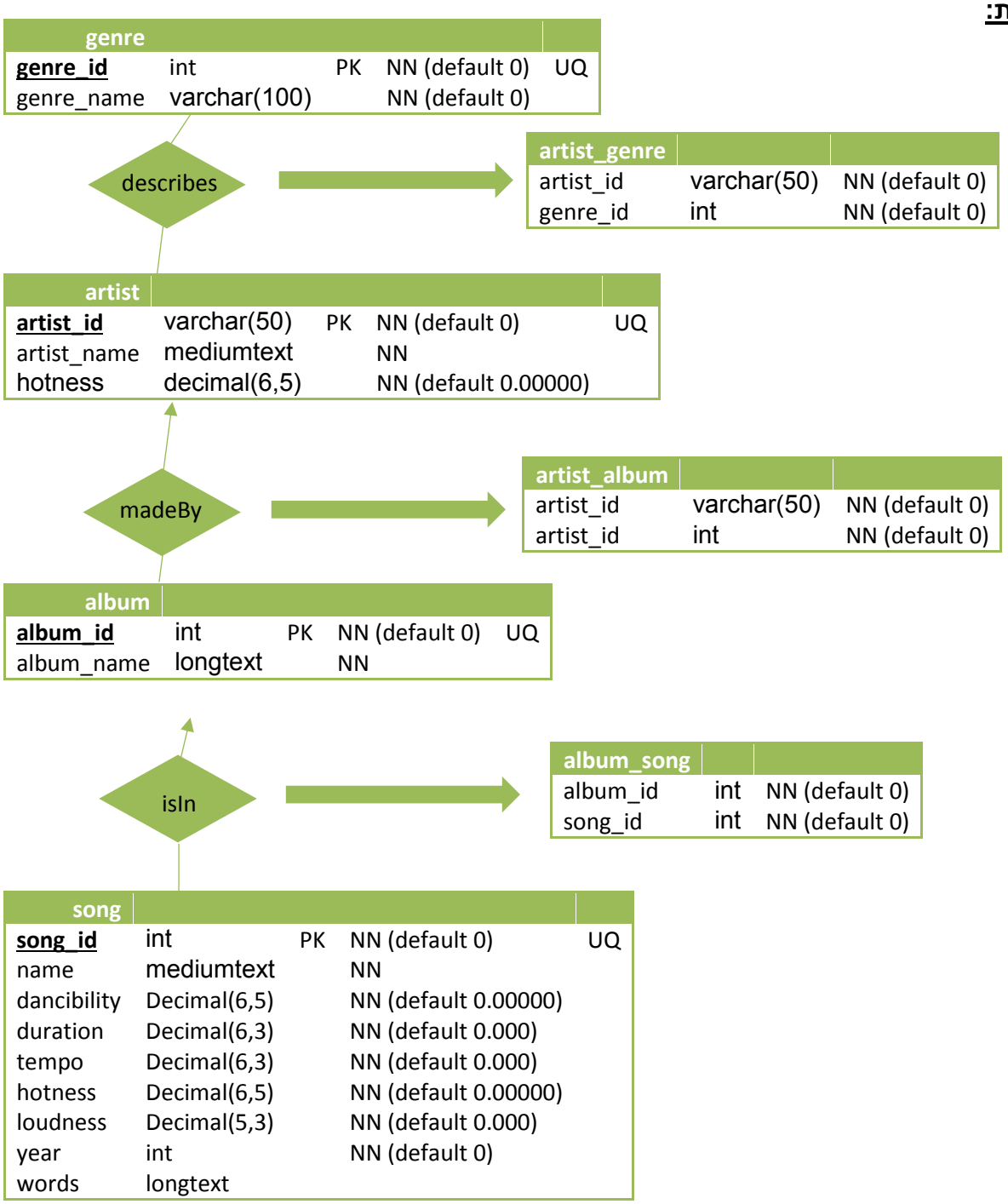
Software documentation

Scheme and Database

סכמות:



תיאור הסכמות

artist:

values:

- artist_id – the unique id of the artist
- artist_name – the name of the artist
- hotness – the popularity of the artist

Indexes and Foreign keys:

- artist_id – an index or artist_id to retrieve the data faster

album:

values:

- album_id – the id of the album
- album_name – the name of the album

indexes and Foreign keys:

- album_id – an index for album_id to retrieve the data faster

artist_album:

values:

- artist_id – the id of the artist
- album_id – the id of the album

indexes and Foreign keys:

- artist_fk_idx – an index for artist_id
- album_fk_idx – an index for album_id
- album_fk – a foreign key to album table -> album_id
  (we weren't able to make fk for artist_id due to duplications in the million song dataset)

song:

values:

- song_id – the id of the song
- name – the name of the song
- dancibility – dancibility rate of the song
- duration – the length of the song (in seconds)
- tempo – the tempo rate of the song
- hotness – the popularity of the song
- loudness – the loudness rate of the song
- year – the year when the song was released
- words – the lyrics of the song

indexes and Foreign keys:

- song_id – the id of the song

album_song:

values:

- album_id – the id of the album
- song_id – the id of the song

indexes and Foreign keys:

- album_idx – an index for album ids to retrieve the data faster
- song_idx - an index for song ids to retrieve the data faster
- album – a foreign key to album table -> album_id
  (we weren't able to make fk for song_id due to duplications in the million song dataset)

כל הטבלאות המקשרות (artist_album, album_song, artist_genre) נועדו להפחית את התלויות בין הטבלאות, וכדי למנוע Delete Anomalies.

