|  |
| --- |
| Note musicali con riempimento a tinta unita  EMOTIONAL  SONGS  Manuale tecnico |
| Progetto Laboratorio Interdisciplinare A – A.A. 2021/22Università degli Studi dell’InsubriaLaurea triennale in Informatica **Sviluppato da:**  • Erik Gurzau, matricola 749400  • Sara Biavaschi, matricola 748698  • Alessia Metaj, matricola 738945 |



Versione 1.0

## Indice

Introduzione …………………………………………………………………………………………………………………………………2

Struttura delle classi …………………………………………………………………………………………………………………2

Playlist …………………………………………………………………………………………………………………………………………….3

PlaylistsManager ……………………………………………………………………………………………………………………….3

Song …………………………………………………………………………………………………………………………………………………4

SongsManager ……………………………………………………………………………………………………………………………4

Emotion ………………………………………………………………………………………………………………………………………….5

EmotionsManager …………………………………………………………………………………………………………………….6

Feedback ………………………………………………………………………………………………………………………………………6

FeedbackManager ……………………………………………………………………………………………………………………7

Person …………………………………………………………………………………………………………………………………………….8

User …………………………………………………………………………………………………………………………………………………8

SecurePassword ……………………………………………………………………………………………………………………….9

UsersManager …………………………………………………………………………………………………………………………….9

FileManager ………………………………………………………………………………………………………………………………10

EmotionalSongs ………………………………………………………………………………………………………………………10

Main ………………………………………………………………………………………………………………………………………………10

Display …………………………………………………………………………………………………………………………………………11

**Struttura delle classi**

L’applicazione è composta dalle seguenti classi:

**Classi principali**

* EmotionalSongs (main)
* Main
* Display

**Managers**

* EmotionsManager
* FeedbackManager
* FileManager
* PlaylistsManager
* SongsManager
* UsersManager

**Classi per la definizione di oggetti**

* Emotion
* Feedback
* Person
* Playlist
* Song
* User

**Altre classi ausiliarie**

* Address
* DisplayColors
* Input
* SecurePassword
* TypeStreet

Nei paragrafi seguenti verranno approfondite nel dettaglio.

**Playlist**

Questa classe è utilizzata per la creazione di oggetti di tipo Playlist, creati a partire dagli input inseriti dall’utente richiesti a terminale, ovvero: nome della playlist e ID delle canzoni da inserire.

Un oggetto di tipo playlist possiede i seguenti campi:

* *userId* di tipo primitivo int
* *name* stringa alfanumerica che rappresenta il nome della playlist scelto dall’utente
* *listSongs* vettore di tipo Integer contenente la lista delle canzoni

L’inserimento delle canzoni all’interno della playlist è gestito dal metodo addSong(), presente all’interno della classe.

**PlaylistsManager**

Questa classe è responsabile della gestione degli oggetti di tipo Playlist, i quali vengono salvati all’interno del file Playlist.txt nel seguente modo:

|  |  |  |
| --- | --- | --- |
| Playlist; | 1; | 45;34;56;67 |

*Nome playlist*

*ID utente*

*ID canzone*

**Struttura dati**

Le playlist sono salvate utilizzando come struttura dati una HashMap (**playlistMap**), che possiede come chiave l’ID utente e come valore la lista di playlist create dall’utente.

**Song**

Questa classe è utilizzata per rappresentare gli oggetti di tipo Song, costituiti dalle singole canzoni presenti nel file Canzoni.txt.

Un oggetto di tipo *song* possiede i seguenti campi:

* *id* di tipo primitivo int, che rappresenta l’id del brano
* *title* di tipo String, che rappresenta il titolo del brano
* *author* di tipo String, che rappresenta l’autore del brano
* *genre* di tipo String, che rappresenta il genere del brano
* *year* di tipo int di 4 cifre, che rappresenta l’anno di produzione del brano
* *duration\_ms* di tipo int di 6 cifre, che rappresenta la durata del brano in millisecondi

All’interno della classe è presente il metodo *millisToTime*, utilizzato per la conversione della durata del brano da millisecondi al formato *minuti:secondi*.

**SongsManager**

Questa classe è responsabile della gestione degli oggetti di tipo Song, che sono salvati all’interno del file Canzoni.txt nel seguente modo:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1; | Titolo; | Autore; | Genere; | Anno; | Durata |

**Struttura dati**

Anche in questo caso viene utilizzata la struttura dati HashMap (**mapSongs**), che possiede come chiave l’ID della canzone e come valore l’oggetto Song.

**Metodi principali**

La classe contiene anche i metodi di ricerca delle canzoni, in particolare *findSongsByTitle()* e *findSongsByAuthorAndYear()*, utilizzati rispettivamente per ricercare una canzone partendo dal titolo o da autore ed anno.

