Politechnika Śląska

Wydział Matematyki Stosowanej

# Audio player/Odtwarzacz audio

Dokumentacja projektu

Dawid Bitner
Marcin Krupa
II/IA 2017/2018 sem. letni

**Część 1** 3
1. Założenia 3
2. Instrukcja obsługi 4

**Część 2** 5
1. Specyfikacja techniczna 5
Kompilacja projektu 5
Podział na pliki 5
Użyte biblioteki 5
Metody użyte w programie 5
2. Szczegóły techniczne 7
a) Działanie programu **Błąd! Nie zdefiniowano zakładki.**
b) Algorytmy/fragmenty kodu 7

**Część 3** 8
Podsumowanie **Błąd! Nie zdefiniowano zakładki.**

# Część 1

---

## 1. Założenia

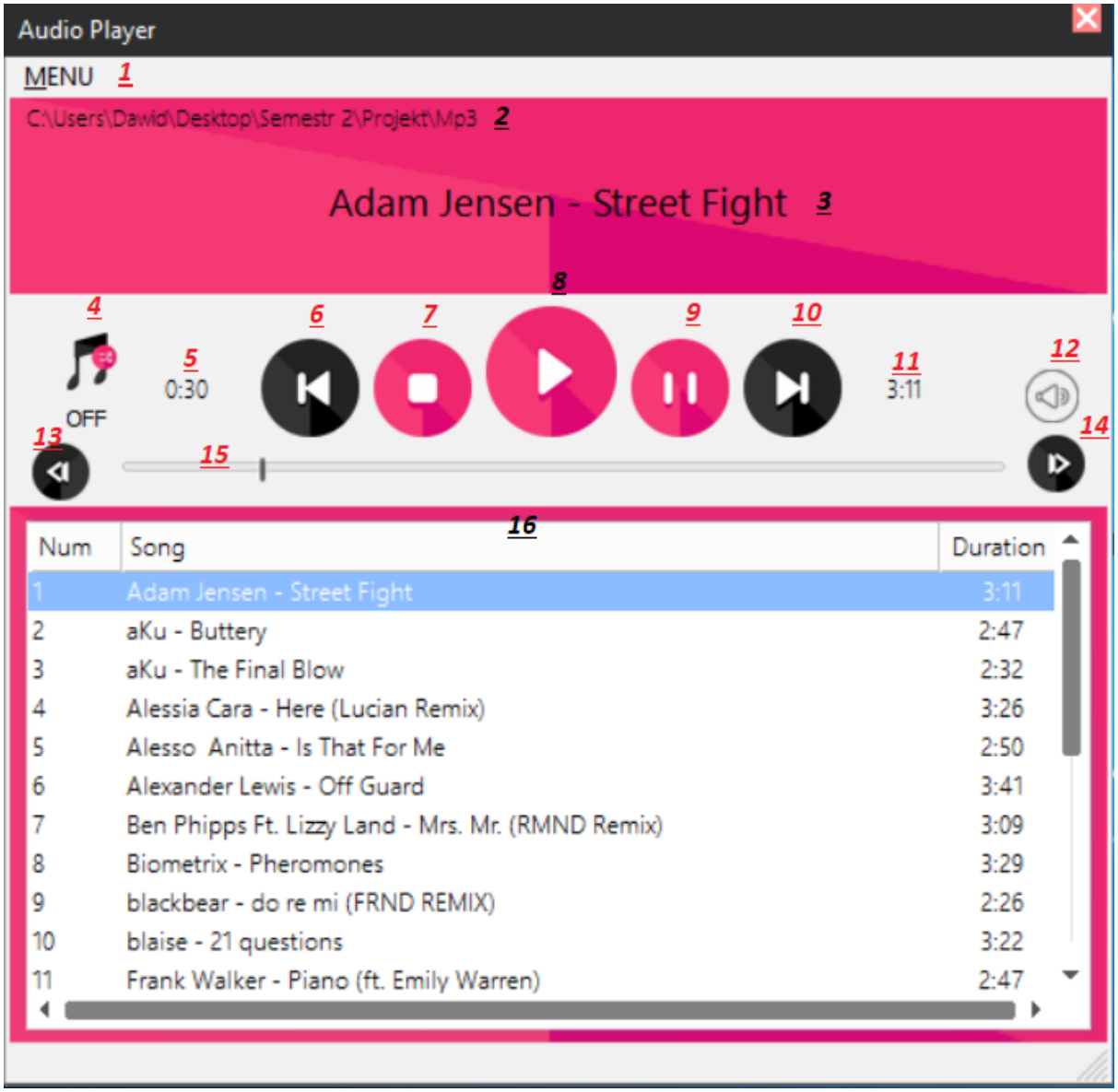
Celem projektu było stworzenie kompaktowego, w pełni działającego odtwarzacza który będzie posiadał najważniejsze funkcje, a także możliwość rozbudowy projektu w przyszłości.

