

**politechnika śląska**

**wydział automatyki, elektroniki i informatyki**

**kierunek teleinformatyka**

##### Projekt inżynierski

# Odtwarzacz muzyczny wykorzystujący API SoundCloud

Autor: Krzysztof Błażełek

Kierujący pracą: dr inż. Adam Gudyś

Gliwice, grudzień 2016

Spis treści

[1. Wprowadzenie do dziedziny 5](#_Toc262680014)

[2. Podstawy teoretyczne 6](#_Toc262680015)

[2.1 Formatowanie tekstu za pomocą stylów 6](#_Toc262680016)

[2.1.1 Styl podstawowy (normalny) 7](#_Toc262680017)

[2.1.2 Tytuły rozdziałów i podrozdziałów projektu inżynierskiego 7](#_Toc262680018)

[2.1.3 Numeracja stron 7](#_Toc262680019)

[2.1.4 Przypisy 7](#_Toc262680020)

[3. Specyfikacja zewnętrzna 8](#_Toc262680021)

-wymagania sprzętowe i programowe

-interfejs użytkownika (opis każdego widoku)

[4. Specyfikacja wewnętrzna 10](#_Toc262680022)

[5. Eksperymenty 11](#_Toc262680023)

[6. Wnioski 12](#_Toc262680024)

1. Wprowadzenie do dziedziny

Wraz z gwałtownym rozwojem Internetu coraz częściej wykorzystywane są media strumieniowe. Dużą popularnością w sieci cieszą się obecnie takie serwisy jak YouTube oraz SoundCloud, które umożliwiają twórcom dostarczenie swoich dzieł do widzów/słuchaczy. SoundCloud jest platformą umożliwiającą artystom dystrybucję oraz promocję własnej muzyki. Serwis ten oprócz udostępniania muzyki pozwala też zarejestrowanym użytkownikom na komentowanie utworów, dodawanie ich do ulubionych oraz na tworzenie list odtwarzania. Oprócz tego serwis udostępnia interfejs programistyczny, który zostaw wykorzystany do napisania odtwarzacza muzycznego będącego tematem tej pracy.

* 1. Porównanie dostępnych technologii

1.1.1 Windows Forms

Jest to biblioteka przeznaczona do tworzenia desktopowych aplikacji z bogatym interfejsem graficznym użytkownika, która jest częścią platformy .NET 3.0. WinForms jest już dopracowaną technologią, która jest utrzymywana, ale nie będą dodawane do niej nowe funkcjonalności. Zastosowany w niej silnik graficzny GDI (Graphics Device Interface) charakteryzuje się prostotą, ale nie oferuje wielu funkcjonalności wspieranych przez takie silniki graficzne jak DirectX oraz OpenGL. Brak wsparcia akceleracji sprzętowej dla grafiki 3D oraz ograniczona optymalizacja sprzętowa wynikająca z zastosowanej architektury zawężają obszar zastosowania GDI. Istnieje możliwość uruchomienia aplikacji napisanej w WinForms na systemach Linux oraz Mac OS za pośrednictwem Mono – alternatywnej implementacji biblioteki .NET opartej na licencji Open Source. Przeportowanie aplikacji wiąże się z licznymi ograniczeniami wynikającymi z ścisłym powiązaniem WinForms z Windows API (WinAPI). Z tego powodu utrudnione jest projektowanie zaawansowanych interfejsów graficznych. Jeżeli istniejące kontrolki (elementy graficzne takie jak przyciski i pola tekstowe) nie spełniają wymagań projektowych, to programista jest zmuszony napisać ich własną implementację lub skorzystać z bibliotek dostarczanych przez innych wytwórców. Wspierane są wszystkie języki programowania kompatybilne z architekturą CLI (Common Language Infrastructure). Najpopularniejsze z nich to C#, VB.NET oraz C++/CLI.

