Corso di laurea in Ingegneria e Scienze informatiche – Università di Bologna
Corso di Programmazione ad Oggetti a.a. 2015-2016

Relazione del progetto “MusicSort”

Rrok Gjinaj (0000651723)

1) Analisi del problema

(PREMESSA INIZIALE)

MusicSort è un programma stile Spotify che permetterà all' utente di avere le funzionalità principali di quest'ultima, ma l'obiettivo primario è di creare questa piattaforma in modo da permettere in più una ricerca (con criteri di ricerca inizialmente preimpostati), organizzazione e riproduzione musicale delle canzoni all' interno del proprio pc offrendo una buona UX .

L'obiettivo futuro è raccogliere più informazioni possibili riguardo i gusti musicali per poterli condividere con gli amici e migliorare l'esperienza di ascolto, inizialmente in locale. Con le informazioni raccolte, questa piattaforma dovrà essere in grado di interconnettere gli utenti che ascoltano lo stesso genere e selezionare le canzoni preferite dei singoli utenti da consigliare a coloro che potrebbe piacere dello stesso genere.

Funzionalità Implementate:

- Creazione di playlist
- Ricerca di canzoni all’ interno del programma tramite la navigazione delle playlist al click
- Aggiungere una canzone in una playlist a scelta in quelle presenti
- rimuovere una canzone da una playlist
- Play/pause/next/prev di una canzone in riproduzione
- apertura di una canzone posizionato nel pc
- visualizzare canzoni di playlist differenti
- rimozione di Playlist con tutte le canzoni all’interno

Features:

- Dare una valutazione ad ogni canzone da 1 a 5 (con una stella) //non implementato

- Gestione utenti con login(collegato alle features) // non implementato

- Aggiungere un insieme di canzoni contenute in una cartella //non implementato

- Usare api spotify

- Condividere le canzoni con gli amici

- Gestione utenti dal web

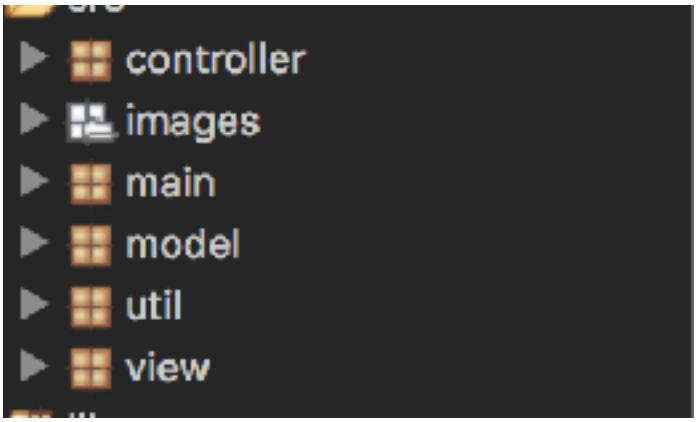
- Consigliare playlist/canzoni di utenti usando criteri: ○ indice di valutazione di ogni utente ○ tempo di ascolto di una canzone con il genere che ascolti

- Condivisione nei social delle proprie preferenze musicali

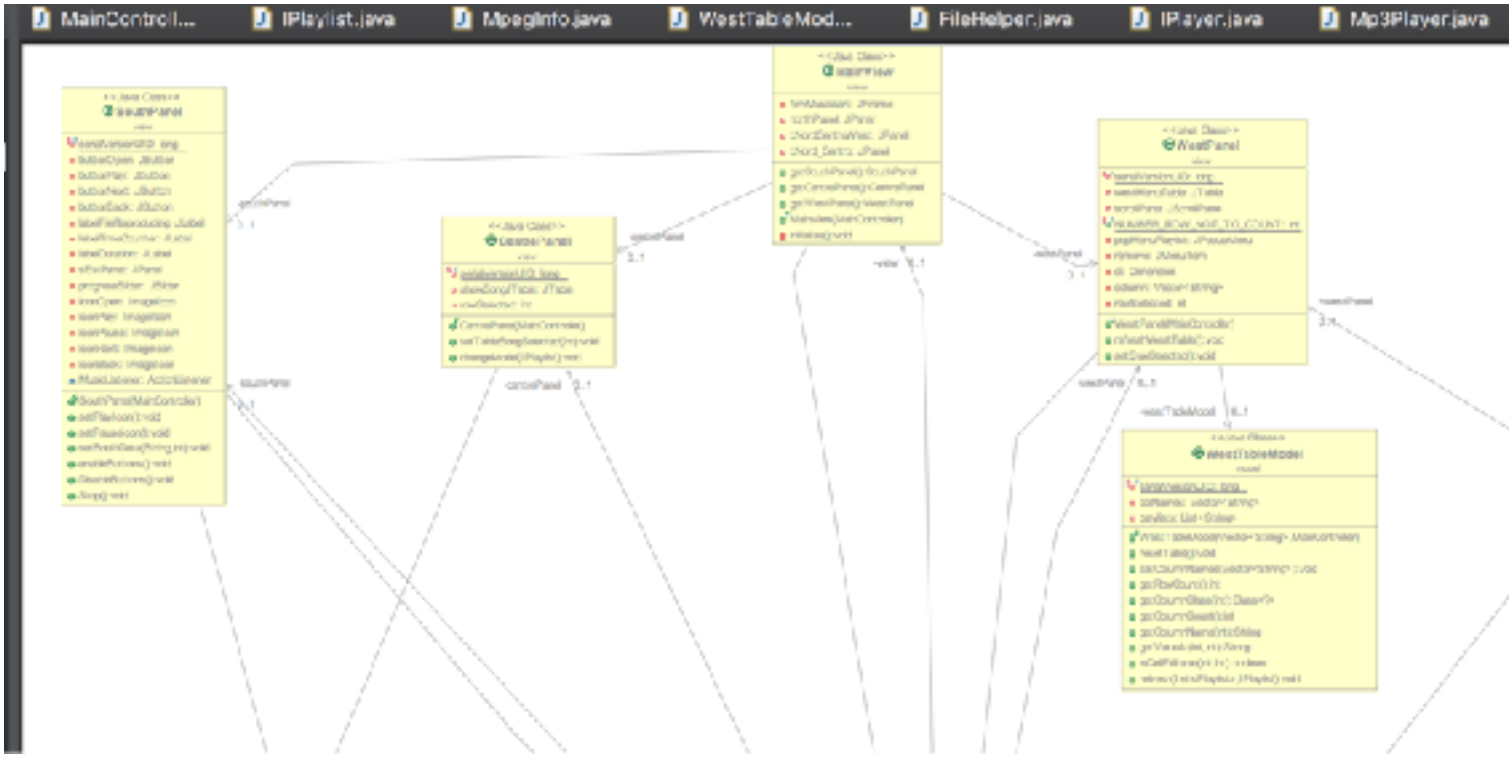
- Ordinare/cercare tutte le canzoni all' interno del programma con criterio: ○ durata ○ data aggiunta ○ tempo di ascolto ○ stelle attribuite ○

2) Organizzazione dei packages

I package sono stati suddivisi tenendo conto del pattern MVC. Sono stati creati 6 packages principali per “view”, “controller” e “model” e ulteriori packages per separare le differenti funzionalità dell’aplicazione.



View



E’ stata creata una mainView la quale contiene tutti pannelli come viene mostrato nell’immagine sopra idicata.

Poiché la view contiene più pannelli, essi sono saltati inseriti dentro delle classi apposite ma istanziati dalla view principale MainView che di base crea la finestra che conterrà gli elementi dichiarati e sette le impostazioni di essa.