Wstępne założenia do projektu :
- odtwarzacz będzie posiadał podstawowe funkcje takie jak:
**1. Pauzowanie, pomijanie, przewijanie utworu do przodu/tyłu.** Zrealizowano.
**2. Wybór utworów z wybranego folderu.** Zrealizowano.
**3. Losowe odtwarzanie utworów.** Zrealizowano.

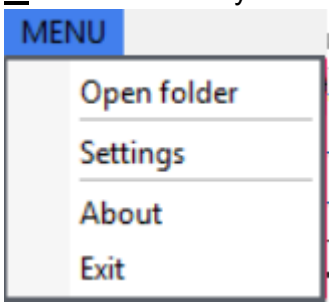
- dodatkowe funkcje:
1. Dodawanie i edycję znaczników (IDv3). Nie zostało wdrożone.
2. Możliwość edycji podstawowego bufora. Nie zostało wdrożone.
3. Audio converter. Nie zostało wdrożone.
4. Dodanie modułu pobierania okładek. Nie zostało wdrożone.

Dodatkowe funkcje nie zostały zrealizowane ponieważ ich wdrożenie jest dużo trudniejsze, oraz zastosowania bardziej rozbudowanych bibliotek co przekłada się na czas potrzebny do ich przestudiowania. Mimo tego odtwarzacz jest w pełni funkcjonalny i można być używany bez żadnych problemów

# 2. Instrukcja obsługi




***1*** Menu otwiera wysuwaną listę




Open folder - wybieramy folder z którego chcemy odtwarzać muzykę
Settings - w obecnej wersji programu przycisk nie działa, w zamyśle miały znajdować się tutaj ustawienia dotyczące m.in.: wyglądu, czy korektor.
About - informacje o autorach
Exit - kończy działanie programu
***2*** Ścieżka do odtwarzanego folderu, z którego odtwarzana jest muzyka
***3*** Nazwa aktualnie odtwarzanego utworu

***4*** Shuffle play - włączanie/wyłączanie losowego odtwarzania
***5*** Aktualny czas odtwarzanego utworu
***6*** Skok do poprzedniego utworu
***7*** Zatrzymaj odtwarzanie
***8*** Odtwarzaj (podczas gdy utwór jest odtwarzany działa jak przycisk pauzy)
***9*** Pauzuj utwór (podczas gdy utwór jest spauzowany działa jak przycisk odtwarzaj)
***10*** Skok do następnego utworu
***11*** Długość odtwarzanego utworu
***12*** Poziom głośności - pasek regulacji zostaje rozwinięty po naciśnięciu przycisku
***13*** Cofa utwór o 10 sekund
***14*** Przewija do przodu utwór o 10 sekund
***15*** Pasek postępu utworu
***16*** Playlista - załadowany folder

# Część 2

---

## 1. Specyfikacja techniczna

### Kompilacja projektu

Projekt stworzony w programie Embarcadero RAD Studio XE4 (C++ Builder XE4) w wersji trial korzystający z kompilatora GNU GCC.