בחלק מהטבלאות המקשרות יהיה מידע שחוזר על עצמו, שכן הן יחס many-to-one או many-to-many (למשל, artist_genre: לכל אמן יכולים להיות כמה ז'אנרים ולכל ז'אנר יכולים להיות כמה אמנים שמתאימים לו.

שתי הטבלאות המקשרות הנוספות מתארות יחס many-to-one:

artist_album – לכל אמן יש כמה אלבומים, אבל כל אלבום שייך לאמן אחד.

album_song – כל אלבום מכיל כמה שירים, וכל שיר שייך בדיוק לאלבום אחד).

טבלאות וערכים

genre

1	pop
2	hip hop
3	rock
4	metal
5	blues
6	jazz
7	punk
8	funk
9	rap
10	classical
11	bluegrass
12	r&b
13	gospel
14	soul
15	salsa
16	techno
17	electronic
18	latin
19	swing
20	progresive
21	country
22	world music
23	disco
24	instrumental
25	bossa nova
26	reggae
27	swing
28	folk
29	other

album

album_id – int

album_name – string

album_song

album_id – int

song_id – int

artist

artist_id – char[50]

artist_name – string

hotness – float(0-1)

artist_album

artist_id – char[50]

album_id – int

artist_genre

artist_id – char[50]

genre_id – int

genre

genre_id – int

genre_name – char[100]

song

song_id – int

name -string

dancibility – float (0-1)

duration – float (000.000 – 999.999)

tempo – float (000.000 – 999.999)

hotness – float (0-1)

loudness – float (-99.999 – 99.999)

year – int

words - string

שאילתות

* כל השאילתות שלנו הם שאילתות select, משום שהאפליקציה שלנו לא צריכה לעדכן את מסד הנתונים.

שאילתות לטבלת song:

הטבלה הזו מחזיקה את כל המידע של שירים כגון שם השיר, אורך וכמובן, מזהה ייחודי.

טבלה זו היא הטבלה המרכזית באפליקציה שלנו, ורוב השאילתות מכוונות אליה (שליפת מידע ממנה).

1. יש לנו החזרת ערכים מטבלת song על ידי כמה פרמטרים:
   - תקופה (era), או שנה וגיל רצויים
   - ז'אנרים (רשימה של ז'אנרים רצויים)
   - קצביות השיר
   - פופולריות השיר
   - אורך השיר
   - שם אמן
   - שם אלבום

הפרמטרים שנמצאים בטבלת song הם קצביות, פופולריות, אורך ושנים, ולכן שאילתות המערבות אחד או יותר מפרמטרים אלו ולא מפרמטרים אחרים, הן שאילתות פשוטות (לכל פרמטר מטבלת song חוץ מהשנים נתנו טווח epsilon שיהיה מרחק מסוים מהתוצאה הרצויה של המשתמש על מנת להציג מגוון תוצאות):

* בקוד עצמו שאילתות אלו מקובצות ביחד משום שנתן לשרשר אותן בקלות כמחרוזת אחת ולצרף בסוף כל שאילתה שצריכה לכך.

נניח לשם נוחות שכל ערך שמחפשים נמצא במשתנה val על מנת להציג את השאילתה.

קיימות לנו שאילתות לכל פרמטר בנפרד:

- Select * from song
  Where tempo between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהקצביות שלהם בין טווח הערכים הרצוי.

- Select * from song
  Where popularity between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהפופולריות שלהם בין טווח הערכים הרצוי.

- Select * from song
  Where duration between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהאורך שלהם בין טווח הערכים הרצוי.

- Select * from song
  Where year between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה שלהם בין טווח הערכים הרצוי.

- Select * from song
  Where year=val

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה שלהם היא השנה הנתונה.

קיימות לנו גם כן שאילתות שהם השילובים בין כל הפרמטרים האלו לדוגמה:

- Select * from song
  Where year=val and duration between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה שלהם היא השנה הנתונה והאורך שלהם בטווח הערכים הרצוי.

- Select * from song
  Where year=val and popularity between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה שלהם היא השנה הנתונה והפופולריות שלהם בטווח הערכים הרצוי.

