Relazione del Progetto di Raccomandazione Musicale

Giaconi Christian, Giacomo Rossi
Matricola: 314045, 314671

6 gennaio 2025

Indice

1 Specifica del problema

Scrivere un programma in Haskell e un programma in Prolog per implementare un sistema di raccomandazione di canzoni. Il sistema suggerisce canzoni a un utente basandosi sulle sue preferenze musicali e utilizza un punteggio di gradimento per ordinare le canzoni più popolari o rilevanti.

2 Analisi del problema

2.1 Dati in ingresso

- *Un file contenente le canzoni nel formato:*

```
Titolo,Artista,Genere,Punteggio
```

- *Una lista di generi preferiti.*
- *Pesi numerici assegnati ai generi preferiti.*

2.2 Dati in uscita

- *Una classifica ordinata di canzoni basata sui punteggi ponderati.*
- *Eventuali messaggi di errore o conferma nelle interazioni utente.*

2.3 Relazioni tra i dati

Ogni canzone è caratterizzata da un titolo, un artista, un genere, e un punteggio. Il punteggio ponderato è calcolato come:

$$Punteggio\ Ponderato = Punteggio \times Peso\_Genere$$

se il genere della canzone appartiene ai generi preferiti, altrimenti è uguale al punteggio originale.

3 Progettazione dell'algoritmo

3.1 Scelte di progetto

- *In Haskell, le canzoni sono modellate come tipi di dati strutturati per una manipolazione chiara e leggibile.*
- *In Prolog, si utilizzano predicati dinamici per rappresentare canzoni, generi preferiti e pesi.*

3.2 Passi dell'algoritmo

1. *Caricare le canzoni.*
2. *Inserire le preferenze dell'utente per i generi e i pesi.*
3. *Calcolare i punteggi ponderati.*
4. *Ordinare le canzoni per punteggio ponderato.*
5. *Stampare la classifica.*