### Podział na pliki

`Main_U.cpp` – plik źródłowy
`Main_U.h` – plik nagłówkowy

### Użyte biblioteki

Wykorzystane biblioteki przy pisaniu programu to:

- `<string>` - obsługuje łańcuchy znaków
- `libzplay.h` - łącznie z własnymi plikami nagłówkowymi funkcji biblioteki
- `<vcl.h>` - biblioteka własna C++ Builder XE4
- `<stdlib.h>` - zawiera deklaracje funkcji służących do przekształcania liczb, przydzielania pamięci i innych zadań
- `<ctime>` - konwersja daty i czasu

### Metody użyte w programie

`void __fastcall TForm1::E1Click(TObject *Sender)` - odpowiada za zamknięcie aplikacji

`void __fastcall TForm1::O2Click(TObject *Sender)` - odpowiada za otwarcie przeglądania folderów
`void __fastcall TForm1::FolderChange(TObject *Sender)` - odpowiada za załadowanie listy utworów do programu
`void __fastcall TForm1::List_SongsDblClick(TObject *Sender)` - odpowiada za odtworzenie utworu po podwójnym kliknięciu go na załadowanej liście
`void TForm1::generateMetadata(String fileName)` - odpowiada za wygenerowanie metadanych
`void __fastcall TForm1::A1Click(TObject *Sender)` - odpowiada za wyświetlenie informacji o autorach
`void TForm1::playThis(int index)` - odpowiada za odtworzenie utworu i załadowaniu informacji o nim
`void __fastcall TForm1::Next_ButtonClick(TObject *Sender)` - odpowiada za przycisk odtworzenia kolejnego utworu
`void __fastcall TForm1::Previous_ButtonClick(TObject *Sender)` - odpowiada za przycisk odtworzenia poprzedniego utworu
`void __fastcall TForm1::Play_ButtonClick(TObject *Sender)` - odpowiada za przycisk Play
`void __fastcall TForm1::Stop_ButtonClick(TObject *Sender)` - odpowiada za przycisk Stop
`void __fastcall TForm1::Pause_ButtonClick(TObject *Sender)` - odpowiada za przycisk Pause
`void __fastcall TForm1::tbVolumeChange(TObject *Sender)` - odpowiada za zmianę poziomu głośności
`void __fastcall TForm1::btnVolumeClick(TObject *Sender)` - odpowiada za przycisk obsługujący zmianę głośności
`void __fastcall TForm1::Timer_tbTimer(TObject *Sender)` - odpowiada za odliczanie czasu
`void __fastcall TForm1::btn_FastForwardClick(TObject *Sender)` - odpowiada za przycisk cofnięcia utworu o 10 sekund
`void __fastcall TForm1::btn_FastBackwardClick(TObject *Sender)` - odpowiada za przycisk przesunięcia utworu o 10 sekund do przodu
`void __fastcall TForm1::tbSongUserChange(TObject *Sender)` - odpowiada za przesuwanie suwaka postępu
`void TForm1::playMode(int mode)` - odpowiada za odtwarzanie utworu
`void TForm1::songTime()` - odpowiada za informacje o czasie
`void TForm1::currentLabel()` - odpowiada za usunięcie rozszerzenia z nazwy aktualnie granego utworu
`void __fastcall TForm1::Shuffle_ButtonClick(TObject *Sender)` - odpowiada za przycisk losowego odtwarzania

# 2. Szczegóły techniczne

## b) Ważniejsze fragmenty kodu

Program w dużej mierze opiera się na prostych algorytmach oraz metodach klas biblioteki libzplay. Przykładem takiego algorytmu jest algorytm który załadowuje utwory do obiektu ListView.

```
void __fastcall TForm1::FolderChange(TObject *Sender)
{
      Folder_Label->Caption=Folder->Text;
      TStringDynArray arr=TDirectory::GetFiles(Folder->Text);

      for (int i=0; i<arr.Length; i++)
      {
         if(ExtractFileExt(arr[i]).LowerCase()==".mp3" || ExtractFileExt(arr[i]).LowerCase()==".way" || ExtractFileExt(a
         {
             list_Files->Add(arr[i]);
             generateMetadata(arr[i]);
             itm=List_Songs->Items->Add();
             a++;
             itm->Caption=a;
             itm->SubItems->Add(ExtractFileName(arr[i]).Delete(ExtractFileExt(arr[i])));
             player->GetStreamInfo(&songInfo);

               if (sec<10) {
                    itm->SubItems->Add(IntToStr(songInfo.Length.hms.minute)+":"+0+IntToStr(songInfo.Length.hms.second));
               } else itm->SubItems->Add(IntToStr(songInfo.Length.hms.minute)+":"+IntToStr(songInfo.Length.hms.second));
         }
      }

      List_Songs->Enabled=true;
      Previous_Button->Enabled=true;
      Next_Button->Enabled=true;
      Play_Button->Enabled=true;
      Pause_Button->Enabled=true;
      Stop_Button->Enabled=true;

}
```

