gastar group

Reproductor de Música

Trabajo Final Informe

Autores:
<Gasparini, Roman;
Tarres, Martin>

Document Version: 1.0.2

Historial de Revisiones

Version	Fecha	Resumen de Cambios	
1.0.0	22/06/2017	Primera versión	
1.0.1	23/06/2017	Diseño e implementación	
1.0.2	24/06/2017	Cambios generales	

Tabla de contenidos

1 Integrantes

El grupo Gastar group está conformado por

Gasparini, Roman	**36587720**	**roman.gasparini@gmail.com**
Tarres, Martín	**36725643**	**martintarres@gmail.com**

2 Release Notes

El documento de las notas de entregas se puede encontrar en la siguiente URL.

3 Manejo de las Configuraciones

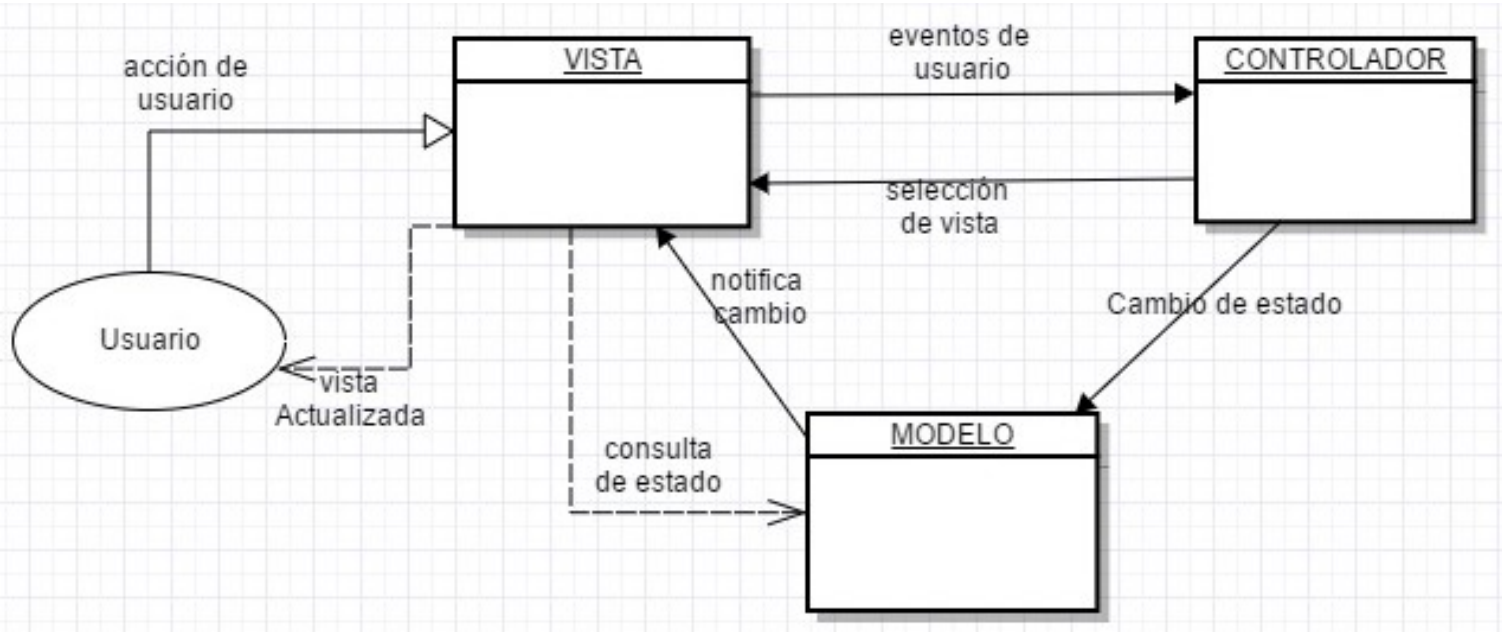
El plan de gestión de las configuraciones se encuentra en la siguiente dirección, URL.

4 Requerimientos

El documento de Especificación de Requerimientos de Software utilizado en el proyecto se encuentra en la siguiente URL.

5 Arquitectura

Diagrama de Arquitectura



En el diagrama anterior podemos observar la arquitectura del sistema, en el cual utilizamos el mas indicado de los patrones para este tipo de proyectos, como lo es el reproductor de música, que es el MVC. Donde separa la interfaz de usuario (vista), de la lógica, es decir del modelo, y del controlador que es el que interpreta las acciones. Como este patrón dispone de un bajo acoplamiento, nos permite poder cambiar de vistas con facilidad, como pasar de la lista principal, a la vista secundaria, es decir de la carpeta contenedora de las pistas de audio, a la vista de la lista de reproducción. De este modo, el modelo puede ser reutilizable por cada vista, y así cada una de estas puede implementar funcionalidades totalmente distintas del modelo.
También debido a su bajo acoplamiento, es posible hacer nuevas implementaciones, o cambiar la vista, de manera muy sencilla.
En el diagrama podemos observar que las lineas continuas (——►) como el flujo principal, donde el usuario tiene interacción con la interfaz gráfica de usuario, el controlador es notificado sobre los cambios, y la orden de este ultimo al modelo.
Las lineas punteadas (- - - - ►) indican el flujo secundario, que hace referencia a la actualización de las vistas.

