Bazy danych - I

Temat projektu: platforma muzyczna z funkcjami społecznościowymi.

Serhii Vynohradov

24 stycznia 2025

1 Opis projektu

1.1 Zdefiniowanie tematu projektu

Tematem projektu jest serwis muzyczny służący do przechowywania informacji o piosenkach, artystach i płytach muzycznych. Każdy użytkownik jest w stanie dodać nową piosenkę, płytę i artystę oraz interagować z wprowadzonymi danymi. Każdą piosenkę lub album można ocenić, a do każdej piosenki można dodać recenzję. Serwis prezentuje również średnią ocenę każdej piosenki i albumu. Strona główna serwisu wykorzystuje układ typu 'grid', losowo wyświetlając zawartość bazy danych, ułatwiając użytkownikom odkrywanie muzyki.

1.2 Analiza wymagań użytkownika

Zaprojektowana baza danych powinna spełniać następujące funkcjonalności:
Użytkownicy serwisu powinni mieć dostęp do informacji o piosenkach, płytach i artystach oraz możliwości wyszukiwania tych danych. Dla zalogowanych użytkowników ważne jest tworzenie i zarządzanie playlistami, a także interakcja z innymi użytkownikami przez obserwowanie, recenzje i playlisty. Niezalogowani użytkownicy powinni mieć możliwość przeglądania danych, oceniania utworów i czytania recenzji.

1.3 Zaprojektowanie funkcji

Podstawowe funkcje realizowane w bazie danych:

- Możliwość dodawania nowych piosenek, artystów, płyt
- Wizualne przedstawienie danych w postaci grid
- Możliwość wyszukiwania wprowadzonych danych
- Możliwość oceniania piosenek/płyt
- Możliwość tworzenia playlist z piosenkami i dzielenie się z pozostałymi użytkownikami
- Możliwosc obserwowania innych użytkowników.

2 Projekt diagramów

2.1 Zdefiniowanie encji (obiektów) oraz ich atrybutów.

Poniżej zostały przedstawione wszystkie tabele wraz z ich atrybutami oraz kluczami:

1. **Tabela:** `bd.artist`
   - **Opis:** Przechowuje informacje o artystach.
   - **Kolumny:**
     - id serial primary key - Unikalne id artysty
     - name varchar(64) unique not null - Imie artysty
     - career_start int not null - Początek kariery artysty
     - career_end int - Koniec kariery artysty
   - **Constraints:** Sprawdzają, by kariera artysty zaczynała się co najmniej w roku 1700, a kończyła się w roku obecnym. Również sprawdzają, by początek nie był już po zakończeniu.

2. **Tabela:** `bd.song`
   - **Opis:** Przechowuje informacje o piosenkach.
   - **Kolumny:**
     - id serial primary key - Unikalne id piosenki
     - name varchar(64) not null - Nazwa piosenki
     - released int - Rok napisania piosenki
   - **Constraints:** Sprawdzają, by piosenka mieściła się w przedziale od 1700 do roku obecnego.

3. **Tabela:** `bd.artist_song`
   - **Opis:** Przechowuje połączenia między piosenką a artystą
   - **Kolumny:**
     - id serial primary key - Unikalne id relacji między piosenką i artystą
     - song_id int not null - ID piosenki
     - artist_id int not null - ID artysty
   - **Constraints:** Sprawdzają, by połączenie artysty i piosenki było unikalne.
   - **Klucze obce:** song_id nawiązuje do tabeli `bd.song`, natomiast artist_id do tabeli `bd.artist`.

4. **Tabela:** `bd.album`
   - **Opis:** Przechowuje informacje o albumach.
   - **Kolumny:**
     - id serial primary key - Unikalne id albumu
     - artist_id int not null - ID artysty
     - name varchar(64) not null - Nazwa albumu
     - released int - Rok napisania albumu
     - cover BYTEA - Okladka albumu
   - **Constraints:** Analogicznie do powyższych założeń, sprawdza się, czy rok napisania mieści się w określonej dacie.
   - **Klucze obce:** Klucz artist_id nawiązuje do tablicy `bd.artist`.

5. **Tabela:** `bd.song_album`

   - **Opis:** Działa podobnie do `bd.artist_song`, przechowując relację między tabelami `bd.artist_song` oraz `bd.album`.
   - **Kolumny:**
     - id serial primary key - Unikalne id relacji między piosenką a artystą
     - song_artist_id int not null - ID artysty
     - album_id int not null - ID płyty
     - run_order int - Porządek w którym występują piosenki
   - **Klucze obce:** Klucz song_artist_id nawiązuje do tablicy `bd.artist_song`, natomiast klucz album_id do tablicy `bd.album`.

6. **Tabela:** `bd.user`

   - **Opis:** Przechowuje dane użytkowników.
   - **Kolumny:**
     - id serial primary key - Unikalne ID użytkownika
     - username varchar(32) not null unique - Unikalne imie użytkownika, używane również jak informacja do logowania
     - password varchar(32) not null – Hasło użytkownika
   - **Constraints** Sprawdzają, by imię użytkownika było unikalne.

7. **Tabela:** `bd.songrating`

   - **Opis:** Przechowuje oceny piosenek.
   - **Kolumny:**
     - id serial primary key - Unikalne id wystawionej piosence oceny
     - user_id int not null - ID użytkownika, który ocenił utwór
     - song_id int not null - ID ocenionej piosenki
     - rating int not null - Ocena
   - **Constraints:** Klucz rating jest liczbą całkowitą od 1 do 5, a dla jednej piosenki użytkownik może mieć maksymalnie jedną ocenę.
   - **Klucze obce:** Klucz user_id nawiązuje do tablicy `bd.user`, natomiast song_id do tablicy `bd.artist_song`

8. **Tabela:** `bd.albumrating`

   - **Opis:** Przechowuje oceny piosenek.
   - **Kolumny:**
     - id serial primary key - Unikalne id wystawionej płycie oceny
     - user_id int not null - ID użytkownika, który ocenił płytę
     - song_id int not null - ID ocenionej płyty
     - rating int not null - Ocena
   - **Constraints:** Analogiczne do poprzedniej tablicy.
   - **Klucze obce:** Klucz user_id nawiązuje do tablicy `bd.user`, natomiast album_id do tablicy `bd.album`

9. Tabela: `bd.playlist`

- **Opis:** Przechowuje informacje o playlistach.
- **Kolumny:**
  - id serial primary key - Unikalne ID utworzonej playlisty
  - name varchar(64) not null unique - Unikalna nazwa playlisty
  - user_id int not null - ID użytkownika który tworzył playlistę
- **Klucze obce:** Klucz user_id nawiązuje do tablicy `bd.user`.

