

Emotional Songs - Manuale Utente

Università degli Studi dell'Insubria – Laurea Triennale in Informatica
Progetto Laboratorio B

Sviluppato da:

Claudio Della Motta, matricola 750667

Diana Cantaluppi, matricola 744457

Edoardo Ballabio, matricola 745115

Joele Vallone, matricola 744775

Indice

1	Specifiche dell'applicativo	4
2	Architettura del sistema	4
2.1	Moduli	5
2.2	RMI - Componenti principali	5
2.3	Database	5
3	Requisiti di sistema	5
4	Linguaggi e librerie utilizzati	6
4.1	Linguaggi	6
4.2	Librerie e linguaggi	6
5	Strumento di Build e IDE	6
6	Strumenti e Tecnologie	6
7	Package	7
8	Scelte progettuali e algoritmiche	7
8.1	Pattern utilizzati	7
8.2	Funzione di ricerca	8
8.3	Strutture dati	8
8.4	Gestione dei dati	8
8.5	Coerenza dei dati	9
8.6	Tag emozionali	9
8.7	Calcolo statistiche	9

9 Use Case Diagram	10
10 State Diagram	10
11 Configurazione DbInIt	11
11.1 Step 1	11
11.2 Step 2	11
11.3 Step 3	12
11.3.1 Popolamento manuale della tabella Song	13
12 Configurazione Server	15
12.1 Step 1	15
12.2 Step 2	15
12.3 Step 3	15
12.4 Step 4	16
12.5 Step 5	17
13 Bibliografia e Sitografia	17

1 Specifiche dell'applicativo

L'obiettivo è sviluppare un'applicazione basata su un sistema client-server che permetta agli utenti di consultare dei brani, creare playlist, registrare e gestire dei feedback legati alle emozioni suscitate dal brano, utilizzando una scala da 1 a 5 per valutare un'emozione specifica.

Le valutazioni si basano su una scala di 9 stati emozionali:

- Stupore (Sensazione di meraviglia o felicità)
- Solennità (Sensazione di trascendenza, ispirazione)
- Tenerezza (Sensualità, affetto, sentimento d'amore)
- Nostalgia (Sensazioni sognanti, malinconiche, sentimentali)
- Calma (Relax, serenità, meditazione)
- Potenza (Sentirsi forte, eroico, trionfante, energico)
- Gioia (Voglia di ballare, sentirsi animato, divertito, vivo)
- Tensione (Sentirsi nervosi, impazienti, irritati)
- Tristezza (Sentirsi depresso, addolorato)

L'applicazione sarà in grado di presentare una visione d'insieme delle etichette emotive assegnate dagli utenti per ogni brano. Questo riassunto mostrerà le medie delle emozioni per ciascuno dei diversi stati emotivi, offrendo agli utenti una migliore comprensione delle reazioni emotive associate ad ogni singola traccia. Inoltre, sarà possibile leggere i commenti rilasciati dagli altri utenti.

L'applicazione dovrà risultare semplice, intuitiva e facile da usare. Per fare ciò il sistema implementa un'interfaccia utente intuitiva per la gestione delle playlist, visione dei brani e l'assegnazione delle valutazioni emozionali.

2 Architettura del sistema

L'applicazione è strutturata su un'architettura client-server, dove il server offre metodi remoti tramite RMI che i client possono invocare per interagire con il sistema. Grazie a RMI, la comunicazione tra gli oggetti distribuiti su macchine differenti è semplificata, consentendo ai metodi di essere chiamati come se fossero locali.

I dati visualizzati e inseriti dagli utenti vengono salvati all'interno di un database PostgreSQL interrogato solo dal server.

L'applicazione in totale si compone di 3 moduli: oltre al client e al server, esiste anche un terzo modulo chiamato `EmotionalSongsDbInit` che deve essere eseguito solo una volta e permette di creare la struttura del database e di popolare le tabelle.

2.1 Moduli

- **Client:** Il client è un'applicazione Java che si connette al server per richiedere servizi e interagire con l'utente attraverso un'interfaccia grafica. La sua responsabilità principale consiste nell'avviare le chiamate remote ai metodi resi disponibili dal server. Questo processo coinvolge l'inizializzazione della comunicazione con il server e l'invocazione dei metodi remoti necessari per soddisfare le richieste dell'utente. Inoltre, il client gestisce la risposta del server e aggiorna l'interfaccia grafica dell'utente.
- **Server:** Il server è un'applicazione Java che svolge un ruolo complementare rispetto ai client. Esso espone metodi remoti che i client possono chiamare e si occupa di interagire con il database per soddisfare le richieste dei client.
- **EmotionalSongsDbInit:** L'EmotionalSongsDbInit è un'applicazione Java che svolge un ruolo fondamentale: crea la struttura del database e popola le tabelle. L'applicazione Java si compone di una sola schermata grafica e si connette al database.

