

Projekt aplikacji „Musify Management Panel”

Opracował: Jakub Toma

Spis treści

[1. Obraz logiczny aplikacji 3](#_Toc115129935)

[1.1. Diagram klas aplikacji 3](#_Toc115129936)

[1.2. Specyfikacja klas 3](#_Toc115129937)

[2. Statyczny opis modelu 4](#_Toc115129938)

[2.1. Diagram klas 4](#_Toc115129939)

[2.2. Implementacja klasy Song 5](#_Toc115129940)

[2.3. Implementacja klasy DBService 6](#_Toc115129941)

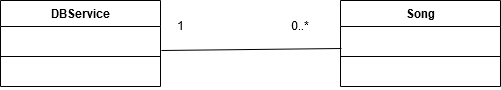
[3. Projekt bazy danych 11](#_Toc115129942)

[2.1. Opis bazy danych 11](#_Toc115129943)

[2.2. Diagram encji 11](#_Toc115129944)

# Obraz logiczny aplikacji

## Diagram klas aplikacji



## Specyfikacja klas

Obraz zawierający stół

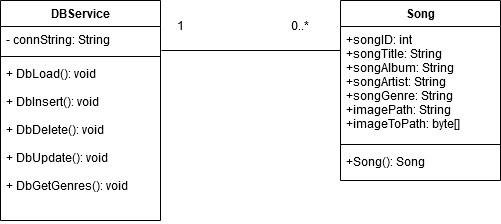
Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

# Statyczny opis modelu

## 2.1. Diagram klas



## 2.2. Implementacja klasy Song

class Song

{

public int songId { get; set; }

public string songTitle { get; set; }

public string songAlbum { get; set; }

public string songArtist { get; set; }

public string songGenre { get; set; }

public string imagePath { get; set; }

public byte[] imageToByte { get; set; }

public Song() { }

public Song(int songId, string songTitle, string songAlbum, string songArtist, string songGenre)

{

this.songId = songId;

this.songTitle = songTitle;

this.songAlbum = songAlbum;

this.songArtist = songArtist;

this.songGenre = songGenre;

}

public Song(Song song)

{

this.songId = song.songId;

this.songTitle = song.songTitle;

this.songAlbum = song.songAlbum;

this.songArtist = song.songArtist;

this.songGenre = song.songGenre;

}

}

## 2.3. Implementacja klasy DBService

class DBService

{

private static string connString = @"Data Source = PC-1\SQLEXPRESS; Initial Catalog = songdb; Trusted\_Connection=True";

SqlConnection cnn = new SqlConnection(connString);

public DBService()

{

}

public void DbLoad(DataGrid dataGrid)

{

SqlCommand cm = new SqlCommand("SELECT \* FROM song;", cnn);

DataTable dt = new DataTable();

cnn.Open();

try

{

SqlDataReader dr = cm.ExecuteReader();

dt.Load(dr);

dataGrid.ItemsSource = dt.DefaultView;

}

catch(SqlException ex)

{

MessageBox.Show("Połączenie nie powiodło się \n"+ ex.Message);

}

}

public void DbInsert(TextBox title, TextBox artist, TextBox album, string genre, string img)

{

cnn.Open();

Song song = new Song();

byte[] imgFile = song.imageToByte = File.ReadAllBytes(img);

SqlCommand cm = new SqlCommand("INSERT INTO song (song\_title, song\_artist, song\_album, song\_genre, song\_img, song\_img\_byte) VALUES (@song\_title, @song\_artist, @song\_album, @song\_genre, @song\_img, @song\_img\_byte)", cnn);

cm.CommandType = CommandType.Text;

cm.Parameters.AddWithValue("@song\_title", title.Text);

cm.Parameters.AddWithValue("@song\_artist", artist.Text);

cm.Parameters.AddWithValue("@song\_album", album.Text);

cm.Parameters.AddWithValue("@song\_genre", genre);

cm.Parameters.AddWithValue("@song\_img", img);

cm.Parameters.AddWithValue("@song\_img\_byte", imgFile);

try

{

cm.ExecuteNonQuery();

MessageBox.Show("Dodano do bazy danych!");

}

catch(SqlException ex)

{

MessageBox.Show("Dodanie do bazy nie powiodło się!\n" + ex.Message);

}

finally

{

cnn.Close();

}

}

public void DbDelete(string id)

{

cnn.Open();

SqlCommand cm = new SqlCommand("DELETE FROM song WHERE ID = @id", cnn);

cm.Parameters.AddWithValue("@id", id);

try

{

cm.ExecuteNonQuery();

MessageBox.Show("Usunięto rekord!");

}

catch (SqlException ex)

{

MessageBox.Show("Operacja nie powiodła się! \n" + ex.Message);

}

finally

{

cnn.Close();

}

}

public void DbUpdate(string id, TextBox title, TextBox artist, TextBox album, string genre, string img)

{

cnn.Open();

Song song = new Song();

SqlCommand cm = new SqlCommand("UPDATE song SET song\_title=@song\_title, song\_artist=@song\_artist, song\_album=@song\_album, song\_genre=@song\_genre, song\_img=@song\_img, song\_img\_byte=@song\_img\_byte WHERE id=@id", cnn);

byte[] imgFile = song.imageToByte = File.ReadAllBytes(img);

cm.Parameters.AddWithValue("@id", id);

cm.Parameters.AddWithValue("@song\_title", title.Text);

cm.Parameters.AddWithValue("@song\_artist", artist.Text);

cm.Parameters.AddWithValue("@song\_album", album.Text);

cm.Parameters.AddWithValue("@song\_genre", genre);

cm.Parameters.AddWithValue("@song\_img", img);

cm.Parameters.AddWithValue("@song\_img\_byte", imgFile);

try

{

cm.ExecuteNonQuery();

MessageBox.Show("Zaktualizowano rekord!");

}

catch(SqlException ex)

{

MessageBox.Show("Operacja nie powiodła się! \n" + ex.Message);

}

finally

{

cnn.Close();

}

}

public void DbGetGenres(ComboBox comboBox)

{

SqlCommand cm = new SqlCommand("SELECT \* FROM genre;", cnn);

DataTable dt = new DataTable();

cnn.Open();

try

{

SqlDataReader dr = cm.ExecuteReader();

dt.Load(dr);

comboBox.ItemsSource = dt.DefaultView;

comboBox.DisplayMemberPath = "genre\_name";

comboBox.SelectedValuePath = "id";

}

catch (SqlException ex)

{

MessageBox.Show("Nie można wczytać tabeli! \n" + ex.Message);

}

finally

{

cnn.Close();

}

}

}

# 3. Projekt bazy danych

## 2.1. Opis bazy danych

Baza danych „songdb” pozwala na przechowywanie informacji o utworach, które są potrzebne do poprawnego działania programu. W tabeli song znajdują się podstawowe informacje o utworze jak i informacje o lokalizacji jego miniaturki. Tabela genres zawiera listę gatunków muzycznych.

## 2.2. Diagram encji

