**Base de datos de música**

Java Proyecto Final

Roberto Alejandro Gutiérrez Guillén

4 de Noviembre de 2015

[Instituto Tecnológico Autónomo de México](https://www.itam.mx/en/)

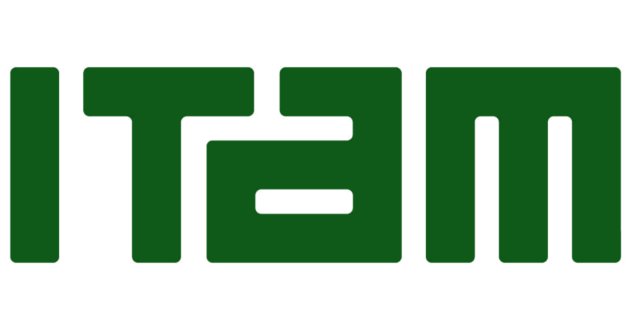
156001

Tabla de Contenido

[Análisis: 2](#_Toc436989126)

[Diseño: 3](#_Toc436989127)

[Descripción de la solución: 3](#_Toc436989128)

[Diagramas UML: 3](#_Toc436989129)

[Cancion 3](#_Toc436989130)

[Artista 4](#_Toc436989131)

[Musica 4](#_Toc436989132)

[MusicaControlador 5](#_Toc436989133)

[ControladorVistaSecundaria 6](#_Toc436989134)

[MusicaVista 6](#_Toc436989135)

[VistaSecundariaAgregarYBorrar 7](#_Toc436989136)

[Manual de usuario: 8](#_Toc436989137)

[Conclusión 8](#_Toc436989138)

# Análisis:

Un grupo de ingenieros en audio de una compañía discográfica quiere implementar una base de datos, la cual debería contener información del artista que incluye su lista de canciones y su nombre. Además cada canción requiere tener guardada su nombre, álbum, duración, velocidad de bits (bit rate) y género. La compañía discográfica requiere que las canciones del artista estén ordenadas alfabéticamente y que toda su información aparezca en una interfaz de usuario, incluyendo que usuario está usando el programa y de que artista son las canciones. El programa debe poder dejar al usuario agregar artistas y canciones, así como dejarlo borrarlas. También, le debe ser posible al usuario buscar canciones y artistas en la interfaz de usuario.

# Diseño:

## Descripción de la solución:

Para solucionar el problema se tienen que crear dos listas de arreglos: la primera va a guardar las canciones del artista en el artista como atributo y como el cliente especifico debe de ser ordenada para que se acomoden las canciones en orden alfabético; la segunda puede ser desordenada ya que el cliente no especifico y va a contener a los artistas dentro de la clase principal música. Para lograr una interfaz de usuario útil, que sea simple para que no abrume al usuario y que pueda ser usada en una variedad de resoluciones

## Diagramas UML:

|  |
| --- |
| Cancion |
| * nombreC: String * album: String * duracion: int * genero: String * bitRate: int |
| * Cancion() * Cancion(String) * Cancion(String, String, int) * Cancion(String, String, int, String, int) * getNombreC(): String * getAlbum(): String * getDuracion(): int * getGenero(): String * getBitRate(): int * hashCode(): int * toString(): String * equals(Object):boolean * compareTo(Cancion): int |

|  |
| --- |
| Artista |
| * nombreA: String * canciones: ListaArregloOrdenada<Cancion> |
| * Arista() * Artista(String) * numCanciones():int * agregaCancion(String, String, int, String, int): boolean * borrarCancion(Cancion): String * borrarCancion(String): String * promDuracionCancion(): double * getCancion(int): String * getCancionToString(String): String * getCancion(String): String * getNombreA():String * setNombreA(String) * hashCode(): int * equals(Object): boolean * compareTo(Artista):int * toString():Stirng |

|  |
| --- |
| Musica |
| * usuario: String * artistas: ListaArregloDesordenada<Cancion> |
| * Musica() * Musica(String) * getUsuario(): String * getArtista(int): String * getArtistaClase(String): Artista * getArtista(String): String * agregaArtista(String): boolean * borrarArtista(Artista): String * borrarArtista(String): String * agregaCancion(String, String, String, int, String, int): boolean * borrarCancion(String, String): String * listaArtistas(): String * hashCode(): int * equals(Object): boolean * compareTo(Musica):int * toString():Stirng |

|  |
| --- |
| MusicaControlador |
| * m: Musica * serialVersionUID : long |
| * MusciaControlador(String, Musica) * EsuchadorBuscArt * EsuchadorBuscCan * EscuchadorBtn * EscuchadorRefreshBtn * lecInicial(): |