- Select * from song
  Where year=val and tempo between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה שלהם היא השנה הנתונה והקצביות שלהם בטווח הערכים הרצוי.

- Select * from song
  Where year between val – epsilon and val + epsilon and duration between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה והאורך שלהם בטווח הערכים הרצוי.

- Select * from song
  Where year between val – epsilon and val + epsilon and popularity between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה והפופולריות שלהם בטווח הערכים הרצוי.

- Select * from song
  Where year between val – epsilon and val + epsilon and tempo between val – epsilon and val + epsilon

בשאילתה זו אנחנו מחזירים את כל השירים שהשנה והקצביות שלהם בטווח הערכים הרצוי.

הרעיון הכללי הוא שיש לנו שאילתה לכל שילוב אפשרי בין הפרמטרים של השיר.
כלומר, לקיחת כל השירים שעומדים בפרמטרים שהמשתמש הכניס.

הפרמטרים הנוספים נמצאים בטבלאות אחרות ולכן השימוש בהם בשאילתה מצריכה גישה ללפחות 2 טבלאות ויותר.
הרעיון הכללי של כל שאילתה המערבת את הפרמטרים הנוספים:

גם לפרמטרים הנוספים יש לנו כל שילוב אפשרי של שאילתות עם הפרמטרים האלו.

כלומר:

- שאילתה שבהינתן רשימה של ז'אנרים, נחזיר את כל השירים של האמנים שיש להם לפחות אחד מז'אנרים אלו.
  שאילתה זו מצריכה מעבר בכל הטבלאות.

- שאילתה שבהינתן שם אמן, נחזיר את כל השירים האמן הנתון.

```
select * from song, album_song, artist_album, artist
where artist.artist_name=given_artist_name
and artist.artist_id=artist_album.artist_id
and artist_album.album_id=album_song.album_id
and album_song.song_id=song.song_id
```

- שאילתה שבהינתן שם אלבום, נחזיר את כל השירים מאלבום זה.

```
select * from song, album_song, album
where album.album_id=album_song.album_id
and album.album_name=given_album_name
and album_song.song_id=song.song_id
```

- שאילתה שבהינתן רשימת ז'אנרים ושם אמן, נחזיר את כל השירים של האמן אם קיים לו לפחות אחד מהז'אנרים הרצויים.

- שאילתה שבהינתן רשימת ז'אנרים ושם אלבום, נחזיר את כל השירים שבאלבום הזה אם לאמן של האלבום קיים לפחות אחד מהז'אנרים הנתונים.

- שאילתה שבהינתן שם אמן ושם אלבום, נחזיר את של השירים מהאלבום הרצוי של האמן הנתון.

```
select * from song, album_song, artist_album, artist, album
where artist.artist_name=given_artist_name
and artist.artist_id=artist_album.artist_id
and artist_album.album_id=album_song.album_id
and album.album_name=given_album_name
and album_song.song_id=song.song_id
```

- שאילתה שבהינתן רשימת ז'אנרים, שם אמן ושם אלבום, נחזיר את כל השירים של האמן הנתון שנמצאים באלבום הרצוי אם לאמן זה קיים לפחות אחד מהז'אנרים הרצויים.

- שאילתה שבהינתן שם שיר ושם אמן, נחזיר את השיר הנתון של האמן המבוקש.

```
select distinct * from song, album_song, artist_album, artist
where artist.artist_name=given_artist_name
and artist.artist_id=artist_album.artist_id
and artist_album.album_id=album_song.album_id
and album_song.song_id=song.song_id
and song.name=given_song_name
```

כמו כן, לכל שאילתה כזו יש גם שילוב עם הפרמטרים של השיר שהזכרנו קודם. כלומר, יש שאילתות עם כל שילוב אפשרי של הפרמטרים שציינו למעלה.

2. יש לנו החזרת ערכים מטבלת artist על פי הפרמטרים הבאים:
   - מזהה שיר
   - שם אמן
   - קצביות
   - אורך שיר
   - פופולריות שיר
   - תקופה
   - ז'אנר

חלק מן השאילתות:

- Select artist_name from artist
  Where artist_name=given_name
  בשאילתה זו אנחנו מחזירים את האמנים עם השם הנתון.

- Select artist_name from artist, artist_album, album
  Where album_name=given_album_name
  And artist.artist_id=artist_album.artist_id
  And artist_album.album_id=album.album_id

בשאילתה זו אנחנו לוקחים את כל האמנים שיש להם את האלבום הנתון.

הרעיון הכללי הוא כמו במקרה של הטבלת song, יש לנו שאילתות לכל שילוב אפשרי של הפרמטרים המצוינים למעלה.

3. יש לנו החזרת ערכים מטבלת album על פי הפרמטרים הבאים:
   - מזהה שיר
   - שם אלבום

השאילתות:

- select distinct * from song, album_song, album
  where album.album_id=album_song.album_id
  and album_song.song_id=song.song_id
  and song.song_id=given_song_id
  שאלתה זו מחזירה את האלבום של שיר נתון.