4 Implementazione dell'algoritmo

4.1 Implementazione in Haskell

Il file `raccomandazioni.hs` *implementa l'algoritmo in Haskell. Un esempio di calcolo dei punteggi ponderati:*

```
-- ############################################################
-- # Corso di Programmazione Logica e Funzionale              #
-- # Progetto di raccomandazione di canzoni                   #
-- # Studente: Giaconi Christian, Giacomo Rossi               #
-- # Matricola: 314045, 314671                                #
-- ############################################################

{- Specifica:
    Scrivere un programma in Haskell per implementare un sistema
       avanzato di raccomandazione di canzoni.
    Il sistema suggerisce canzoni a un utente in base a:
    - Preferenze per uno o pi  generi musicali specificati.
    - Un sistema di punteggio ponderato per dare priorit  a
       canzoni pi  rilevanti.
    L'utente deve fornire un file di testo con le canzoni nel
       seguente formato:
        Titolo,Artista,Genere,Punteggio
    Dove "Punteggio"    un intero da 1 a 10.
    Le canzoni saranno ordinate in base al punteggio ponderato e
       filtrate per genere.
-}

module Main where

import Data.List (sortOn, nub, intercalate)
import Data.Maybe (mapMaybe)
import Data.Ord (Down(..))
import qualified Data.Map as Map
import System.IO.Error(isDoesNotExistError)
import Control.Exception (catch, IOException)
import Text.Read (readMaybe)

-- ############################################################
-- Definizioni dei tipi di dati
-- ############################################################

-- | La struttura 'Canzone' rappresenta una canzone con:
-- - titolo: il titolo della canzone.
-- - artista: l'artista che la interpreta.
-- - genere: il genere musicale della canzone.
-- - punteggio: un punteggio assegnato (da 1 a 10).
data Canzone = Canzone
    { titolo    :: String
    , artista   :: String
    , genere    :: String
    , punteggio :: Int
    } deriving (Show, Eq)

-- | PesiGeneri    una mappa che associa un genere musicale a un
   peso
-- che influenza la priorit  delle raccomandazioni.
type PesiGeneri = Map.Map String Double
```

```haskell
-- ############################################################
-- Main: Menu interattivo
-- ############################################################

-- | Funzione principale che avvia il menu interattivo.
main :: IO ()
main = menuLoop Nothing Map.empty

-- | Gestisce il menu principale, mantenendo lo stato del sistema:
-- - maybeCanzoni: un elenco opzionale delle canzoni caricate.
-- - pesi: i pesi dei generi preferiti, gestiti dall'utente.
menuLoop :: Maybe [Canzone] -> PesiGeneri -> IO ()
menuLoop maybeCanzoni pesi = do
    putStrLn "\n--- Sistema di Raccomandazione Musicale ---"
    putStrLn "1. Carica un file con le canzoni"
    putStrLn "2. Gestisci i generi preferiti (aggiungi o modifica)"
    putStrLn "3. Stampa la classifica delle canzoni"
    putStrLn "4. Stampa i generi preferiti con il relativo
       punteggio"
    putStrLn "5. Esci"
    putStrLn "Seleziona un'opzione:"
    scelta <- getLine
    case scelta of
        "1" -> caricaCanzoni >>= (`menuLoop` pesi) . Just
        "2" -> selezionaGeneriPreferitiEImpostaPesi maybeCanzoni
           pesi >>= menuLoop maybeCanzoni
        "3" -> raccomandaCanzoni maybeCanzoni pesi >> menuLoop
           maybeCanzoni pesi
        "4" -> visualizzaGeneriPreferiti pesi >> menuLoop
           maybeCanzoni pesi
        "5" -> putStrLn "Grazie per aver usato il sistema di
           raccomandazione. Arrivederci!"
        _   -> putStrLn "Opzione non valida. Riprova." >> menuLoop
           maybeCanzoni pesi

-- ############################################################
-- Funzioni di caricamento e gestione dei dati
-- ############################################################

-- | Carica un file di testo, legge i dati delle canzoni e li
-- trasforma in una lista di Canzone.
-- Il file deve avere un formato valido: Titolo,Artista,Genere,
   Punteggio.
caricaCanzoni :: IO [Canzone]
caricaCanzoni = do
    nomeFile <- chiediNomeFile
    contenuto <- readFile nomeFile
    let canzoni = mapMaybe analizzaCanzone (lines contenuto)
    if null canzoni
        then putStrLn "Errore: il file non contiene dati validi!
           Riprova." >> caricaCanzoni
        else putStrLn "File caricato con successo!" >> return
           canzoni

-- | Richiede all'utente di inserire il nome del file con le
   canzoni
```

```haskell
-- e ne effettua una validazione dell'input tramite la funzione
   validaFile.
chiediNomeFile :: IO FilePath
chiediNomeFile = do
    putStrLn "Inserire il nome del file:"
    nomeFile <- getLine
    esito_lettura <- validaFile nomeFile
    case esito_lettura of
        Right () -> return nomeFile  -- Restituisce il nome del
           file se valido
        Left err -> do
            putStrLn $ "Errore: " ++ err
            chiediNomeFile

-- | Controlla se il nome del file    espresso
-- correttamente e se tale file esiste.
validaFile :: FilePath -> IO (Either String ())
validaFile nomeFile =
    catch (readFile nomeFile >> return (Right ()))
          (\e -> if isDoesNotExistError e
                 then return $ Left "File non trovato!"
                 else return $ Left "Errore durante l'apertura del
                    file.")

-- | Permette all'utente di scegliere
-- i generi preferiti e assegnare un peso a ciascuno di essi.
selezionaGeneriPreferitiEImpostaPesi :: Maybe [Canzone] ->
   PesiGeneri -> IO PesiGeneri
selezionaGeneriPreferitiEImpostaPesi Nothing pesi = do
    putStrLn "Errore: nessun file caricato. Carica un file prima di
        continuare."
    return pesi
selezionaGeneriPreferitiEImpostaPesi (Just canzoni) pesi = do
    let generiDisponibili = nub $ map genere canzoni