- `Folder_Label->Caption=Folder->Text` - pozwala na wyświetlanie aktualnego folderu w obiekcie typu Label.
- `TStringDynArray arr=TDirectory::GetFiles(Folder->Text);` - deklaracja dynamicznej tablicy, która pobiera pliki ze wskazanego folderu.
- Następnie mamy pętle która będzie wykonywała się do momentu, w którym “przebada” wszystkie pliki ze wskazanego folderu.
- Wewnątrz pętli znajduje się instrukcja warunkowa `if`, która pozwala na wczytanie wyłącznie plików o rozszerzeniu audio obsługiwanym przez bibliotekę libzplay.
- `list_Files->Add(arr[i]);` - `list_Files` jest obiektem klasy `TStringList` służącej do uporządkowanego umieszczenia obiektów typu String na liście. Dodajemy te elementy tablicy, które mają interesujące nas rozszerzenie.
- generateMetadata(arr[i]); - funkcja generująca metadane o pliku.
- `itm=List_Songs->Items->Add();` - `itm` jest obiektem klasy `TListItem`, pozwalającym na dodawanie i edytowanie “przedmiotów” w obiekcie `ListView` - w tym przypadku nazwanym `List_Songs`
- `itm->Caption=a;` oraz `itm->SubItems->Add(ExtractFileName(arr[i]).Delete(ExtractFileExt(arr[i])));` pozwalają na dodanie właśnie takich przedmiotów do obiektu `ListView`, wcześniej już podzielonego na kolumny. W pierwszej kolumnie wyświetla on index pliku audio, w drugiej nazwę pliku bez rozszerzenia.
- Kolejne linie kodu mają za zadanie dodanie w ostatniej kolumnie czas trwania utworu. W tym celu wykorzystaliśmy funkcję biblioteki libzplay `GetStreamInfo();`

która pozwala na wygenerowanie danych o strumieniu, takich jak minuty, sekundy czy godziny trwania pliku audio.

- ➢ Po zakończeniu pętli, funkcja umożliwia klikanie w przyciski odtwarzacza oraz wybór z listy `ListView`.

# Część 3

---

## Kolejność prac nad projektem

1. Dodawanie obiektów TMainMenu, TsDirectoryEdit, TsListView.
2. Zaprogramowanie działania przycisku Exit, About oraz Open folder połączonego z dodawaniem plików do obiektu TsListView (funkcja generateMetadata oraz metoda FolderChange).
3. Metoda List_SongsDblClick pozwalająca na odtwarzanie utworu przez podwójne kliknięcie użytkownika aplikacji.
4. Zaprogramowanie funkcji playMode() która ma pozwalać na zmianę trybu odtwarzania.
5. Dodanie i zaprogramowanie przycisków Play, Pause, Stop, Previous oraz Next.
6. Dodanie i zaprogramowanie przycisku Volume.
7. Dodanie i zaprogramowanie TrackBar’a do wyświetlania pozycji strumienia audio, oraz suwaka do jej zmiany przez użytkownika.
8. Dodanie i zaprogramowanie przycisków FastForward oraz FastBackward do przewijania i cofania pliku audio o 10 sekund.
9. Dodanie funkcji currentLabel() oraz songTime() wyświetlających nazwę pliku bez rozszerzenia oraz czas pliku audio w obiektach typu Label.
10. Dodanie i zaprogramowanie przycisku Shuffle.
11. Zmiany kosmetyczne w aplikacji.

## Podsumowanie

Udało nam się zrealizować wszystkie wstępne założenia. Niestety czas nie pozwolił na dodanie kilku innych funkcji które mieliśmy w głowach, a które rozbudowały by funkcjonalność aplikacji. Niemniej jednak jesteśmy zadowoleni z rezultatu, z postępu prac nad projektem które przebiegały bezproblemowo, możliwości głębszego poznania środowiska RAD Studio XE4.