10. Tabela: `bd.playlistsong`

- **Opis:** Przechowuje połączenia między playlistami a piosenkami.
- **Kolumny:**
  - id serial primary key - Unikalne ID połączenia mięzy piosenką a playlistą
  - playlist_id int not null - ID playlisty
  - song_id int not null - ID piosenki
- **Constraints:** Połączenie playlist_id i song_id ma być unikalne, czyli w jednej playliście nie znajdziemy powtarzających się piosenek.
- **Klucze obce:** Klucz playlist_id nawiązuje do tablicy `bd.playlist`, natomiast song_id do tablicy `bd.artist_song`.

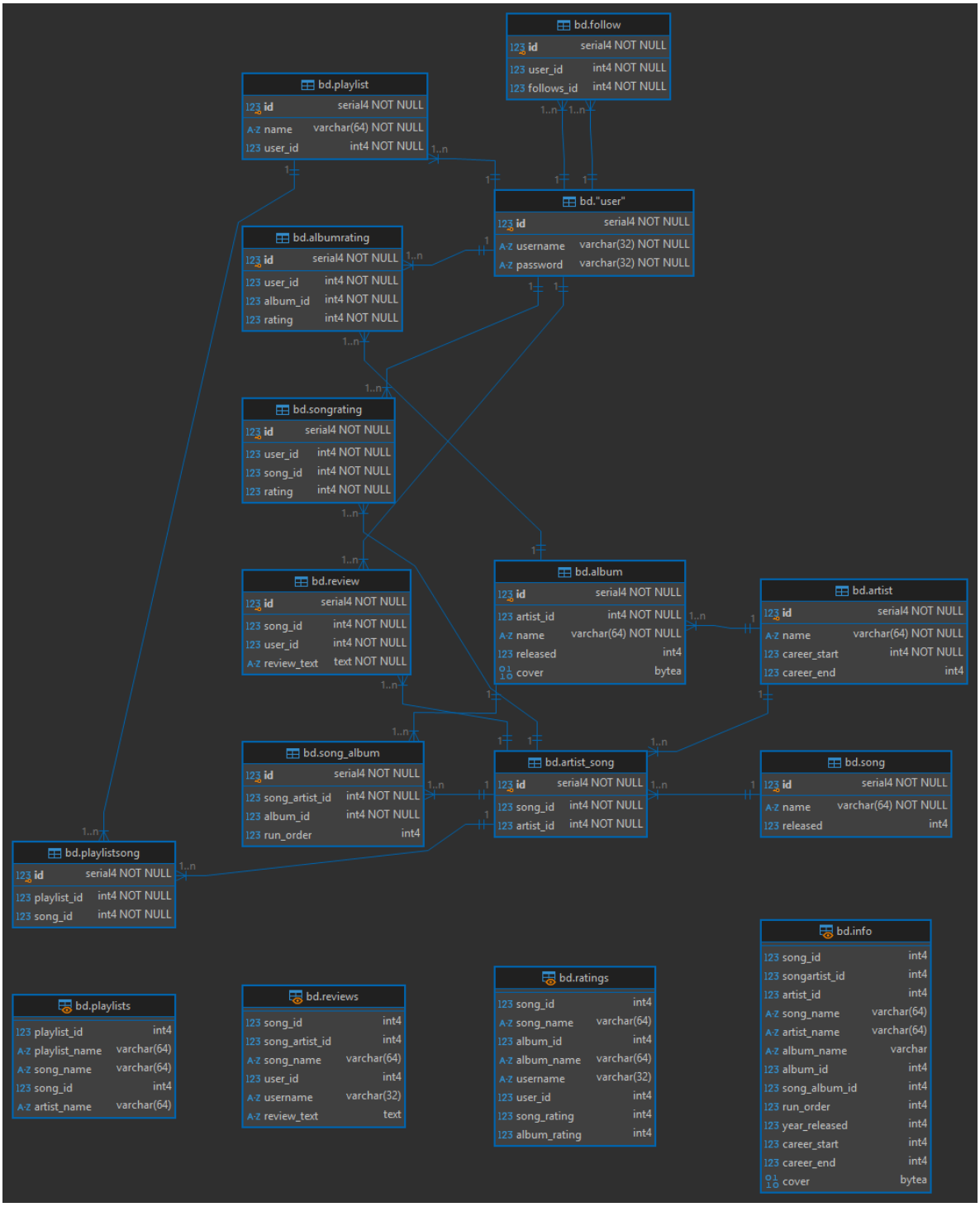
11. Tabela: `bd.review`

- **Opis:** Przechowuje recenzje piosenek.
- **Kolumny:**
  - id serial primary key - Unikalne ID napisanej recenzji
  - song_id int not null - ID piosenki
  - user_id int not null - ID użytkownika,który napisał recenzję
  - review_text text not null - Tekst recenzji
- **Klucze obce:** Klucz user_id nawiązuje do tablicy `bd.user(id)`, natomiast song_id do tablicy `bd.artist_song`

12. Tabela: `bd.follow`

- **Opis:** Przechowuje informacje o tym, kogo obserwuje użytkownik.
- **Kolumny:**
  - id serial primary key - Unikalne ID tabeli pośredniczącej
  - user_id int not null - ID użytkownika, który obserwuje
  - follows_id int not null - ID użytkownika, którego obserwują
- **Constraints:** Użytkownik nie może obserwować samego siebie,
- **Klucze obce:** Klucze user_id oraz follows_id nawiązują do tablicy `bd.user`

2.2 Zaprojektowanie relacji pomiędzy encjami: Diagram ERD.



Rysunek 1: Diagram ERD projektu.

