Emotional Songs

Manuale tecnico

Versione 1.0

26/05/2022

Università degli Studi dell'Insubria – Laurea Triennale in Informatica

Progetto Laboratorio A: Emotional Songs

Sviluppato da: Erba Lorenzo, matricola 748702, Cacciarino Matteo, matricola 748231, Ferialdo Elezi, matricola 749721, Zancanella Alessandro, matricola 751494

Indice

1. Introduzione

Emotional Songs un applicativo sviluppato nell'ambito del progetto di Laboratorio A per il corso di Informatica dell'Università degli Studi dell'Insubria. Il progetto è sviluppato in Java 16, utilizza un'interfaccia grafica basata su JavaSwing ed è stato effettuato il testing sul sistema operativo Windows 10 e 11.

1.1 Struttura generale delle classi

Il progetto si basa su 2 principali macrogruppi di classi, le main classes e le gui classes. Il primo gruppo fa riferimento a tutte le classi utilizzate per l'elaborazione dati back-end, come lettura e scrittura su file, mentre il secondo gruppo gestisce l'interazione front-end tra l'utente e la parte grafica dell'applicativo.

- Main classes
  - Canzone
  - Emozioni_Dati
  - Emozione_Collection
  - Emozioni_Canzone
  - OperazioniXML
  - OperazioniXMLwrite
  - Playlist
  - Repository
  - Utente
- Gui classes
  - EmotionalSongs (main)
  - emotionsSummary_Gui
  - GUI
  - Gui_CreaPlaylist
  - Gui_ElencoBrani
  - GUI_login
  - Gui_Playlist
  - Gui_VisualizzaPlaylist
  - notes_gui
  - PannelloEmozioni
  - repositoryChoice_Gui

2. Main Classes

2.1 Canzone

La classe Canzone permette la gestione di oggetti rappresentanti il brano presente nel repository. Per tale motivo la classe Canzone prevede i seguenti attributi:

- Titolo (String), rappresentante il titolo della canzone.
- Autore (String), rappresentante l'autore della canzone.
- Anno (String), rappresentante l'anno di uscita del brano
- Linkyt (String), rappresentante il collegamento ipertestuale al brano su youtube.

Complessità stimate e strategie progettuali

La classe Canzone è composta da soli metodi getter and setter i quali hanno complessità O(1). Durante la progettazione della classe Canzone si è deciso di non tenere traccia del genere, dell'album e della durata in quanto non sono stati richiesti come criteri di filtraggio durante la ricerca delle canzoni.

2.2 Emozioni_Dati

La classe Emozioni_Dati permette la gestione di oggetti rappresentanti ogni singola emozione (in totale 9) suscitata durante l'ascolto di un brano. Per tale motivo la classe Emozioni_Dati prevede i seguenti attributi:

- Emozione (String), rappresentante il nome dell'emozione.
- Punteggio (int), rappresentante il punteggio relativo all'emozione, rilasciato dall'utente.
- Note (String), rappresentante il commento relativo all'emozione, rilasciato dall'utente.

Complessità stimate e strategie progettuali

La classe Emozioni_Dati è composta da soli metodi getter i quali hanno complessità O(1). Durante la progettazione della classe Emozioni_Dati si è deciso di non implementare metodi setter in quanto non necessari per la realizzazione del progetto.

2.3 Emozione_Collection

La classe Emozioni_Collection permette la gestione di oggetti rappresentante la recensione rilasciata dall'utente relativa a una canzone. Per tale motivo la classe Emozione_Collection possiede i seguenti attributi:

- Idutente (String), rappresentante l'identificativo dell'utente che ha rilasciato la recensione.
- Emozioni_lst (ArrayList<Emozioni_Dati>), rappresentante la lista delle emozioni rilasciate dall'utente, con i relativi punteggi e note.

Complessità stimate e strategie progettuali

La classe Emozioni_Collection è composta da metodi getter e setter i quali hanno complessità O(1).

