Emotional Songs

Manuale Tecnico

Università degli Studi dell'Insubria – Laurea Triennale in Informatica

Progetto Laboratorio A: Emotional Songs

Sviluppato da:

Edoardo Rossi, matricola 749089
Mattia Sindoni, matricola 750760
Gabriele Todeschini, matricola 750767
Matteo Argiolas, matricola 75125

Sommario

Introduzione 2
Librerie esterne utilizzate 2
Struttura generale 3
Classe EmotionalSongs 3
start 3
main 3
Classe Emozioni 3
Emozioni 3
Add 3
getMedia 4
getValutazioni 4
calcMedia 5
readFile 5
Classe Utente 5
Login 5
registrazione 5
CheckDuplicatedUsers 6
Classe HelloController 6
getRes 6
LoadRepositoryList 6
PrintMedia 6
PrintReview 6
dataList 6
search 6
PasswordValidator 7
idTAXcodeValidator 7
emailValidator 7
openSignin 7
signIN 7
logIN 7
StartPlaylist 7
StopPlaylist 7
canc 7
reset 7
AddSong 8
Classe CreaModificaPlaylist 8

getID 8
GetRepository 8
CreaPlaylistVuota 8
AggiungiBrani 8
Gestione File 8
Canzoni.dati.csv 8
Emozioni.dati.csv 8
UtentiRegistrati.dati.csv 9
currentUsrID.txt 10
Cartella Utenti e file playlist 10
getID(String str) 10
GetRepository(String PathRepository) 10

Introduzione

Emotional Songs è un progetto sviluppato nell'ambito del progetto di Laboratorio A per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria.

Il progetto è sviluppato in Java 17, usa un'interfaccia grafica costruita con OpenJFx 18 ed è stato sviluppato su Windows 10 e 11 e testato sui sistemi operativi Windows 10, 11, MacOS High Sierra, Catalina e Linux Ubuntu 22 e Linux Mint 21.

Librerie esterne utilizzate

Per lo sviluppo di questo progetto è stata utilizzata una libreria di terze parti: openjfx

Java FX fornisce delle librerie per realizzare e gestire interfacce grafiche, realizzabili direttamente da codice o tramite file di formato fxml, simile al xml col quale è possibile realizzare interfacce grafiche e collegarci funzioni in linguaggio java

Struttura generale

- ## Classe EmotionalSongs

  Questa classe (sottoclasse di Application) è la prima ad essere eseguita all'avvio del programma.
  Viene utilizzata per caricare il layout grafico fissandone le dimensioni e lo mostra sullo schermo.
  Metodi di EmotionalSongs:

  - ### start

    Il file StartWindows.fxml viene caricato all'interno dell'ogetto fxmlLoader di tipo FXMLLoader; dopo di che viene creata una nuova scena utilizzando lo stesso fxmlLoader impostando le dimensioni della finsetra a 1300px * 800px.
    La scena viene inserita all'interno dello stage impostando il parametro setResizable a false per bloccare il ridimensionamento della finestra per poi visualizzarlo a schermo.

  - ### main

    esegue il metodo launch(); ereditato dalla superclasse Application, la quale fa parte della libreria JavaFX

- ## Classe Emozioni

  Questa classe gestisce tutte le funzioni che riguardano l'elaborazione, la memorizzazione e la lettura delle emozioni.
  Metodi "**public**" di Emozioni:

  - ### Emozioni

    Costruttore della classe Emozioni, richiede come parametro una Stringa (p) che contiene il percorso della directory "data" e lo salva in una variabile interna chiamata "path"

  - ### Add

    Salva le valutazioni degli utenti all'interno del file Emozioni.dati.csv
    Al metodo Add vengono passati 4 parametri: 1 array di 9 int che rappresenta le valutazioni da 1 a 5 per le singole emozioni
    1 array di 9 elementi che contiene i commenti per ogni emozione, e 2 interi che rappresentano l'ID della canzone e l'Id dell'utente.