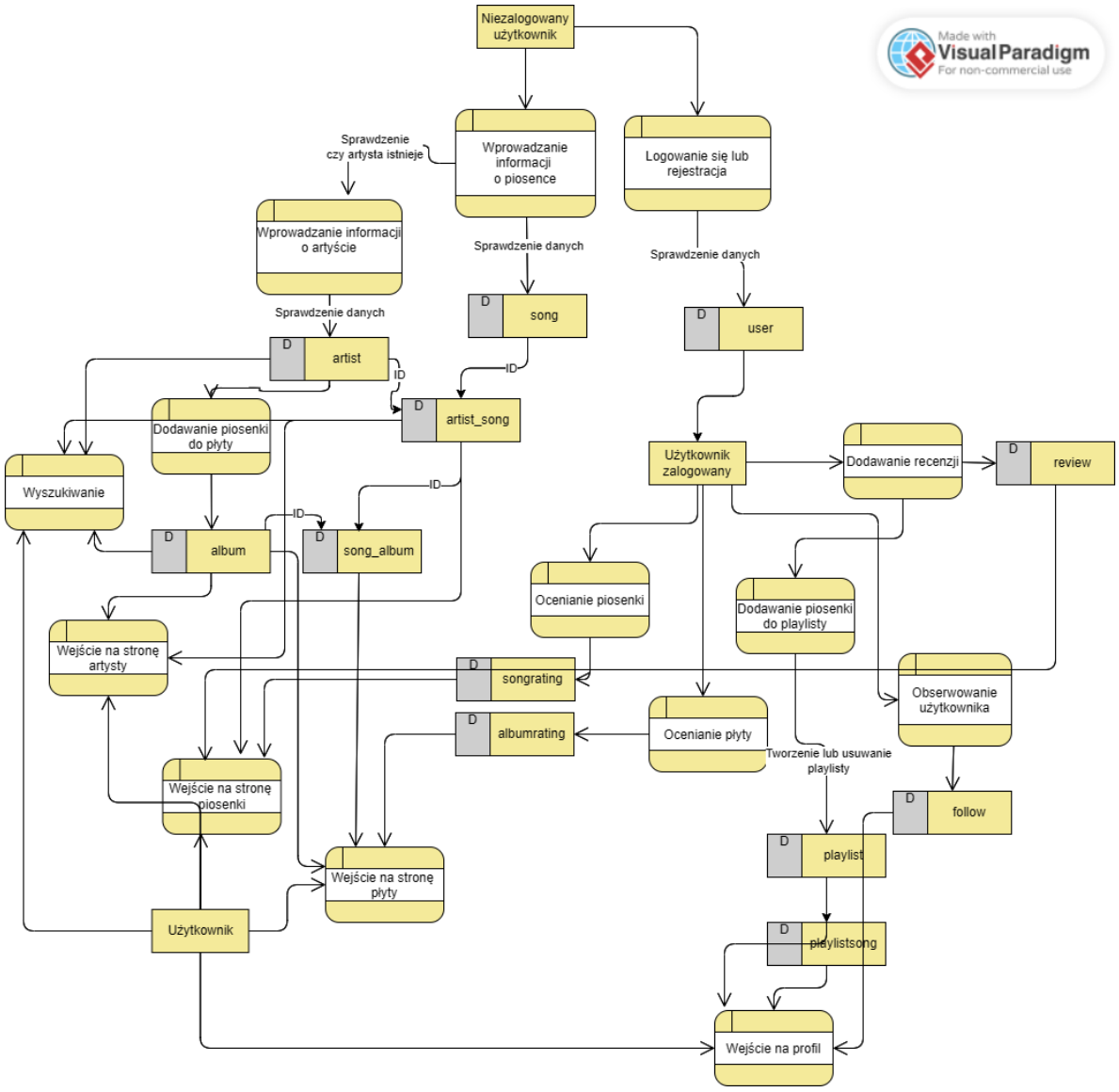
Na diagramie dokładnie widzimy, że są dwie tabelki z największą ilością połączeń:

- Tabela `user`, używana do realizacji funkcjonalności socjalnej. Połączenia do tabel `songrating`,`albumrating`, `follow`, `rewiew` oraz `playlist` są typu *1..N*
- Tabela `artist_song`, używana do powiązania tabel z informacjami o artyście, piosenkach, płytach oraz ocenami i recenzjami, a także playlistami.

Relację *N:M* realizują tabeli:

- bd.playlistsong - jedna piosenka może znajdować się w wielu playlistach, a w jednej playliście - wiele różnych piosenek
- bd.songrating - wielu użytkowników mogą ocenić tą samą piosenkę, lub jeden użytkownik może ocenić wiele rożnych piosenek
- bd.albumrating - wielu użytkowników mogą ocenić tą samą płytę, lub jeden użytkownik może ocenić wiele rożnych płyt
- bd.follow - jeden użytkownik może obserwować wielu innych użytkowników, lub inni użytkownicy mogą obserwować tego samego użytkownika

2.3 Diagram DFD



Rysunek 2: Diagram DFD

2.4 Widoki

Dla uproszczenia wyszukiwania danych były użyte widoki:

- Widok `info` zbiera w sobie wszystkie informacje o piosence, takie jak:
  - song_id
  - songartist_id
  - artist_id
  - song_name
  - artist_name
  - album_name
  - album_id
  - song_album_id
  - run_order
  - year_released
  - career_start
  - career_end
  - cover
- Widok `ratings` zbiera w sobie wszystkie zostawione oceny piosenek i płyt:
  - song_id
  - song_name
  - album_id
  - album_name
  - username
  - user_id
  - song_rating
  - album_rating
- Widok `playlists` zbiera w sobie wszystkie playlisty i piosenki w nich:
  - playlist_id
  - playlist_name
  - song_name
  - song_id
  - artist_name
- Widok `reviews` zbiera w sobie wszystkie napisane recenzje oraz informację o użytkowniku, takie jak:
  - song_id
  - song_artist_id
  - song_name
  - user_id
  - username
  - review_text

3 Projekt logiczny

3.1 Projektowanie tabel, kluczy, indeksów

Plik SQL tworzący tabele oraz zawierający użyte funkcje i tryggery został zamieszczony w katalogu sql. Struktura projektu została opisana w poprzednich punktach.

3.2 Słowniki danych

Dane zostały przedstawione w punkcie 2.2.