Inoltre possiede i seguenti metodi:

- addEmozioniDatiToArrayList, che effettua un inserimento in coda nella lista emozioni_lst, quindi complessità O(1).
- checkEmozioniEqualsToZero, che controlla se tutte le emozioni rilasciate dall'utente hanno punteggio 0, complessità di O(n).
- getEmoCanzonePunteggioToArray, metodo getter che restituisce il contenuto della lista emozioni_lst in formato Vector<String> formattato in HTML, complessità O(n).
- getAllNotes, metodo getter che restituisce l'insieme delle note di tutte le emozioni rilasciate dall'utente formattate in HTML, complessità O(n)

Durante la progettazione della classe Emozioni_Collection si è deciso di utilizzare come struttura dati una lista in quanto garantiva per gran parte delle operazioni complessità O(1) oppure lineare al caso peggiore.

2.4 Emozioni_Canzone

La classe Emozioni_Canzone permette la gestione di oggetti rappresentanti l'insieme delle recensioni di tutti gli utenti relativi a un singolo brano. Per tale motivo la classe Emozioni_Canzone possiede i seguenti attributi:

- Titolo (String), rappresentante il nome dell'emozione.
- Autore (String), rappresentante il punteggio relativo all'emozione, rilasciato dall'utente.
- Emocanzone_lst (ArrayList<Emozione_Collection>), rappresentante la lista delle recensioni, rilasciate dagli utenti.
- avrgAmazement (int), rappresentante la media dell'emozione "Amazement".
- avrgSolemnity (int), rappresentante la media dell'emozione "Solemnity".
- avrgTenderness (int), rappresentante la media dell'emozione "Tenderness".
- avrgNostalgia (int), rappresentante la media dell'emozione "Nostalgia.
- avrgCalmness (int), rappresentante la media dell'emozione "Calmness".
- avrgPower (int), rappresentante la media dell'emozione "Power.

- avrgJoy (int), rappresentante la media dell'emozione "Joy".
- avrgTension (int), rappresentante la media dell'emozione "Tension".
- avrgSadness (int), rappresentante la media dell'emozione "Sadness".

Complessità stimate e strategie progettuali

La classe Emozioni_Canzone è composta da metodi getter e setter i quali hanno complessità O(1).

Inoltre possiede i seguenti metodi:

- addEmozioneCollectionToArray, metodo che permette di aggiungere in coda alla lista una recensione rilasciata dall'utente, complessità O(1).

Durante la progettazione della classe Emozioni_Canzone si è deciso di utilizzare come struttura dati una lista in quanto garantisce per tutte le operazioni complessità O(1).

2.5 OperazioniXML

La classe OperazioniXML permette la gestione dei file .xml, lettura e scrittura dei dati da/su file. Per tale motivo la classe OperazioniXML possiede i seguenti attributi:

- PATH_FILE (final String), rappresentante il percorso relativo del file .xml.
- FILE_NOME (final String), rappresentante il nome del file .xml.
- PATH_XSD (final String), rappresentante il percorso relativo del file .xsd.

Complessità stimate e strategie progettuali

La classe OperazioniXML possiede i seguenti metodi:

- addPlaylist, metodo che aggiunge la playlist creata sotto il rispettivo idUtente e nome della relativa playlist, complessità O(n).
- aggiornaXML, metodo che viene chiamato dal Panel PannelloEmozioni per svolgere la lettura, e riscrittura del file, complessità O(1)
- checkIfIsError, metodo che controlla se l'oggetto passato come parametro è istanza di String, complessità O(1).
- checkIfPlaylistExists, metodo che controlla se l'utente ha almeno una playlist, complessità O(n).
- createFilePlaylist, metodo che crea il file playlist, complessità O(1).
- fileExist, metodo che controllla se un file è già esistente, complessità O(1).
- findCanzone, Metodo che controlla se la canzone indicata è presente nel file emozioni.xml, complessità O(n).
- getCanzoneFiltrataAnnoAutore, metodo che controlla se l'elemento passato come parametro coincide coi filtri inseriti, complessità O(1).
- getCanzoneFiltrataTitolo, metodo che controlla se l'elemento passato come parametro coincide coi filtri inseriti, complessità O(1).