Diagrama Deployment

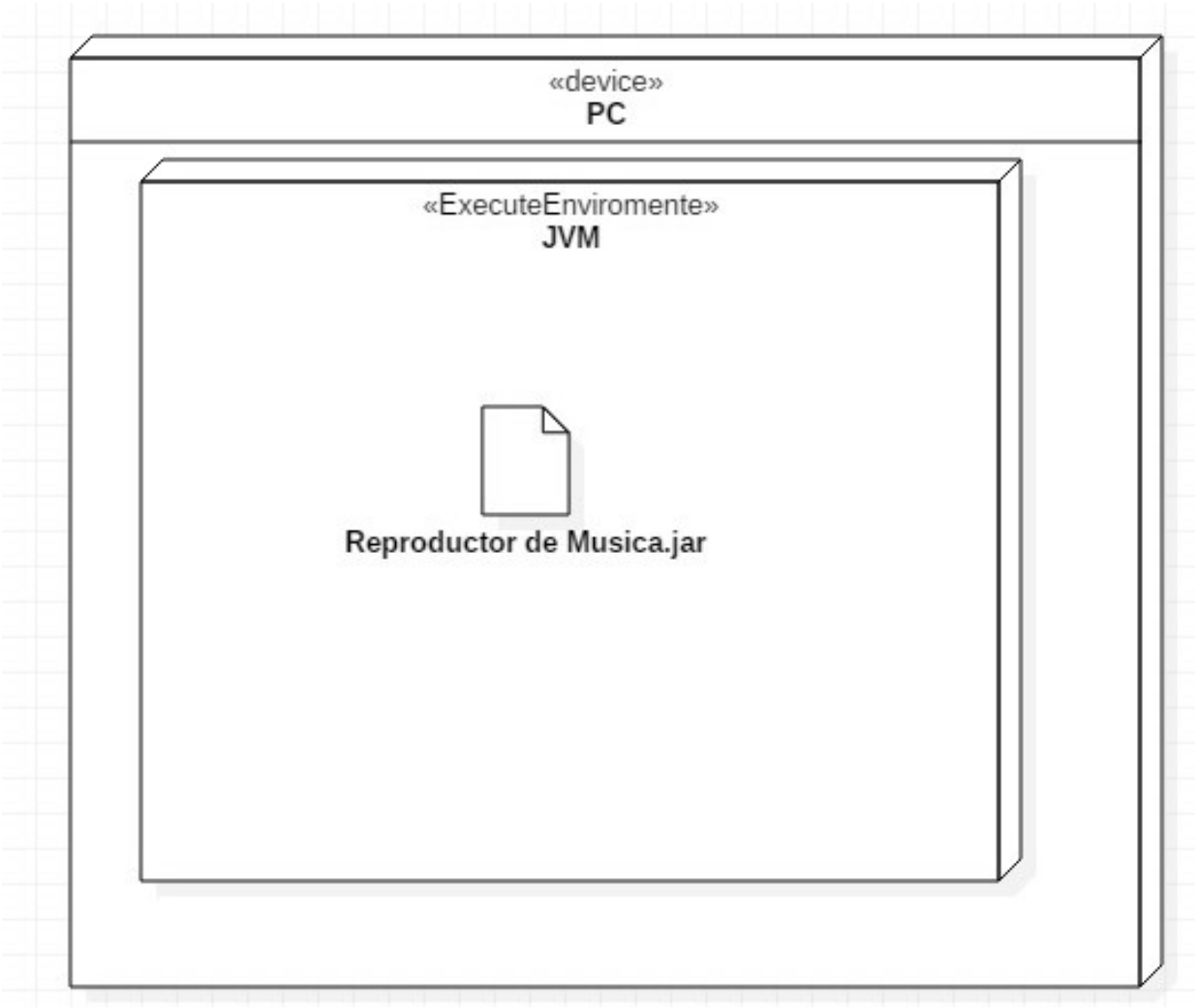
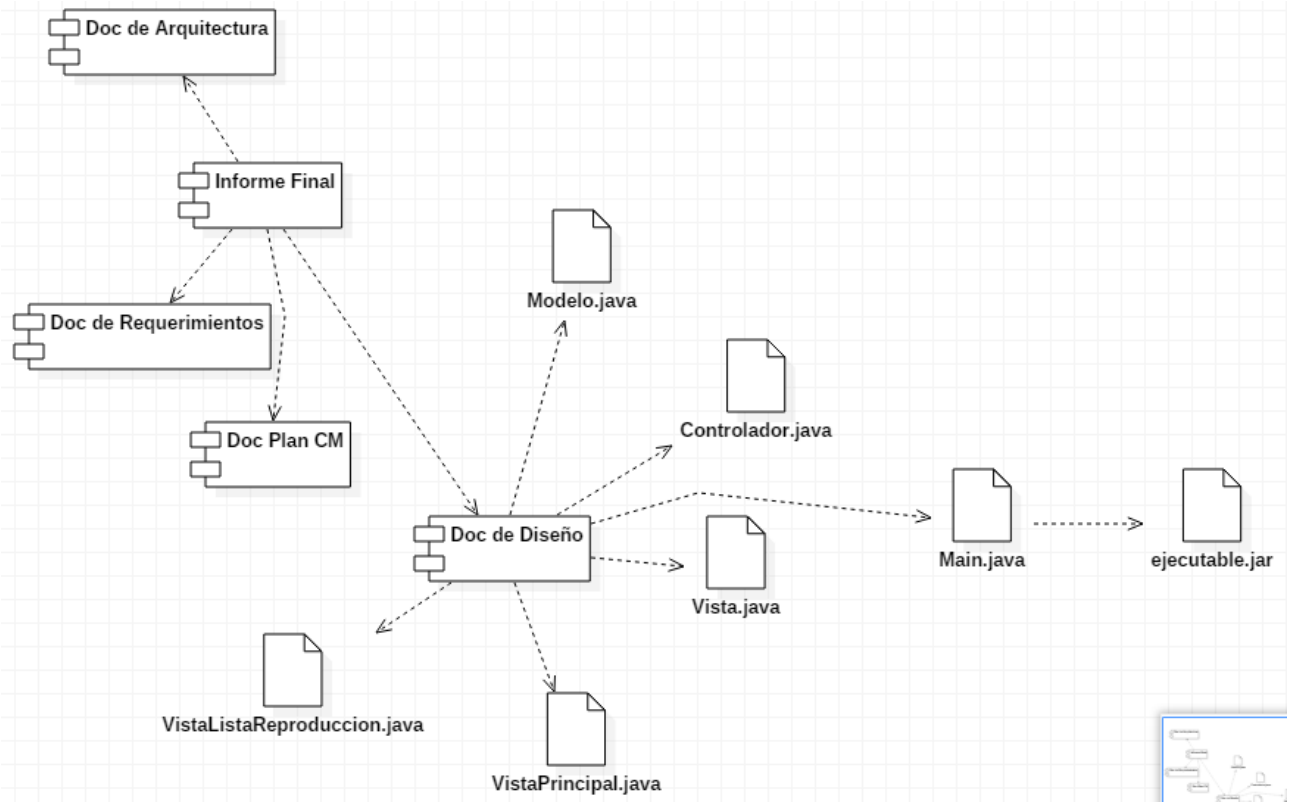


Diagrama de Componentes



6 Diseño e Implementación

En el siguiente apartado mostraremos los diagramas que conforman el diseño del proyecto.