3.3 Zaprojektowanie operacji na danych

3.3.1 Wstawienie piosenki

Wstawianie piosenki przez użytkownika wygląda następująco: użytkownik ma podać nazwę piosenki, imię artysty oraz rok jej napisania. Wstawienie piosenki oraz walidacja danych są kontrolowane poprzez dwa triggery. W jednym, który jest wywoływany przed wstawianiem, sprawdzamy, czy wprowadzone pola nie są puste, czy podany rok mieści się w ramach kariery artysty lub czy taka piosenka już istnieje w bazie. Trygger wywoływany po walidacji używa się do powiązywania piosenki i artysty w tabeli `artist_song`.

3.3.2 Wstawianie artysty

Wstawianie piosenki przez użytkownika wygląda następująco: użytkownik ma podać nazwę artysty, rok rozpoczęcia oraz rok zakończenia kariery. Jeżeli artysta wciąż jest aktywny, ostatnie pole pozostawia puste. Artysta jest zawsze tworzony wraz z piosenką, więc odpowiedni trygger najpierw sprawdza, czy rok napisania podanej wcześniej piosenki zgadza się z latami aktywności artysty.

3.3.3 Wstawianie płyty

Przy wstawianiu piosenki użytkownik ma opcję wskazania, czy dana piosenka jest singlem czy pochodzi z albumu. Użytkownik musi podać jedynie nazwę albumu oraz okładkę, która będzie wykorzystywana. Wstawianie odbywa się poprzez funkcję, która wyszukuje odpowiednie id w bazie danych, później jest uruchamiany trygger, który utworzy relację między płytą a piosenką. Możemy zobaczyć, jak odbywa się wstawianie:

```
CREATE OR REPLACE FUNCTION bd.insertAlbum(_artist_name varchar(64),
   _song_name varchar(64), _album_name varchar(64),int)
RETURNS VOID
LANGUAGE plpgsql AS $$
DECLARE
    _song_id int;
    _album_id int;
    _artist_id int;
BEGIN
    select bd.getArtistID(_artist_name) into _artist_id;
    select bd.getAlbumID(_album_name,_artist_id) into _album_id;
    select bd.getSongID(_song_name,_artist_name) into _song_id;
    INSERT INTO bd.song_album (song_artist_id, album_id,run_order)
       VALUES (_song_id, _album_id,$4);
END;
$$;
```

3.3.4 Dodawanie ocen

Dodawanie oceny przez zalogowanego użytkownika realizowane poprzez wybieranie odpowiedniej ilości gwiazdek na stronie. Zostaje wtedy wywołana oddzielna funkcja, która sprawdza, czy ocena już istnieje; w zależności od tego, tworzy nowy wpis lub zmienia istniejący:

```
CREATE OR REPLACE FUNCTION bd.addRating(item_id int ,_user_id int,
   _rating int,mode char)
RETURNS VOID
LANGUAGE plpgsql AS $$
BEGIN
    if mode = 's' then
        if exists (select 1 from bd.songrating where user_id =
           _user_id and song_id = item_id) then
            update bd.songrating set rating = _rating where user_id =
               _user_id and song_id = item_id;
        else
            insert into bd.songrating(user_id, song_id, rating) values
                (_user_id, item_id, _rating);
        end if;
    elsif mode = 'a' then
        if exists (select 1 from bd.albumrating where user_id =
           _user_id and album_id = item_id) then
            update bd.albumrating set rating = _rating where user_id =
                _user_id and album_id = item_id;
        else
            insert into bd.albumrating(user_id, album_id, rating)
               values (_user_id, item_id, _rating);
        end if;
    end if;
END;
$$;
```

3.3.5 Dodawanie playlisty lub piosenki do playlisty

Po wejściu na stronę piosenki zalogowany użytkownik ma opcję dodania tej piosenki do playlisty. Wtedy otwiera się małe okienko dialogowe, z opcją wybrania istniejącej playlisty lub utworzenia nowej. Po potwierdzeniu wyboru przyciskiem 'OK' piosenka będzie dodana do odpowiedniej playlisty lub zostanie utworzona nowa o podanej nazwie. Nowa playlista będzie widoczna w zakładce 'Profile'.

3.3.6 Pisanie recenzji

Na tej samej stronie użytkownik może dodać recenzję. Okienko z recenzjami pełni rolę 'Shoutboxa', co oznacza, że pisać możemy kilkakrotnie, na przykład prowadząc dyskusję.

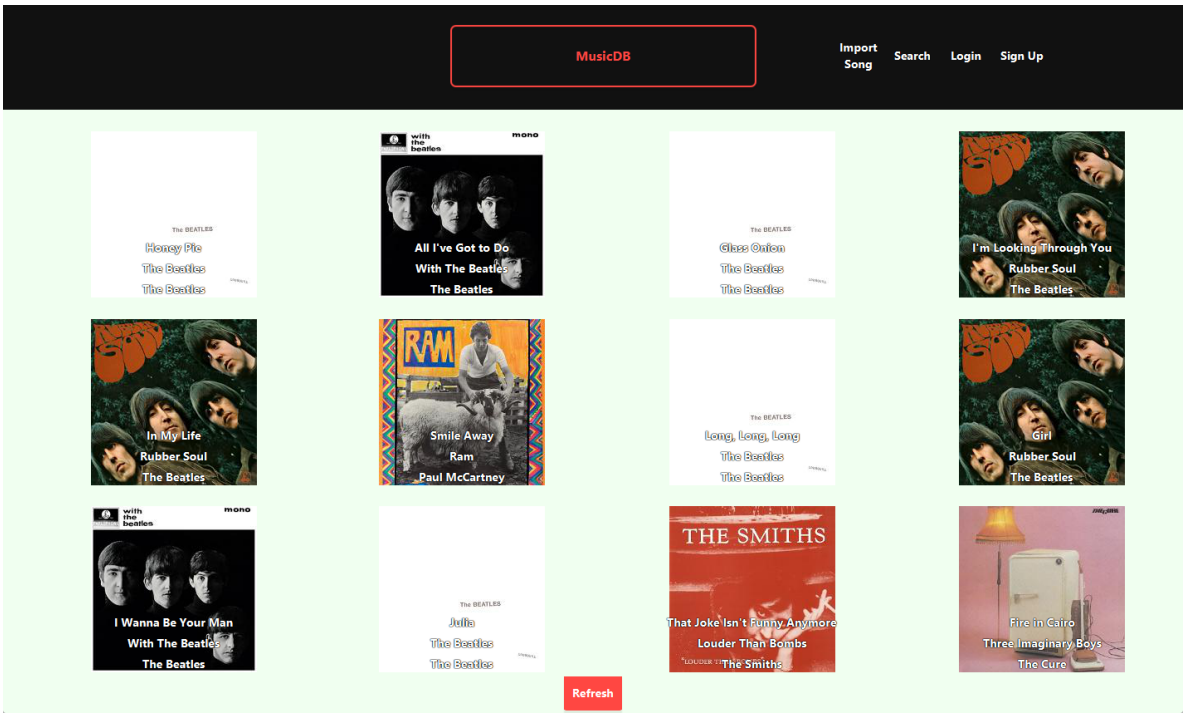
3.3.7 Dodawanie do listy obserwujących

Po wejściu na profil innego użytkownika pojawia się przycisk 'Follow', naciskając który użytkownik jest dodawany do tablicy `bd.follow`.