- getEmozioneEUtente, metodo che legge le emozioni di un singolo utente, complessità O(1).
- getEmozioni, metodo che legge tutte le emozioni della canzone indicata, complessità O(n).
- getFloatValue, metodo che restituisce il valore float dell'elemento figlio specificato, complessità O(1).
- getIntValue, metodo che restituisce il valore intero dell'elemento figlio specificato, complessità O(1).
- getTextValue, metodo che restituisce il valore string dell'elemento figlio specificato, complessità O(1).
- getUtente, metodo che restituisce l'utente letto, complessità O(1).
- leggiFile, metodo che permette la lettura di tutto il file, e inserirlo all'interno di un'ArrayList di tipo Emozioni_Canzone, che servirà per la scrittura (viene inserito anche il la nuova votazione), complessità O(n).
- leggiRepository, metodo che legge il file contenente la repository di canzoni disponibili per poter creare la playlist, complessità O(n).
- parseDocument, metodo che permette di leggere il file degli utenti registrati, complessità O(n).
- parseDocumentCanzoniFiltrate, metodo che effettua il parsing del file canzoni.xml filtrando tramite il titolo della canzone, complessità O(n).
- parseDocumentEmozioni, metodo che effettua il parsing del file emozioni.xml, complessità O(n).
- readPlaylist, metodo che legge da file tutte le playlist relativa all'IdUtente passato, complessità O(n).
- scriviXMLEmozioni, metodo che aggiorna il file XML con le nuove emozioni, riscrivendolo da capo, complessità O(n).
- updateAvrg, metodo che aggiorna le medie delle emozioni, complessità O(n).
- writeFile, metodo che scrive la lista degli utenti registrati sul file UtentiRegistrati.xml, complessità O(n).
- createDOMTree, metodo che realizza il dom dell'albero nel nuovo file, complessità O(n).
- createUtenteElement, metodo che realizza l'elemento utente all'interno del file, complessità O(1).
- appendWrite, metodo che scrive in modalità append nel file UtentiRegistrati.xml, complessità O(1).
- printToFile, metodo che sovrascrive il file specificato, complessità O(1).

2.6 Playlist

La classe Playlist permette la gestione di oggetti rappresentanti la playlist dell'utente. Per tale motivo la classe Playlist presenta i seguenti attributi:

- Nomeplaylist (String), rappresentante il nome della playlist.
- Brani (ArrayList<Canzone>), rappresentante la lista di brani presenti nella playlist.

Complessità stimate e strategie progettuali

La classe Playlist è composta da metodi con complessità O(1).

2.7 Repository

La classe Repository permette la gestione di oggetti rappresentanti il repository di canzoni, ovvero l'insieme delle canzoni rese disponibili dall'applicazione. Per tale motivo la classe Repository presenta i seguenti attributi:

- Repository (ArrayList<Canzone>), rappresentante la lista dei brani presenti nel repository.

Complessità stimate e strategie progettuali

La classe Repository è composta da metodi con complessità O(1)

2.8 Utente

La classe Utene permette la gestione di oggetti rappresentanti l'utente che ha utilizzato o utilizza l'applicazione. Per tale motivo la classe Utente presenta i seguenti attributi:

- Nome (String), rappresentante il nome dell'utente.
- Cognome (String), rappresentante il cognome dell'utente.
- Cf (String), rappresentante il codice fiscale dell'utente.
- Email (String), rappresentante l'email dell'utente.
- Indirizzo (String), rappresentante l'indirizzo dell'utente.
- Userid (String), rappresentante l'identificativo numerico dell'utente.
- Password (String), rappresentante la password dell'utente.

Complessità stimate e strategie progettuali

La classe Utente è composta da metodi con complessità O(1). Durante la progettazione della classe Utente si è deciso di generare l'Userid come un numero intero a 5 cifre e di utilizzarlo in fase di login in quanto creato univocamente senza duplicati. Inoltre si è deciso di non impostare particolari requisiti per la creazione della password.

3. GUI Classes

Per la realizzazione dell'interfaccia grafica è stato utilizzato il linguaggio Java_Swing, una particolare estensione del linguaggio Java che fa utilizzo di costrutti grafici per la realizzazione di progetti con grafica utente. Fra i vari oggetti grafici utilizzati troviamo:

- JButton
- JPanel
- JFrame
- JLabel
- JTextField
- JTable
- JTextPane
- JTabbedPane
- JOptionPane

4. Formato dei file

Per quanto riguarda la memorizzazione dei dati utilizzati dall'applicazione, si è deciso di ricorrere all'utilizzo di file xml in quanto risultano essere:

- efficaci dal punto di vista implementativo consentendo di realizzare funzioni con complessità lineare e alcune volte in tempo costante.
- chiari e ordinati dal punto di vista visivo, in quanto rispettano una struttura gerarchica dei tag che rendono più facile la lettura/scrittura dei file.