Diagrama de paquetes

A continuación podemos observar el diagramas de paquetes, este esta compuesto por los paquetes para cada tipo de clases, y las librerías que estos utilizan. Dentro del paquete vistas se encuentran las clases Vista, VistaListaReproduccion y VistaPrincipal.

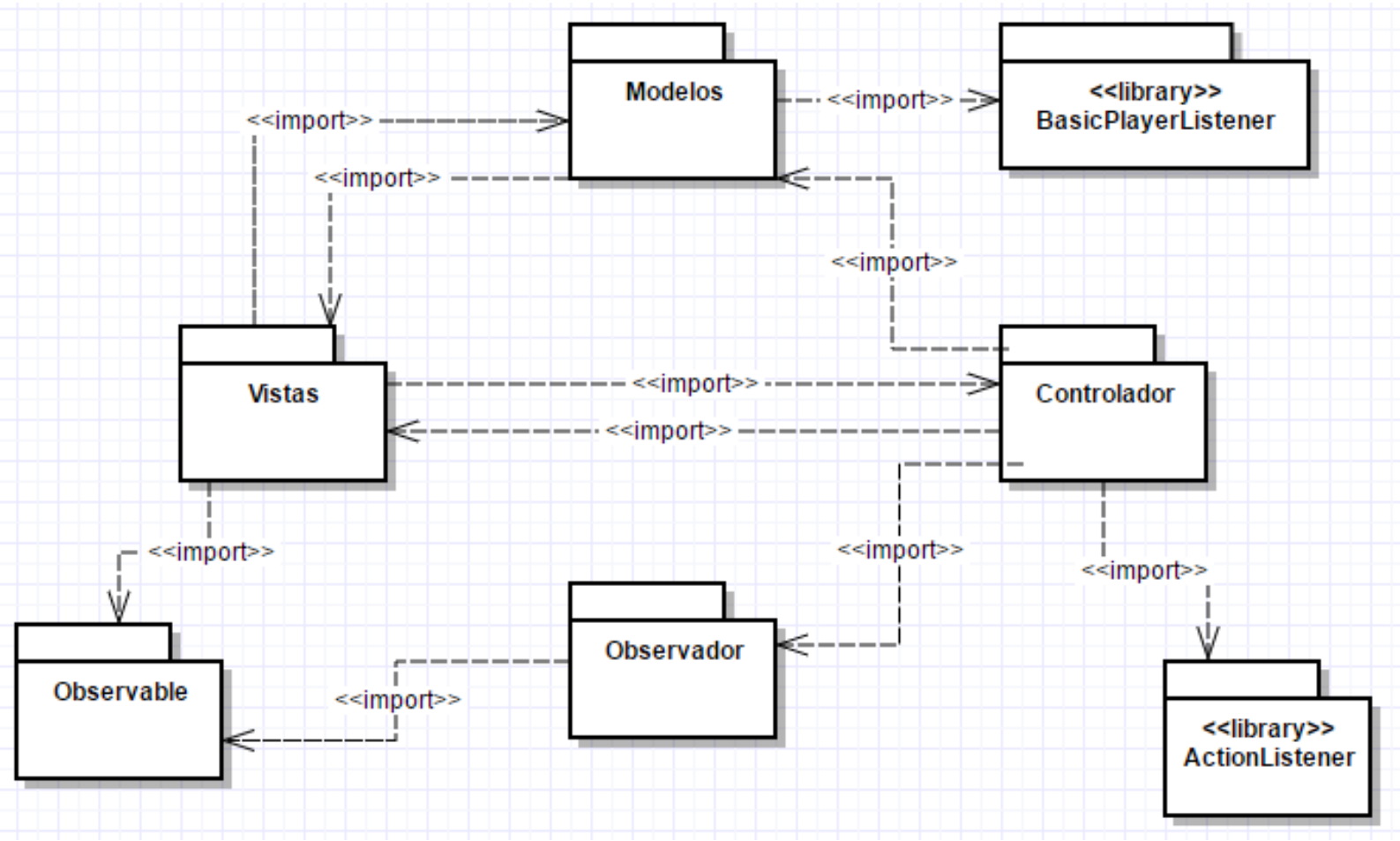


Diagrama de clases

La siguiente imagen nos muestra el diagrama de clases, el cual contendrá las clases, Main, la clase principal, modelo, vista, Observador, vistaListaReproduccion, vistaPrincipal.