2.2 RMI - Componenti principali

- **Classe ClientHandler:** La classe gestisce la comunicazione con il server. Al suo interno sono contenuti tutti i metodi che, invocati da altre classi, richiedono informazioni al server. Questi metodi rappresentano le diverse funzionalità e operazioni disponibili che il server espone al client.
- **Interfaccia ServerHandler:** L'interfaccia ServerHandler definisce i metodi remoti che il server espone ai client. Questi metodi rappresentano le diverse funzionalità e operazioni disponibili per i client tramite la comunicazione remota. Utilizzando questa interfaccia, i client sono in grado di interagire con il server e richiedere l'esecuzione di determinate azioni.
- **Classe Server:** La classe implementa tutti i metodi contenuti all'interno dell'interfaccia **ServerHandler**. Questa a sua volta chiama i metodi contenuti nelle classi che interrogano effettivamente il database.

2.3 Database

Il database è il sistema utilizzato per l'archiviazione e la gestione dei dati all'interno dell'applicazione. Il server e l'EmotionalSongsDbInit, attraverso l'utilizzo delle classi `Connection`, `PreparedStatement` e `Statement` contenute nel package `java.sql`, comunicano con il database per effettuare operazioni come il recupero e l'aggiornamento dei dati in risposta alle richieste provenienti dai client.

3 Requisiti di sistema

Per il corretto funzionamento dell'applicazione è necessario avere un sistema operativo Windows (versione 8 o successive), oppure un sistema operativo Linux con architettura a 64 bit, oppure Mac OS (versione 10.14 o successive) con architettura 64 bit o AArch64.

È necessario avere installata una versione di Java (JDK 17 o successive) e un Database Postgres (versione 14 o successive).

4 Linguaggi e librerie utilizzati

4.1 Linguaggi

- **Java:** linguaggio di programmazione scelto per lo sviluppo dell'intero progetto
- **SQL:** linguaggio utilizzato per permettere al server di interagire con il database PostgreSQL
- **PLPGSQL:** linguaggio utilizzato da PostgreSQL all'interno delle funzioni

4.2 Librerie e linguaggi

Nome	Libreria	Versione	Descrizione
JavaFX	javafx-controls e javafx-fxml	17.0.2	libreria utilizzata per la creazione delle interfaccia grafiche presenti all'interno del progetto. È usata per rendere l'interazione dell'utente con il programma il più user-friendly possibile. Tramite l'uso di FXML, separiamo la parte logica dalla sua rappresentazione grafica tramite l'utilizzo di markup XML
PostgreSQL	postgresql	42.7.0	impiegato per la connessione al database PostgreSQL

5 Strumento di Build e IDE

Il progetto è stato sviluppato con l'IDE IntelliJ IDEA. Per la build del progetto invece è stato usato Maven che semplifica la build dei progetti e la gestione delle dipendenze, facilitando il processo di compilazione e l'aggiunta di librerie esterne al progetto.

6 Strumenti e Tecnologie

- **Scene Builder:** Strumento che offre un'interfaccia utente intuitiva, utilizzata per progettare e creare l'interfaccia grafica dell'applicazione

- **draw.io:** Strumento utilizzato per la creazione dello schema Entità-Relazione (ER) e i diagrammi UML
- **pgAdmin 4:** Utilizzato come strumento per la gestione del database PostgreSQL. Ha permesso di interagire con il database postgres attraverso l'esecuzione di query e comandi psql
- **Java RMI:** Tecnologia implementata nella comunicazione tra client e server, questa consente al client di invocare metodi remoti sul server come se fossero locali, permettendo al client l'utilizzo delle funzionalità del server. Classi e interfacce contenute nel package `java.rmi`
- **Java SQL:** Abbiamo utilizzato il pacchetto `java.sql` per gestire le interazioni con il database PostgreSQL. Questo consente l'esecuzione di query, aggiornamenti e altre operazioni sul database in modo efficiente. Classi e interfacce contenute nel package `java.sql`

7 Package

- **Client:** Il package `"clientES"` contiene tutti gli elementi necessari per gli utenti che utilizzano l'applicazione. Questo include i file per la configurazione dell'interfaccia utente, identificati dall'estensione `"fxml"`, e i relativi controller per il loro controllo. Inoltre, fornisce un'interfaccia per richiamare i metodi in modalità remota. Infine, include anche altre classi Java indispensabili per garantire il corretto funzionamento dell'applicazione
- **Server:** Il package `"serverES"` è fondamentale per la configurazione del server. Al suo interno si trovano sia l'interfaccia che consente agli utenti connessi di invocare i metodi da remoto, sia l'implementazione di tali metodi. Quest'ultima permette anche di stabilire la comunicazione con il database.
- **EmotionalSongsDbInit:** Il package `"com.example.emotionalsongs"` è essenziale per avviare la creazione della base di dati. Include sia i file dedicati alla struttura del database sia quelli per il controllo dell'interfaccia.

8 Scelte progettuali e algoritmiche

8.1 Pattern utilizzati

Nel progetto "Emotional Songs", abbiamo implementato il design pattern `Singleton`, per assicurarci che esista soltanto un'istanza di determinate classi chiave. Questo approccio è stato adottato principalmente per gestire con efficienza la connessione al database.