    Il metodo associa gli elementi dei 2 array passati a delle variabili locali e successivamente conta gli elementi di canzoni.dati.csv e li salva in una variabile, che verrà usata per creare un array contenente per ogni elemento una riga del file emozioni.dati.csv.
    Dopodiché controlla se il file emozioni.dati esiste, se non dovesse esistere il tag newFile verrebbe impostato a true, e verrebbe generato il file contenente

per ogni canzone la stringa "id, array di media" dove array di media è composto da 9 elementi impostati a zero
Salva nell'array le righe del file e seleziona l'elemento corrispondente alla canzone selezionata.
A questa riga aggiunge id dell'utente che ha commentato, valutazioni e i commenti.
Successivamente controlla che non si tratti della prima valutazione per la canzone selezionata verificando se la media è uguale a 0.
Nel caso in cui fosse la prima valutazione per la canzone selezionata la stringa corrispondente alla media diventa uguale alla stringa contenente le valutazioni, altrimenti viene chiamato il metodo calcMedia,
che passando un array di stringhe dove in ogni elemento è presente una valutazione ritorna una stringa formattata correttamente che verrà poi sostituita all'elemento media precedente.
Infine l'array contenente tutte le valutazioni e la media inserito nell'array degli elementi nella posizione della canzone scelta, per poi essere interamente stampato sul file.

- getMedia

  Il metodo get media riceve come parametro l'id della canzone da cui andare ad estrarre l'array della media.
  Viene creato un array vuoto, che conterrà la media, quest'ultimo viene poi controllato se la variabile che contiene il contenuto del file è nullo, in tal caso viene chiamata la funzione readFile, che provvede a salvare nella variabile della funzione l'intero contenuto del file Emozioni.dati.csv
  di conseguenza procede a accedere all'elemento dell'array del file nella posizione corrispondente all'id della canzone, per poi, tramite la funzione split separare i blocchi di valutazioni (separati tramite ;) e poi accedere al blocco della media e separare le singole medie da aggiungere all'array media, che viene infine ritornato

- getValutazioni

  Il metodo get valutazioni riceve come parametro l'id della canzone da cui andare ad estrarre la matrice contenente per ogni riga una recensione e nelle colonne id dell'utente e le 9 valutazioni.
  il metodo verifica se la variabile della contenente l'intero file è diversa da null, ovverosia se il file è già stato letto e salvato, nel caso contrario viene chiamato il metodo readFile che procederà alla lettura e al salvataggio.
  Successivamente legge la riga corrispondente alla canzone selezionata, indicandone l'id come posizione, per poi contare il numero di "blocchi" sottraendo i primi 2, che corrispondono a id canzone e all'array della media
  Una volta trovato il numero di recensioni lasciate dagli utenti le scorre una per una e le salva in un array.
  In fine per ogni elemento dell'array (recensione) verranno separate le singole valutazioni / recensioni per poi salvare il tutto dentro a una matrice che verrà ritornata.

Metodi "**private**" di Emozioni:

- calcMedia
- readFile

  Il metodo readFile permette di salvare in un array il contenuto del file emozioni.dati.csv
  Il metodo inizia calcolando il numero di righe in canzoni.dati.csv, per poi proseguire a leggere le righe e salvarle per ogni riga dell'array.

Classe Utente

iL costruttore utente riceve come parametro una stringa, che associa alla variabile path, che corrisponde alla directory della cartella data.
Metodi di Utente:

- Login

  Il metodo Login permette di autenticare l'utente tramite email o username e password
  Il metodo riceve come parametri la credenziale inserita (indirizzo email o username) e password
  Apre il file UtentiRegistrati.dati.csv, dal quale estrai una riga alla volta, separa le informazioni tramite la funzione split e poi verifica se la posizione 4 o la posizione 6 (rispettivamente username e email) corrispondono alla credenziale inserita
  Una volta trovato l'utente a cui corrispondono quelle credenziali compara la password passata con quella dell'utente, se le due password dovessero corrispondere le credenziali verrebbero salvate in variabili interne alla classe e verrebbe ritornato l'id dell'utente
  Altrimenti se l'autenticazione non dovesse venire viene ritornato il valore -1, che rappresenta la mancata esecuzione della procedura.

- registrazione

  Il metodo registrazione permette di salvare le credenziali dell'utente all'interno del file utentiRegistrati.dati.csv
  Al metodo vengono passati 7 parametri: nome, cognome, codice fiscale, username, indirizzo, email e password.
  Il metodo procede a salvare i dati all'interno delle proprie variabili, per poi aprire il file UtentiRegistrati.dati.csv, verificandone innanzitutto l'esistenza.
  Se il file non dovesse esistere verrebbe creato e l'id utente verrebbe impostato a 0.
  Nel caso in cui il file esista, verrebbero lette le righe incrementando una variabile nel ciclo di lettura, e il risultato corrisponde all'id utente
  poi viene creata una stringa con i dati nell'ordine prescelto, aggiunti ad una stringa che verrà poi accodata al file tramite il tag append impostato a true nell'oggetto di tipo FileWriter
  Dopo aver inserito la stringa viene aggiunta, nella directory data una cartella chiamata Utenti, dentro la quale verrà generata un ulteriore cartella col nome dell'id utente, che verrà successivamente usata per salvare le playlist dell'utente creato.