Diagrama de clases simplificado

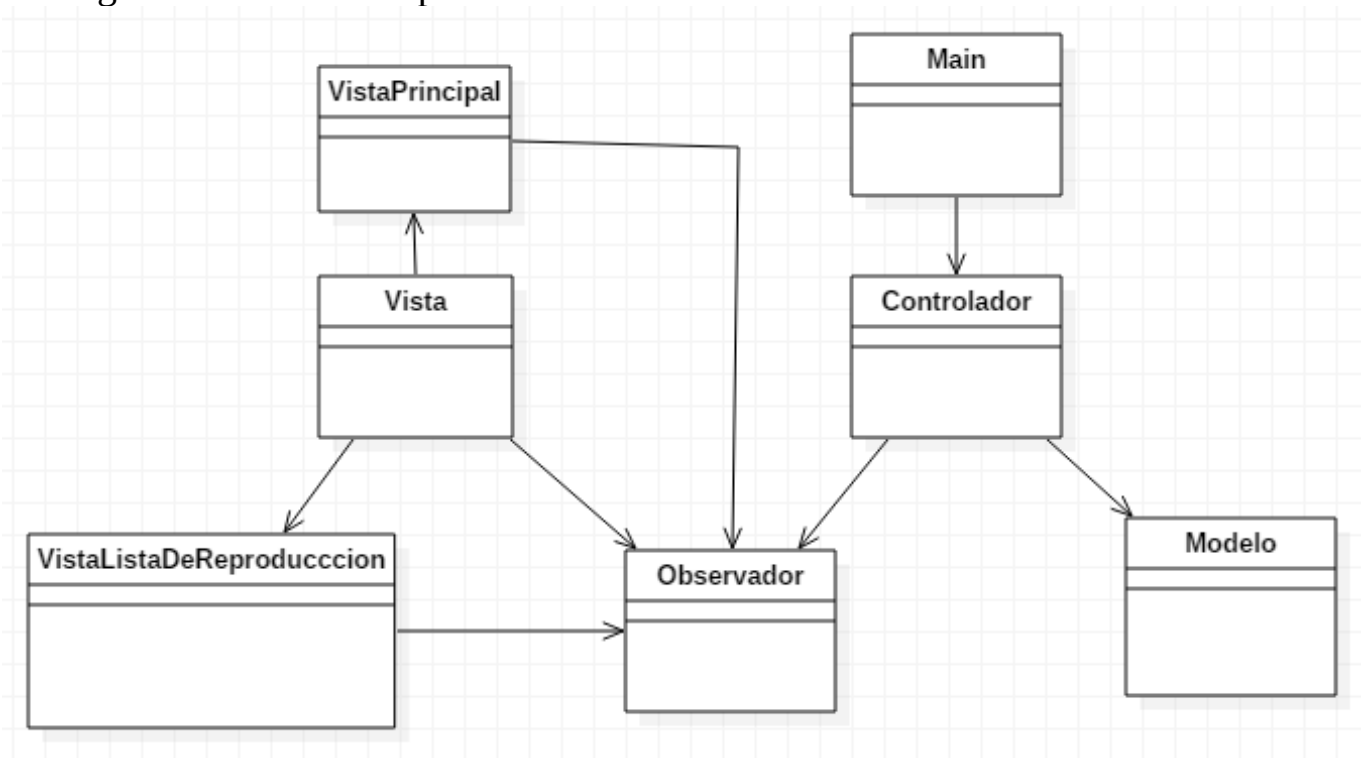
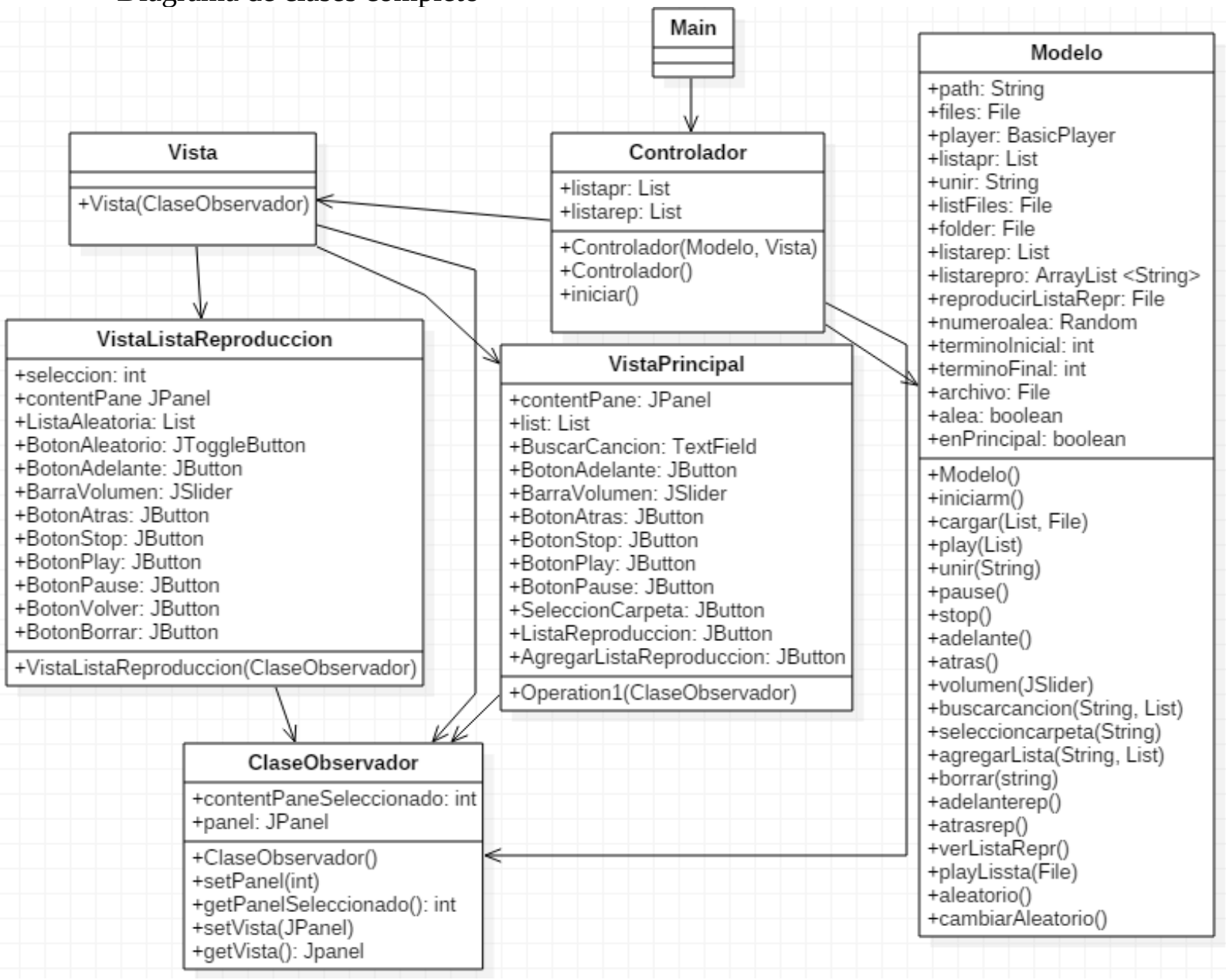