- select distinct * from album
  where album.album_id=given_album_id
  שאילתה זו מחזירה את האלבום שניתן המזהה הייחודי שלו.

4. יש לנו שאילתה שבהינתן שם טבלה, מחזירה את כל הערכים בטבלה:

Select * from given_table

<u>הערות:</u>

כלל השאילתות שמצוינות פה הם תמצות של מה שהכנסנו לקוד עצמו.
משום שכפי שציינו, יש המון שילובים של הפרמטרים שנמצאים בשימוש, ובקוד עצמו המקרים החופפים עברו אופטימיזציה וייעול על מנת לא לחזור יותר מדי על השאילתות בקוד.
* יש שימוש ב-count, שאילתות מכוננות, orderBy ועוד (מופיעים בקוד).
לא היה צורך ב-preparedStatements משום שבמקרה שלנו פונקציות תפקדו יותר טוב בחיסכון בשאילתות כפולות.

Code Structure

GUI

הסבר על המחלקות השונות:

ה GUI package בנוי מסט של מחלקות ודפים כאשר עבור כל חלון יש לנו דף fxml, מחלקה של controller ולעיתים גם דף css. דף ה fxml מכיל את שלד החלון כפי שיוצג. מחלקת הcontroller שולטת על הפיצ'רים השונים בדף הfxml ומקשרת אותם לפונקציות מתאימות. דף הcss אחראי על עיצוב החלון.

הגבלנו כל חלון לגודל המוצג שלו על מנת לא לגרום לבעיות תצוגה אם המשתמש בטעות ישחק עם גודל החלון. כמו כן, טיפלנו בכך שאם כפתור מסוים לא תקין פונקציונלית בזמן נתון הוא לא יהיה לחיץ (למשל כפתור לדף הקודם יהיה לא זמין אם זה הדף הראשון).

כעת נתעמק במחלקות הקיימות:

Main:
זוהי מחלקת הmain של האפליקציה כולה.
המחלקה מעלה את דף הפתיחה של התפריט, יוצרת את החיבור לController ונותנת חיווי למשתנה במקרה של תקלה.
בנוסף, בעת סגירת האפליקציה במחלקה זו יסגר גם החיבור.

Connection:
מחלקה זו היא למעשה Singleton כי אנו מאפשרים יצירה של מופע אחד שלה בלבד. כל בקשה (נוספת לראשונה) ליצירה תקבל את המופע הקיים.
המחלקה יוצרת את הcontrollerים השונים שבשימוש(song, artist, album).
כיוון שהמחלקה היא סינגלטון גם מחלקות אלה יווצרו רק פעם אחת.
כאשר פונים אל מחלקה זו עם בקשה, היא מתווכת את הבקשה ל-controller המתאים ומחזירה את התוצאות.

About Controller:
קורא מקובץ חיצוני את המידע על האפליקציה ומציג אותו.
מכיל ניווט חזרה לדף הבית.

Centralizer:
מטרת מחלקה זו היא לתת כלי למרכוז חלון. בכלי זה כל החלונות המוצגים משתמשים.

Alerter:
מטרת המחלקה היא לספק התראות שונות עבור מצבים שונים.
במידת הצורך החלונות השונים ישתמשו בהתראות המתאימות האלו. למשל, התראה על חוסר יכולת להתחבר לDB.

Menu Controller:
שולט על הדף של התפריט. מכיל כפתור אודות, התחלה של חיפוש ויציאה מהאפליקציה.
לחיצה על כפתור של חלון תגרום להצגתו(או סגירה במקרה של exit).

Search:

מחלקה שמכילה פונקציות ומשתנים אשר אחידים עבור שני החיפושים (הרגיל והמתקדם) על מנת למנוע שכפול קוד. כמו כן, המחלקה מכילה פונקציה שתפקידה לשמר ערכים במעבר בין הדפים(חיפוש רגיל ומתקדם), וזאת על מנת שמעבר לחיפוש המתקדם מהחיפוש הרגיל (או ההפך) לא ימחק את הפרטים שהמשתמש כבר מילא בטופס.
בנוסף, המחלקה מכילה פונקציה שבודקת האם הערכים שהמשתמש הכניס תקינים, במידה ולא היא תציג Alert מתאים למשתמש בעזרת המחלקה Alerter.
החיפוש גם מכיל את ההצגה של חלון ההמתנה כאשר מבקשים תוצאות.

Search Controller:
מכיל את המעבר לחיפוש המתקדם והוספת הערכים שהתקבלו על ידי המשתמש בטופס למפה שתישלח בהמשך לשאילתה. מרחיב את המחלקה Search ולכן בעל הפונקציונאליות שנמצאת שם גם כן.