- CheckDuplicatedUsers

  Il metodo CheckDuplicatedUsers permette di verificare se, tramite il passaggio di username e indirizzo email l'utente esiste già
  Viene creato un arraylist contenente l'intero repository contenente gli utenti, il cui percorso viene passato come parametro
  successivamente viene verificato che nessuna delle due credenziali sia presente all'interno dell'arraylist, e nel caso in cui esista già un utente con queste credenziali verrebbe ritornato il valore false e true nel caso opposto.

Classe HelloController

La classe HelloController è la classe che principalmente gestisce tutta la parte grafica dell'applicazione EmotionalSongs.
I suoi metodi sono i seguenti:

- getRes

  una volta richiamato restituisce un ArrayList<String>
  che contiene tutto il repository di canzoni.

- LoadRepositoryList

  si occupa di
  1) far visualizzare all'interno della Listview ListCanzoni l'intero repository di brani.
  2) far visualizzare all'interno di 2 label la canzone selezionata.
  3) far visualizzare la media delle recensioni della canzone selezionata.

- PrintMedia

  metodo utilizzato per far visualizzare all'interno delle varie label il punteggio medio di tutte le recensioni create dai vari utenti.

- PrintReview

  metodo utilizzato per far visualizzare all'interno delle varie label la valutazione completa di un brano selezionato (solo nel caso in cui almeno un utente abbia creato una playlist che contenga quella specifica canzone).

- dataList

  serve per far visualizzare all'interno della users review list solo il nome degli utenti che hanno aggiunto la canzone selezionata allinterno delle varie playlist.

- search

  richiamato nel momento in cui viene eseguita la ricerca di un brano.
  Questo metodo prende la stringa inserita allinterno della barra di ricerca searchBar1 che viene confrontata con tutti gli elementi presenti nel repository di canzoni tramite l'utilizzo di un ciclo (for(String str: arraylist che contiene il repository). Ogni volta che viene trovata una corrispondenza il brano correente viene aggiointo ad un nuovo ArrayList<String> che verrà successivamente visualizzato nella listview dei risultati.

- PasswordValidator

  restituisce true se la password inserita al momento del sign in rispetta i requisiti del pattern.

- idTAXcodeValidator
  restituisce true se il codice fiscale inserito al momento del sign in rispetta i requisiti del pattern.

- emailValidator
  restituisce true se l'email inserita al momento del sign in rispetta i requisiti del pattern.

- openSignin
  serve per aprire la finestra di sign in tramite la creazione di un nuovo FXMLLoader(contiene il file grafico fxml) che verrà caricato allinterno di un nuovo stage.

- signIN
  metodo che prende i vari dati inseriti nei textfield (neme, cognome, indirizzo, email ecc), controlla che siano giusti e se lo sono li passa al metodo Registrazione() della classe Utenti che aggiungerà i dati del nuovo utente all'interno del file UtentiRegistrati.dati.csv.

- logIN
  metodo che prende i vari dati inseriti nei textfield(username e password) e li passa al metodo Login() della classe Utenti che restituisce -1 se se i dati non esistono all'interno del file UtentiRegistrati.dati.csv oppure l'id dell'utente in caso di riscontro positivo.
  Se il valore ritornato dalla classe è >=0 verrà creato un nuovo AnchorPane dentro la quale verrà caricato il file Home.fxml.

- StartPlaylist
  metodo che prende prende il nome della playlist dal textfield GetnamePlaylist e il percorso dell'utente che ha appena effettuato l'accesso.
  Il nome della playlist e il percorso vengono passati al metodo CreaPlaylistVuota() della classe CreaModificaPlaylist che si occuperà di creare un nuovo file.csv (con il nome della playlist appena creata) all'interno del percorso dell'utente che ha appena effettuato l'accesso.

- StopPlaylist
  serve per interrompere la creazione di una playlist, viene richiamato solo se la playlist appena creata contenga almeno un brano.

- canc
  richiama il metodo reset().

- reset
  serve per riportare allo stato iniziale i vari radiobutton e textfield contenuti nella finestra di creazione delle playlist.