Diagrama de clases completo



Diagramas de secuencia

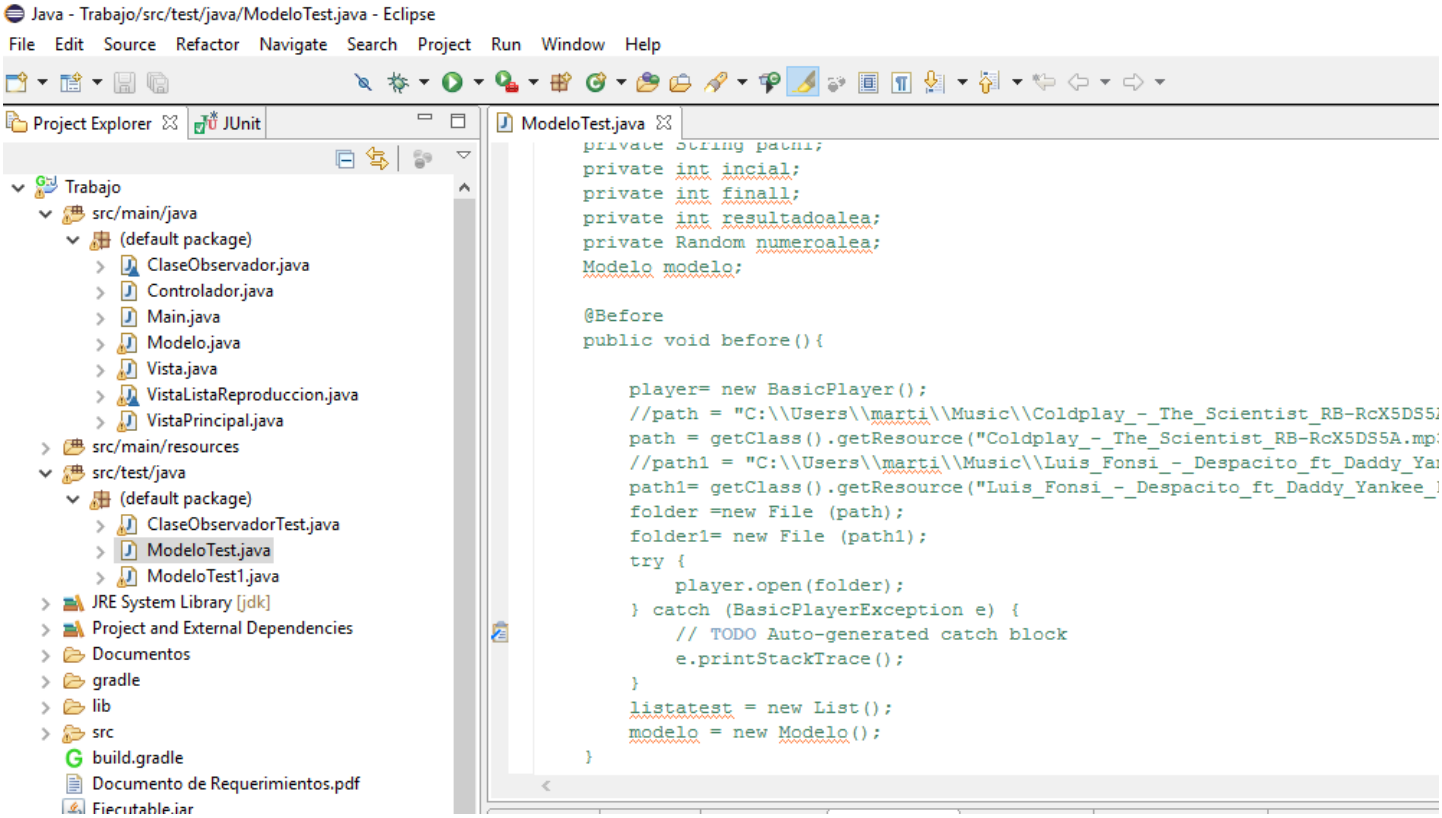
En la siguiente imagen podremos observar las primeras acciones que hace el programa.

7 Pruebas Unitarias y del Sistema

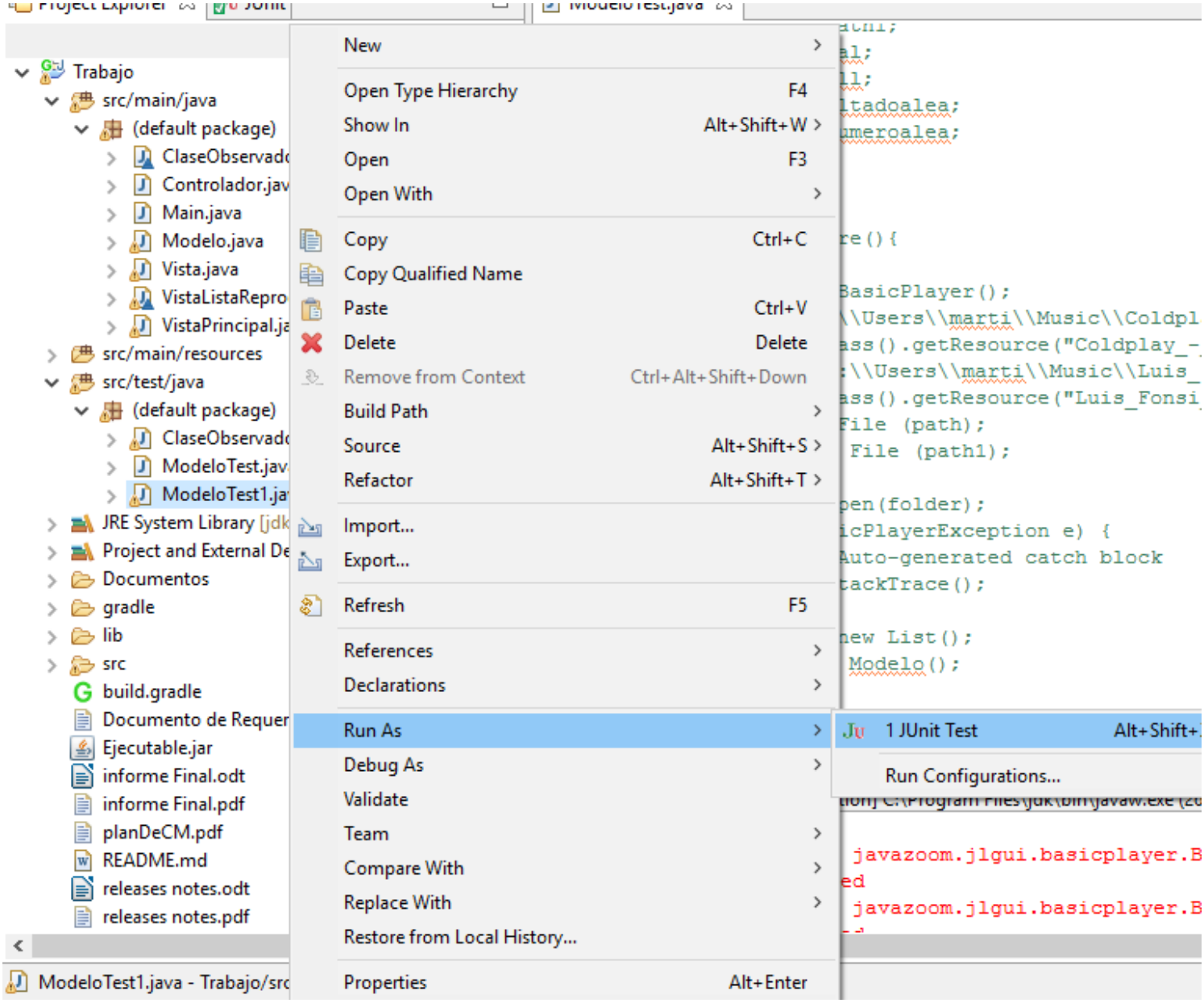
Se generaron pruebas unitarias automáticas basadas en JUnit. Las mismas contribuyen a verificar el correcto funcionamiento de las clases de nuestro sistema. Se realizaron test a las clases Modelo y Observador.