Dentro questa view, per offrire il resizing automatico, grazie a BorderLayout sono stati dei contenitori per inglobare i pannelli suddivisi

Il concetto seguito è il seguente:

Il pannello sud è quello effettivo mentre è stato creato un pannello nordwestCenter che contiene i pannelli del titolo stesso

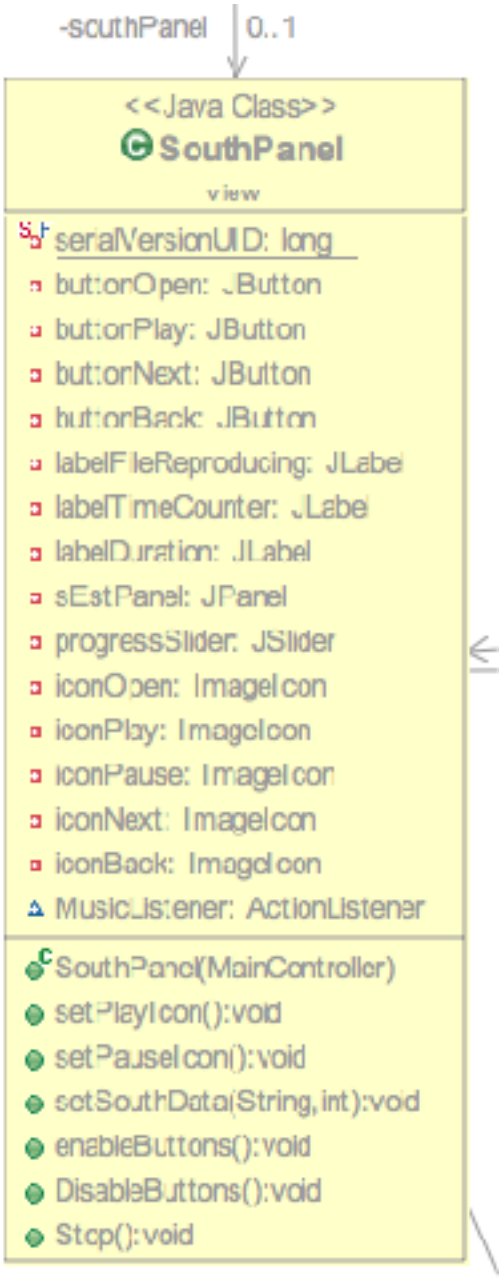
il pannello west è stato salvato nella BorderLayout.WEST mentre nel centro è stato creato un'altro contenitore che contiene il pannello nord posizionato on BorderLayout.NORD e il center panel BorderLayout.CENTER

Questa soluzione permette di avere sotto controllo qualsiasi componente ed eventualmente l'aggiunta di nuove future senza toccare gli altri componenti

L'implementazione è consistita nella divisione più possibile dei moduli della view in pannelli ogni uno con funzionalità differenti in modo da essere riusata o modificati senza interferire con gli altri.

I pannelli principali sono :

> South panel



Questo pannello mette a disposizione i bottone play, pause, next e prev
i quali inizialmente partono spenti per far scegliere all'utente quale canzone riprodurre.
All'interno del south panel viene utilizzato un BorderLayout per posizionare gli elementi all'interno

così permettendo all'utente il resizing della view senza perdere la qualità della del pannello.

Viene usato una seek bar e quindi essendo necessario il refresh della stessa, questo compito viene delegato ad un thread che si occupa del refresh in modo indipendente usando

```
SwingUtilities.invokeAndWait(new Runnable() {
                        @Override
                        public void run() {
                            southP.revalidate();
```

Il thread della progress bar viene inizializzato dal south panel e poi in base ai dati che gli fornisce il controller, imposta i suoi dati necessari

ProgressBarThread

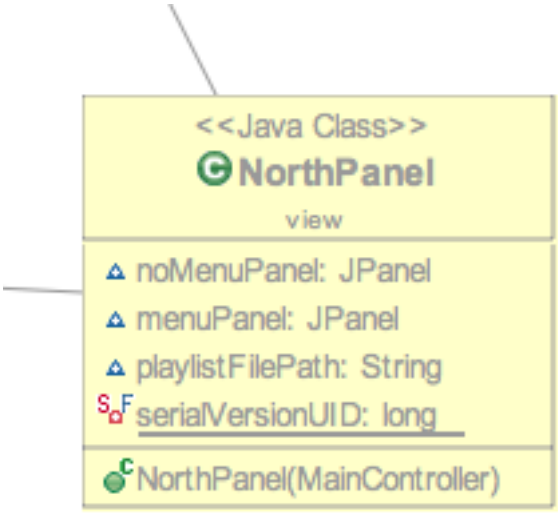
La progress bar sfrutta un dato fornito da un'altro thread che si occupa solo di incrementare un valore ogni secondo, utile per tenere traccia dei secondi che passano e quindi aggiornare la posizione del della seek bar. Questa soluzione non elegante ma funzionale, è stata necessaria poiché il basicPlayler forniva un dato sulla posizione corrente della canzone irregolare e non stabile compromettendo il funzionamento della bar stessa.

Infatti la fine della canzone, il controller Principale(main Controller) la verifica grazie a questi due thread i quali hanno il compito di mantenere traccia della posizione attuale della canzone, ridisegnare la seekbar ed aggiornare i dati della stessa chiedendo al controller quali siano i nuovi dati

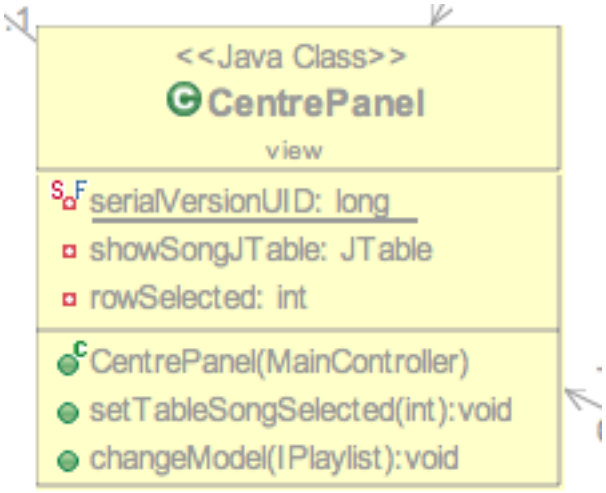
Il pannello sud mette a disposizione alcuni metodi che permettono al controllerò di settore la view in caso di necessità

> North panel

Questo panello risulta importante poiché permette di creare una nuova playlist e in futuro cercare canzoni o ordinare canzoni con criterio.
Attualmente quello con minori funzionalità ma che acquisirà importanza quando verrano implementate le nuove funzionalità.



>Centre panel



E' il pannello più importante dove vengono visualizzate tutte le canzoni della coda(recenti) oppure le canzoni della playlist che si sta ascoltando
In questo pannello sono state messe a disposizioni delle funzioni basi come il doppio click sulla riga della tabella che corrisponde ad una canzone per mandarla in riproduzione.
Questo pannello da la possibilità di aggiungere alla playlist, tra quelle disponibili, con il click destro oltre che la riproduzione.
La view per fare queste operazioni va a richiamare ovviamente le funzionalità messe a disposizione dal controller.