**Emotion**

Questa classe è utilizzata per rappresentare oggetti di tipo Emotion, costituiti dalle emozioni contenute nel file ListaEmozioni.txt.

Un oggetto di tipo *emotion* possiede i seguenti campi:

* *id* di tipo int, che rappresenta l’id dell’emozione
* *category* di tipo String, che rappresenta il nome dell’emozione
* *explanation* di tipo String, che rappresenta la spiegazione dell’emozione

Le emozioni presenti in totale sono 8, ovvero:

|  |  |  |
| --- | --- | --- |
| 1 | **Amazement** | Feeling of wonder or happiness |
| 2 | **Solemnity** | Feeling of transcendence, inspiration. Thrills |
| 3 | **Tenderness** | Sensuality, affect, feeling of love |
| 4 | **Nostalgia** | Dreamy, melancholic, sentimental feelings |
| 5 | **Calmness** | Relaxation, serenity, meditativeness |
| 6 | **Power** | Feeling strong, heroic, triumphant, energetic |
| 7 | **Joy** | Feels like dancing, bouncy feeling, animated, amused |
| 8 | **Tension** | Feeling nervous, impatient, irritated |
| 9 | **Sadness** | Feeling depressed, sorrowful |

**EmotionsManager**

Questa classe è responsabile della gestione degli oggetti di tipo Emotion, che sono salvati all’interno del file ListaEmozioni.txt nel seguente modo:

|  |  |  |
| --- | --- | --- |
| 1; | Nome emozione; | Spiegazione emozione |

**Struttura dati**

In questo caso la lista di emozioni viene salvata all’interno di un oggetto di tipo *Vector<Emotion>*, ovvero array dinamico contenente oggetti di tipo Emotion.

**Metodi principali**

È presente il metodo *getEmotion* che permette di risalire ad una determinata emozione partendo dall’id.

**Feedback**

Questa classe è utilizzata per rappresentare oggetti di tipo Feedback, per gestire l’inserimento di recensioni dei brani da parte dell’utente. Tutte le recensioni vengono salvate nel file Emozioni.txt.

Un oggetto di tipo playlist possiede i seguenti campi:

* *namePlaylist* di tipo String, che rappresenta il nome della playlist in cui è contenuto il brano che l’utente desidera recensire
* *userId* di tipo int, che rappresenta l’id dell’utente che crea la recensione
* *songId* di tipo int, che rappresenta l’id del brano da recensire
* *emotionId* di tipo int, che rappresenta l’id dell’emozione
* *score* di tipo int, che rappresenta il punteggio da 1 a 5 assegnato all’emozione

Il campo *note* di tipo String, che rappresenta l’eventuale commento rilasciato dall’utente per una determinata emozione, è gestito in maniera differente in base al costruttore utilizzato, poiché viene data possibilità all’utente di non inserire alcun commento.

**FeedbackManager**

Questa classe è responsabile della gestione degli oggetti di tipo Feedback, che sono salvati all’interno del file Emozioni.txt nel seguente modo:

Recensione senza nota

Recensione con nota

|  |  |  |  |
| --- | --- | --- | --- |
| NomePlaylist; | ID utente; | 45,1,5, commento; | 67,1,2; |

1 2 3

*1 – ID canzone*

*2 – ID emozione*

*3 – punteggio emozione*

**Struttura dati**

Anche in questo caso viene utilizzata la struttura dati HashMap (**mapFeedback**), che possiede come chiave un codice composto dal nome della playlist e dall’ID utente e come valore la lista di recensioni dell’utente.

**Metodi principali**

All’interno della classe sono presenti i metodi *countFeedback()* e *totScoreFeedback()*, che vengono successivamente utilizzati per il calcolo del punteggio medio associato ad ogni emozione.

Il primo metodo permette di contare il numero totale di recensioni, mentre il secondo calcola la somma dei punteggi assegnati ad una determinata emozione per ogni singolo brano.

Un ulteriore metodo è *listNotes()*, che permette di salvare all’interno di un vettore la lista di tutti i commenti rilasciati dagli utenti riguardo ad un’emozione di un brano.

**Person**

Questa classe viene utilizzata per rappresentare oggetti di tipo Person, che definiscono una persona prima *(che sia completata la sua registrazione???)* che diventi un utente dell'applicazione.

Un oggetto di tipo *Person* possiede i seguenti campi:

* *name*di tipo String, che rappresenta il nome della persona
* *surname* di tipo String, che rappresenta il cognome della persona
* *cf* di tipo String, che rappresenta il codice fiscale della persona e che deve avere una lunghezza di 16 caratteri
* *address* di tipo Address, che rappresenta l’indirizzo della persona

L’oggetto di tipo Address è definito nella classe Address, ed è costituito dai campi typeStreet, nameStreet, houseNumber, postalCode, city e province.