    putStrLn $ "Generi disponibili: " ++ intercalate ", "
       generiDisponibili
    generiSelezionati <- raccogliGeneri generiDisponibili
    aggiornaPesi generiSelezionati pesi

-- | Consente all'utente di inserire i generi
-- preferiti uno alla volta, terminando con "fine".
raccogliGeneri :: [String] -> IO [String]
raccogliGeneri generiDisponibili = do
    putStrLn "Inserisci i generi preferiti uno alla volta. Scrivi '
       fine' per terminare."
    loop []
  where
    loop acc = do
        putStrLn "Inserisci un genere preferito:"
        input <- getLine
        if input == "fine"
            then return (nub acc)
            else if input `elem` generiDisponibili
                 then putStrLn ("Genere '" ++ input ++ "' aggiunto
                    ai preferiti.") >> loop (input : acc)
                 else putStrLn "Genere non valido. Riprova." >>
                    loop acc
```

```haskell
-- | Consente all'utente di modificare i pesi dei generi preferiti.
-- Se il genere ha già un peso, l'utente può scegliere di
   mantenerlo o aggiornarlo.
aggiornaPesi :: [String] -> PesiGeneri -> IO PesiGeneri
aggiornaPesi [] pesi = return pesi
aggiornaPesi (g:gs) pesi = do
    let pesoCorrente = Map.findWithDefault 1.0 g pesi
    putStrLn $ "Peso corrente per il genere '" ++ g ++ "': " ++
       show pesoCorrente
    putStrLn "Vuoi aggiornare il peso? (s/n)"
    risposta <- getLine
    if risposta == "s"
        then do
            putStrLn $ "Inserisci il nuovo peso per il genere '" ++
                g ++ "':"
            nuovoPeso <- leggiPesoValido
            aggiornaPesi gs (Map.insert g nuovoPeso pesi)
        else do
            putStrLn $ "Peso per il genere '" ++ g ++ "' invariato.
                "
            aggiornaPesi gs pesi

-- ############################################################
-- Raccomandazioni
-- ############################################################

-- | Genera e stampa una lista di canzoni consigliate
-- basandosi sui pesi dei generi e sui punteggi delle canzoni.
raccomandaCanzoni :: Maybe [Canzone] -> PesiGeneri -> IO ()
raccomandaCanzoni Nothing _ = putStrLn "Errore: nessun file
   caricato. Carica un file prima di continuare."
raccomandaCanzoni (Just canzoni) pesi = do
    let raccomandate = raccomanda pesi canzoni
    if null raccomandate
        then putStrLn "Nessuna canzone trovata con i pesi attuali."
        else stampaClassifica raccomandate

-- ############################################################
-- Funzioni ausiliarie
-- ############################################################

-- | Converte una riga di testo in un oggetto Canzone.
-- Restituisce Nothing se la riga non è formattata correttamente.
analizzaCanzone :: String -> Maybe Canzone
analizzaCanzone riga =
    case separaTaglia ',' riga of
        [titolo, artista, genere, punteggioStr]
            | "" `notElem` [titolo, artista, genere, punteggioStr]
                -- Controlla che tutte le parti siano non vuote
            , Just punteggio <- readMaybe punteggioStr  -- Prova a
               leggere il punteggio
            , punteggio >= 1 && punteggio <= 10 -> Just (Canzone
               titolo artista genere punteggio)  -- Verifica che il
                punteggio sia valido
        _ -> Nothing  -- Restituisce Nothing se la riga non
           valida
```

```haskell
-- | Divide una stringa in una lista di stringhe, usando un
   delimitatore.
separa :: Char -> String -> [String]
separa _ "" = []
separa delimiter string =
    let (primo, resto) = break (== delimiter) string
    in primo : case resto of
        [] -> []
        x -> separa delimiter (dropWhile (== delimiter) (tail x))

-- | Divide una stringa in campi separati, pulendo gli spazi.
separaTaglia :: Char -> String -> [String]
separaTaglia delimiter string = map (filter (/= ' ')) (separa
   delimiter string)

-- | Legge un valore di peso valido inserito dall'utente.
leggiPesoValido :: IO Double
leggiPesoValido = do
    input <- getLine
    case readMaybe input of
        Just peso | peso > 0 -> return peso
        _ -> putStrLn "Peso non valido. Riprova." >>
            leggiPesoValido

-- | Calcola il punteggio ponderato per ogni canzone e le ordina.
raccomanda :: PesiGeneri -> [Canzone] -> [(Double, Canzone)]
raccomanda pesi canzoni =
    let arricchite = arricchisci pesi canzoni
    in sortOn (Down . fst) arricchite

-- | Calcola il punteggio ponderato per ogni canzone.
arricchisci :: PesiGeneri -> [Canzone] -> [(Double, Canzone)]
arricchisci pesi canzoni =
    [ (fromIntegral (punteggio c) * Map.findWithDefault 1.0 (genere
        c) pesi, c) | c <- canzoni ]

-- | Stampa le canzoni ordinate con il loro punteggio ponderato.
stampaClassifica :: [(Double, Canzone)] -> IO ()
stampaClassifica raccomandate =
    mapM_ stampaConPosizione (zip [1..] raccomandate)
    where
        stampaConPosizione (pos, (punteggioPonderato, Canzone
            titolo artista genere _)) = do
            putStrLn $ "#" ++ show pos ++ " - " ++ titolo
            putStrLn $ "   Artista: " ++ artista
            putStrLn $ "   Genere: " ++ genere
            putStrLn $ "   Punteggio ponderato: " ++ show
                punteggioPonderato
            putStrLn "----------------------------------------"

-- | Visualizza i generi preferiti e i pesi associati.
visualizzaGeneriPreferiti :: PesiGeneri -> IO ()
visualizzaGeneriPreferiti pesi
    | Map.null pesi = putStrLn "Nessun genere ancora definito."
    | otherwise = do
        putStrLn "I tuoi generi preferiti e pesi associati sono:"
        mapM_ stampaGenere (Map.toList pesi)
```

```haskell
-- | Stampa il genere, concatenato al peso suo relativo
stampaGenere :: (String, Double) -> IO ()
stampaGenere (genere, peso) = putStrLn $ genere ++ ": " ++ show
    peso
```