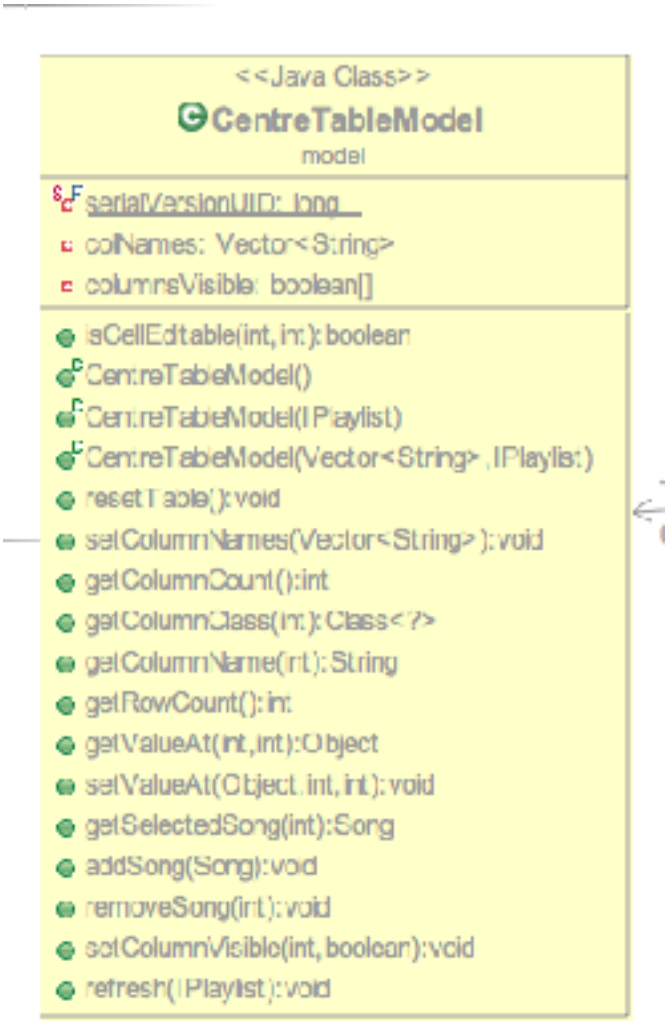
Vengono sfruttate le opzioni della AbstractTableModel dove impostando come modello della JTable showSongJTable si hanno a disposizione funzioni come l'aggiunta di una canzone e l'eliminazione di essa semplicemente richiamando fireTableRowsDeleted e richiamando il metodo fireTableChanged.

Quest'impostazione permette di avere anche più future da implementare in seguito.

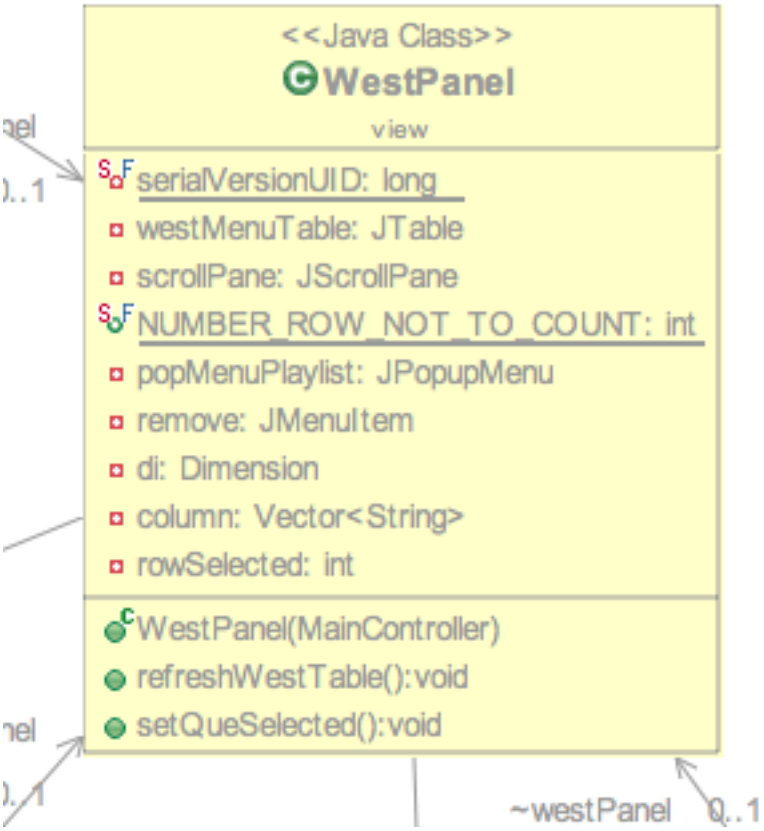
Il click destro viene gestito da un popUP menu gestito da una classe esterna dove una volta catturato l'evento del MouseAdapter viene passato al JPopupMenu.

Ad il click del popup demo richiama una classa apposta listenerAddToPlaylist alla quale passiamo rowSelected, cioè la canzone, il controller e il testo per individuare a quale playlist aggiungere la canzone

Il modello della JTable viene gestito da una tabella a parte la quale ha le sue funzioni che permettono l'aggiornamento, inserimento e tutte le funzioni messe a disposizione dalla abstract Table model. La classe che estende L'abstractTableModel viene implementa i metodi seguenti:



>West Panel



Questo pannello ha il compito di visualizzare le playlist dove con il click su di essa, mostra le canzoni presenti nel pannello centrale le canzoni presenti dentro quella playlist, usa l'ottica del pannello centrale ma a differenza di esso, per il modello viene sfruttata la classe anonima del DefaultTableModel creata all'interno della classe del pannello per capire le differenze con l'AbstractTableModel

Questo pannello mette a disposizione diverse funzionalità che verranno implementate in seguito le quali

-rinominare playlist(non compleatata in accordo con MV)

-eliminare playlist

Sia il pannello centrale che questo, sono sottoposti ad un layout BoxLayout(panel, BoxLayout.X_AXIS)) i quali permettono consistenza nel resizing della tabella senza dover mettere delle altezze e larghezze preferred.

Questa soluzione offre una gui elegante ed elastica permettendo l'inserimento di un numero anche elevato di canzoni o paylist per scorrerle con la rotella del proprio mouse.

Inizialmente era stato impostato il modello della tabella con default table model ma poi per difficoltà di refresh del modello in fase di aggiornamento con nuove playlist, scoperti in fase di implementazione. E' stato scelto l'abstract table model per poter usufruire dei suoi metodi poiché avendo dei metodi base, consentiva una gestione più sotto controllo dei dati all'interno.

Sia la tabella del pannello laterale che quella centrale, usufruiscono di JPopupMenu per facilitare all'utente la l'utilizzo dell'applicazione mettendo a dizione un menu che permette la rimozione o l'aggiunta della canzone, o la rimozione della playlist.
RightClickCentreTable è la class infatti che estende JPopUpMenu la quale implementa le funzioni dette.

Nel pannello west invece non risultava necessario creare una classe a parte poiché risultava sufficiente una classe anonima per svolgere la funzione di rimozione della playlist.

Main

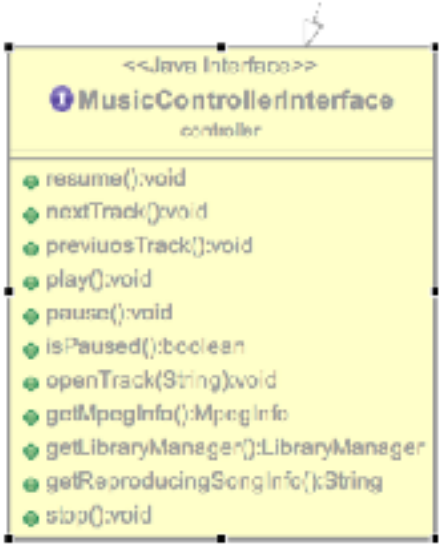
E’ stato creato un package solo per istanziare controllerò view e model chiamato main il quale mette in comunicazione questi oggetti e facendo partire la classe che a cascata richiama e istanza tutti i pannelli cioè la main view passando come argomento il controller.