1.1.2 Windows Presentation Foundation

WPF jest również częścią platformy .NET oraz następcą Windows Forms. Technologia ta utworzyła spójne środowisko dla projektowania bogatych, desktopowych aplikacji graficznych wykorzystujących grafikę 2D, 3D, dokumenty, animacje oraz media. Do definiowania interfejsu graficznego wprowadzono język XAML (Extensible Application Markup Language). Jest to deklaratywny język oparty na języku XML w którym za pomocą tagów i atrybutów opisywane są (głównie) elementy graficzne i ich właściwości, np.

***<Button x:Name="myButton" Content="Click me" Background="Aqua" />***

Parser języka XAML następnie generuje kod w którym tworzone są odpowiadające obiekty i powiązane z nimi właściwości. Hierarchiczna struktura języka XAML, style oraz szablony pozwalają na łatwiejsze projektowanie zaawansowanych interfejsów graficznych. Kontrolki nie są już powiązane z WinAPI jak to miało miejsce w WinForms. W WPF zastosowano również nowoczesny silnik graficzny – DirectX. Pozwala on na wyświetlanie bardziej zaawansowanej grafiki niż to miało miejsce w przypadku silnika GDI. Dodatkowo silnik ten oferuje większą wydajność poprzez lepsze wykorzystanie dostępnego sprzętu (głównie karty graficznej). W WPF zastosowany został mechanizm powiązania danych (data binding). Umożliwia on utworzenie połączenia pomiędzy elementem UI a obiektem znajdującym się w logice biznesowej. Połączenie takie umożliwia wzajemną aktualizację obu stron w momencie nastąpienia zmiany. Technika ta zapewnia separację logiki biznesowej od definicji interfejsu graficznego, co z kolei wpływa na czytelność i łatwość utrzymania aplikacji. WPF jest już również dojrzałą technologią i Microsoft nie planuje dalszych zmian (poza poprawkami błędów).

1.1.3 Silverlight

Silverlight jest technologią, która zadebiutowała w 2007 roku. Wykorzystuje ona wiele funkcjonalności dostępnych w WPF, ale aplikacje napisane w tej technologii uruchamiane są poprzez przeglądarkę internetową (np. Mozilla Firefox lub Internet Explorer). Technologia ta umożliwiała uruchamianie aplikacji na wielu systemach operacyjnych (m.in. macOS oraz Linux poprzez alternatywną implementację Moonlight). Silverlight wraz z kolejnymi wersjami miał problemy z kompatybilnością z przeglądarkami internetowymi oraz z wspieranymi systemami operacyjnymi. Obecnie Microsoft jak i większość przeglądarek zakończyło/planuje zakończenie wsparcia dla tej technologii.

1.1.4 Windows Runtime

WinRT jest interfejsem programistycznym, który został wprowadzony wraz z debiutem systemu Windows 8. Technologia ta nie bazuje już tak jak wcześniej wymienione na bibliotece .NET, lecz zastępuje ona wcześniej wykorzystywany interfejs WinAPI. Pomimo oderwania się od platformy .NET większość mechanizmów w niej zawartych zostało przeniesionych na WinRT. Nowy interfejs programistyczny wymusza na programiście stosowanie wywołań asynchronicznych przy operacjach wejścia/wyjścia w celu uniknięcia blokady interfejsu graficznego lub niepotrzebnego blokowania wątku podczas oczekiwania na dane. WinRT natywnie oferuje wsparcie dla procesorów x86 oraz ARM. Odłączenie się od platformy .NET wiąże się również z zmianą wspieranych języków programowania. Są to C#, VB.NET, JavaScript oraz C++/CX. Interfejsy graficzne natomiast projektuje się za pomocą wcześniej opisanego języka XAML oraz HTML/CSS. Języki te są natywnie wspierane przez system operacyjny. Aplikacje w WinRT są uruchamiane w odseparowanym środowisku (sandbox) kontrolowanym przez system operacyjny. Ma to zapewnić większe bezpieczeństwo poprzez ścisłą kontrolę dostępu aplikacji do zasobów systemu takich jak dysk, sieć oraz do podłączonych urządzeń. Urządzenia mobilne korzystają z odmiany WinRT o nazwie Windows Phone Runtime. Aplikacje WinRT są dystrybuowane poprzez sklep Windows (Windows Store). Podczas instalacji użytkownik proszony jest o zezwoleniu danej aplikacji na dostęp do chronionych części systemu (o ile takich aplikacja wymaga).