3.4 Projekt funkcjonalny

3.4.1 Interfejsy do prezentacji, edycji i obsługi danych

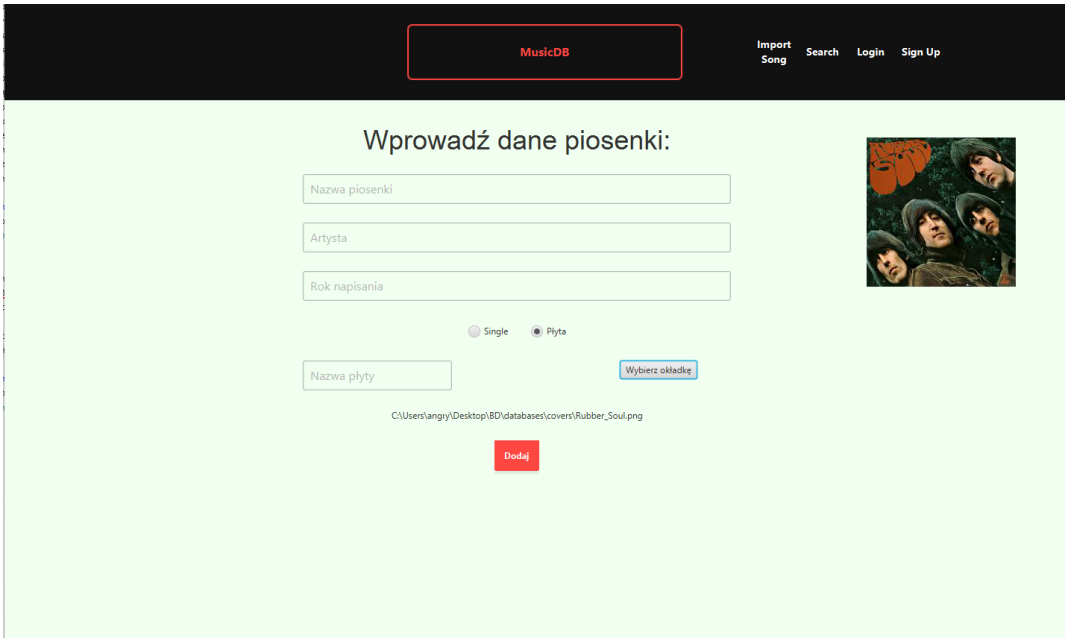
Strona główna prezentuje się gridem, który zachęca użytkownika do eksploracji bazy danych. Wygląda to następująco:



Rysunek 3: Strona główna aplikacji.

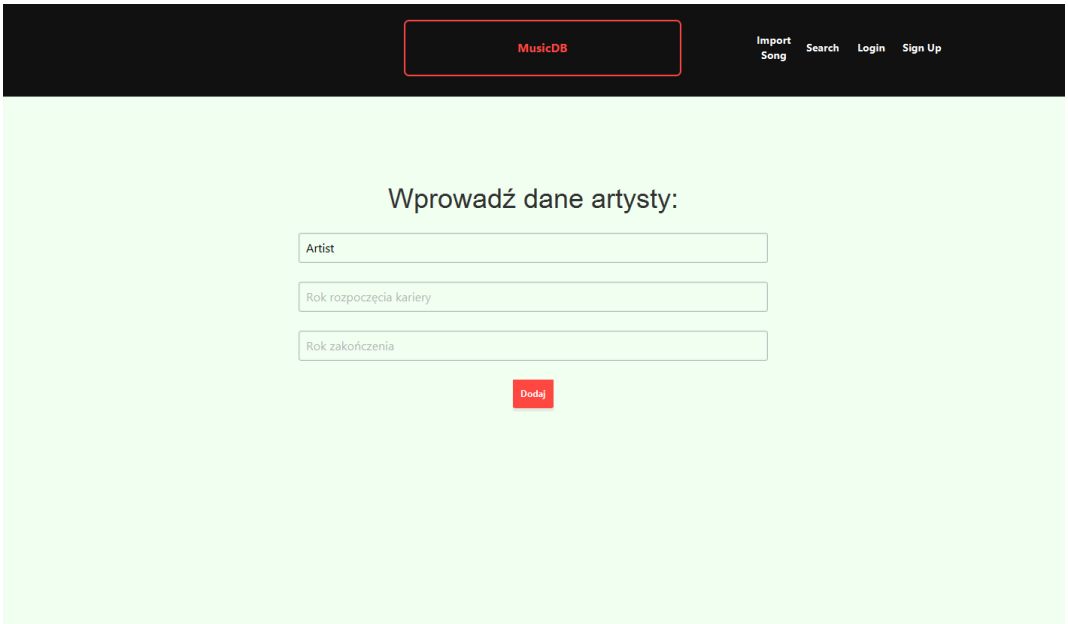
Możemy zauważyć na górze cztery przyciski do sterowania aplikacją, oraz jeden przycisk na dole. Przycisk 'Refresh' odnawia grid. Przycisk 'MusicDB' na górze, który jednocześnie jest logo, przenosi użytkownika z powrotem na stronę główną. Do pierwszego interfejsu prowadzi przycisk 'Import Song'.

Po wybraniu typu 'Płyta' pojawiają się dodatkowe pola do wprowadzenia danych o płycie. A po dodaniu okładki widzimy ścieżkę do pliku i wybrany obrazek. Strona prezentuje się następująco:



Rysunek 4: Strona wprowadzania danych piosenki.