Le classi che implementano questo pattern sono la `DbConnection` contenuta nel package `serverES` e la classe `ClientHandler` contenuta nel package `clientES`.

In futuro, è possibile considerare l'eliminazione del pattern `Singleton` applicato alla classe `DbConnection` per implementare la gestione di un pool di connessioni, sfruttando ad esempio una libreria fornita da Apache.

8.2 Funzione di ricerca

La ricerca di un brano è stata implementata direttamente lato database, non ci sono algoritmi di ricerca implementati lato codice Java e non avviene nessun caricamento massivo di dati in memoria, gli unici dati caricati sono quelli relativi alla ricerca.

Per velocizzare il processo di ricerca sono stati aggiunti degli indici sulla tabella del database.

A ogni ricerca viene interrogato il database e la condizione `WHERE` della query di ricerca viene costruita dinamicamente in base alla richiesta dell'utente. La costruzione della condizione avviene all'interno di un'apposita funzione valutando i parametri forniti e consentendo di individuare solo le canzoni che soddisfano i criteri di ricerca inseriti.

Il risultato finale visto in tabella vede le canzoni ordinate per *Titolo* e *Autore* in modo ascendente, il tutto è fatto sempre a livello di query all'interno della `ORDER BY` clause.

8.3 Strutture dati

All'interno del progetto per gestire gli elenchi di oggetti della stessa tipologia, utilizziamo le strutture dati fornite dalla libreria Java `java.util`

Nella maggior parte dei casi vengono utilizzate le classi `List` e `ArrayList` e, solo in un caso specifico, usiamo le mappe, nello specifico le classi `Map` e `HashMap`.

Le liste e la mappa sono usate anche come “*oggetto di scambio*” tra client e server, ovvero il server attraverso le opportune query di interrogazioni prende la collezione di dati restituita dal database, li elabora e salvati all'interno della lista/mappa di oggetti della stessa tipologia e li restituisce al client. Ogni richiesta gestisce una lista di oggetti diversa basata sulle necessità del client.

Altre strutture dati usate sono le `ObservableList`, sfruttate all'interno delle classi “*Controller*” per la gestione dei dati contenuti nelle tabelle e nei menù a tendina.

8.4 Gestione dei dati

Come anticipato all'interno del capitolo 8.2 “*Funzione di ricerca*”, non avviene il caricamento in memoria dell'elenco delle canzoni, ma più in generale non viene effettuato un caricamento massivo di dati in memoria, viene tutto calcolato e caricato all'occorrenza. L'unico caricamento di dati che viene effettuato in memoria all'avvio di client e server, è il caricamento dei dati legati alle emozioni (tabella `emotion_details` del database).

Per rendere il processo di ricerca efficiente, abbiamo inserito a livello di database gli indici. Gli indici, all'interno della soluzione, hanno come vantaggio una maggiore velocità nell'interrogazione della tabella, ma portano come svantaggio un lieve aumento della dimensione della tabella, quindi una maggiore dimensione del database. Ad oggi, per la mole di dati contenuta all'interno delle tabelle dove gli indici sono stati aggiunti e per la dimensione del database, l'aumento di dimensione della tabella è trascurabile. Di seguito viene riportata un'analisi dello spazio occupato della tabella `song`, che attualmente si presenta come la tabella con maggior dimensione:

- Dimensione totale della tabella: 101 MB
- Dimensione dei dati contenuti nella tabella: 48 MB

I dati sopra sono ricavati dalla seguente query:

```
select pg_size_pretty(pg_total_relation_size('emotional_songs.song')) as total_size,  
pg_size_pretty(pg_relation_size('emotional_songs.song')) as just_data_size;
```

L'unico svantaggio che si potrebbe verificare a causa di questa scelta implementativa, è un rallentamento del sistema causato da un eventuale traffico verso il database. Ad oggi questo non si verifica perché l'applicazione gestisce pochi dati e il database ha delle dimensioni moderate.

8.5 Coerenza dei dati

Ad ogni modifica dei dati (inserimento, modifica o cancellazione) effettuata dall'utente, il client chiama il server e gli comunica la modifica e in seguito avviene un ricalcolo delle informazioni, ovvero il client richiama il server chiedendo i dati aggiornati. Questo garantisce all'utente di vedere sempre la versione aggiornata e corretta dei dati.

8.6 Tag emozionali

La classe `Emotions` è l'unica classe che carica in memoria dei dati "massivamente" all'avvio del client e all'avvio del server.