1.1.5 Universal Windows Platform

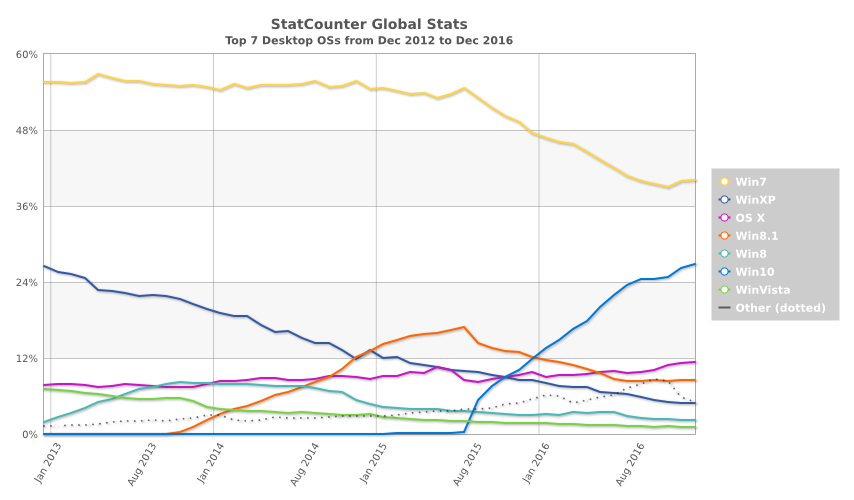
Wraz z premierą Windows’a 10 zadebiutowała nowa platforma - Universal Windows Platform (UWP). Technologia ta pozwala na tworzenie aplikacji, które można uruchomić na wszystkich urządzeniach wspierających system Windows 10. Obecnie są to następujące rodziny urządzeń:

* PC
* Mobilne
* IoT (np. Raspberry PI)
* Xbox
* HoloLens

Programista jest w stanie napisać 1 aplikację, która działa na wszystkich tych urządzeniach jednocześnie. Pomimo wieloplatformowości wciąż można wykorzystać funkcjonalności specyficzne dla danej rodziny urządzeń, np. operacje na pinach w urządzeniach IoT lub zintegrowane przyciski w urządzeniach mobilnych. Możliwe jest to dzięki rozszerzeniom SDK (Extension SDK). Dostarczają one programiście API pozwalające wykorzystać te dodatkowe funkcje jeżeli będą dostępne. Inną kluczową cechą UWP jest adaptacyjny interfejs użytkownika. Zależy on od wielkości ekranu i jego rozdzielczości – DPI, sposobu komunikacji użytkownika z urządzeniem oraz typowej odległości w jakiej użytkownik korzysta z niego. W rezultacie programista podając wymiary elementów operuje na efektywnych pikselach (a nie fizycznych). Taki sposób definiowania rozmiarów powoduje, że fizycznie większy rozmiar czcionki wyświetlanej na ekranie telewizora będzie z perspektywy użytkownika odbierany tak samo jak fizycznie mniejszy rozmiar czcionki na ekranie smartfona, który znajduje się bliżej użytkownika. Dodatkowo programista ma szereg narzędzi pozwalających na uzyskanie jeszcze lepszego interfejsu użytkownika, np. dostosowujący się w zależności od dostępnego na ekranie miejsca lub ze względu na sposób komunikacji użytkownika z urządzeniem.

* 1. Uzasadnienie wyboru technologii.

Do napisania odtwarzacza muzycznego wykorzystano Universal Windows Platform. Jest to najnowsza technologia do tworzenia aplikacji z bogatym interfejsem użytkownika na urządzenia z zainstalowanym systemem Windows 10. Architektura UWP umożliwia uruchomienie aplikacji na wielu rodzinach urządzeń oraz zawiera szereg udogodnień zapewniających odpowiedni wygląd aplikacji na nich. Analizując dane z Rysunku 1 można zauważyć, że system Windows 10 posiada trend rosnący, a najpopularniejszy obecnie desktopowy system Windows 7 ma trend malejący. W styczniu 2020 r. Microsoft kończy wsparcie dla systemu Windows 7[[1]](#footnote-1).