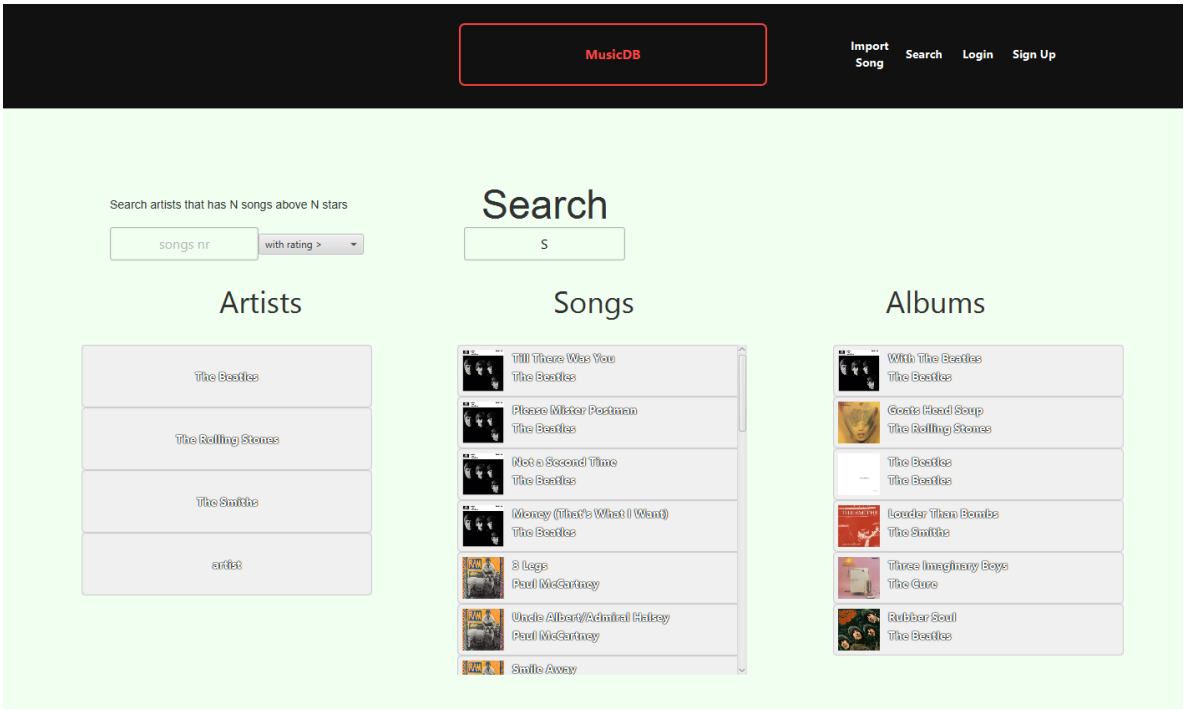
Po wciśnięciu przycisku 'Dodaj' program sprawdzi, czy podany artysta już istnieje w bazie danych. Jeżeli jednak nie, poprosi o uzupełnienie brakujących informacji. Otworzy się nowa strona, do której wpisujemy dane artysty:



Rysunek 5: Strona wprowadzania danych artysty.

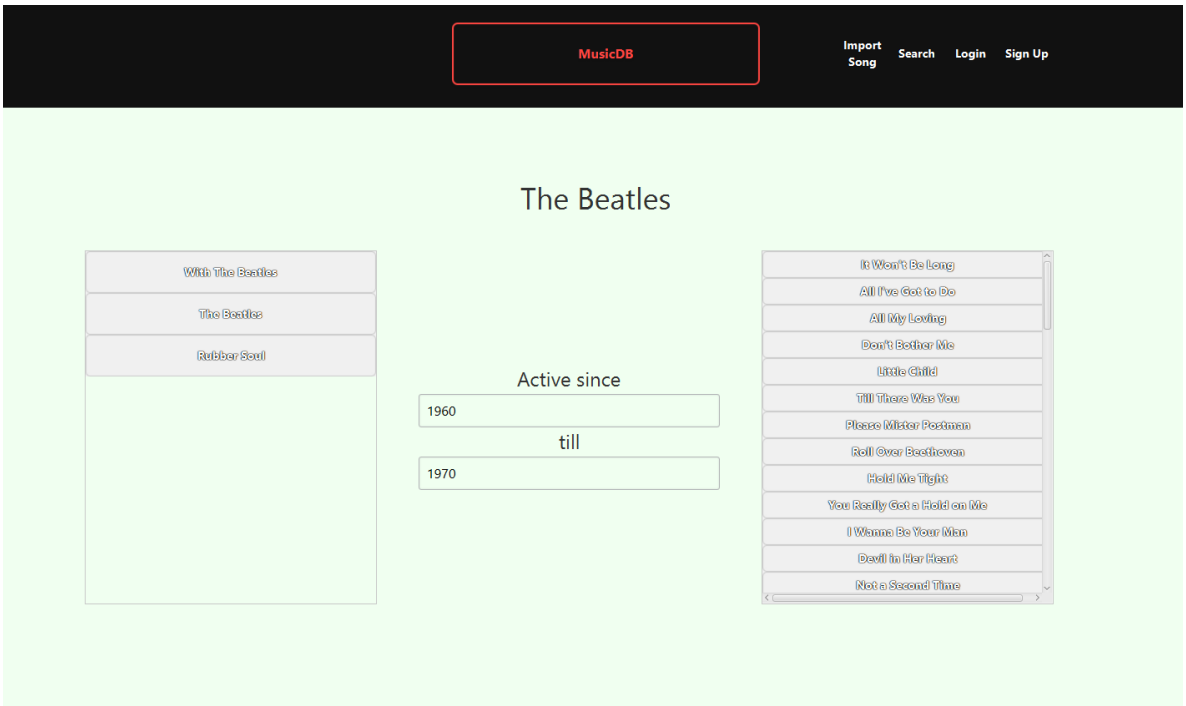
Pole nazwy artysty będzie już uzupełnione, więc pozostaje nam tylko wprowadzenie danych o latach jego kariery. Po dodaniu artysty jesteśmy z powrotem przekierowani na stronę główną.