Advanced Search Controller:
מכיל את המעבר לחיפוש הרגיל והוספת הערכים שהתקבלו על ידי המשתמש בטופס למפה שתישלח בהמשך לשאילתה. מרחיב את המחלקה Search ולכן בעל הפונקציונאליות שנמצאת שם גם כן.

Waiting Controller:
מבצע את השאילתה שניתנה לו, כאשר השאילתה מסתיימת הוא סוגר את חלון ההמתנה ופותח את חלון התוצאות.

Results Controller:
מציג את חלון התוצאות עבור השאילתה המבוקשת שהתבצעה. מאפשר לחזור חזרה לחלון הבית או לחלון החיפוש.
מאפשר דפדוף בין דפי התוצאות, כל דף יציג עד 50 תוצאות.
כמו כן, מכיל פיצ'ר "back to the future" אשר מחזיר תוצאות עבור אותם קריטריונים של חיפוש עשור קדימה.

Song Display Data:
מכיל את כל המשתנים האפשריים עבור שיר ומאפשר לקבל אותם במידת הצורך.

Song Info Controller:
מציג את המידע עבור שיר ספציפי שנבחר. ומאפשר לחזור לתוצאות החיפוש.

Controller

ה-Controller אחראי על החיבור בין ה-GUI וה-Model. כלומר - הוא מעביר בקשות מה-GUI אל ה-Model, ובחזרה.

ה-Controller מורכב מ:

1). Interface - ControllerInterface
כל הממשמ אינטרפייס זה מחוייב לממש את המתודה :

```
public TableInfo getInfoFromGUI(Map<String, ArrayList<String>>
infoFromGUI) throws SQLException;
```

המתודה הזו היא האחראית על העברת המידע הנחוץ ל-Model וקבלת והעברת תוצאת החיפוש אל ה-GUI

2). מחלקה אבסטרקטית - ControllerAbstract הממשת את האינטרפייס.

כל אובייקט המרחיב את המחלקה, מחויב לממש את המתודה של האינטרפייס.

תפקידה של המחלקה הזו הוא להיות האחראית היחידה של פתיחת וסגירת החיבור מול ה-Model והחזקת אובייקט הModel. בנוסף, ישנן מתודות משותפות עבור כל סוג של controller למשל - עיבוד המידע שהתקבל מה-GUI. שימוש במחלקה אבסטרקטית מפחית את החזרה על קוד.

<u>3). 3 מחלקות המרחיבות את ControlleAbstract, אשר אחראיות על חיפוש לפי:</u>

שיר (SongController), אמן (ArtistController) או אלבום (AlbumController).

כל אחת מהמחלקות כמובן מממשת את המתודה getInfoFromGUI

כל מחלקה משתמשת ב- QueryInfo המתאים לה: ArtistQueryInfo, AlbumQueryInfo, SongQueryInfo (מידע המסביר על אובייקטים אלו מופיע תחת המחלקות הנ"ל. כאן נסביר את השימוש בהם ב-Controller מתחת לכותרת ההסבר של החיבור בין הcontroller לmodel).

בנוסף, שלושת המחלקות משתמשות ב-dataContainer (מידע מורחב יופיע במחלקה הנ"ל. כאן נסביר את שימושו בcontroller).

<u>בכל אחת מהמחלקות המרחיבות את המחלקה האבסטרקטית, כך מתבצע אופן החיבור:</u>

- מה-**GUI** ל-**Controller**

ה-GUI שולח ל-Controller מידע בתוך MAP. ה-key מכיל את השדות חיפוש שהמשתמש מעוניין בהם. ה-value של כל מפתח הוא הערך המתאים איתו נבצע את החיפוש.
למשל: "key = "era" וה-"value = "20's". כלומר - המשתמש רוצה לחפש שירים לפי שדה era והערך שהוא מעוניין בו הוא כל השירים שיצאו בשנות ה20.

כעת על ה-Controller לעבד את המידע שקיבל מה-GUI לתבנית קבלה המתאימה אצל ה-Model.

- מה-**Controller** ל-**Model**

הcontroller יוצר אובייקט מסוג QueryInfo (המתאים לפי סוג הcontroller) ומכניס אליו את כל המידע הדרוש שקיבל מהGUI, ודרוש כדי לבצע שאילתא נכונה. את הQueryInfo שולחים למודל והוא מחזיר אלינו את הData שחוזר מהחיפוש בDB.

- מה-**Model** ל-**Controller**

הcontroller יוצר אובייקט מסוג DataContainer. לאובייקט זה חוזר המידע מה-Model . ה-DataContainer מכיל בתוכו את תוצאות החיפוש החוזרות מהDB.

כעת על הcontroller לעבד את המידע שקיבל מהmodel לתבנית קבלה המתאימה אצל הGUI.