Controller

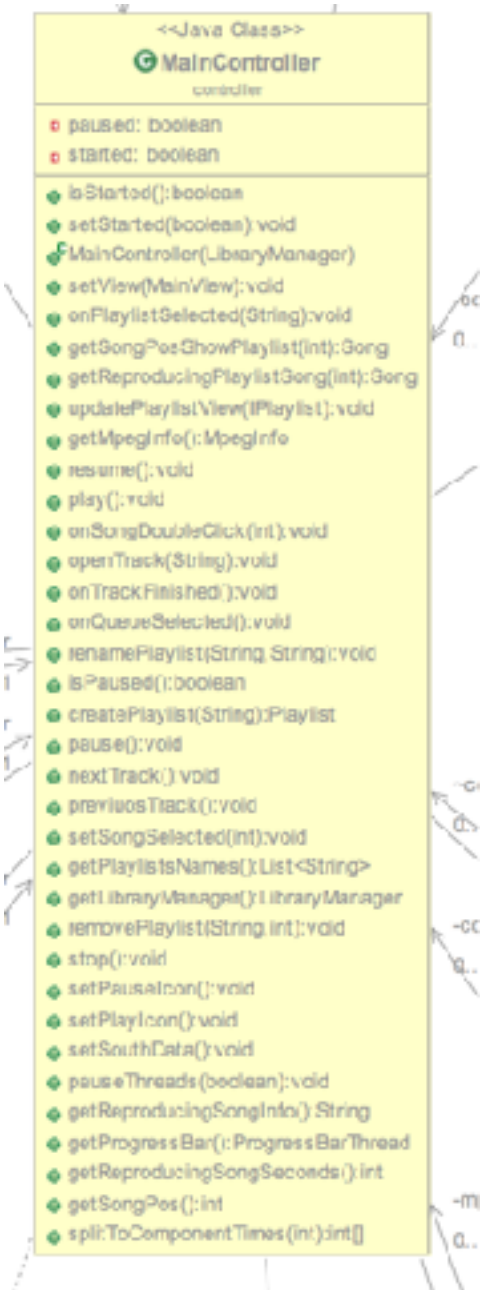
Il controller dei questa applicazione si basa perlopiù sul mainController che implementa l’interfaccia MusicControllerInterface

MusicControllerInterface:

Quest’interfaccia ha come i metodi principali di un riproduttore musicale, mostrati nella seguente foto:



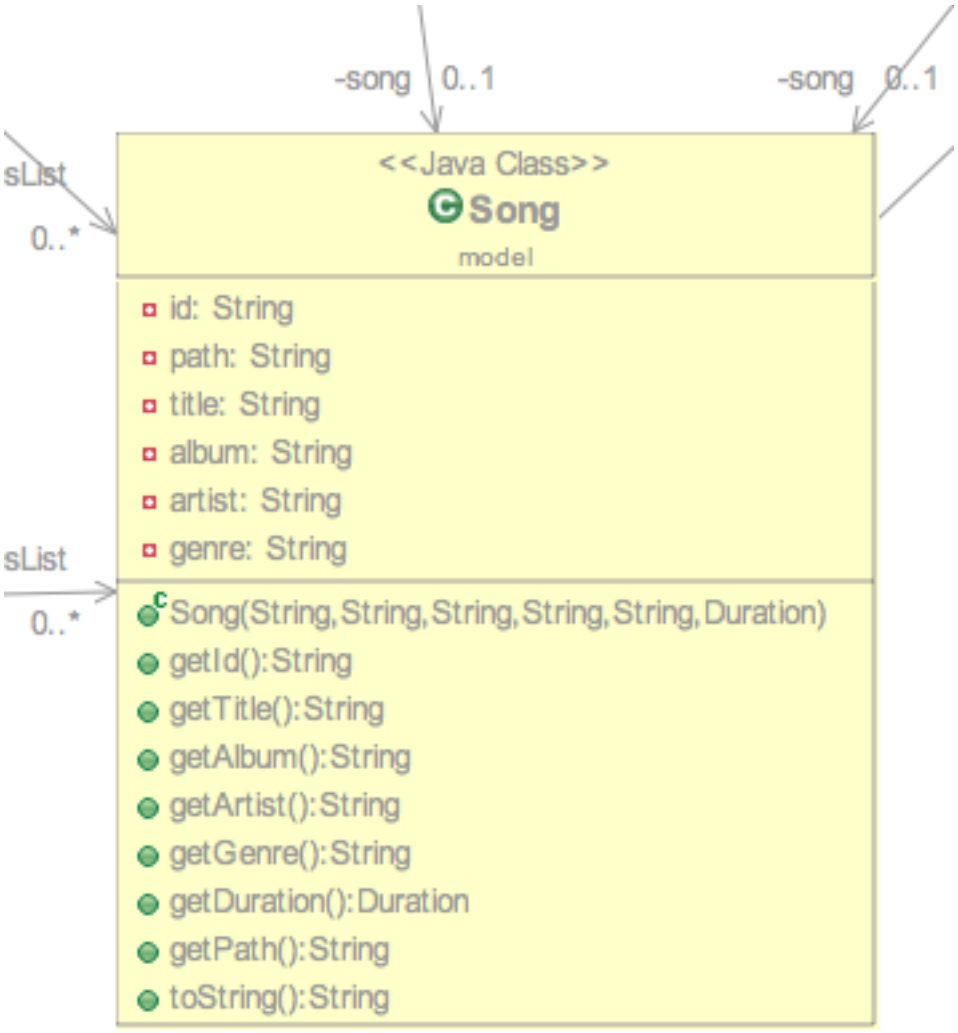
Il main controller contiene diversi metodi oltre a quelli dell'interfaccia MusicController
Ad esempio ha come l'aggiunta di una canzone, ricevere eventi dalla view e operare di conseguenza.
Essendo il controllore su cui si basa tutta l'architettura, è stato preferito per questioni anche di tempo, implementare tutti i suoi metodi al suo interno.



Si occupa di utilizzare i metodi messi a disposizione dei vari modelli per aggiornarli.
Riceve tutti i comandi dell'utente tramite la view e sfrutta i metodi messo a disposizione del Library manager.