Następnie możemy spróbować wyszukać dodaną piosenkę. Wprowadzamy nazwę do odpowiedniego pola i wybieramy z otrzymanych artystów, piosenek oraz albumów:



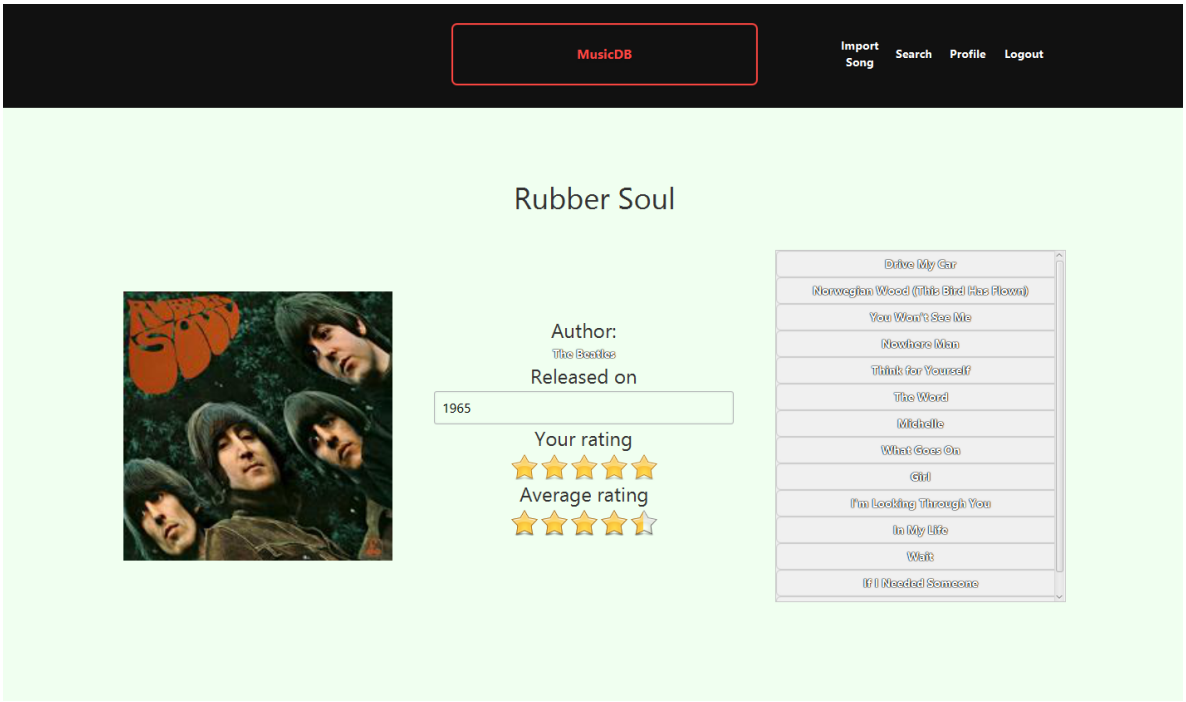
Rysunek 6: Strona wyszukiwania.

Użytkownik może nacisnąć na każdy tekst, po czym zostanie przekierowany na odpowiednią stronę. Również została dodana funkcja wyszukiwania artystów, którzy mają N piosenek powyżej określonej oceny. Na przykład, tak prezentuje się strona artysty:



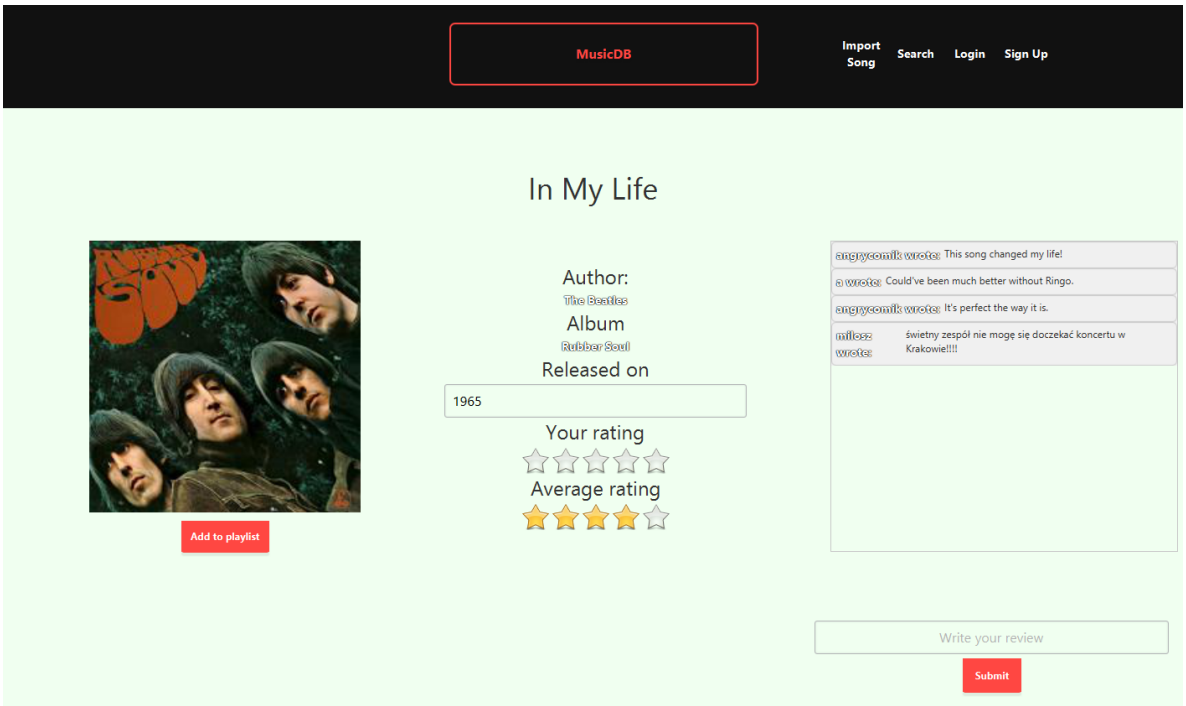
Rysunek 7: Strona artysty.