Rysunek - procentowy udział desktopowych systemów operacyjnych w okresie od 12.2012 do 12.2016. Dane zebrane na podstawie odwiedzin stron w sieci StatCounter[[2]](#footnote-2).

Do definiowania interfejsu użytkownika został użyty język XAML, który sprawdził się w poprzednich technologiach (WPF, Silverlight, WinRT). Wadą i jednocześnie zaletą UWP jest konieczność dystrybucji aplikacji poprzez Windows Store. Dla programisty wiąże się to z koniecznością założenia konta i uiszczenia jednorazowej opłaty wynoszącej $19 USD dla konta indywidualnego lub $99 USD dla konta firmowego[[3]](#footnote-3). Z drugiej strony aplikacje w Windows Store są sprawdzane pod kątem bezpieczeństwa, wydajności oraz kompatybilności. Dodatkowo użytkownik aplikacji jest jawnie informowany do jakich zasobów systemu operacyjnego dana aplikacja wymaga dostępu.

2. Podstawy teoretyczne.

1. Specyfikacja zewnętrzna.
   1. Instalacja

Aplikacja jest dostępna poprzez Windows Store i jej instalacja wymaga jedynie kliknięcia w przycisk „Install”.

* 1. Wymagania sprzętowe i programowe

|  |  |
| --- | --- |
| System operacyjny | Windows 10 |
| Procesor | 1 GHz lub szybszy |
| Pamięć RAM | 2 GB lub więcej |
| Wolne miejsce na dysku twardym | 100 MB lub więcej |
| Karta graficzna | Wspierająca Directx 9 lub nowsze z sterownikiem WDDM w wersji 1.0 lub nowszej |
| Wyświetlacz | O rozdzielczości minimalnej 800x600 |
| Połączenie z Internetem | 1 Mb/s |
| Urządzenia peryferyjne[[4]](#footnote-4) | Klawiatura, mysz |

* 1. Opis funkcjonalności

Odtwarzacz muzyczny TraxxPlayer posiada następujące funkcjonalności:

-odtwarzanie muzyki pochodzącej z serwisu SoundCloud

-dodawanie utworów do ulubionych

-tworzenie i modyfikacja list odtwarzania

-dodawanie utworów do list odtwarzania z uwzględnieniem ich kolejności

-wyszukiwanie muzyki znajdującej się w serwisie SoundCloud

-tworzenie lokalnych kont użytkownika

-przeglądanie ostatnich odtwarzanych przez nas utworów  
Specyfikacja zewnętrzna to opis funkcjonalności dostępnych dla użytkownika końcowego. Opisuje ona interfejs użytkownika oraz możliwości aplikacji.

3.4 Użytkowanie aplikacji

W specyfikacji zewnętrznej powinno się znaleźć wszystko to, co przeciętny użytkownik powinien wiedzieć o programie w celu jego prawidłowego użytkowania. Produkt opisywany jest z punktu widzenia użytkownika, więc należy się wystrzegać terminów z zakresu specyfikacji wewnętrznej.