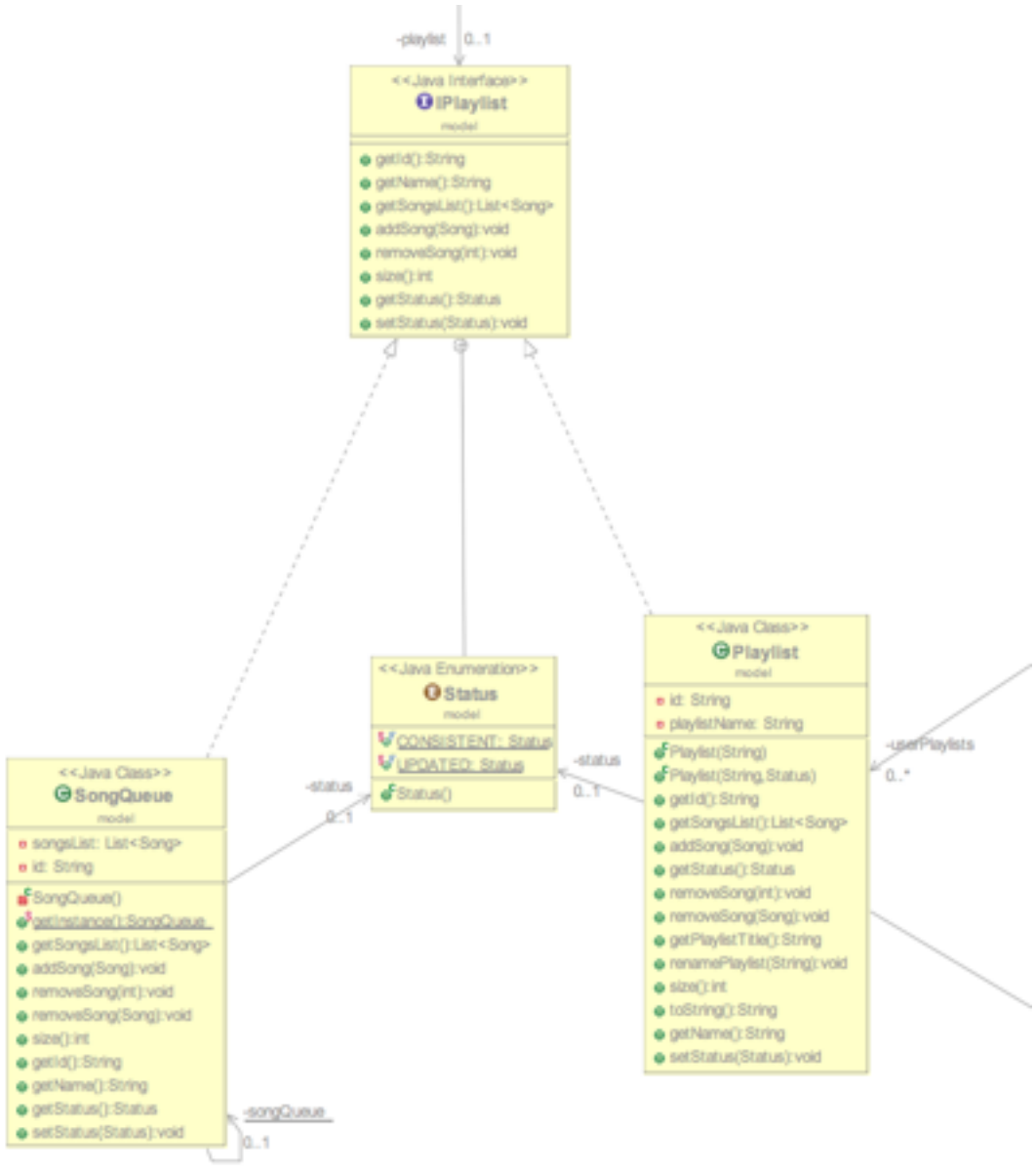
Model

Si è diviso in più sezioni.
Le classi principali del model sono ovviamente Song e Playlist le quali vengono usate dal library manager che si occupa di aggiornare il model quando il controller lo informa che compito eseguire.



Questa classe contiene i le informazioni più importanti di una canzone e è stato deciso di differenziarla runtime da un id univoco generato da java.rmi.server.UID.
In questa maniera, quando è stato necessario fare dei compare tra le canzoni delle playlist,

Playlist



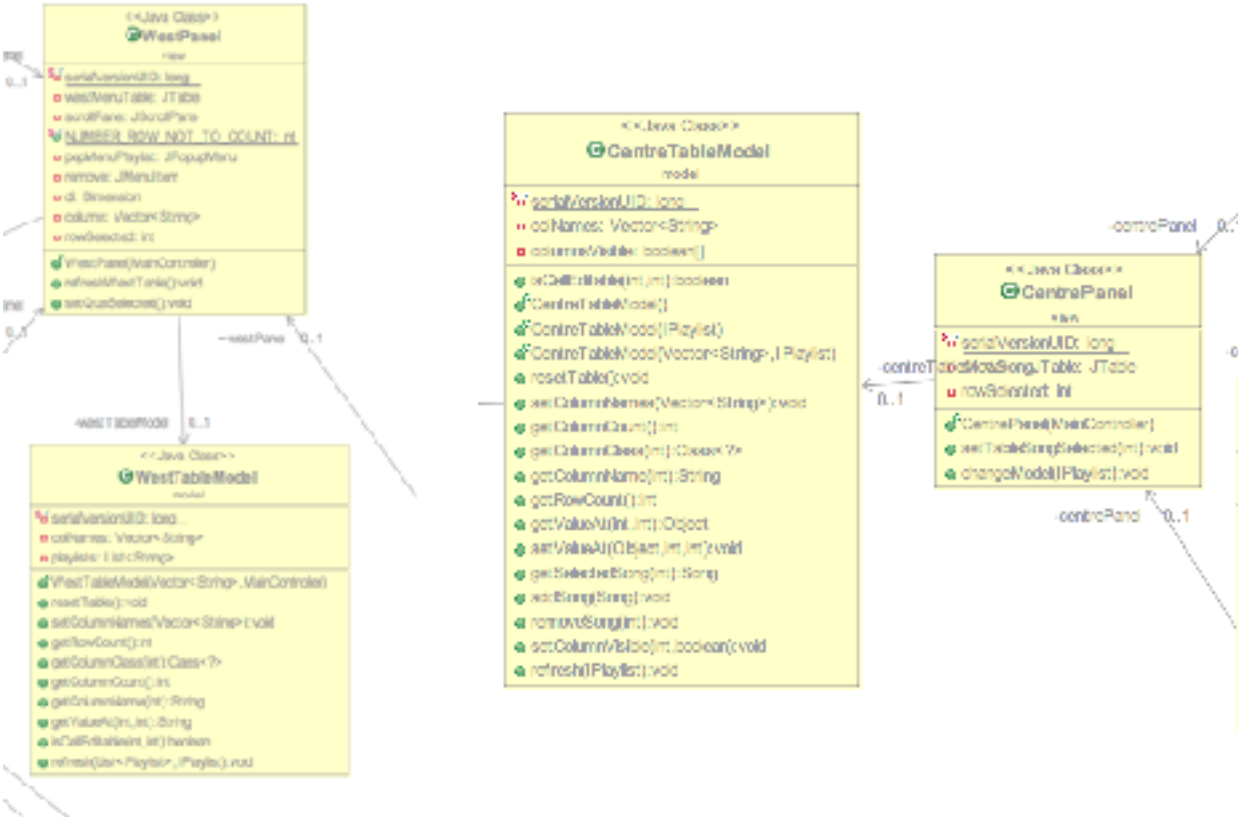
Il modello delle playlist invece viene differenziato da una playlist che ha creato l'utente e la Queue che invece è sempre una playlist ma non è eliminabile, infatti con

Eè stato usato per la queue il pattern singleton, se non è stata creata(la prima volta che si apre l'applicazione) viene creata e non è eliminabile e deve essere unica.

Playlist e SongQueue implementano tutti e due IPlaylist perché hanno tanti metodi in comune.

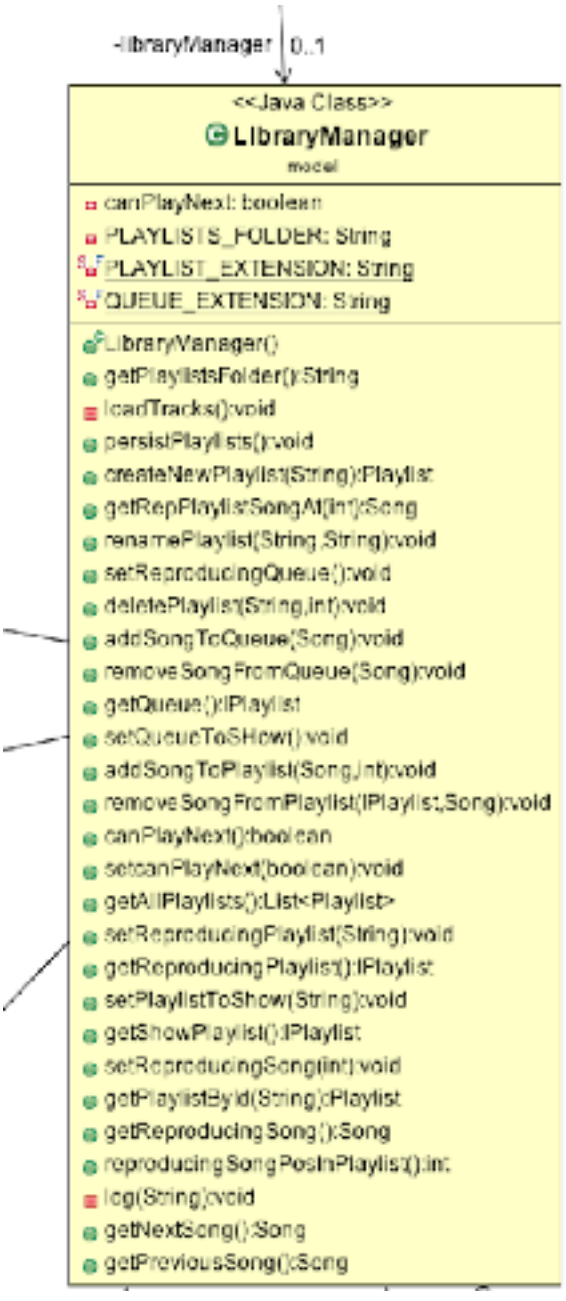
Come si può notare tutte e due le classi hanno uno stato quindi in base alle modifiche di una playlist runtime o della queue, viene modificato lo stato.
Per esempio l'aggiunta di una canzone o la creazione di una nuova modifica lo stato di una playlist in updated.