Listing 1: raccomandazioni.hs

4.2 Implementazione in Prolog

Il file `raccomandazioni.pl` *implementa l'algoritmo in Prolog. Esempio di ordinamento delle canzoni:*

```prolog
classifica_ordinata(Ordinata) :-
    findall(Punteggio-Titolo, punteggio_ponderato(Titolo, Punteggio),
        Punteggi),
    sort(1, @>=, Punteggi, Ordinata).
```

Listing 2: raccomandazioni.pl

5 Testing

5.1 Testing del programma in Haskell

Test 1

```
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:

Opzione non valida. Riprova.
```

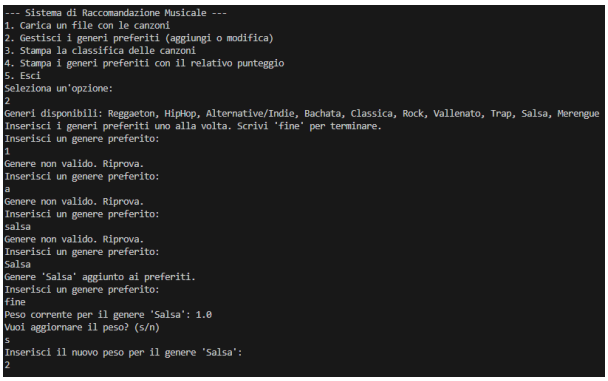
Test 2

```
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
1
Inserire il nome del file:
pippo
Errore: File non trovato!
```