Il campo typeStreet è un’oggetto di tipo TypeStreet definito all’interno di una classe enumerativa contenente i seguenti valori: “via”, “largo”, “piazza”, “corso”, “vicolo”, “viale”.

**User**

Questa classe estende la classe Person e viene utilizzata per rappresentare oggetti di tipo User, che definiscono gli utenti registrati all’interno dell’applicazione assegnandogli uno userID.

Un oggetto di tipo *User* possiede i seguenti campi:

* *name*, *surname*, *cf*, *address*, che vengono ereditati da Person
* *userId* di tipo int, che rappresenta l’ID dell’utente
* *email* di tipo String, che rappresenta l’email dell’utente
* *psw* di tipo String, limitato ad una lunghezza compresa tra 6 e 16 caratteri definiti dai campi final MIN\_LENGTH\_PSW e MAX\_LENGTH\_PSW

**Metodi principali**

All’interno della classe sono presenti i metodi per il controllo della validità dell’e-mail e della password, che devono rispettare determinati requisiti, ovvero:

* isPswValid: lunghezza compresa tra i limiti, presenza di un carattere maiuscolo, di un numero e di un carattere speciale
* isEmailValid: il nome utente deve contenere lettere o numeri, non deve contenere solo numeri e non può iniziare con un numero; deve essere presente il DNS

**SecurePassword**

Questa classe viene utilizzata per criptare la password associata ad un utente al momento della registrazione.

Possiede inoltre il metodo *genPsw()*, che consente la generazione di una password casuale che verrà visualizzata solamente una volta al momento della registrazione.

**UsersManager**

Questa classe è responsabile della gestione degli oggetti di tipo User, che sono salvati all’interno del file UtentiRegistrati.txt nel seguente modo:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID utente; | e-mail; | password criptata; | nome; | cognome; | CF; | indirizzo |

**Struttura dati**

Anche in questo caso viene utilizzata la struttura dati HashMap (**mapUsers**), che possiede come chiave l’e-mail e come valore l’oggetto User.

**Metodi principali**

All’interno di questa classe troviamo i seguenti metodi:

* *nextUserId()*, che assegna ad un nuovo utente che si vuole registrare l'ultimo ID disponibile;
* *register*, che registra un nuovo utente nell'applicazione;
* *login*, che effettua i controlli per l’autenticazione dell’utente a partire da e-mail e password.

**FileManager**

Questa classe si occupa dell’importazione e dell’elaborazione dei file di testo forniti come input.

Viene infatti utilizzata all’interno degli altri manager presenti nell’applicazione per salvare oppure leggere i dati all’interno dei file .txt presenti nella cartella “data”, creando oggetti FileManager che ricevono come input il percorso del file necessario.

**EmotionalSongs**

Questa classe contiene il metodo *main()*, che permette l’avvio e l’esecuzione dell’applicazione.

Il costruttore della classe contiene tutti i gestori e permette l’utilizzo dei relativi metodi in essi definiti.

All’interno della classe sono inoltre implementati i metodi login e logout, che permettono agli utenti registrati di accedere all’applicazione oppure di scollegarsi dall’account.

**Main**

All’interno di questa classe è definito il menu principale dell’applicazione. Sono presenti nove possibili opzioni, che possono essere selezionate dall’utente digitando il numero corrispondente.

**Funzionalità per utenti non registrati**

In particolare, gli utenti non registrati possono usufruire solamente di alcune funzionalità, ovvero:

* ricercare brani con titolo oppure autore ed anno (6)
* registrarsi all’applicazione (2)
* visualizzare un report delle emozioni associate alla canzone selezionata (7)
* visualizzare tutte le canzoni (8)
* uscire dall’applicazione (0)

**Funzionalità per utenti registrati**

Gli utenti registrati che effettuano l’accesso possono invece:

* effettuare il login/logout (1)
* creare le proprie playlist (4)
* visualizzare un report emozionale di tutte le canzoni presenti all’interno della playlist selezionata (3)
* recensire le canzoni inserendo l’intensità delle emozioni provate durante l’ascolto (5)

**Display**

È la classe che gestisce l’**interfaccia utente**.

Permette di visualizzare il menu all’avvio dell’applicazione e di formattare gli output nel terminale in diversi colori e stili, avvalendosi della classe *DisplayColors*, contenente i possibili colori.

Questa classe dispone inoltre di metodi per stampare sotto forma di tabella le liste delle canzoni, delle playlist e dei report emozionali e per visualizzare il nome e l’e-mail dell’utente che ha effettuato l’accesso.