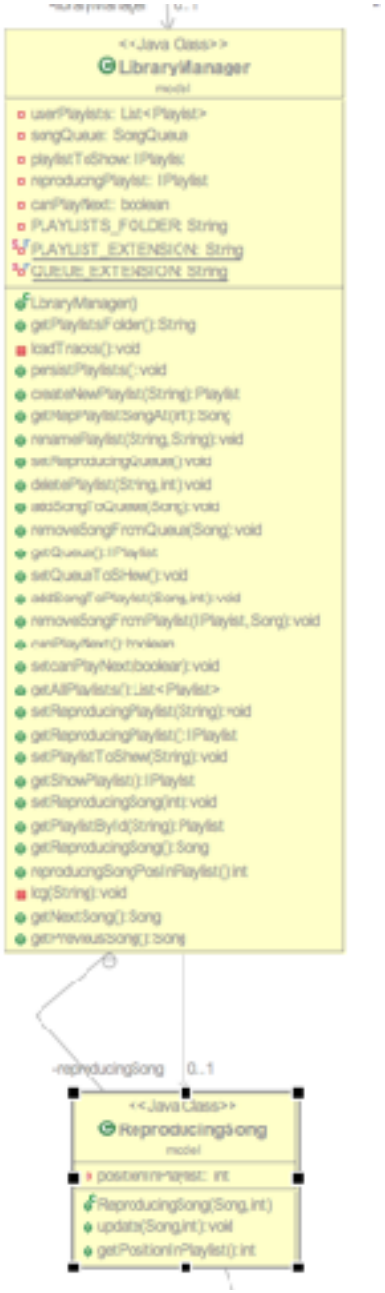
Modelli JTABLE:



Anche i modelli delle tabelle(quella centrale dove vengono visualizzate le canzoni e quella dove vengono visualizzate le playlist) sono contenuti in questo package poiché fanno sempre parte del modello però di visualizzazione.

Library manager





Questa classe supporta il controller per fornire tutti dati necessari sia runtime che chiusura o apertura del programma. Questa classe si occupa di gestire le playlist, la queue e le canzoni.

Ha il compito di eliminare, aggiungere, salvare nuove playlist.

Tiene traccia anche della canzone in riproduzione e la cambia ogni volta che il controller informa che la canzone corrente deve essere un’altra.

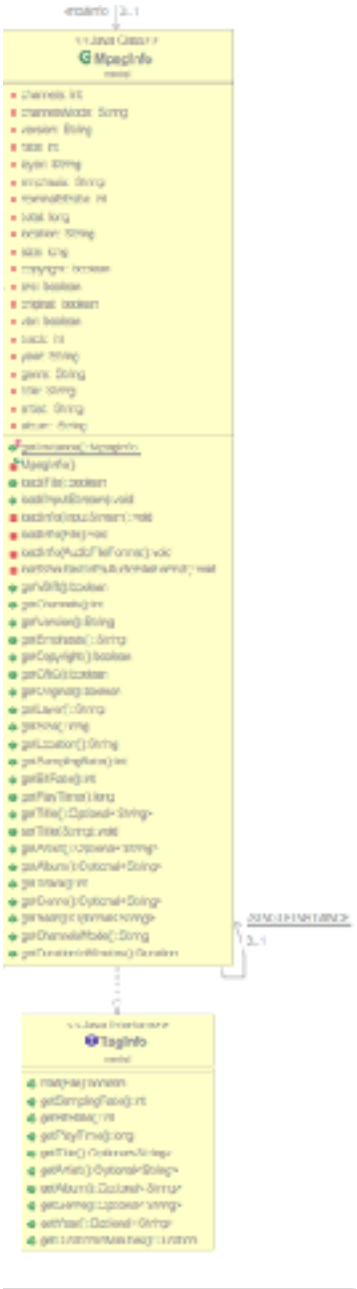
Usa i contenuti di util per caricare inizialmente e gestirli runtime. Sfrutta infatti il file Helper che si occupa della lettura dei delle canzoni.
In pratica legge i file che all’interno contengono le path delle canzoni.

Mp3Info

E’ la classe che fornisce le informazioni di un file mp3. sfrutta le seguenti librerie:

```
org.tritonus.share.sampled.file.TAudioFileFormat;
javax.sound.sampled.AudioFileFormat;
javax.sound.sampled.AudioSystem;
javax.sound.sampled.UnsupportedAudioFileException;
```

Implementa TagInfo



Anche qui viene usato il pattern singleton.

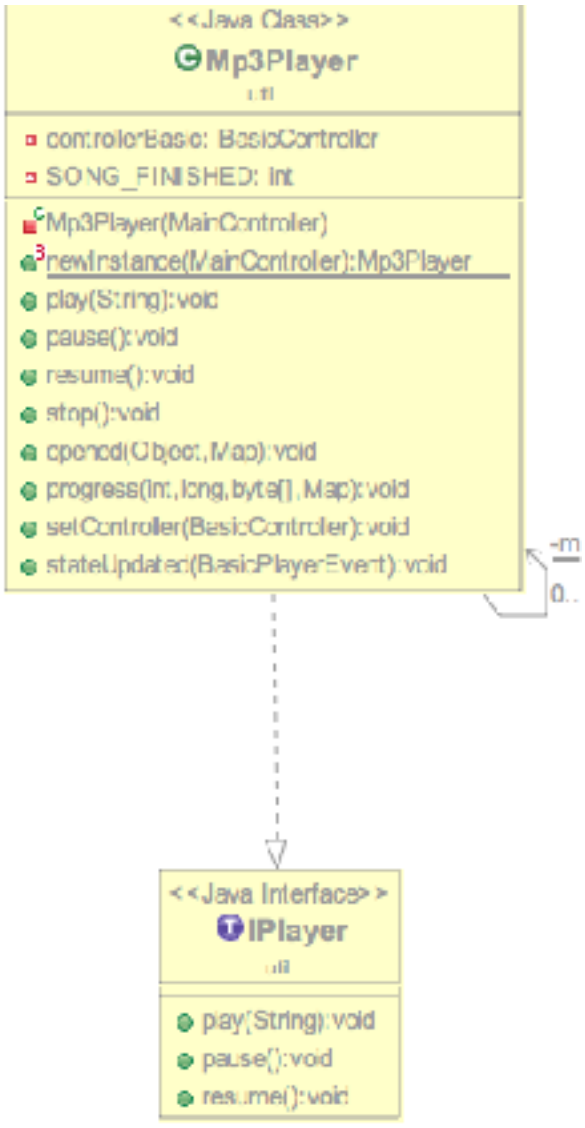
Util

Dentro il pack util si trovano altre classi molto imporanti come file helper che di fatto aiuta l'applicazione per:

- il caricamento delle canzoni
- salvare le playlist(persist Playlist)
- cancellare le playlist
- e rinominare la canzone

E’ stata messa a disposizione un’interfaccia IPlayer per permettere in futuro nuove estensioni del programma nella riproduzione non solo di mp3 ma di altri formati.
Mp3Player

E’ la classe che implementa IPlayer, BasicPlayerListener e si occupa di riprodurre il brano mp3 effettivo.
Si basa sulla libreria
sfrutta le librerie messe a disposizione da www.javazoom.net/jlgui/jlgui.html per la riproduzione



Threads

Sono stati creati due thread che si occupano di tenere traccia della posizione dalla canzone.
La loro creazione è dovuta al fatto che poiché il dato ricevuto dal MP3Player della posizione corrente, risultava impreciso. Le funzioni che hanno i due thread sono molto semplici:

Position song thread si occupa di incrementare al un intero ogni secondo

Progress Bar thread invece ha un compito molto più importante ed è stato necessario poiché sfrutta il metodo southP.revalidate() dentro il il metodo run per ridisegnare la posizione della progress bar all'interno del panel.
Questo thread chiede al controller la conversione dei dati da secondi a minuti e ore per stampare il risultato giusto nella Jlabel che mostra il conteggio dei minuti.