- AddSong
  serve per aggiungere al file Emozioni.dati.csv l'id dell'utente che ha appena effettuato l'accesso con un array che contiene tutte le valutazioni da 1 a 5 per ogni emozione e un array che contiene tutti i commenti (facoltativi) per ogni emozione.

- Classe CreaModificaPlaylist
  - getID
    metodo che prende in input una stringa(una riga del file csv con le canzoni)e ne restituisce l'id mediante l'utilizzo di un pattern.
  - GetRepository
    crea un ArrayList<String> che andrà a contenere tutto il file Canzoni.dati.csv.
  - CreaPlaylistVuota
  - AggiungiBrani

Gestione File

Emotional Songs utilizza diversi file per salvare i dati, i più importanti in formati csv, con come separatore principale '**;**'.

Utilizza inoltre un file in formato txt per salvare informazioni minori.

I file utilizzati sono:

- Canzoni.dati.csv

  il file Canzoni.dati.csv è la repository delle canzoni e contiene tutte le informazioni sui brani presenti nel programma

  il file contiene una canzone per ogni riga, per un totale di 2076 brani

  Ogni brano si compone di 6 elementi separati da '**;**', ovvero

  - ID della canzone
  - Titolo del brano
  - Album
  - Artisti, a loro volta separati da '**,**'
  - Data di rilascio in formato AAAA-MM-GG
  - Durata in minuti

- Emozioni.dati.csv

  il file Emozioni.dati.csv contiene tutte le valutazioni lasciate dagli utenti per ogni brano.

  Per ciascuna canzone il file contiene almeno due elementi separati da '**;**', ovvero l'ID della canzone e un array che contiene la media delle valutazioni, composto da 9 elementi separati da '**,**'

che rappresentano la media delle valutazioni per le 9 emozioni, che nel caso in cui la canzone non sia mai stata valutata corrispondono a una stringa di 0 (0,0,0,0,0,0,0,0,0)

Ogni volta che un utente recensisce un brano viene ricalcolata la media (tenendo conto del fatto che sia la prima valutazione per la canzone oppure no) e viene aggiunto un elemento sempre separato da ';' che corrisponde alla valutazione e si compone di:

- ID dell'utente che ha valutato
- 9 recensioni da 1 a 5
- 9 stringhe contenenti i commenti, che in caso non vengano lasciati saranno contati come uno spazio (' ')

Questi 19 elementi sono separati con ','

In ogni stringa (della media, delle valutazioni e delle recensioni) l'ordine delle 9 emozioni è sempre:

- Amazement
- Solemnity
- Tenderness
- Nostalgia
- Calmness
- Power
- Joy
- Tension
- Sadness

o UtentiRegistrati.dati.csv

il file UtentiRegistrati.dati.csv contiene le informazioni degli utenti registrati nell'applicazione.

Ogni riga corrisponde ad un utente, e si compone di 8 elementi separati da ';':

- ID dell'utente
- Nome
- Cognome
- Codice Fiscale (o Tax Code)
- Username
- Indirizzo di residenza
- indirizzo Email
- Password

- ### currentUsrID.txt

  il file currentUsrID contiene l'id dell'utente che sta utilizzando il programma o dell'ultimo utente che lo ha utilizzato

- ### Cartella Utenti e file playlist

  per ogni utente viene creata una cartella (in **data/Utenti**) che contiene tutte le playlist che prendono il nome di "*nome della playlist*.**csv**"che contiene le informazioni delle canzoni aggiunte alla playlist nello stesso formato del file Canzoni.dati.csv

- ### getID(String str)

  metodo che prende in input una stringa(una riga del file csv con le canzoni)e ne restituisce l'id mediante l'utilizzo di un pattern.

- ### GetRepository(String PathRepository)

  crea un ArrayList<String> che andrà a contenere tutto il file Canzoni.dati.csv

- ### CreaPlaylistVuota(String filePath)

  crea un file csv nella path passata come parametro(String filePath).

- ### AggungiBrani(String PathRepository, String filePath, int index)

  metodo utilizzato per aggiungere un brano ad un file csv (la playlist che si stà creando) tramite l'utilizzo di 2 ArrayList<String> (catalogo e playlist).

  Catalogo contiene il repository e playlist contiene il file csv gia esistente(playlist gia esistente). Dato un indice(index) verrà aggiunto all'ArrayList<String> playlist lelemento di catalogo in posizione index.(playlist.add(catalogo.get(index));)