- מה-**Controller** ל-**GUI**

ה-Controller משתמש באובייקט מסוג TableInfo שלתוכו מוכנסים כל הערכים שחזרו מהחיפוש בDB, לפי התבנית קבלה המתאימה אצל הGUI.

<u>4). מחלקה SongCoparator המשמשת לqueue שהקונטרולרים משתמשים בו על מנת לשלוף שורות של מידע שחוזר מהמודל.</u>

המחלקה מממשת את האינטרפייס של Comparator ואת המתודה compare שלה.

הקומפרטור לוקח את הערכים שהמשתמש בחר (אורך שיר, פופולריות וקצביות – כל הערכים המספריים שנתונים לבחירתו) ומחשב את המרחק שלהם מהערכים של התוצאות שהתקבלו, וכך הוא ממיין אותם לפי סכום המרחקים מכל פרמטר. כלומר, אם הפרמטר נבחר ע"י המשתמש אז הקומפרטור יתחשב בו. מכך נקבל שהתוצאות יוצגו למשתמש לפי הקרבה לתוצאה הרצויה (בעצם ידורגו לפי רלוונטיות המידע).

מה שמעניין:

ה-Controller נכתב תוך התחשבות במודולריות:

- בעזרת שימוש באינטרפייס ומחלקה אבסטרקטית ניתן :
  1. להרחיב את סוגי הControllers שניתן ליצור, במידה והאפליקצייה תתרחב ונרצה להעביר עוד מידע מעבר לשיר,אמן ואלבום.
  2. ניתן להחזיר מידע לGUI בכמה דרכים שונות. כל קונטרולר מחזיר את המידע כפי שנחוץ, ובמידה ונצטרך לשנות דרך של קונטרולר מסויים -השינוי יהיה מזערי ונקודתי, ולא נפגע בשאר הקונטרולרים.
  3. מחיקה של קונטרולר לא תפגע בשאר הקונטרולרים.
- הComparator נכתב בצורה מודולרית: הוא אינו יודע מהם הערכים שנשלחים אליו. הוא רק יודע שעליו לשלוף את המידע המתאים, ולחשב מרחק לפי הפונקציית מרחק שלו.

התבנית של המידע המוחזר מה- SongController נועדה להקל על הGUI:

בעזרת שימוש בcomparator, priority queue ו arraylist המידע שמוחזר ממוין כדי להציג למשתמש את המידע שהוא ביקש בצורה הטובה ביותר (מחזיר קודם כל את הפרמטרים הקרובים ביותר לפרמטרים שהמשתמש הזין על פי פונקציה מסוימת של מרחק מהתוצאות), ומוחזר בבלוקים של 50 שורות, כך שהתצוגה תהיה גם הטובה והנוחה ביותר.

Model

<u>תיאור כללי:</u>

המודל הוא החלק אשר מתקשר עם ה-DataBase**.** הוא היחיד שיוצר איתו קשר ומנהל את החיבור אליו.

המודל מקבל מידע מה-Controller עם פרמטרים שאיתם הוא אמור לבצע שאילתות ל-DataBase**.**

<u>* ה-package שבו יושב החלק של המודל היא DBConnection**.**</u>

<u>המחלקות בחבילה:</u>

- AlbumQueries
- ArtistQueries
- DBModel – המודל עצמו
- Executor
- Model – האינטרפייס

- QueryBuilder
- SongQueries

<u>פירוט המחלקות:</u>

<u>-AlbumQueries</u>

המחלקה אחראית על ביצוע כל השאילתות לטבלת album.

המודל מתקשר עם המחלקה הזו וקורא לפונקציות הנצרכות שלה בהתאם לפרמטרים נתונים.

המחלקה מחזיקה פונקציות שכל אחת מהן היא שאילתה נצרכת לטבלה.

<u>-ArtistQueries</u>

המחלקה אחראית על ביצוע כל השאילתות לטבלת artist.

המודל מתקשר עם המחלקה הזו וקורא לפונקציות הנצרכות שלה בהתאם לפרמטרים נתונים.

המחלקה מחזיקה פונקציות שכל אחת מהן היא שאילתה נצרכת לטבלה.

<u>-DBModel</u>

המחלקה העיקרית של המודל. מחלקה זו מקבלת מהקונטרולרים את המידע הנצרך לשאילתות. המודל בודק מידע זה ועל פיו קורא לפונקציות המתאימות במחלקות שמחזיקות את הפונקציות שמבצעות את השאילתות.

המודל:

- מנתב את המידע שמתקבל לפונקציות המתאימות על מנת לבצע אותן.
- מחזיר את המידע שהתקבל מביצוע השאילתות ושולח בחזרה לקונטרולרים.
- לוקח את הפרמטרים של שנה, אורך שיר, קצביות ופופולריות ונותן טווח ערכים חדש על ידי הוספה והחזרה של אפסילון מוגדר לכל ערך. כך המשתמש יקבל טווח ערכים גדול הקרובים לתוצאות הרצויות שלו.