Conclusioni:

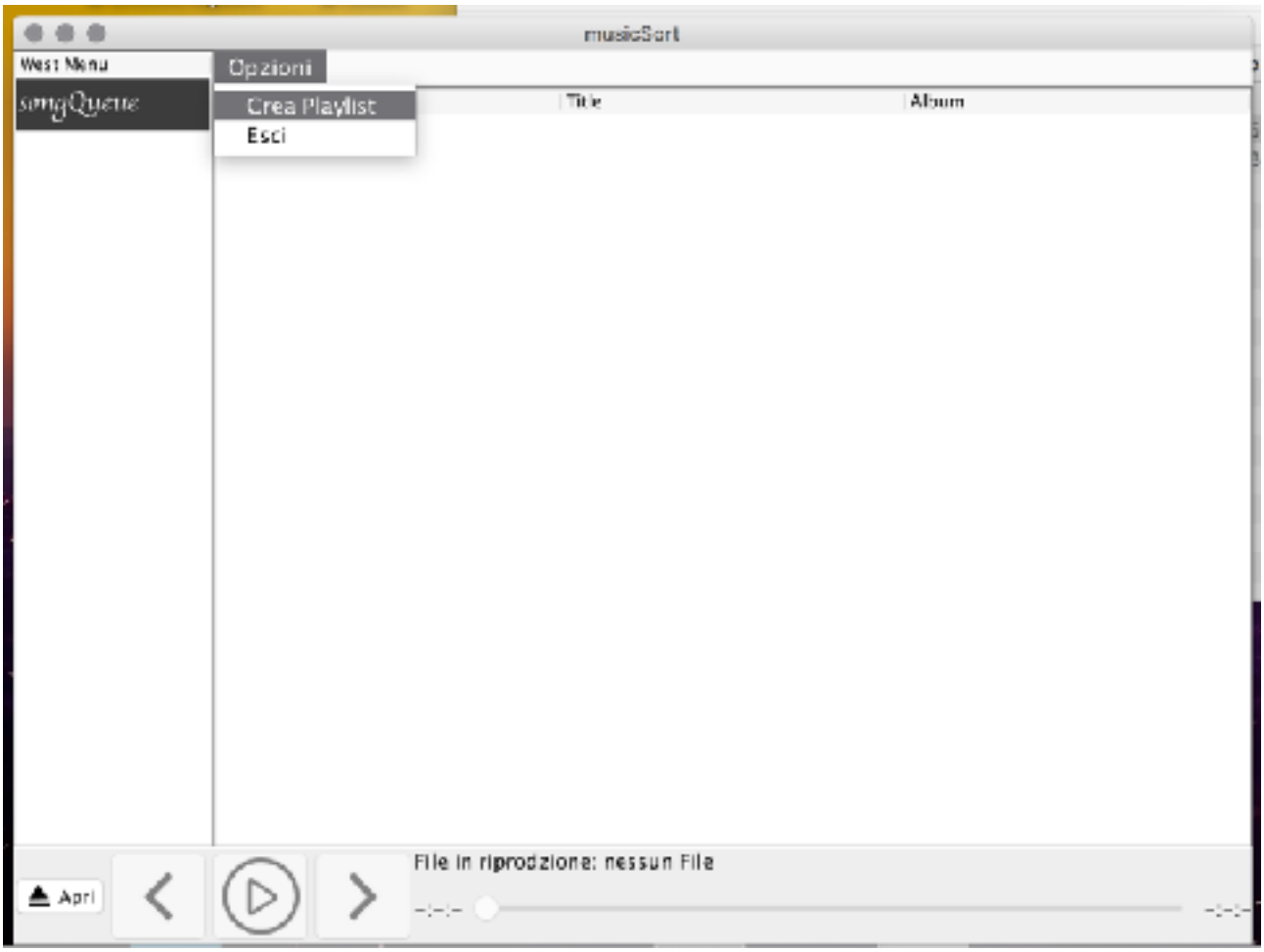
Purtroppo non è stato possibile completare tutte le funzioni promesse in partenza. In accordo con Viroli, dopo aver analizzato il risultato del progetto, è stato deciso di tralasciare alcune funzionalità inserendole tra le future.

Il risultato ottenuto è frutto di intenso lavoro svolto a partire da luglio.
Il risultato della gui è particolarmente impostato come inizialmente pensato.
Questa base di progetto permette la continuazione e l'implementazione di nuove funzionalità ai pannelli messi a disposizione.
Per inesperienza, la piattaforma di bitbucket non è stata impostata come speravo perché ho avuto diversi problemi con la configurazione di mercurial e i push che continuamente mi dava errori sul terminale. Dopo diversi tentativi di configurazione tra sourceTree e mercurial come plugIn di eclipse, è stato possibile procedere con i commit e push ma sempre con errori implicando push forzati.

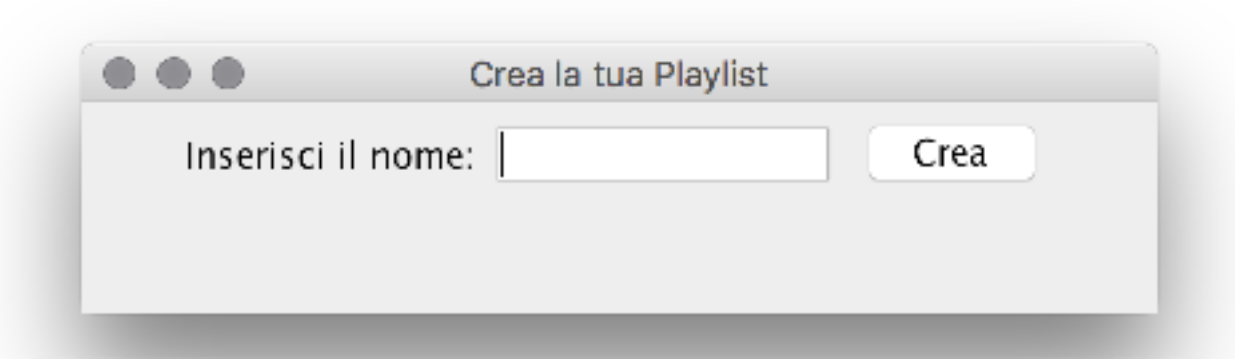
Nonostante le difficoltà sono molto soddisfatto del risultato ottenuto e delle conoscenze apprese.
Purtroppo non ho potuto condividere quest'esperienza con la mia collega che non è riuscita a partecipare.