Możemy zobaczyć wcześniej wprowadzone dane oraz listę wszystkich piosenek i płyt. Następnie przejdziemy na stronę płyty:



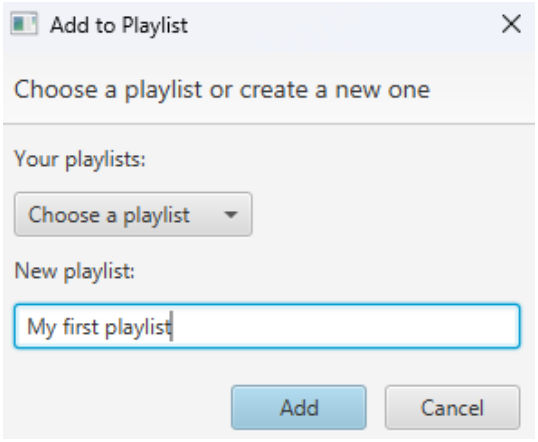
Rysunek 8: Strona płyty.

Na tej stronie już pojawił się rating. Każdy użytkownik może dodać ocenę klikając na odpowiednią ilość gwiazdek. Wybieramy piosenkę z panelu po prawej stronie:

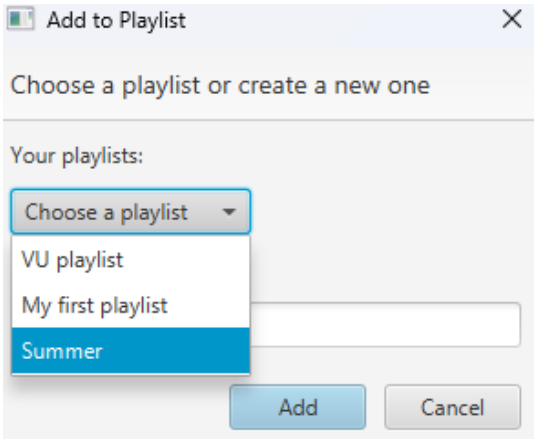


Rysunek 9: Strona piosenki.

Po lewej stronie widzimy przycisk 'Add to Playlist'. Otwiera on okienko dodawania piosenek do playlisty:

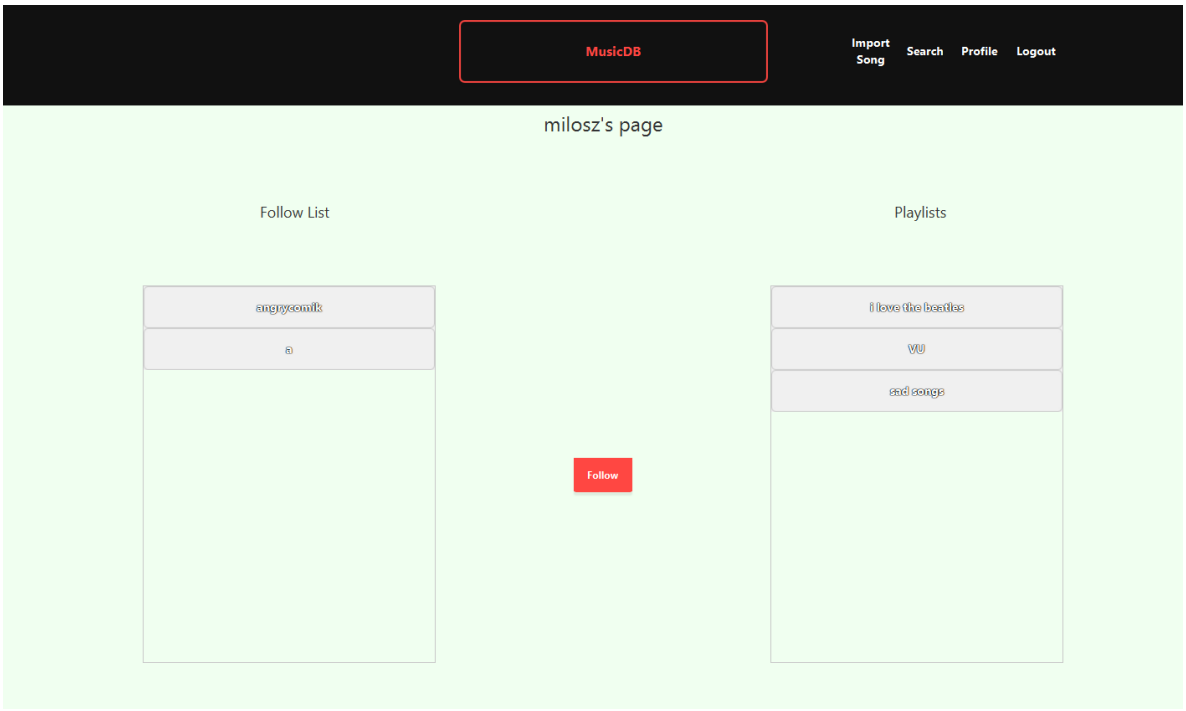


(a) Dodawanie playlisty.



(b) Wybieranie playlisty.

Po prawej natomiast znajduje się pole z recenzjami. Zalogowany użytkownik może napisać recenzję i wysłać ją przyciskiem 'Submit'. Wyżej znajduje się panel z recenzjami. Po naciśnięciu na imię użytkownika jesteśmy przekierowani na jego stronę osobistą:



Rysunek 11: Strona osobista.

Przycisk 'Follow' umożliwia obserwowanie użytkownika, po lewej stronie widzimy kogo obserwuje użytkownik, po prawej - jego playlisty.

4 Dokumentacja

4.1 Wprowadzanie danych

W momencie utworzenia bazy danych dane są wprowadzane poprzez odpowiednią klasę ReadJsonData, używając pliku JSON o odpowiedniej strukturze, wraz z plikami okładek.

4.2 Uruchomienie aplikacji

Aplikacja została wykonana przy użyciu JavaFX oraz programu Scene Builder. Ponieważ JavaFX, zaczynając od wersji Java 8, nie jest w składzie standardowego pakietu Java, uruchomić program będzie można przy pomocy polecenia `java -jar Projekt.java` (wtedy zostanie wypisany warning, który nie ma wpływu na działanie programu) lub przy użyciu specjalnej wersji Javy, która zawiera w sobie JavaFX, na przykład Zulu:
https://www.azul.com/downloads/?version=java-23os=windowspackage=jre-fxzulu

4.3 Opracowanie dokumentacji technicznej

Dokumentacja wygenerowana w postaci HTML znajduje się w katalogu doc.

4.4 Wykaz literatury

www.vojtechruzicka.com/javafx-getting-started/
materiały z wykładu