Los pasos para correr los test son los siguientes:

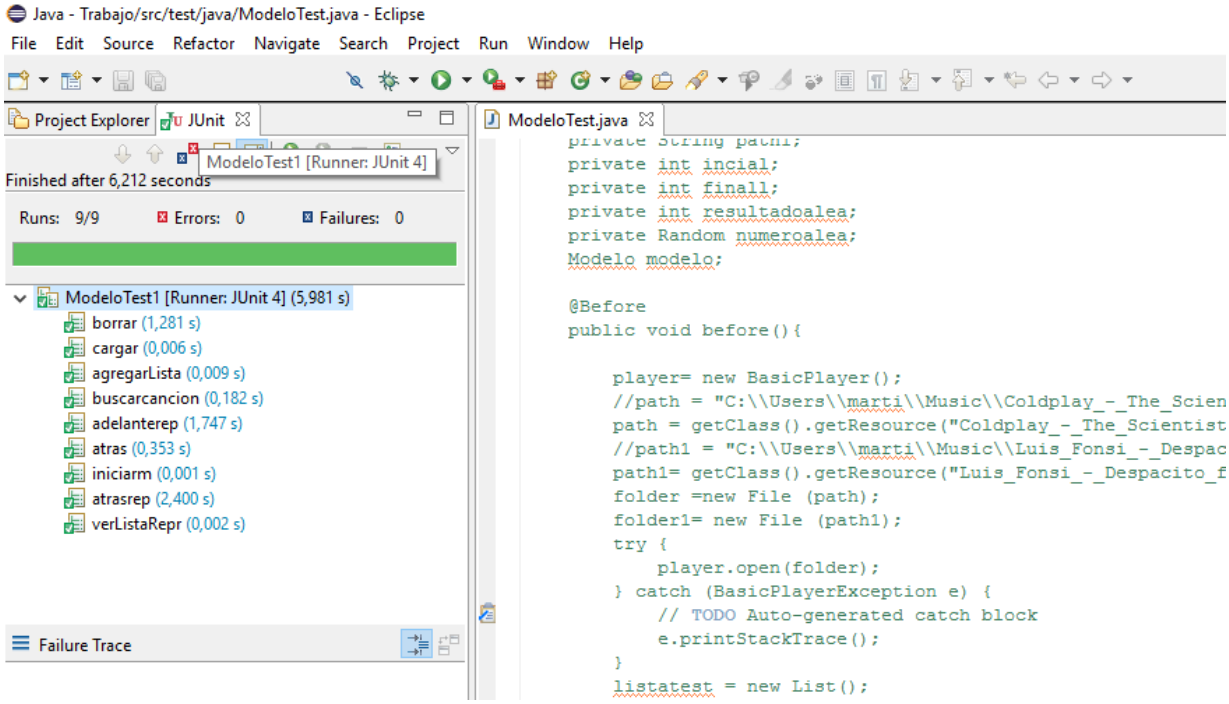
1. Importar el proyecto en cualquier IDE como Eclipse o Intellij.
2. Buscar las clases test en el explorador.
3. A la clases que se quiera probar abrirla.



