Projekt Programowanie Komputerów 4

System Zarządzania Bazami Danych

18.05.2021

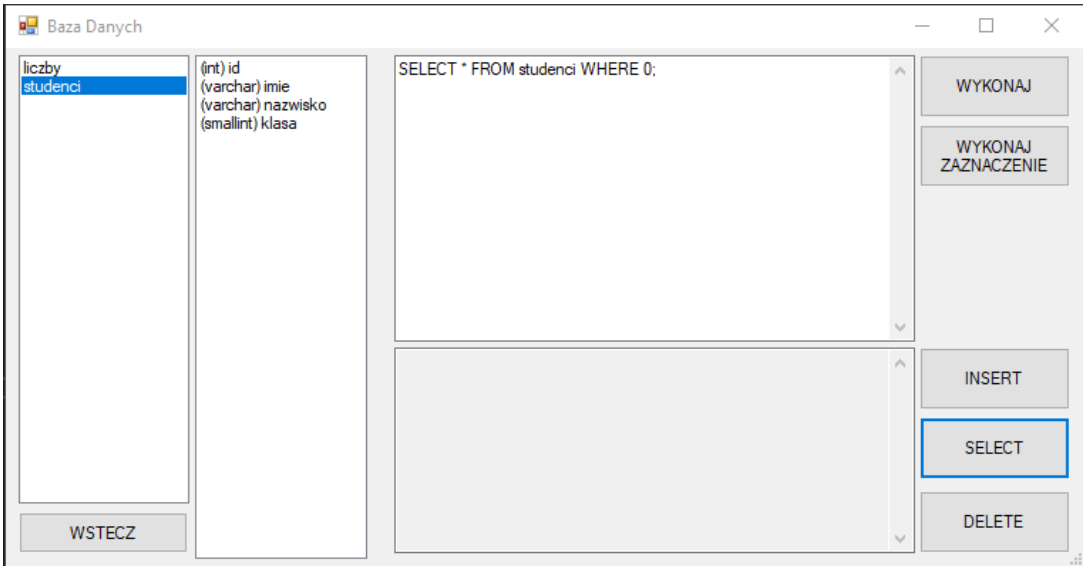
autor:
Rafał Jurczyk

prowadzący:
Krzysztof Pasterak

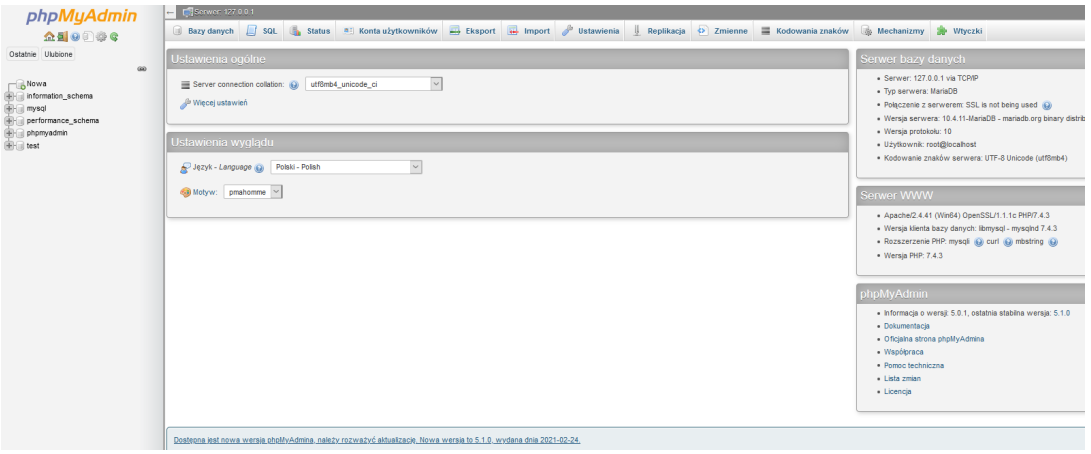
1. Analiza Tematu

1.1 Przedstawienie tematu

Projekt *'System Zarządzania Bazami Danych'* ma na celu stworzenie własnego systemu zarządzania bazami danych bez zbędnych funkcji tak, aby ktoś nie obeznany z komputerem mógł skorzystać z mojej aplikacji w celu szybkiej komunikacji z lokalną bazą danych.