<u>-Executor</u>

המחלקה המוציאה לפועל של השאילתות.

כל מחלקה שמחזיקה פונקציות של שאילתות בונה את השאילתות ושולחת אותם למחלקה הזו.

המחלקה מבצעת את השאילתה ומאגדת את הנתונים שחוזרים בצורה נוחה לשימוש.

<u>-Model</u>

האינטרפייס של המודל. מבטא את הפונקציונליות של המודל.

כל מי שמרחיב את האינטרפייס צריך לממש את הפונקציות:

- openConnection
- closeConnection
- getData – לכל סוג מחזיק דאטה שיש (שלושה)

-QueryBuilder

מחלקה של Builder Design Pattern. המחלקה בונה שאילתה עם הפונקציות הפנימיות שלה.

השימוש במחלקה נעשה בחלק מהשאילתות הפשוטות יותר.

המחלקה נבנתה לבניית כמעט כל סוג של שאילתה (לדוגמא: מכילה where, and, between וגם leftJoin).

פותחים את השאילתה ביצירה של האובייקט ששם הוא משרשר "select <fields> from <tables>", כאשר fields אלו השדות הרצויים המוזנים לבנאי וtables אלו הטבלאות המוזנות לבנאי. לאחר שמסיימים לבנות את השאילתה מפעילים את פונקציית הtoString ומקבלים את השאילתה המוכנה כ-String.

-SongQueries

המחלקה אחראית על ביצוע כל השאילתות לטבלת song.

המודל מתקשר עם המחלקה הזו וקורא לפונקציות הנצרכות שלה בהתאם לפרמטרים נתונים.

המחלקה מחזיקה פונקציות שכל אחת מהן היא שאילתה נצרכת לטבלה.

Resources

ה-package של ה-resources זו החבילה שבה נמצאים כל המחלקות המשותפות בין החלקים השונים. כגון, מחזיקי מידע המשמשים להעברת נתונים.

המחלקות בחבילה זו:

-Container

אינטרפייס גנרי שכל מי שמרחיב אותו חייב לממש את הפונקציה getValue. מי שממש אותו נחשב מחזיק של מידע.

כל המחלקות המממשות את container-

המחלקות:

- ArtistContainer
- AlbumIdContainer
- DurationContainer
- FamiliarityContainer
- GenreContainer
- LyricsContainer
- PopularityContainer
- SongIdContainer
- TempoContainer

כל המחלקות האלו הם מחזיקים של מידע כפי שמתואר בשם שלהם. הסיבה לכך היא משום שהמודל צריך להבדיל בין סוגי מידע שונים שיכולים להיות מאותו טיפוס (בעיקר String). משמשים אך ורק על מנת להעביר מידע.

-AlbumQueryInfo

מועבר למודל מה-AlbumController ומחזיק את כל המידע שיכול להיות לגבי שאילתה לטבלת ה-album.

-ArtistQueryInfo

מועבר למודל מה-ArtistController ומחזיק את כל המידע שיכול להיות לגבי שאילתה לטבלת ה-artist.

-SongQueryInfo

מועבר למודל מה-SongController ומחזיק את כל המידע שיכול להיות לגבי שאילתה לטבלת ה-song.

-AlertMessages

מחלקה שמחזיקה הודעות כלליות לגבי שגיאות שיכולות לקרות. משמש על מנת שההודעות יהיו אחידות ומתקבלות ממקור אחד. מצמצם שינויים עתידיים במידת הצורך.

-DataContainer

מחלקה המחזיקה מידע לגבי תשובות של שאילתה שהתבצעה.

מוחזר ל-controllers מהמודל.

מחזיק את המידע עצמו, מספר השורות והעמודות ושאת השדות שהוחזרו.

-DBConnectionException

סוג exception חדש. משמש על מנת להבדיל בין סוגי שגיאות שקורות.

לדוגמא:

אם השאילתה החזירה תשובה ריקה אז נזרוק exception עם ההודעה "empty results".

-TableInfo

מחלקה המחזיקה את המידע שחוזר מהמודל לאחר שעבר מניפולציה של ה-controller. המידע תחילה עובר ב-controller, ואז הוא מחלץ את המידע, מעבד אותו ומכניס אותו לאובייקט חדש מסוג זה. לאחר מכן שולח את האובייקט ל-GUI שמציג את התוכן שלו.

Data Source

השתמשנו במסד הנתונים million song dataset (ב subset).

השתמשנו בסקריפטים מ github של המסד בשביל לחלץ את המידע מקבצי H5 לקובץ CSV.

המידע שהתקבל נכתב לקובץ SongCSV.csv. מהקובץ יצרנו קבצי CSV נוספים כאשר כל אחד מהם מכיל את העמודות הרלוונטיות לטבלה אחרת. את הקבצים הללו הכנסנו ל database בעזרת ה MySQL workbench.