Questa classe al suo interno contiene una mappa che ha come chiave un oggetto di tipo `Long` che corrisponde all'id dell'emozione e come valore l'oggetto `Emotion` che rappresenta l'emozione corrispondente. I metodi `getter`, `setter` e la mappa `Map<Long, Emotion> emotionMap` stessa sono statici. Questo ci permette di non dover istanziare la classe, ma al bisogno richiamiamo solo i metodi, di seguito un esempio:

```
Map<Long, Emotion> emotionMap = server.loadEmotionMap();  
Emotions.setEmotionMap(emotionMap);
```

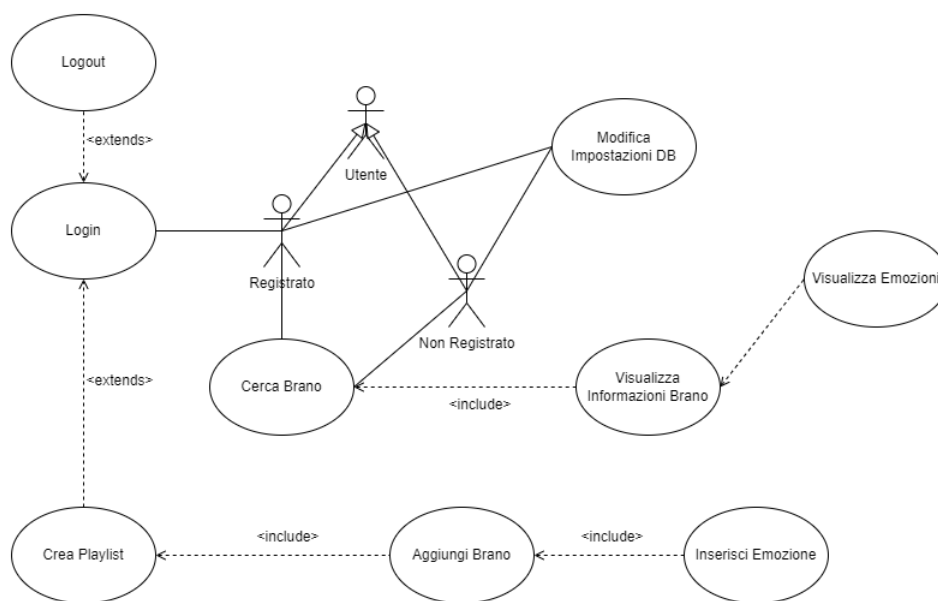
8.7 Calcolo statistiche

Il client offre tramite la classe `EmotionFeedbackStatistics` la possibilità di accedere alle statistiche dettagliate delle canzoni, come la media degli score assegnati per ciascuna emozione e i commenti rilasciati dagli utenti.

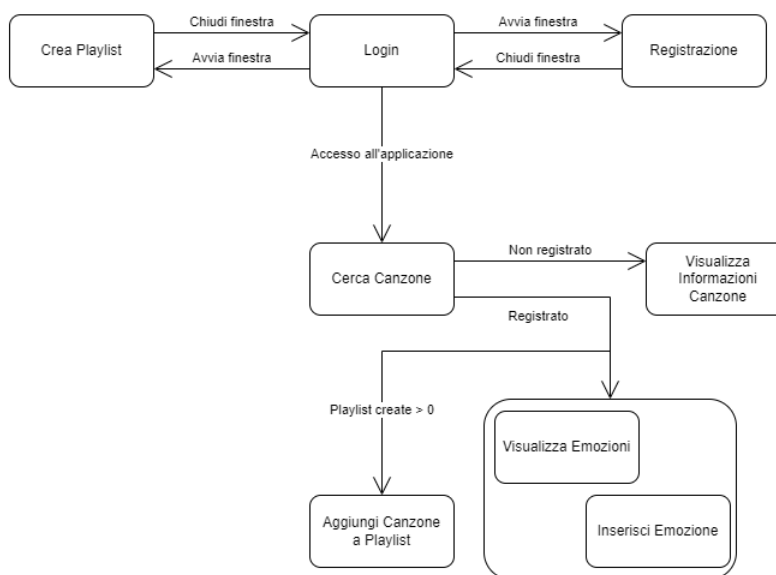
Il calcolo delle statistiche è fatto lato query SQL: la media degli score è calcolata tramite la funzione `AVG` e il numero di utenti che hanno espresso una valutazione è calcolata dalla funzione `COUNT`. Questo viene reso possibile attraverso l'utilizzo della clausola `GROUP BY`, insieme alle funzioni di aggregazione `AVG` e `COUNT`, per ottenere in modo efficiente la media e il conteggio degli score.

Inoltre, per visualizzare i commenti relativi ad una specifica canzone, viene sfruttata la funzione `stringAgg()` messa a disposizione da PostgreSQL, che raggruppa i commenti in una singola stringa. Successivamente, questa stringa viene divisa utilizzando il metodo `split` della classe `String`. Al metodo `split` viene passato come parametro il carattere usato nell'aggregazione (`###`), permettendo così di mostrare a video i singoli commenti.

9 Use Case Diagram



10 State Diagram



11 Configurazione DbInit

Prima di cominciare con la configurazione del database e l'avvio del server è necessario assicurarsi di soddisfare tutti i "Requisiti di sistema" elencati nel capitolo 3. Inoltre, è necessario che per la prima parte di configurazione non ci sia nessun utente connesso al database del progetto (database con nome "dbes").

11.1 Step 1

Assicurarsi che il database postgres sia acceso e che, però, non ci sia nessun utente collegato al database con nome "dbes".