Rys. 1. Przykładowe zdjęcie autorskiego programu do komunikacji z bazą danych



Rys. 2. Przykładowe zdjęcie strony phpmyadmin do komunikacji z bazą danych

Jak widzimy na zdjęciu 1 oraz 2 powyżej, zarówno program jak i strona mogą służyć do tego samego - wpisywaniu zapytań SQL do naszej bazy danych - jednak ktoś kto pierwszy raz wszedłby na przykładową stronę *phpmyadmin* mógłby być przytłoczony informacjami widocznymi na ekranie.

1.2 Wybór klas, algorytmów, bibliotek

Do napisania programu został użyty język C# w połączeniu z Windows Forms do stworzenia interfejsu graficznego. Jest to kombinacja, która w prosty i szybki sposób umożliwia stworzenie małych aplikacji.

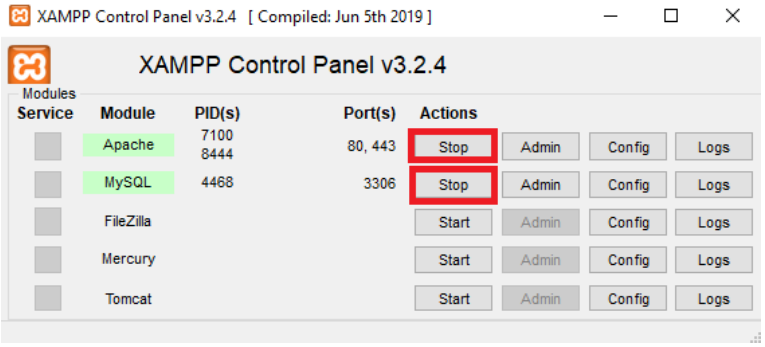
Ważnym powodem wykorzystania C# i pisania programu w środowisku Microsoft Visual Studio jest gotowa biblioteka MySQL.Data.MySqlClient, która pokrywa większość logiki podczas komunikacji z bazą danych. Z pomocą tej biblioteki można w prosty sposób połączyć się z bazami danych, zajmuje się ona występującymi błędami, przekazuje zapytania, itp.

W projekcie należało wykorzystać tematy przerabiane na laboratoriach. Użyte zostały:

- kontenery STL - np. tworzenie List<T>
- wyjątki - try{}catch{} idealnie nadawały się do przekazywania zapytań do bazy danych. W przypadku błędu wyrzucanego z bazy, catch{} łapie wyjątek i wyrzuca treść błędu
- algorytmy SQL - np. List.Sort()

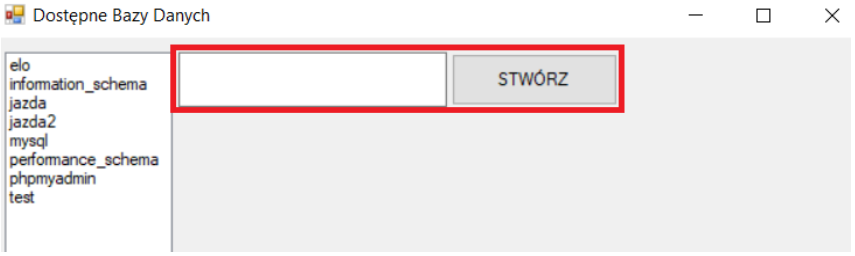
2. Specyfikacja Zewnętrzna

Aby korzystać z programu potrzebujemy przede wszystkim gotowej (może być pusta) bazy danych. Z założenia programu będziemy korzystać z bazy danych stworzonej lokalnie, a pomoże nam w tym program XAMPP (link do pobrania). Na podanej stronie powinniśmy też znaleźć instrukcję do instalacji. Po uruchomieniu go należy włączyć usługę Apache oraz MySQL tak jak na zdjęciu.