את העמודות הנוספות הכנסנו בעזרת סקריפט שכתבנו בפייתון CreateLyricsTable.py.

External Packages

השתמשנו ב 2 חבילות חיצוניות בשביל החיבור של java ל SQL:

1. MySQLConnector
2. JDBC

זרימת האפליקציה

המשתמש מתחיל במסך הבית. במסך זה ישנם שלושה כפתורים, כפתור About, כפתור Search וכפתור Exit.

בלחיצה על:

- About, המשתמש יועבר לחלון המציג מידע כללי על האפליקציה.
- Exit, התוכנית תיסגר.
- Start, המשתמש יועבר לחלון החיפוש הרגיל בו יוכל להזין את הפרמטרים הרצויים לשירים שהוא רוצה. כמו כן, יוכל ללחוץ על כפתור ה-Advanced והוא יועבר לחלון חיפוש מתקדם בו יוכל להזין פרמטרים נוספים לחיפוש השירים.

לאחר שהמשתמש מזין את הפרמטרים של החיפוש הוא לוחץ על כפתור ה-Go to results והפרמטרים שלו יקלטו באפליקציה ויעברו עיבוד.

אחרי שהאפליקציה עיבדה את נתוני החיפוש המשתמש יועבר לחלון התוצאות בו יוצגו למשתמש תוצאות החיפוש שלו.

המשתמש יכול לראות את כל תוצאות החיפוש בקבוצות של 50 (50 לכל דף), והוא יכול לבחור שיר ספציפי על מנת לראות עליו פרטים נוספים (יועבר לחלון תצוגה של שיר בודד). מכאן המשתמש יכול לחזור לדף התוצאות.

מדף התוצאות ניתן לחזור לעמוד הבית או לדף החיפוש. כמו כן, ניתן ללחוץ על כפתור Back to the future ולקבל תוצאות של אותו חיפוש על שירים מהעשור הבא.

אלגוריתמים ומתודות

מימשנו comparator אשר אחראי על למיין את התוצאות המתקבלות מהשאילתה על פי נתונים מסוימים מהשאילתה.

אנחנו נותנים ל-comparator את הערכים המספריים שהמשתמש הכניס שהם אורך שיר, קצביות, ופופולריות.

ה-comparator לוקח כל ערך כפי שציינו (אם הוזן) ומחסיר ממנו את הערך המקביל של השיר הנוכחי שעוברים עליו. לאחר ההחסרה על כל ערך עושים על התוצאה ערך מוחלט וסוכמים את התוצאות, וזהו המרחק.

לדוגמא:

הערכים הרצויים הם אורך שיר 240 שניות, פופולריות 0.1 וקצביות 0.1 (ערכים בדיוניים).

כעת נניח שאנחנו עוברים על שיר שהתקבל מהשאילתה עם הערכים: אורך שיר 230, פופולריות 0.5 וקצביות 0.1.

נחשב את המרחק-

$$|240-230|+|0.1-0.5|+|0.1-0.1|=10+0.4+0=10.4$$

ולכן המרחק של השיר הנתון מהפרמטרים הרצויים של המשתמש הוא 10.4.

אנחנו נותנים את ה-comparator ל-queue על מנת שיפעל על פיו. וכאשר אנחנו שולפים ממנו, הערכים יוצאים ממוינים על פי המרחק שלהם ממה שהמשתמש רצה, שזה בעצם כמו דירוג התוצאות הכי קרובות למבוקש. מה שיהיה יותר קרוב יוצג קודם.

Special features:

- חלון המתנה בזמן שהמשתמש ממתין שהאפליקציה תעבד את המידע שלו ותחזיר תוצאות.
  החלון הוא חלון אקטיבי (יש בו gif של טעינה).
  על מנת ליצור חלון זה ועל מנת שהוא יתחבר ויעבוד היה צורך בשימוש ב-threads ושימוש בקוד מיוחד המאפשר לקוד ב-thread להתחבר חזרה לui thread (ה-thread הראשי).
- כל חלון בתצוגת האפליקציה עוצב אישית על ידי חברות הצוות.
- בחלון התוצאות ניתן לבחור שיר על ידי לחיצה עליו ועל כפתור ה-display או על ידי לחיצה כפולה על השורה בתוצאות. פיצ'ר זה דרש לערוך את ה-event של הלחיצה של כל שורה שמכניסים לטבלת התוצאות.
- ישנם 2 חלונות חיפושים וניתן לעבור ביניהם. טיפלנו בכך שיהיה מעבר מידע בין החלונות על מנת לשמר את הבחירה של המשתמש.
- יש חלון אודות על האפליקציה. את המידע שאנחנו מציגים שם אנחנו קוראים מקובץ חיצוני. דבר זה מאפשר לנו לשנות את תוכן החלון ללא שינוי בקוד.