Scompattare la cartella compressa e accedere alla cartella Lab-B-Server all'interno della cartella DbInit. Aprire il command prompt e lanciare il comando `cd {path}/DbInit/Lab-B-Server` sostituendo a {path} il path corretto, nel mio caso:

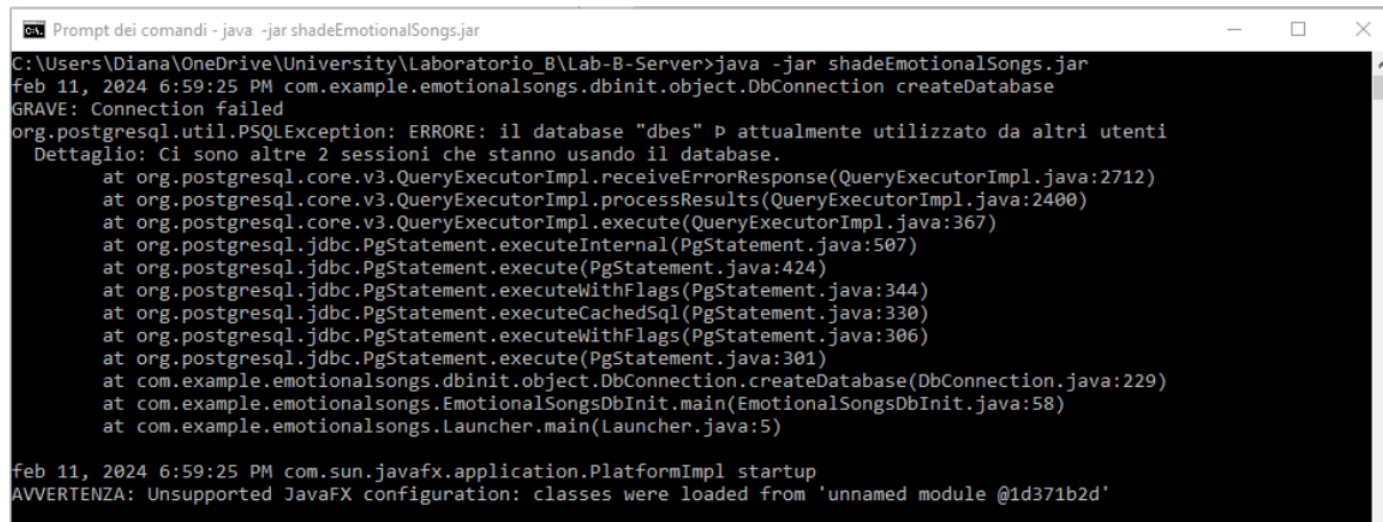
```
cd C:\Users\Diana\OneDrive\University\Laboratorio_B\Lab-B-Server
```

11.2 Step 2

Lanciare il comando `java -jar shadeEmotionalSongs.jar` per avviare il db-init.

Attenzione: la finestra del command prompt NON va chiusa

Attenzione: se quando si avvia l'applicazione con il comando viene loggata un'eccezione del genere (vedi screen sotto), significa che c'è un utente che sta usando il database dbes, è necessario scollegare l'utente per configurare il database. Inoltre è necessario chiudere e riavviare l'applicazione.



```
cmd Prompt dei comandi - java -jar shadeEmotionalSongs.jar
C:\Users\Diana\OneDrive\University\Laboratorio_B\Lab-B-Server>java -jar shadeEmotionalSongs.jar
feb 11, 2024 6:59:25 PM com.example.emotionalsongs.dbinit.object.DbConnection createDatabase
GRAVE: Connection failed
org.postgresql.util.PSQLException: ERRORE: il database "dbes" è attualmente utilizzato da altri utenti
Dettaglio: Ci sono altre 2 sessioni che stanno usando il database.
    at org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2712)
    at org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:2400)
    at org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:367)
    at org.postgresql.jdbc.PgStatement.executeInternal(PgStatement.java:507)
    at org.postgresql.jdbc.PgStatement.execute(PgStatement.java:424)
    at org.postgresql.jdbc.PgStatement.executeWithFlags(PgStatement.java:344)
    at org.postgresql.jdbc.PgStatement.executeCachedSql(PgStatement.java:330)
    at org.postgresql.jdbc.PgStatement.executeWithFlags(PgStatement.java:306)
    at org.postgresql.jdbc.PgStatement.execute(PgStatement.java:301)
    at com.example.emotionalsongs.dbinit.object.DbConnection.createDatabase(DbConnection.java:229)
    at com.example.emotionalsongs.EmotionalSongsDbInit.main(EmotionalSongsDbInit.java:58)
    at com.example.emotionalsongs.Launcher.main(Launcher.java:5)
feb 11, 2024 6:59:25 PM com.sun.javafx.application.PlatformImpl startup
AVVERTENZA: Unsupported JavaFX configuration: classes were loaded from 'unnamed module @1d371b2d'
```

11.3 Step 3

Verrà aperta una schermata chiamata “DATABASE”

The screenshot shows a window titled "DATABASE" with a subtitle "DATABASE CONNECTION INFO". Below the subtitle, there is a paragraph: "Se non si eseguisse l'inizializzazione del database compilando gli appositi campi, verranno usati i valori di default per effettuare la connessione." Below this, there are five input fields: "Username", "Password", "DB Host", "DB Port", and "DB Name". The "DB Name" field contains the text "dbes". At the bottom of the window, there are six buttons: "INIT DB CONNECTION INFO", "TEST CONNECTION", "CREATE DB STRUCTURE", "FILL SONG TABLE", "COMMAND TO FILL SONG TABLE MANUALLY", and "FILL OTHER TABLE".