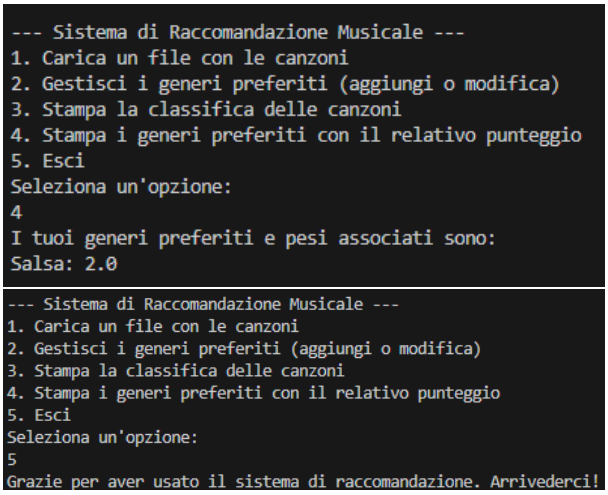
Test 3

```
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
1
Inserire il nome del file:
canzoni.txt
File caricato con successo!
```

Test 4


--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
2
Generi disponibili: Reggaeton, HipHop, Alternative/Indie, Bachata, Classica, Rock, Vallenato, Trap, Salsa, Merengue
Inserisci i generi preferiti uno alla volta. Scrivi 'fine' per terminare.
Inserisci un genere preferito:
1
Genere non valido. Riprova.
Inserisci un genere preferito:
a
Genere non valido. Riprova.
Inserisci un genere preferito:
salsa
Genere non valido. Riprova.
Inserisci un genere preferito:
Salsa
Genere 'Salsa' aggiunto ai preferiti.
Inserisci un genere preferito:
fine
Peso corrente per il genere 'Salsa': 1.0
Vuoi aggiornare il peso? (s/n)
s
Inserisci il nuovo peso per il genere 'Salsa':
2

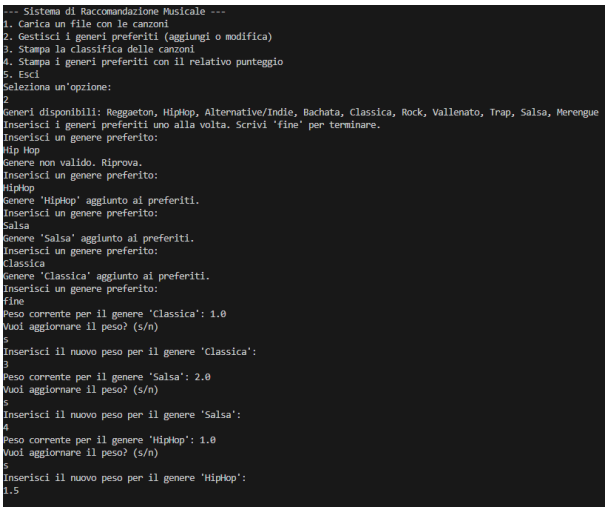
Test 5


--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
4
I tuoi generi preferiti e pesi associati sono:
Salsa: 2.0
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
5
Grazie per aver usato il sistema di raccomandazione. Arrivederci!

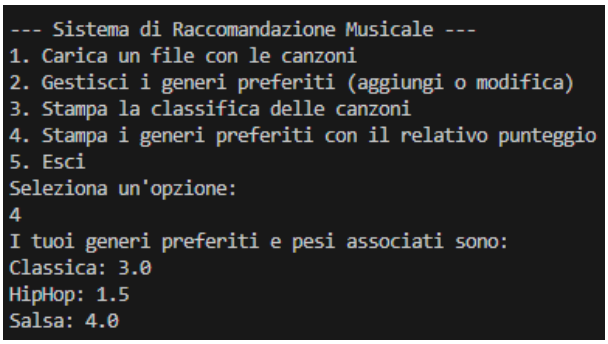
Test 6

```
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
3
#1 - Suavemente
   Artista: ElvisCrespo
   Genere: Merengue
   Punteggio ponderato: 10.0
----------------------------------------
#2 - ManyMen(WishDeath)
   Artista: 50Cent
   Genere: HipHop
   Punteggio ponderato: 10.0
----------------------------------------
#3 - DanzaKuduro
   Artista: DonOmar
   Genere: Reggaeton
   Punteggio ponderato: 9.0
----------------------------------------
#4 - BachataRosa
   Artista: JuanLuisGuerra
   Genere: Bachata
   Punteggio ponderato: 9.0
----------------------------------------
#5 - Stronger
   Artista: KanyeWest
   Genere: HipHop
   Punteggio ponderato: 9.0
----------------------------------------
#6 - ElCantante
   Artista: HectorLavoe
   Genere: Salsa
   Punteggio ponderato: 9.0
----------------------------------------
#7 - LaVaca
   Artista: LosTorosBand
   Genere: Merengue
   Punteggio ponderato: 9.0
----------------------------------------
#8 - Despacito
   Artista: LuisFonsi
   Genere: Reggaeton
   Punteggio ponderato: 8.0
----------------------------------------
#9 - AllEyezOnMe
   Artista: Tupac
   Genere: HipHop
   Punteggio ponderato: 8.0
----------------------------------------
#10 - FreeBird
   Artista: LynyrdSkyryd
```