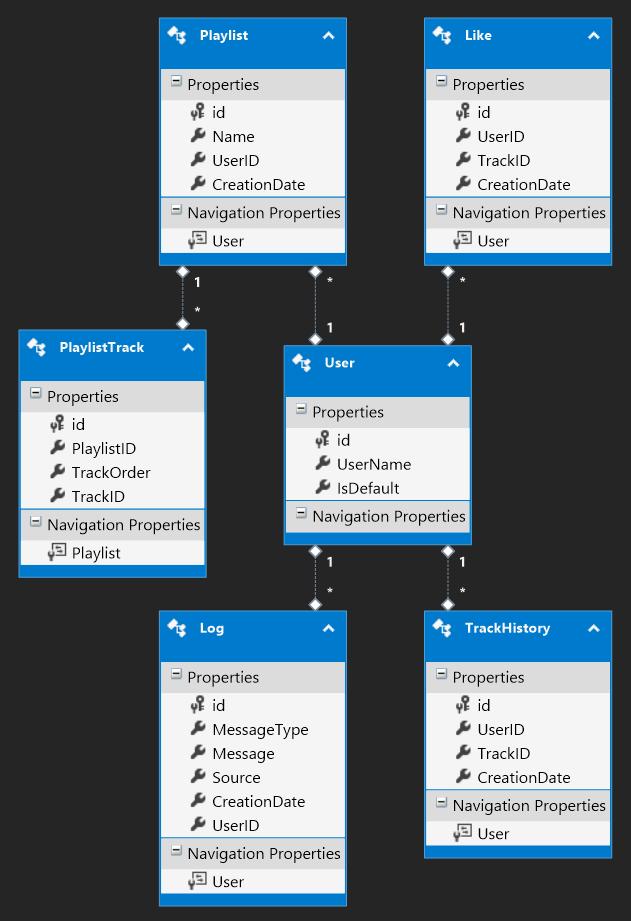
Porównanie technologii:

<https://developers.soundcloud.com/docs/api/sdks> napisac, ze nie ma wrappera dla c#  
4. Specyfikacja wewnętrzna.

4.1 Baza danych

Do stworzenia bazy danych wykorzystano system do zarządzania bazą danych SQLite oraz narzędzie typu ORM (Object Relational Mapping) Entity Framework Core. Rozważono również użycie systemu Microsoft SQL Server, jednak w okresie pisania pracy nie był on wspierany dla technologii UWP[[5]](#footnote-5). System SQLite okazał się dobrym rozwiązaniem, ponieważ w przeciwieństwie do MS SQL Server jest darmowym rozwiązaniem i ma bardzo mało zależności. SQLite do działania na docelowym urządzeniu nie wymaga żadnego dodatkowego oprogramowania. Większość systemów bazodanowych ma architekturę typu klient-serwer, a do komunikacji korzystają z protokołu TCP-IP. SQLite nie posiada serwera, a komunikacja aplikacji z bazą danych jest bezpośrednia[[6]](#footnote-6). Dzięki temu silnik ten charakteryzuje się wysoką wydajnością oraz nie wymaga żadnej konfiguracji do prawidłowego działania. Baza danych w SQLite jest zawarta w 1 pliku co umożliwia przenoszenie jest w łatwy sposób z jednego urządzenia na drugie. Istotną zaletą SQLite jest również wieloplatformowość, co doskonale łączy się z platformą UWP.

Narzędzie Entity Framework Core zostało użyte do utworzenia tabeli wraz z relacjami na podstawie klas w programie (podejście Code First). Do modyfikacji bazy danych został wykorzystany mechanizm migracji tworzący zapytania aktualizujące schemat bazy danych na podstawie zmian w modelu (klasach).



Rysunek - schemat bazy danych

User – tabela zawierająca użytkowników. Aby ułatwić korzystanie z aplikacji zrezygnowano z przechowywania hasła dla użytkownika.

Pola

|  |  |  |
| --- | --- | --- |
| User | | |
| Tabela zawierająca listę użytkowników aplikacji | | |
| Nazwa | Typ | Opis |
| id | Integer | Klucz główny |
| UserName | String | Nazwa użytkownika |
| IsDefault | Boolean | Flaga oznaczająca domyślnego użytkownika |

|  |  |  |
| --- | --- | --- |
| Playlist | | |
| Tabela zawierająca listy odtwarzania dla każdego użytkownika | | |
| Nazwa | Typ | Opis |
| id | Integer | Klucz główny |
| Name | String | Nazwa listy odtwarzania |
| UserID | Integer | Klucz obcy do tabeli User |
| CreationDate | DateTime | Data utworzenia listy odtwarzania |

|  |  |  |
| --- | --- | --- |
| PlaylistTrack | | |
| Tabela zawierająca listę utworów dla każdej listy odtwarzania | | |
| Nazwa | Typ | Opis |
| id | Integer | Klucz główny |
| PlaylistID | Integer | Klucz obcy do tabeli Playlist |
| TrackOrder | Integer | Kolejność utworu w liście odtwarzania |
| TrackID | Integer | ID utworu w serwisie SoundCloud |