La sequenza delle operazioni da eseguire è la seguente:

1. Compilare i campi username, password, BD Host e DB Port le configurazioni del proprio database sono diverse da quelle standard (username: postgres, password: postgres, host: localhost, porta: 5432) e premere il bottone “INIT DB CONNECTION INFO”
2. Testare la connessione al db premendo il bottone “TEST CONNECTION”
3. Creare la struttura del database premendo il bottone “CREATE DB STRUCTURE”
4. Popolare la tabella delle canzoni premendo il pulsante “FILL SONG TABLE”. Nel caso in cui l'operazione non vada a buon fine vedere il capitolo “**11.3.1 Popolamento manuale della tabella Song**” prima di proseguire con il punto successivo. Non bisogna chiudere né l'applicazione né la finestra del command prompt
5. Popolare le altre tabelle con il bottone “FILL OTHER TABLE” finché a video non verrà visualizzata la scritta seguente

DB Port

DB Name

OK: procedure correctly performed, database is ready

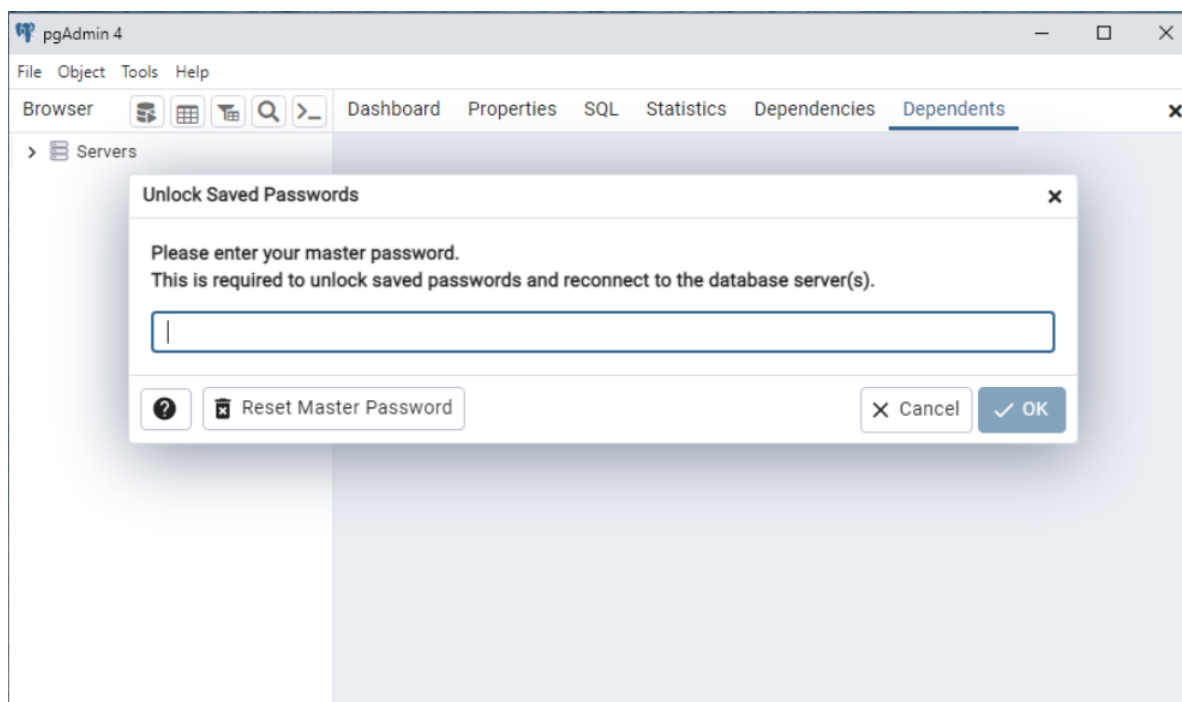
INIT DB CONNECTION INFO **TEST CONNECTION** **CREATE DB STRUCTURE**

FILL SONG TABLE **COMMAND TO FILL SONG TABLE MANUALLY** **FILL OTHER TABLE**

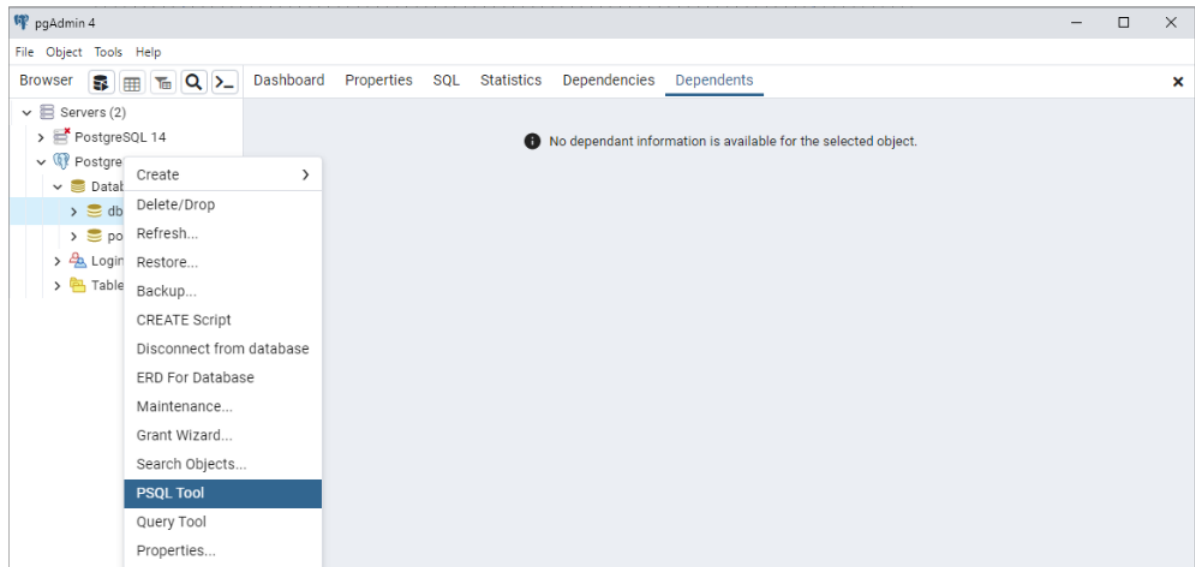
11.3.1 Popolamento manuale della tabella Song

La tabella song non è stata popolata perchè postgres non può accedere al file csv con i dati da inserire nella tabella. Per popolare la tabella è necessario quindi:

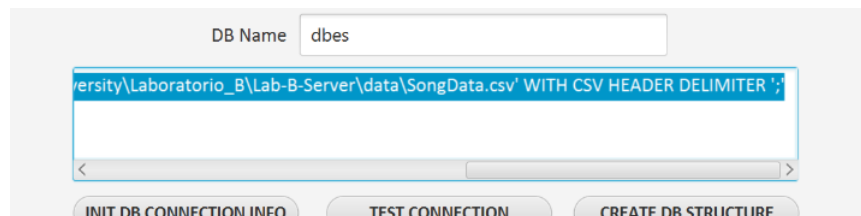
1. Aprire pgAdmin
2. Collegarsi al database



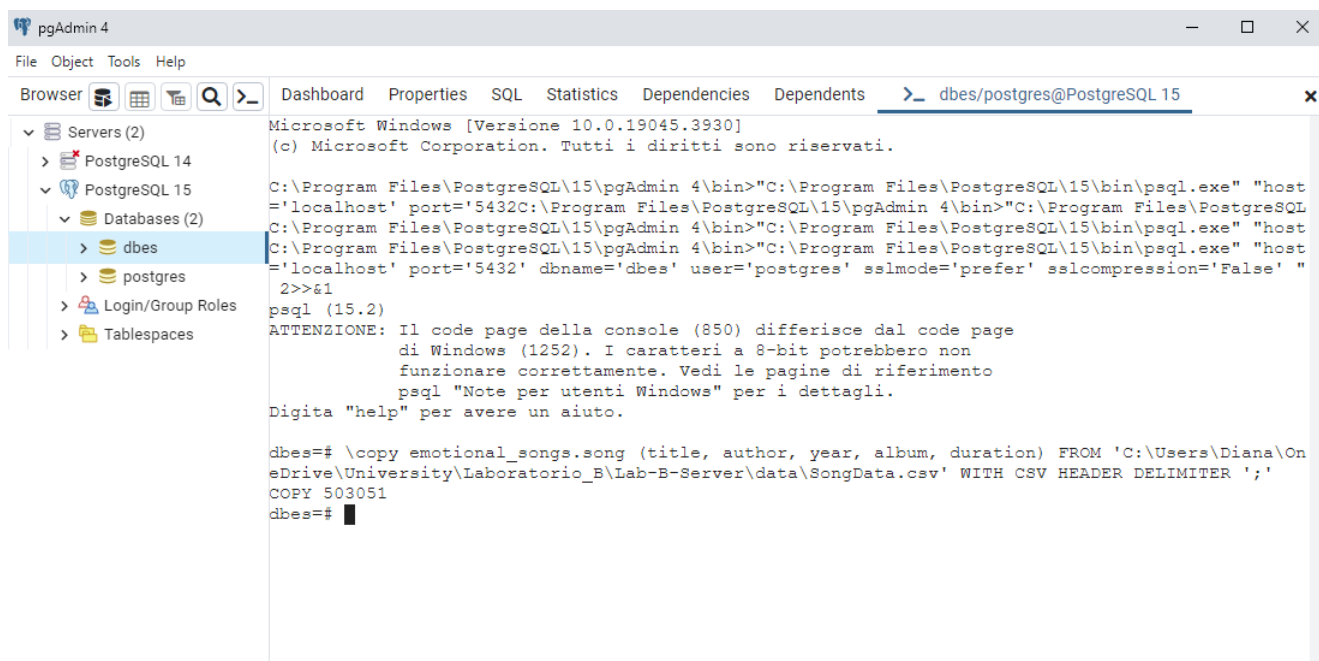
3. Aprire il dettaglio del database nella spalla destra e fare tasto destro sul database con nome "dbes", in seguito selezionare la voce "PSQL Tool"



