Emotional Songs - manuale tecnico

Edoardo Picazio
Federico Ligas
Università degli Studi
dell'Insubria

Sommario

L'obiettivo di questo manuale è di illustrare da un punto di vista tecnico il funzionamento del programma Emotional Songs, spiegandone le soluzioni e gli approcci adottati.

1 Introduzione

Il progetto Emotional Songs è un lavoro sviluppato nel corso "Laboratorio A" del corso di informatica dell'università dell'Insubria.
Lo scopo dell'applicazione è di permettere all'utente di consultare una libreria di canzoni per valutare le emozioni che esse trasmettono.

1.1 Compatibilità

L'applicazione è stata sviluppata per funzionare su sistemi operativi Windows e Linux based; la compatibilità con MacOS è possibile perché è un sistema UNIX, però non è stata testata e quindi non ne è garantito il funzionamento.

1.2 Librerie Utilizzate

Per la creazione dell'applicazione abbiamo fatto ricorso a:

- Open JDK: la libreria standard per lo sviluppo in linguaggio Java
- OpenJFX 19: una libreria che permette lo sviluppo di interfacce grafiche in Java

1.3 Classi utilizzate

Il progetto è composto da due tipi di classi: le *core* classes che si occupano della effettiva elaborazione dei dati e dello svolgimento delle funzioni; poi seguono tutte le classi adibite alla gestione della UI(User Interface).

- Core Classes:
  - Song: la classe che si occuppa della gestione delle canzoni, permette di cercarle nella repository, vederne i dati e infine di valutarne le emozioni trasmesse.
  - Playlist: la classe che si occupa della creazione e gestione delle playlist.
  - Login: la classe che si occupa di gestire gli utenti: permette di effettuare registrazione e accesso, che sono richieste per accedere ad alcune funzioni del programma.
- UI Classes:
  - AddSongToPlaylistController.java
  - Controller.java
  - CreazionePlaylistController.java
  - Emozioni.java
  - InserisciCommentoController.java
  - InserisciEmozioniController.java
  - LoginController.java
  - Main.java
  - MenuInizialeController.java
  - MenuUtenteController.java
  - NomePlaylistController.java
  - RegistrazioneController.java
  - RicercaAvanzataController.java
  - RicercaInRepositoryController.java
  - SearchSongTableController.java
  - SelezionaCanzoneController.java
  - SelezionaPlaylistController.java

- SongTableController.java
- VisualizzaCommentiController.java
- VisualizzaEmozioniController.java
- VisualizzaEmozioniDati.java

Di seguito discuteremo nel dettaglio il funzionamento delle core classes, spiegandone la struttura e i metodi.

2 Analisi Core Classes

2.1 Classe Song

La classe song si occuppa di gestire le canzoni, ovvero l'oggetto alla base dell'applicazione perciò è quella più articolata e ricca di funzioni.

2.1.1 Struttura

La classe è composta dai valori che compongono una canzone, ovvero:

- Name: il nome del brano
- Album: il nome dell'album di appartenenza
- Author: il nome dell'artista o del gruppo
- Duration: la durata del brano
- Year: l'anno di uscita del brano
- Id: è la posizione della canzone all'interno della repository, è utile per identificare le canzoni e per trovarle all'interno della repository.

Adesso che abbiamo specificato la struttura della classe possiamo esaminare il funzionamento dei metodi.

2.1.2 Ricerca canzone - searchSong()

Una delle operazioni più importanti in assoluto è quella di ricerca delle canzoni: la ricerca può essere effettuata in base al titolo, all'autore, all'autore e l'anno e all'album. Al metodo deve essere passato il tipo di ricerca che si vuole effettuare e la lista dei parametri da utilizzare. Per trovare i brani desiderati ci si limita a scorrere ogni elemento nella repository, alla fine del processo vengono restituiti tutti i brani coerenti con la ricerca. Poiché si scorre tutta la repository per intero, l'unico processo di durata variabile è l'aggiunta delle canzoni che è compresa fra 0 e 5000(tutte le canzoni della repo.), perciò la complessita della funzione di ricerca è $\theta(1)$.

2.1.3 Ottenimento dati canzone - getSongData()

Questo metodo è utile al costruttore nel caso in cui l'unico dato noto sia l'id della canzone(per esempio quando dobbiamo identificare le canzoni in una playlist). Il metodo si limita ad andare alla riga n =id e ne restituisce i dati. Siccome l'esecuzione di questa funzione è limitata ad un banale ciclo *for* la cui lunghezza è sempre compresa tra 1 e 5000(ultima canzone della repo.) la complessità è: $\theta(1)$.

2.1.4 Valutazione Emozioni - RegistraEmozioni()

La valutazione delle emozioni consiste semplicemente nel prendere in input l'id della canzone e dell'utente assieme ad una lista di voti, e eventuali commenti, e salvarli in memoria. Si effettua sempre lo stesso numero di operazioni nello stesso ordine, perciò la complessità è $O(1)$.