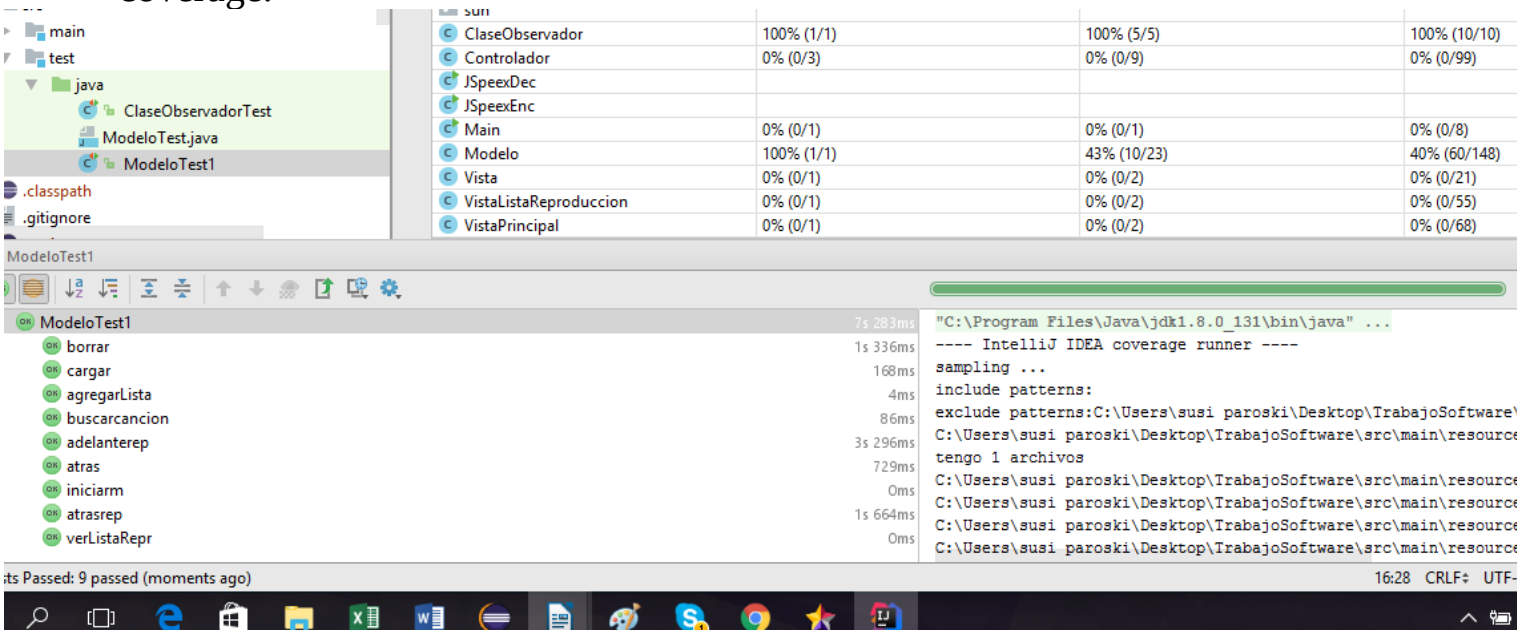
4. Correr la clase como JUnit. Darle click derecho sobre la clase test a probar y en Run As elegir JUnit Test.



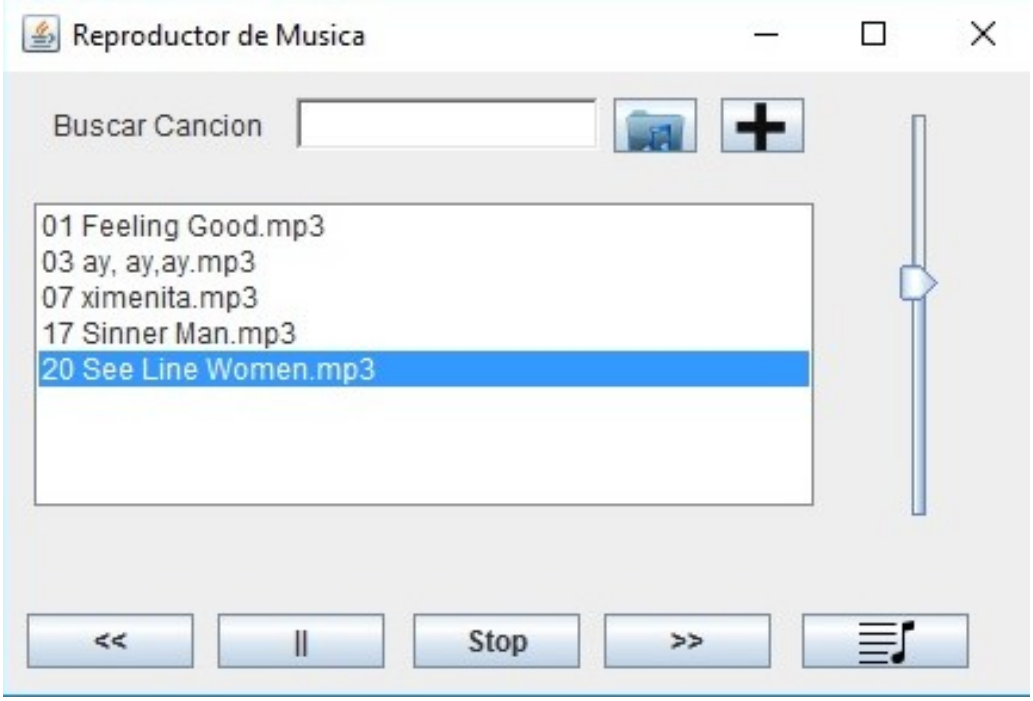
5. Hacer click sobre la pestaña junit y en cuestión de segundos apareceran los test corridos.



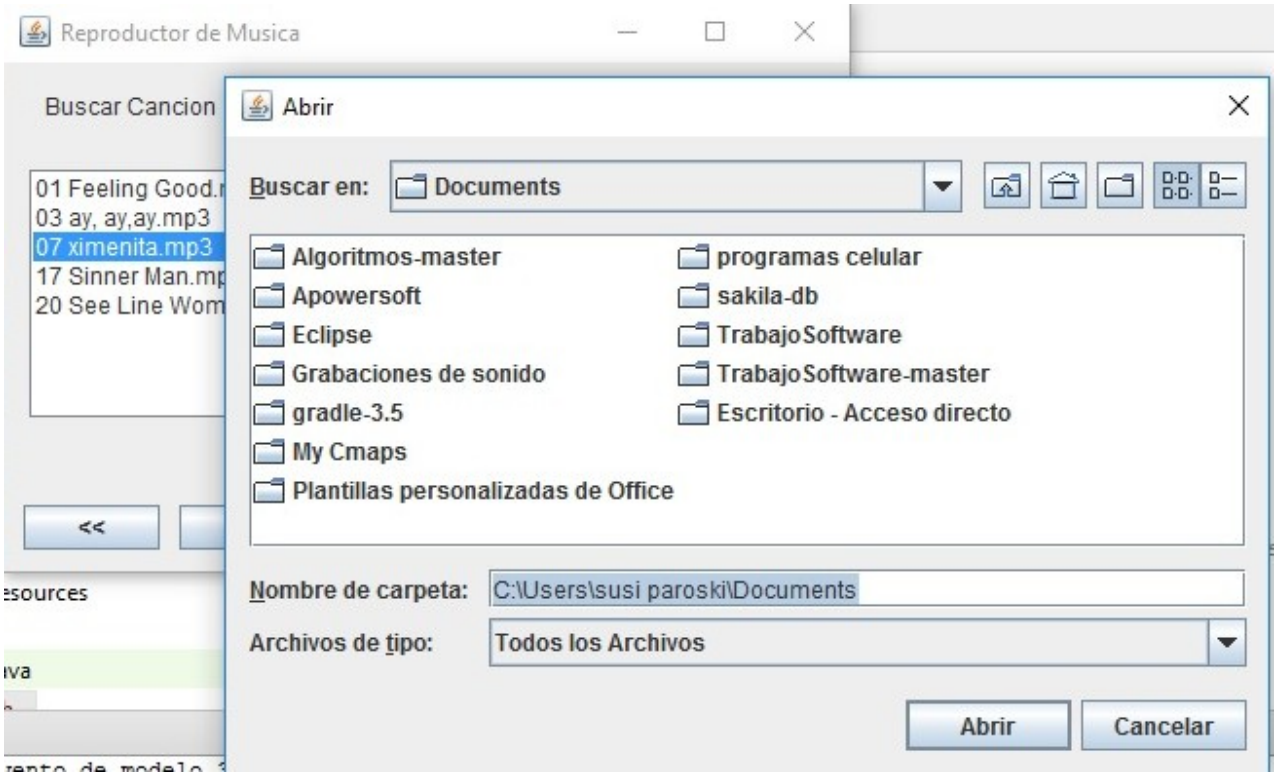
En la siguiente imagen se muestran las pruebas corridas en Intellij Idea con coverage.