|  |  |  |
| --- | --- | --- |
| Like | | |
| Tabela zawierająca listę polubionych utworów dla każdego użytkownika | | |
| Nazwa | Typ | Opis |
| id | Integer | Klucz główny |
| UserID | Integer | Klucz obcy do tabeli User |
| TrackID | Integer | ID utworu w serwisie SoundCloud |
| CreationDate | DateTime | Data dodania utworu do ulubionych |

|  |  |  |
| --- | --- | --- |
| Log | | |
| Tabela przechowująca dziennik zdarzeń aplikacji | | |
| Nazwa | Typ | Opis |
| id | Integer | Klucz główny |
| MessageType | Integer | Typ wiadomości: 0 – INFO (informacja) 1 – WARNING (ostrzeżenie) 2 – ERROR (błąd) |
| Message | String | Wiadomość |
| Source | String | Miejsce w programie, które dodało wiadomość |
| CreationDate | DateTime | Data dodania wiadomości |
| UserID | Integer | Klucz obcy do tabeli User |

|  |  |  |
| --- | --- | --- |
| TrackHistory | | |
| Tabela przechowująca historię odtwarzanych utworów | | |
| Nazwa | Typ | Opis |
| id | Integer | Klucz główny |
| UserID | Integer | Klucz obcy do tabeli User |
| TrackID | Integer | ID utworu w serwisie SoundCloud |
| CreationDate | DateTime | Data odtworzenia utworu |

Struktura polucji

TraxxPlayer.Common

Projekt ten zawiera klasy i stałe używane przez projekty TraxxPlayer.BackgroundAudioTask oraz TraxxPlayer.UI.

Enums and constans – typy wyliczeniowe i stałe. Przykładową klasą ze stałymi jest SoundCloudConstants zawierająca unikalny identyfikator aplikacji w SoundCloud oraz składowe adresu używane w innych miejscach aplikacji do pobierania danych poprzez interfejs programistyczny SoundCloud.

Exceptions – wyjątki używane do sygnalizacji błędów w aplikacji.

Helpers – klasy pomocnicze. Np.:  
SoundCloudHelper – metody pobierające określone dane z serwisu SoundCloud

Logger – metody służące do zapisywania wiadomości do dziennika zdarzeń w bazie danych (tabela Log).

Messages – wiadomości wymieniane pomiędzy aplikacją pierwszoplanową, a wątkiem w tle odpowiedzialnym za odtwarzanie muzyki. Przykładem wiadomości jest SkipNextMessage, która jest wysyłana do odtwarzacza w tle w momencie gdy użytkownik kliknie przycisk odpowiedzialny za przejście do następnego utworu w liście odtwarzania.

Models – klasy, które są używane przy deserializacji danych odebranych w formacie JSON z serwisu SoundCloud. Przykładem jest klasa SoundCloudTrack, do której są mapowane dane utworu otrzymane z serwisu SoundCloud.

TraxxPlayer.Data

Jest to projekt bazy danych.

Migrations – automatycznie generowane klasy pomocnicze służące do utworzenia zapytań mających na celu zmianę bazy danych na podstawie zmian w modelu w aplikacji. Dzięki mechanizmowi migracji możliwa jest aktualizacja istniejącej bazy danych bez konieczności usuwania jej (i przechowywanych w niej danych).

Models – klasy, które są odwzorowane na tabele w bazie danych. Przykład:

public class PlaylistTrack

{

public int id { get; set; }

[ForeignKey("Playlist")]

public int PlaylistID { get; set; }

public int TrackOrder { get; set; }

public int TrackID { get; set; }

public Playlist Playlist { get; set; }

}

TraxxPlayerContext – klasa odpowiedzialna za dostęp do bazy danych. Poprzez nią wykonywane są takie operacje jak dodawanie, usuwanie i modyfikacja rekordów.

TraxxPlayer.Services

Projekt stanowiący warstwę dostępu do danych (Data Access Layer). Jest to jedyny projekt w aplikacji wykonujący operacje na bazie danych.