Guida all’utente:

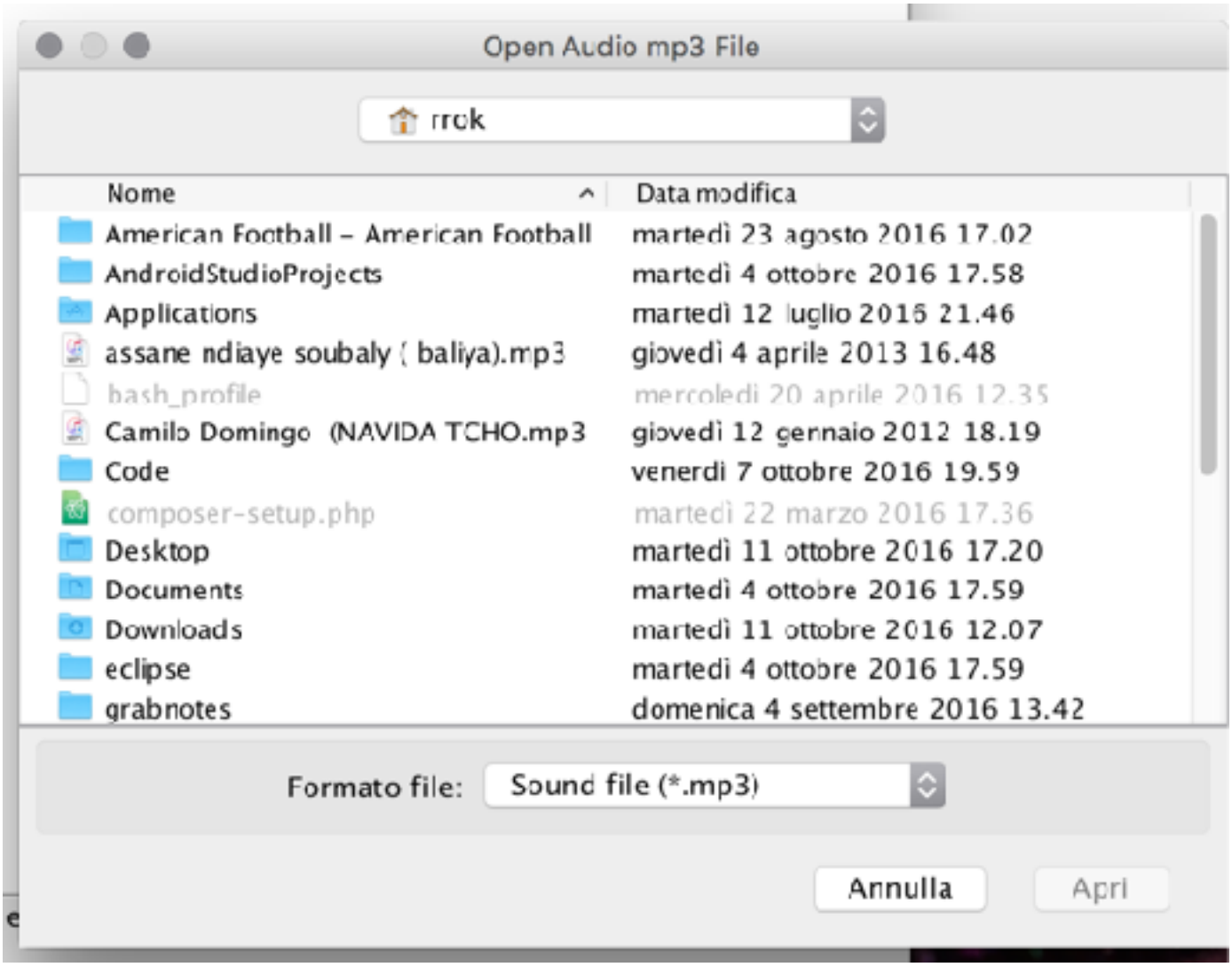
Creazione di una playlist



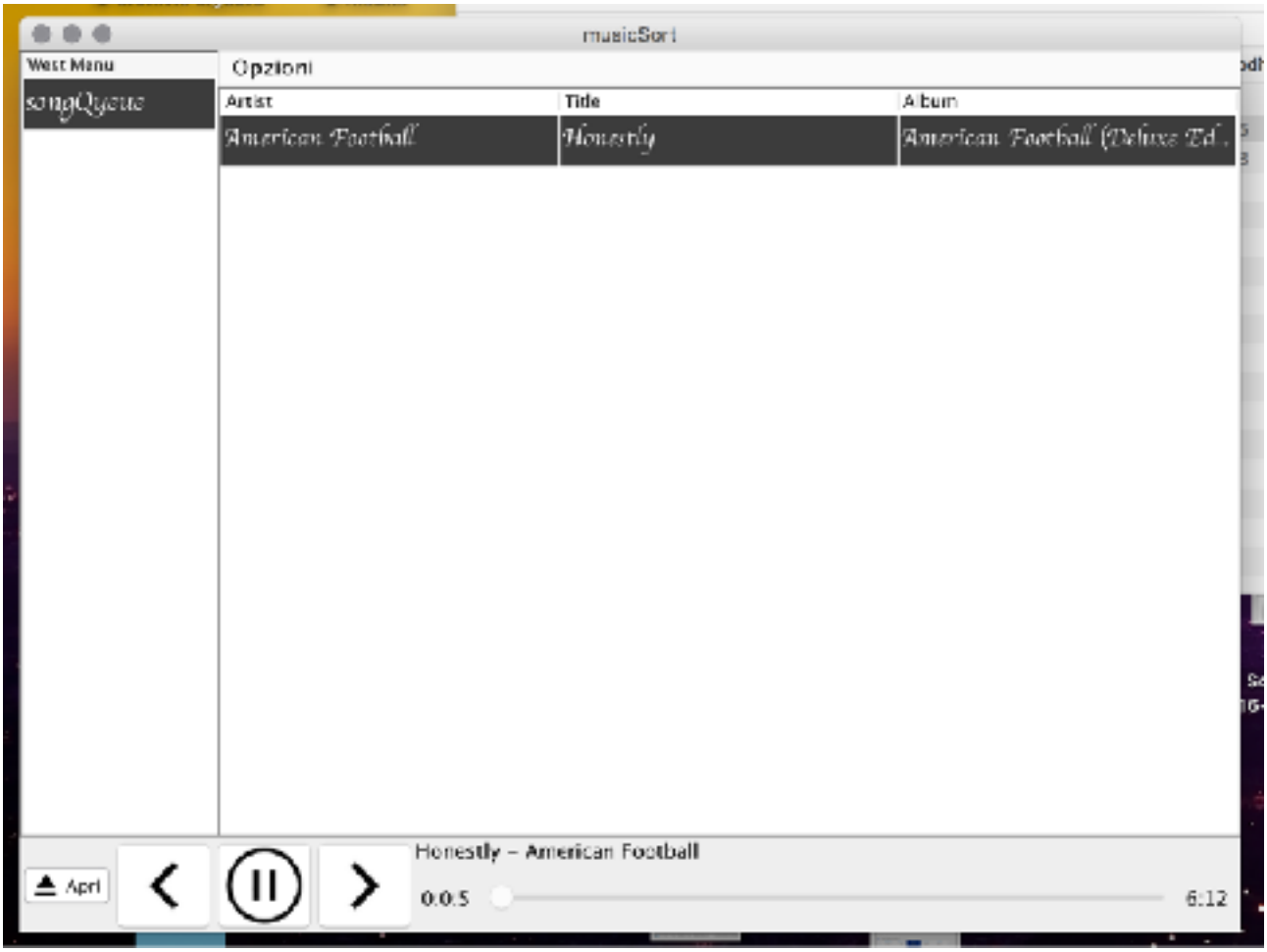
Inserimento nome:



Apertura di file mp3 dal bottone apri in basso a sinistra fornisce questa finestra per la ricerca della canzone.



Il brano aggiunto finirà nella song queue che subito la parà partire.



La canzone creata si potrà aggiungere alla playlist nuova oppure rimuovere.