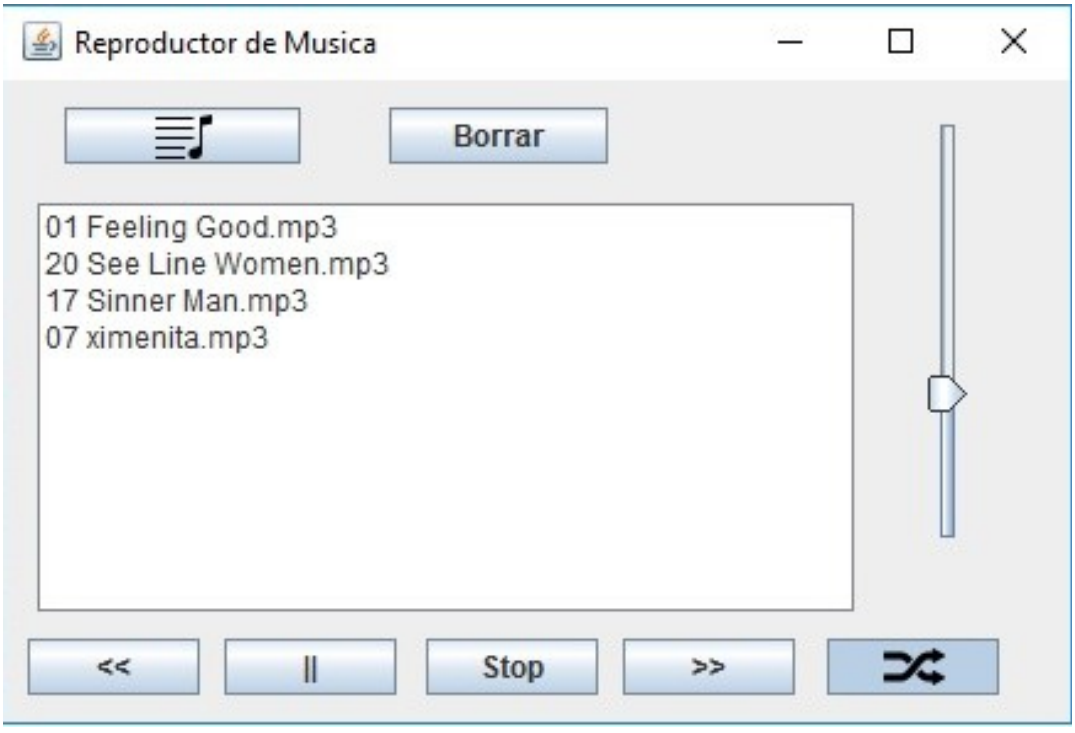
A continuación se mostrarán unas capturas de la aplicación en tiempo de ejecución.
Vista de todas las canciones de la carpeta en el reproductor.



Vista de busqueda de carpeta de pistas de audio.



Vista lista de reproduccion



8 Datos Históricos

Para poder realizar este trabajo tuvimos que debatir varios temas, buscar información sobre otros temas que no estábamos al tanto. Discutir ideas, y optar por la mas conveniente, decidir acciones, testear lo codificado, documentar todo lo realizado, algo se torna un poco tedioso cuando es mucho el material.
El esfuerzo y tiempo invertido por cada integrante fue el siguiente.

Diseño: 10 horas.
Codificación: 24 horas.
Testing: 8 horas aprox.
Documentación: 12 horas.

El esfuerzo personal por cada integrante fue:
Gasparini, Roman: 22 hs.
Martin, Tarres: 32 hs

9 Información adicional

Durante la elaboración de este trabajo práctico aprendimos a trabajar en equipo en un proyecto concreto, aplicando las prácticas de ingeniería de software vistas en clase.
Aprendimos a utilizar herramientas como Intellij idea, Travis CI, GitHub, Gradle, hacer testings, etc. Tuvimos que averiguar sobre como se codificaba
Aprendimos a separar la lógica de un programa de su interfaz de usuario, para poder tener un proyecto fácilmente expandible utilizando el patron de arquitectuira MVC.
También leímos sobre patrones de diseño e integramos algunos a nuestro trabajo.
La lección más importante aprendida durante la realización del práctico fue trabajar en equipo, dividir las tareas, compartir ideas e implementarlas.
Nos equivocamos seguido al programar, cuando resolvíamos algunos problemas, paralelamente nos surgían otros.
Cuando corregiamos bugs en el programa las cosas no resultaban como se esperaba.
Además, nos dimos cuenta la dificultad y el trabajo necesario para llevar a cabo un proyecto de software, tarea que suele ser subestimada.