4. Dall'applicazione Dblnit premere il bottone "COMMAND TO FILL SONG TABLE MANUALLY" e copiare il contenuto della text area



5. Da pgAdmin incollare il comando e premere invio, al termine dell'elaborazione dei dati pgAdmin comunicherà quante righe sono state inserite



```

Microsoft Windows [Versione 10.0.19045.3930]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Program Files\PostgreSQL\15\pgAdmin 4\bin>"C:\Program Files\PostgreSQL\15\bin\psql.exe" "host
='localhost' port='5432' dbname='dbes' user='postgres' sslmode='prefer' sslcompression='False' "
C:\Program Files\PostgreSQL\15\pgAdmin 4\bin>"C:\Program Files\PostgreSQL\15\bin\psql.exe" "host
='localhost' port='5432' dbname='dbes' user='postgres' sslmode='prefer' sslcompression='False' "
2>>&1
psql (15.2)
ATTENZIONE: Il code page della console (850) differisce dal code page
di Windows (1252). I caratteri a 8-bit potrebbero non
funzionare correttamente. Vedi le pagine di riferimento
psql "Note per utenti Windows" per i dettagli.
Digita "help" per avere un aiuto.

dbes=# \copy emotional_songs (title, author, year, album, duration) FROM 'C:\Users\Diana\One
Drive\University\Laboratorio_B\Lab-B-Server\data\SongData.csv' WITH CSV HEADER DELIMITER ';'
COPY 503051
dbes=#
  
```

12 Configurazione Server

Prima di cominciare con la configurazione e l'avvio del server è necessario assicurarsi di soddisfare tutti i “Requisiti di sistema” elencati nel capitolo 3 e assicurarsi che il database postgres sia acceso.

12.1 Step 1

Scompackare la cartella compressa e accedere alla cartella Server-Lab-B.. Aprire il command prompt e lanciare il comando `cd {path}/Server-Lab-B` sostituendo a {path} il path corretto, nel mio caso :

```
cd C:\Users\Diana\OneDrive\University\Laboratorio_B\Server-Lab-B
```


12.2 Step 2

Lanciare il comando `java -jar shadeEmotionalSongs.jar` per avviare il server.

Attenzione: la finestra del command prompt **NON** va chiusa.

12.3 Step 3

Verrà aperta una schermata chiamata “DATABASE”

 DATABASE

— □ ×

DATABASE CONNECTION INFO

Se non si eseguisse l'inizializzazione del database compilando gli appositi campi, verranno usati i valori di default per effettuare la connessione.

Username

Password

DB Host

DB Port

DB Name

INIT DB CONNECTION INFO

TEST CONNECTION

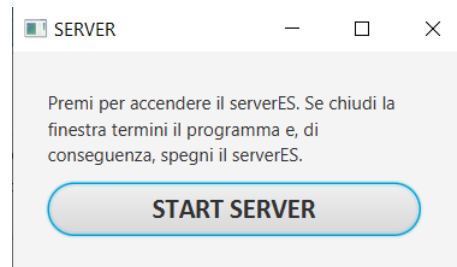
GO TO SERVER

La sequenza delle operazioni da eseguire è la seguente:

1. Compilare i campi username, password, BD Host e DB Port le configurazioni del proprio database sono diverse da quelle standard (username: postgres, password: postgres, host: localhost, porta: 5432) e premere il bottone "INIT DB CONNECTION INFO"
2. Testare la connessione al db premendo il bottone "TEST CONNECTION"
3. Premendo il bottone "GO TO SERVER"

12.4 Step 4

Premere sul bottone "START SERVER" per far partire effettivamente il server.



Attenzione: la finestra del command prompt e l'applicazione NON devono essere chiuse altrimenti il cliente non parte.

12.5 Step 5

Per spegnere il server basta chiudere la finestra.

13 Bibliografia e Sitografia

1. <https://www.oracle.com/>
2. <https://mvnrepository.com/>
3. <https://mvnrepository.com/artifact/org.openjfx/javafx-fxml>
4. <https://mvnrepository.com/artifact/org.openjfx/javafx-controls>
5. <https://mvnrepository.com/artifact/org.controlsfx/controlsfx>
6. <https://mvnrepository.com/artifact/org.postgresql/postgresql>
7. <https://www.postgresql.org/docs/current/indexes.html>
8. <https://www.postgresql.org/docs/current/sql-copy.html>