2.1.5 Ottenimento valutazioni Aggregate - VisualizzaEmozioniBrano()

Questa funzione permette di ottenere un prospetto riassuntivo delle valutazioni che gli utenti hanno fatto su una canzone. Per fare ciò si scorrono tutte le valutazioni di una canzone e si calcolano valori medi e numero di utenti che hanno partecipato al voto. Data una canzone S con n valutazioni la funzione scorrerà sempre tutte le valutazioni, quindi la complessità è $O(n)$

2.2 Classe Playlist

La classe palylist serve per la gestione delle playlist, per lo più bisonga occuparsi della loro creazione. Essa è composta da:

- Nome della playlist
- Id dell'utente creatore
- Lista degli Id delle canzoni che fanno parte della playlist

2.2.1 Creazione Playlist

La creazione dell'istanza viene fatta semplicemente prendendo in input il nome della playlist, l'id dell'utente che l'ha creata, e la lista degli id delle canzoni che ne fanno parte.

2.2.2 Registrazione Playlist - RegistraPlaylist()

Consiste nello scrivere nel file *Playlist.dati.csv* gli attributi della playlist; quest'operazione dipende dal numero di canzoni nella playlist e perciò la tempo impiegato cresce linearmente insieme a queste, quindi la complessità è $O(n)$

2.3 Classe Login

La classe login si occupa di gestire accesso e registrazione degli utenti; è composta da:

- UserId: il nome utente
- Password: la password dell'account
- Lista delle playlist: la lista delle playlist create dall'utente.

2.3.1 Registrazione - Login()

La registrazione è un'operazione molto semplice, avviene grazi al costruttore Login() che non fa altro che prendere un insieme di dati in input(userId, password, email etc.) e salvarli in memoria scrivendoli nel file UtentiRegistrati.dati.csv. L'operazione ovviamente ha tempo di esecuzione costante e quindi complessità $O(1)$.

2.3.2 Accesso - Login()

Il metodo permette all'utente di accedere al suo account inserendo userId e password. La verifica avviene eseguendo una scansione di tutte le righe del file UtentiRegistrati.dati.csv fino a che non si trova una coppia di credenziali che corrisponde a quella inserita.
L'esecuzione di questo metodo in un contesto dove sono registrati n utenti può richiedere da 1 a n cicli, da ciò ne segue che la complessità è $O(n)$.

2.3.3 Ottieni playlist utente - getPlaylistUtente()

Il metodo permette di ricevere la lista di tutte le palylist create dall'utente. Il metodo funziona estraendo le playlist dell'utente dalla lista di tutte le palylist salvate in memoria, che legge sempre per intero; quindi la complessità, date n palylist salvate è di $O(1)$. Di seguito vengono illustrati diversi metodi di funzionamento analogo che hanno tutti lo scopo di verificare l'unicità dei dati inseriti dall'utente durante la registrazione.

2.3.4 Verifica dei dati di registrazione

I metodi checkCF(), checkUserId() e checkEmail() verificano che non ci sia un altro utente già registrato con gli stessi dati. Il loro funzionamento consiste nello scorrere i dati della lista di utenti registrati e vedere che non ce ne siano di corrisponenti. L'esecuzione di questi metodi scorre sempre tutta la lsita di utenti a meno che non si trovi un dato non valido, di conseguenza la sua complessità è $O(n)$.

3 Analisi Interfaccia Grafica

Le classi controller gestiscono l'inizializzazione di una nuova finestra e l'esecuzione degli eventi prodotti dai vari componenti presenti nella scena della finestra considerata. JavaFx utilizza una specializzazione di XML, FXML, che permette di separare meglio l'aspetto grafico vero e proprio (tipologia di elementi, posizioni all'interno della scena, dimensioni ecc...) e la gestione degli eventi generati (ad esempio, il metodo x che viene chiamato quando si preme il pulsante y). Il file FXML, quindi, può essere creato indipendentemente, magari con l'ausilio di tool specializzati come, ad esempio, SceneBuilder, e successivamente "caricato" in un oggetto di tipo Scene che sta all'interno della finestra, che in JavaFx è rappresentata dalla classe Stage. Le classi controller implementano l'interfaccia Initializable: oggetti che implementano questa interfaccia hanno un metodo initialize(), che viene chiamato subito dopo che il file fxml è stato propriamente caricato nella scena. Di "caricare" questi file fxml se ne occupa un oggetto di tipo FXMLLoader, attraverso il metodo load(). È possibile, alla creazione di un file fxml, associare già un controller adibito alla gestione degli eventi di quella scena, oppure si può optare per l'associazione manuale prima di chiamare il metodo load().

3.1 Styling con CSS

Per abbellire l'interfaccia grafica abbiamo deciso di utilizzare delle componenti in linguaggio di CSS che sono state di grande aiuto. I file CSS possono essere utilizzati nelle scene grazie al metodo getStyleSheets().