R15: Añadir artistas, como parte del minijuego, este consiste en llenar encuestas que representan “anuncios” con los cuales se pueden ganar más puntos, el usuario puede escoger un artista de los que aparecen y añadirlo a la lista.

R16: Añadir Compositores, como parte del minijuego, este consiste en llenar encuestas que representan “anuncios” con los cuales se pueden ganar más puntos, el usuario puede escoger un compositor de los que aparecen y añadirlo a la lista.

Requerimientos no funcionales:

R1: Verificar que no se pueda registrar dos apodos iguales dentro del sistema y en caso tal lanzar una excepción de tipo UserAlreadyExistsException.

R2: Los usuarios serán manejados con la estructura de datos ArrayList.

R3: Las canciones serán manejadas con Listas doblemente enlazadas.

R4: El minijuego se manejará con Arboles Binarios.

R5: Se implementará librerías de Musica para poder manejar la reproducción de canciones.

R6: Para verificar que el usuario existe, se utiliza el método de búsqueda binaria.

R7: El método de búsqueda de Usuario por apodo, se realiza con búsqueda binaria.

R8: Los datos ingresados por los usuarios, los artistas y los puntos recolectados serán guardados en un archivo serializable.

R9: Si una persona no se encuentra dentro del sistema y se intenta buscar, se lanza una excepción de tipo UserDoesntExistException.

R10: No se puede canjear puntos si no tiene suficientes, por lo que de intentar hacer esto, se lanzara una excepción de tipo InsufficientPointsException.

R11: Los usuarios se ordenarán por método de ordenamiento de inserción.

R12: Las canciones se van a ordenar por el método de ordenamiento de burbuja.

R13: Todos los campos del formulario de registro de usuario son OBLIGATORIOS, por ende, al no registrar algún dato, el sistema no permitirá el registro y lanzará una excepción de tipo RequiredFieldsException.

R14: La suma de puntos se realiza con un método de acumulación naturalmente recursivo.

R15: Se usa Comparable como parámetro de comparación el apodo del usuario.

R16: Si una persona intenta agregar una canción que ya existe lanza una excepción de tipo SongAlreadyExistsException y se le impedirá agregarla.

R17: La lista de canciones será cargada con persistencia de archivos planos.

R18: Si la lista de canciones está vacía, lanzara una excepción de tipo NullPointerException.

R19: Habrá un hilo para abrir el minijuego, también habrá otro para cargar la partida y un último para reproducir la canción.

R20: Habrá un hilo para reproducir la siguiente canción

R21: Cuando la ventana principal se cierre, todas las demás abiertas se cerrarán también.

R22: Por defecto, las canciones se reproducen aleatoriamente.

R23: Si el usuario no tiene puntos suficientes, el botón de agregar canciones no estará activado

R24: Si no hay un usuario que haya iniciado sesión no se puede mostrar información ni comprar puntos.

R25: Habrá un hilo que mueva un mensaje de bienvenida en la pantalla principal

Pruebas Unitarias:

Configuración de los Escenarios de la clase Game:

Nombre	Clase	Escenario
setUpStage1	GameTest	User name= “Fernanda”, nickname= “fernandarojas152”, gender=”femenine”, password= “fernanda”

Nombre	Clase	Escenario
setUpStage2	GameTest	No se ha creado ningún jugador, la lista está vacía y solo se inicializa el juego.

Nombre	Clase	Escenario
setUpStage3	GameTest	User name= “Fernanda”, nickname= “fernandarojas152”, gender=”femenine”, password= “fernanda” User2 name= “Angelica”, nickname= “angelica2013”, gender=”femenine”, password= “angelica”

Nombre	Clase	Escenario
setUpStage4	GameTest	User name= “Fernanda”, nickname= “fernandarojas152”, gender=”femenine”, password= “fernanda” User2 name= “Angelica”, nickname= “angelica2013”, gender=”femenine”, password= “angelica” User3 name= “Amanda”, nickname= “amandaR”, gender=”femenine”, password= “angelica” User4 name= “Saru”, nickname= “saruhiko”, gender=”male”, password= “angelica”

Nombre	Clase	Escenario
setUpStage5	GameTest	User name= “Fernanda”, nickname= “fernandarojas152”, gender=”femenine”, password= “fernanda”

Nombre	Clase	Escenario
setUpStage6	GameTest	Artist a= new Artist("Ed Sheeran", "England", "Atlantic Records"); Artist b= new Artist("One Direction", "England", "SYCO Music"); Artist c= new Artist("Taylor Swift", "USA", "Universal Music");

Diseño de Casos de Prueba:

Objetivo de la Prueba:. El programa es capaz de verificar que existen usuarios registrados previamente dentro del usuario y no permite ingresar uno con el mismo apodo.				
Clase	Método	Escenario	Valores de Entrada	Resultado
Game	addUser()	setUpStage1	User2 name= "Fernanda", nickname= "fernandarojas152", gender="femenine", password= "fernanda"	UserAlreadyExistsException

Objetivo de la Prueba:. El programa busca un usuario dentro del sistema y si no lo encuentra avisa al usuario de que este no existe aún.				
Clase	Método	Escenario	Valores de Entrada	Resultado
Game	searchUser()	setUpStage1	ninguno	UserDoesntExistException

Objetivo de la Prueba:. El programa verifica si la lista que almacena a los usuarios tiene alguno dentro de esta, por lo que busca dentro de una lista vacía a una persona y manda un mensaje avisando al usuario				
Clase	Método	Escenario	Valores de Entrada	Resultado
Game	searchUser()	setUpStage2	ninguno	UserDoesntExistException

Objetivo de la Prueba:. Se verifica que el usuario no pueda agregar una nueva cuenta si este no rellena todos los campos de registro.

Clase	Método	Escenario	Valores de Entrada	Resultado
Game	addUser()	setUpStage5	User3 name= “”, nickname= “amandaR”, gender=””, password= “amanda”	RequiredFieldsException

Objetivo de la Prueba:. El programa es capaz de ordenar una lista de usuarios en orden alfabetico.

Clase	Método	Escenario	Valores de Entrada	Resultado
Game	sortUsers()	setUpStage4	ninguno	Angelica2013

Objetivo de la Prueba:. El programa es capaz de buscar a un usuario y verificar que este existe

Clase	Método	Escenario	Valores de Entrada	Resultado
Game	searchUser()	setUpStage3	ninguno	True, angelica2013 existe

Objetivo de la Prueba:. El programa es capaz de buscar un usuario y al no ser encontrado notificarlo por medio de un mensaje.

Clase	Método	Escenario	Valores de Entrada	Resultado
Game	searchUser()	setUpStage3	ninguno	UserDoesntExistException

Objetivo de la Prueba:. El programa añade artistas

Clase	Método	Escenario	Valores de Entrada	Resultado
Game	addArtist()	setUpStage2	Artist a= new Artist("Ed Sheeran", "England", "Atlantic Records");	true

Objetivo de la Prueba:. El programa es capaz de ordenar un artista en preorden				
Clase	Método	Escenario	Valores de Entrada	Resultado
Game	preOrderSort()	setUpStage6	ninguno	true

Objetivo de la Prueba:. El programa es capaz de ordenar un artista en posorden				
Clase	Método	Escenario	Valores de Entrada	Resultado
Game	posOrderSort()	setUpStage6	ninguno	true

Objetivo de la Prueba El programa es capaz de ordenar un artista en inorden				
Clase	Método	Escenario	Valores de Entrada	Resultado
Game	inOrderSort()	setUpStage6	ninguno	true

Configuración de los Escenarios de la clase User:

Nombre	Clase	Escenario
setUpStage1	User	User name= “Fernanda”, nickname= “fernandarojas152”, gender=”femenine”, password= “fernanda” User.accumulatePoints(5)

Nombre	Clase	Escenario
setUpStage2	User	User name= “Fernanda”, nickname= “fernandarojas152”, gender=”femenine”, password= “fernanda” User.accumulatePoints(10) User.setPoints(user.getPoints()-20)

Diseño de Casos de Prueba:

Objetivo de la Prueba:. Verifica que se vaya acumulando los puntos obtenidos por el usuario.				
Clase	Método	Escenario	Valores de Entrada	Resultado
User	acumulatePoints()	setUpStage1	User.accumulatePoints(10)	25

Objetivo de la Prueba:. Verifica que se eliminen puntos cuando estos se gastan.				
Clase	Método	Escenario	Valores de Entrada	Resultado
User	acumulatePoints()	setUpStage2	ninguno	0

Configuración de los Escenarios de la clase de la clase MusicLibrary:

Nombre	Clase	Escenario
setUpStage1	MusicLibrary	Se instancia la nueva librería musical.

Nombre	Clase	Escenario
setUpStage2	MusicLibrary	Se añade una nueva canción. “./src/give_me_love.mp3”

Diseño de Casos de Prueba:

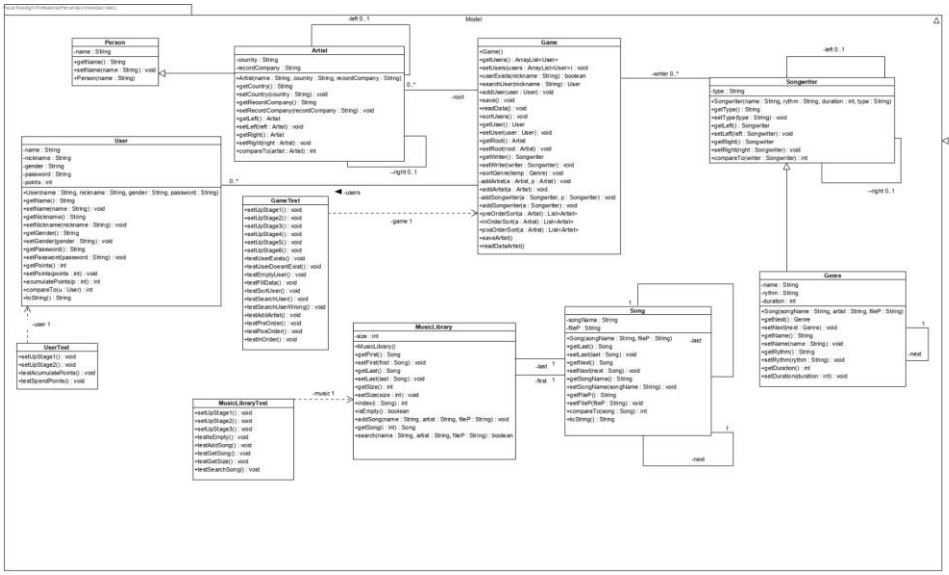
Objetivo de la Prueba:. El programa es capaz de agregar una nueva canción				
Clase	Método	Escenario	Valores de Entrada	Resultado
MusicLibrary	addSong()	setUpStage1	“./src/give_me_love.mp3”	True, se agrega correctamente

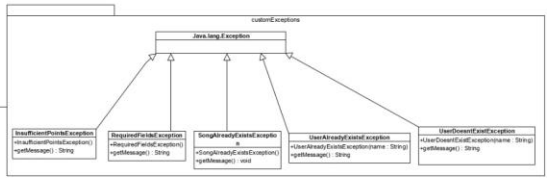
Objetivo de la Prueba:. El programa no permite ingresar dos canciones iguales dentro de la lista de reproducción para la ruleta				
Clase	Método	Escenario	Valores de Entrada	Resultado
MusicLibrary	addSong()	setUpStage2	“./src/give_me_love.mp3”	SongAlreadyExistException

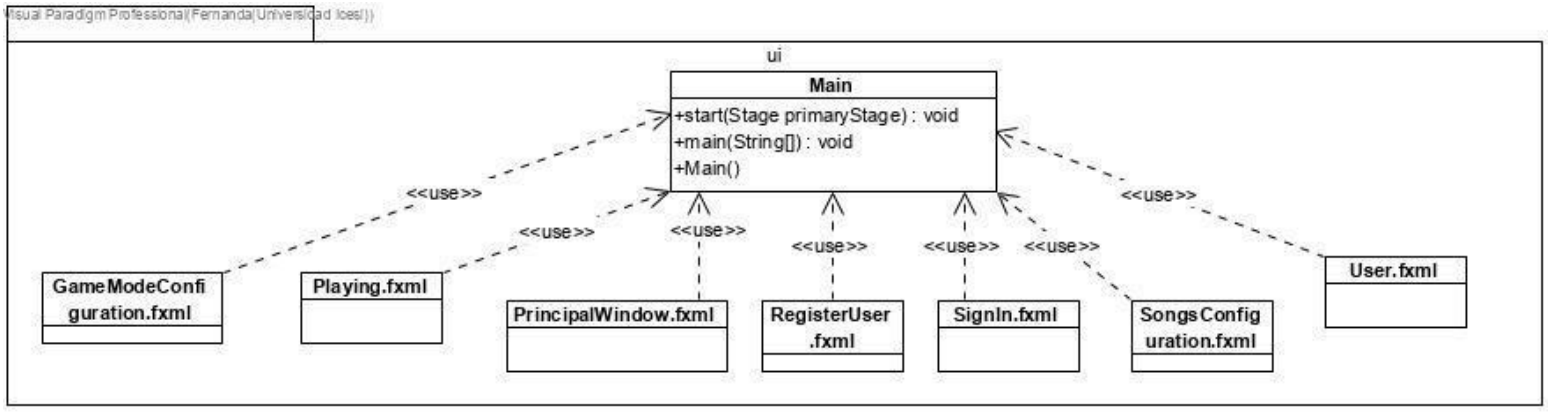
Objetivo de la Prueba:. El programa es capaz de buscar una canción dentro de la lista de reproducción.				
Clase	Método	Escenario	Valores de Entrada	Resultado

MusicLibrary	search()	setUpStage2	ninguno	True, la canción existe.

DIAGRAMAS DE CLASE:







- **<u>Los requerimientos que están implementados ahora son: agregar usuario y agregar canción, buscar usuario y buscar canción, acumular puntos, verificar si el usuario existe, ordenar usuarios.</u>**