Test 7


--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
2
Generi disponibili: Reggaeton, HipHop, Alternative/Indie, Bachata, Classica, Rock, Vallenato, Trap, Salsa, Merengue
Inserisci i generi preferiti uno alla volta. Scrivi 'fine' per terminare.
Inserisci un genere preferito:
Hip Hop
Genere non valido. Riprova.
Inserisci un genere preferito:
HipHop
Genere 'HipHop' aggiunto ai preferiti.
Inserisci un genere preferito:
Salsa
Genere 'Salsa' aggiunto ai preferiti.
Inserisci un genere preferito:
Classica
Genere 'Classica' aggiunto ai preferiti.
Inserisci un genere preferito:
fine
Peso corrente per il genere 'Classica': 1.0
Vuoi aggiornare il peso? (s/n)
s
Inserisci il nuovo peso per il genere 'Classica':
3
Peso corrente per il genere 'Salsa': 2.0
Vuoi aggiornare il peso? (s/n)
s
Inserisci il nuovo peso per il genere 'Salsa':
4
Peso corrente per il genere 'HipHop': 1.0
Vuoi aggiornare il peso? (s/n)
s
Inserisci il nuovo peso per il genere 'HipHop':
1.5

Test 8


--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
4
I tuoi generi preferiti e pesi associati sono:
Classica: 3.0
HipHop: 1.5
Salsa: 4.0

Test 9

```
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
3
#1 - ElCantante
   Artista: HectorLavoe
   Genere: Salsa
   Punteggio ponderato: 36.0
----------------------------------------
#2 - Notturnoop55no1
   Artista: Chopin
   Genere: Classica
   Punteggio ponderato: 18.0
----------------------------------------
#3 - ManyMen(WishDeath)
   Artista: 50Cent
   Genere: HipHop
   Punteggio ponderato: 15.0
----------------------------------------
#4 - Stronger
   Artista: KanyeWest
   Genere: HipHop
   Punteggio ponderato: 13.5
----------------------------------------
#5 - AllEyezOnMe
   Artista: Tupac
   Genere: HipHop
   Punteggio ponderato: 12.0
----------------------------------------
#6 - Suavemente
   Artista: ElvisCrespo
   Genere: Merengue
   Punteggio ponderato: 10.0
----------------------------------------
#7 - DanzaKuduro
   Artista: DonOmar
   Genere: Reggaeton
   Punteggio ponderato: 9.0
----------------------------------------
#8 - BachataRosa
   Artista: JuanLuisGuerra
   Genere: Bachata
   Punteggio ponderato: 9.0
----------------------------------------
#9 - LaVaca
   Artista: LosTorosBand
   Genere: Merengue
   Punteggio ponderato: 9.0
----------------------------------------
#10 - Despacito
   Artista: LuisFonsi
```

Test 10

```
--- Sistema di Raccomandazione Musicale ---
1. Carica un file con le canzoni
2. Gestisci i generi preferiti (aggiungi o modifica)
3. Stampa la classifica delle canzoni
4. Stampa i generi preferiti con il relativo punteggio
5. Esci
Seleziona un'opzione:
5
Grazie per aver usato il sistema di raccomandazione. Arrivederci!
```

5.2 Testing del programma in Prolog