Projekt z PROI

Playlists

Mateusz Czarniewicz-Cyfra

Andrii-Stepan Pryimak

Hanna Zarzycka

# Opis projektu

Funkcjonalności:

* Baza danych z piosenkami
* Możliwość tworzenia playlist przez użytkowników
* Możliwość losowania playlist
* Możliwość dodawania piosenek
* Administrator może zarządzać piosenkami
* Baza danych użytkowników

Nasz projekt to program dziający w trybie klient-serwer umożliwiający zbieranie muzyki w playlisty. Użytkownik po zalogowaniu się na swoje konto ma możliwość wybrania dostępnych piosenek z bazy danych i zapisania ich do playlisty lub wygenerowania jej losowo. Ma też specjalną playlistę na ulubione piosenki. Następnie może wyświetlić swoje playlisty i ich zawartość.

Użytkownik ma możliwość w każdym momencie zostać artystą. Po takiej decyzji dostaje też uprawnienia do dodawania własnych piosenek do bazy danych.

Administrator może wyświetlać użytkowników i nadawać im uprawnienia administratora.

Piosenka ma swój identyfikator, tytuł, artystę, gatunek, rok wydania, czas trwania. Można ją odtworzyć, dodać do ulubionych albo do playlisty.

Playlistę można odtworzyć - po kolei włączać piosenki. Można dodać nową piosenkę albo usunąć.

Dane są ładowane z bazy po otwarciu sesji i zapisują się na bieżąco przez użytkownika.

# Klasy podstawowe

Zawarte w bibliotekach SongLib i UserLib

Song

- zawiera pola: id, nazwa, artysta, gatunek, czas trwania i rok

- pozwala wyświetlać i formatować przechowywane dane

Playlist

- zawiera id, nazwę i vector piosenek, udostępnia iterator do jego obsługi

- przechowuje iterator do aktualnej piosenki

- pozwala wyświetlać i formatować przechowywanych danych

- pozwala „odtwarzać” piosenki

VirtualUser <- User

- zawiera imię, login, hasło, vector playlist, playlistę ulubionych

- zawiera polę output, które zawiera wiadomość do wyświetlenia po stronie klienta

- inicjalizuje mapy instrukcji

- zawiera metody do wykonywania operacji na playlistach, wylogowania się, zostania artystą, wyświetlania piosenek

VirtualUser <- User <- Artist

- pozwala dodawać własne piosenki i ewentualnie je usuwać

VirtualUser <- User <- Admin

- może wyświetlać wszystkich użytkowników i nadawać prawa administratora

VitualUser <- User <- Artist, Admin <- AdminArtist

# Struktura i obsługa bazy danych

Klasy zawarte w bibliotece DatabaseLib:

- SongDataInterface

- SongDataInterface <- PlaylistDataInterface

- UserDataInterface

Pozwalają na czytanie i zapisywanie danych z wyżej wymienionych klas do bazy danych. Po odczytaniu danych tworzą odpowiednie obiekty.

Struktura:

Directories

root

|

|-|users

| |

| |-|{user.login}

|   |

|   |-|userPlaylists

|   | |

|   | |{playlistId1}.txt

|   | |{playlistId2}.txt

|   |

|   |credentials.txt

|

|-|public

| |

| |-|songs

| | |

| | |-|{songId.txt}

|

|id.txt

Budowa plików

Playlist

{playlistId1}.txt - space separated songIds

User

login must be uniqe

credentials.txt

  {password} {name} {accessLevel}

Song

info.txt

  {name} {artist} {genre} {duration} {year}

# Realizacja Klient-Serwer

Połączenie klient-serwer jest realizowane przez protokół TCP z pomocą biblioteki Boost. Wszystkie akcje takie jak obsługa bazy danych, tworzenie obiektów i wykonywanie na nich metod są wykonywane po stronie serwera i jedynie ich efekty w formie tekstu są zwracane do klienta. Klient przekazuje do serwera numery instrukcji z argumentami.

# Jak uruchomić projekt

Przed zbudowaniem projektu należy pobrać bibliotekę Boost ze strony <https://www.boost.org/> (dokładna instrukcja znajduje się w reedme projektu).

Projekt należy zbudować przy użyciu cmake. Po zbudowaniu należy najpierw otworzyć Server.exe (./build/Server/Server.exe), potem w oddzielnym terminalu Client.exe (./build/Client/Client.exe).

Po stronie klienta można zalogować się na konto użytkownika znajdującego się w bazie danych lub utworzyć nowe. Następnie można wybierać opcje z listy instrukcji poprzez podanie numeru instrukcji oraz wymaganych argumentów.

Mogą wystąpić problemy z działaniem programu przy zbyt dużym obciążeniu pamięci operacyjnej komputera.