Helpers – klasy pomocnicze używane przy dodawaniu, modyfikowaniu i wyświetlaniu rekordów z bazy danych. Przykładowo klasa LikeToAdd pośrednicząca przy dodawaniu rekordu wygląda następująco:

public class LikeToAdd

{

public int UserID { get; set; }

public int TrackID { get; set; }

}

Klasa LikeToDisplay natomiast używana jest przy zwracaniu rekordów z bazy danych:

public class LikeToDisplay

{

public int id { get; set; }

public int UserID { get; set; }

public int TrackID { get; set; }

public DateTime CreationDate { get; set; }

}

Klasy z przyrostkiem Service – zawierają metody CRUD (Create Read Update Delete) wykonujące operacje dodawania, odczytu, aktualizacji i usuwania rekordów z bazy danych. Przykładowo klasa LikeService zawiera m.in. następujące metody:  
IEnumerable<LikeToDisplay> GetLikes(int userID)

void DeleteLike(int likeID)

void AddLike(LikeToAdd like)

void ModifyLike(LikeToDisplay like)

TraxxPlayer.UI

Projekt graficznego interfejsu użytkownika. Zawiera klasy i widoki odpowiedzialne za interakcję z użytkownikiem.

Assets – zawiera pliki graficzne wykorzystywane w aplikacji.

Converters – klasy pośredniczące pomiędzy daną zmienną, a jej prezentacją graficzną w aplikacji. Przykładowo klasa MillisecondsToSecondsConverter:

Pole CurrentPosition (typ double) zawiera aktualną odtwarzaną pozycję w utworze w milisekundach. Aby wyświetlić tą pozycję w przyzwoitej dla użytkownika formie klasa MillisecondsToSecondsConverter zamienia tę liczbę na ciąg tekstowy reprezentujący minutę i sekundę, np. wartość 80 000 zamieniana jest na ciąg „01:20”.

Styles – style wykorzystywane przez aplikację do definiowania wyglądu wybranych elementów.

ViewModels – klasy zawierające logikę biznesową. Tutaj przetwarzane są dane, które widoki później wyświetlają użytkownikowi. Przykładowo klasa NowPlayingViewModel zawiera pola:

ImageSource AlbumImage – grafika albumu

ImageSource PlayPauseImage – obraz wyświetlający grafikę odtwarzania lub pauzy

String TrackName – nazwa utworu

String AlbumTitle – nazwa albumu

Double TrackDuration – długość utworu

Double CurrentPosition – aktualna pozycja utworu

Metody:

void PlayButtonClicked() – kliknięcie przycisku odtwarzania / pauzy

void NextButtonClicked() – kliknięcie przycisku odtwarzania następnego utworu w liście odtwarzania

void PreviousButtonClicked() – kliknięcie przycisku odtwarzania poprzedniego utworu w liście odtwarzania

Views – widoki zawierające opis graficznego interfejsu użytkownika. Tutaj znajdują się pliki napisane w języku XAML zawierające definicję obiektów graficznych i ich właściwości.

Package.appxmanifest – jest to plik xml zawierający ogólne informacje o aplikacji, np.:

-nazwa aplikacji  
-wspierane orientacje ekranu  
-opis aplikacji  
-logo naszej aplikacji w różnych miejscach  
-funkcjonalności z jakich nasza aplikacja korzysta (np. kamera)  
-wersja naszej aplikacji

5. Eksperymenty.

6. Wnioski.

1. https://support.microsoft.com/en-us/help/13853/windows-lifecycle-fact-sheet [↑](#footnote-ref-1)
2. http://gs.statcounter.com/ [↑](#footnote-ref-2)
3. https://msdn.microsoft.com/pl-pl/windows/uwp/publish/account-types-locations-and-fees [↑](#footnote-ref-3)
4. Dotyczy tylko urządzeń klasy PC [↑](#footnote-ref-4)
5. https://msdn.microsoft.com/pl-pl/windows/uwp/data-access/entity-framework-7-with-sqlite-for-csharp-apps [↑](#footnote-ref-5)
6. https://sqlite.org/different.html [↑](#footnote-ref-6)