|  |
| --- |
| ControladorVistaSecundaria |
| * m: Musica * serialVersionUID : long |
| * ControladorVistaSecundaria(String, Musica) * EscuchadorAggCan * EscuchadorBorrCan * EscuchadorAggArt * EscuchadorBorrArt |

|  |
| --- |
| MusicaVista |
| * serialVersionUID : long * scroll: JScrollPane * musicaLabel : JLabel * buscArtLabel : JLabel * buscCanLabel : JLabel * label1Btn1 : JLabel * label2Btn1 : JLabel * musicaInfoArea : JTextArea * artResBuscArea : JTextArea * canResBuscArea : JTextArea * buscArtField : JTextField * buscCanField : JTextField * btn : JButton * refreshBtn : JButton |
| * MusicaVista(String) |

|  |
| --- |
| VistaSecundariaAgregarYBorrar |
| * serialVersionUID : long * nombreATituloLabel : JLabel * cancionTituloLabel : JLabel * artistaLabel : JLabel * nombreACanLabel : JLabel * nombreCLabel : JLabel * albumLabel : JLabel * duracionLabel : JLabel * generoLabel : JLabel * bitRateLabel : JLabel * agCBtnLabel1 : JLabel * agCBtnLabel2 : JLabel * borrCBtnLabel1 : JLabel * borrCBtnLabel2 : JLabel * agABtnLabel1 : JLabel * agABtnLabel2 : JLabel * borrABtnLabel1 : JLabel * borrABtnLabel2 : JLabel * scroll: JScrollPane * nombreATxt : JTextField * nombreAParaCTxt : JTextField * nombreCTxt : JTextField * albumTxt : JTextField * duracionTxt : JTextField * generoTxt : JTextField * bitRateTxt : JTextField * agregarCBtn : JButton * agregarABtn : JButton * borrarCBtn : JButton * borrarABtn : JButton * infoArea : JTextArea |
| * VistaSecundariaAgregarYBorrar(String) |

## C:\Users\Mac\Desktop\boton refresh.PNGManual de usuario:

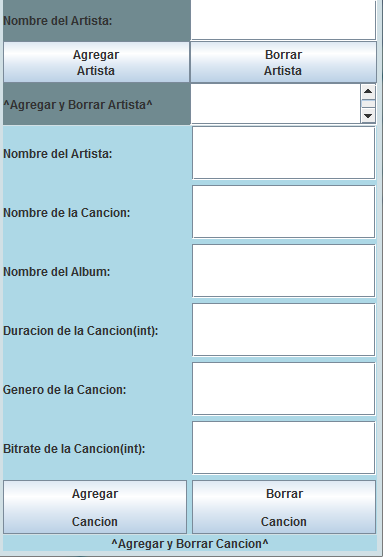
Cuando el usuario hace clic en el botón, ” refresh” el área de texto que esta abajo se actualiza con la nueva información de las canciones y los artistas, actualiza el toString de la clase principal que es Música.

C:\Users\Mac\Desktop\busqueda.PNG

El recuadro superior es para que el usuario haga búsquedas en la librería, el usuario introduce la información en la primera columna de texto y la información ya sea de la canción o del artista aparece en la segunda columna de recuadros. El programa avisa al usuario si hace falta información o si el artista o canción no existe.

C:\Users\Mac\Desktop\boton agregar.PNG

El botón “Agregar o borrar, canción o artista” que se ve arriba abre la ventana que se puede ver abajo.



La ventana que se puede ver a la izquierda es la que se abre cuando el usuario hace clic en el botón de “Agregar o borrar, canción o artista”. Cuando la ventana secundaria se cierra la información introducida sigue guardada y el usuario puede seguir haciendo uso del programa. En el recuadro con una barra de desplazamiento es en donde aparece si la alta fue exitosa o no. El programa se encarga de que el usuario introduzca toda la información necesaria para agregar una canción o artista, en caso de que el usuario no introduzca todo el programa le avisara al usuario en el recuadro y no se cerrará.

Conclusión:

El programa claramente cumple los requerimientos de la compañía discográfica, hace uso de dos listas de arreglos una ordenada y otra desordenada, además tiene una lectura de archivo al inicio del programa. Adicionalmente, el programa tiene una interfaz fácil de usar que se compone de dos vistas: la principal en donde se muestra toda la información de las canciones y del artista y que contiene los buscadores; la segunda, es en donde se agregan o borran canciones y artistas, el algoritmo se encarga de que el usuario introduzca la información correcta o no lo dejara continuar, esto es para evitar de que el programa se cierre por un problema.