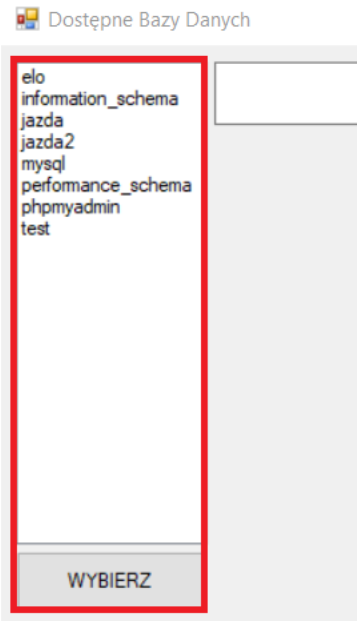
Rys. 3. Pomoc wizualna do aktywowania potrzebnych usług w programie XAMPP

Jeżeli chcemy możemy wejść na stronę `127.0.0.1/phpmyadmin` w celu tworzenia baz danych, jednak już teraz możemy połączyć się z nią naszym programem. Jeżeli nie mamy stworzonych żadnych baz danych możemy takową zainicjować w oknie zaznaczonym poniżej wpisując nazwę bazy danych, a następnie kliknięciu przycisku ‘Stwórz’.



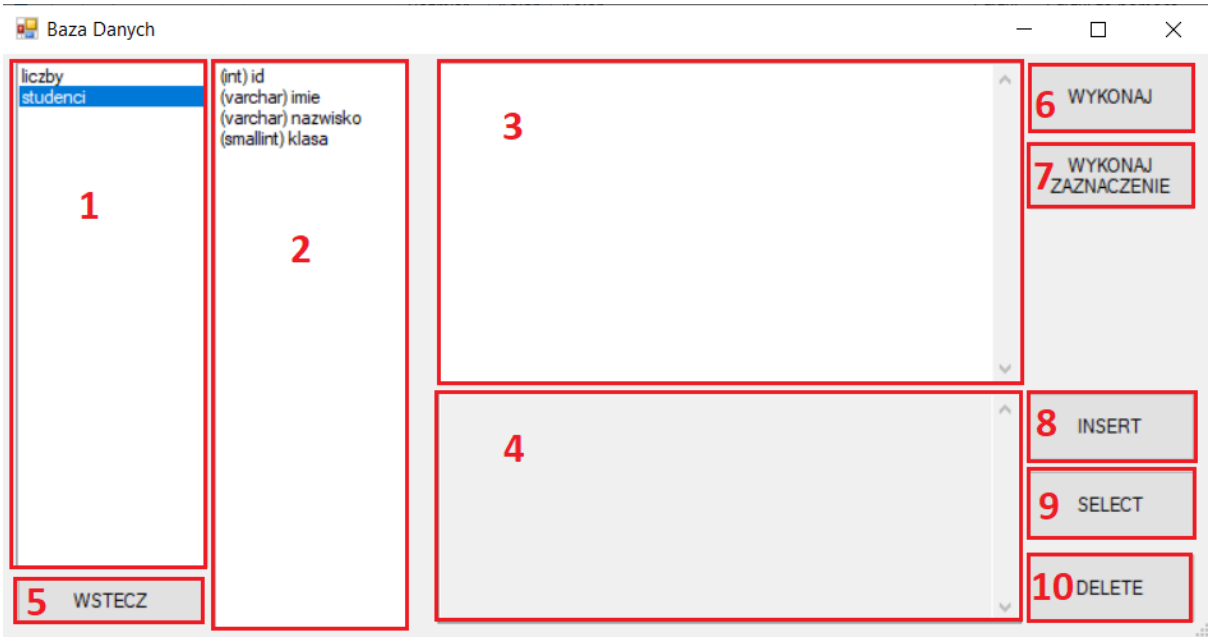
Rys. 4. miejsce do wpisania nazwy bazy danych do utworzenia

Po lewej stronie widzimy widoczne i dostępne już bazy danych, do których możemy się połączyć zaznaczając interesującą nas, a następnie klikając ‘Wybierz’. Bazy ‘information_schema’, ‘mysql’, ‘performance_schema’ oraz ‘phpmyadmin’ to bazy danych utworzone automatycznie. Ich widoczność możemy wyłączyć w opcjach programu (sekcja 3. Specyfikacja Wewnętrzna).



Rys. 5.Lista z której wybieramy interesującą nas bazę danych

Następnym krokiem po wybraniu bazy danych z którą chcemy się połączyć jest operowanie na niej. Pod spodem umieszczone jest zdjęcie z adnotacjami w postaci cyfr, które są opisane w dalszej części.

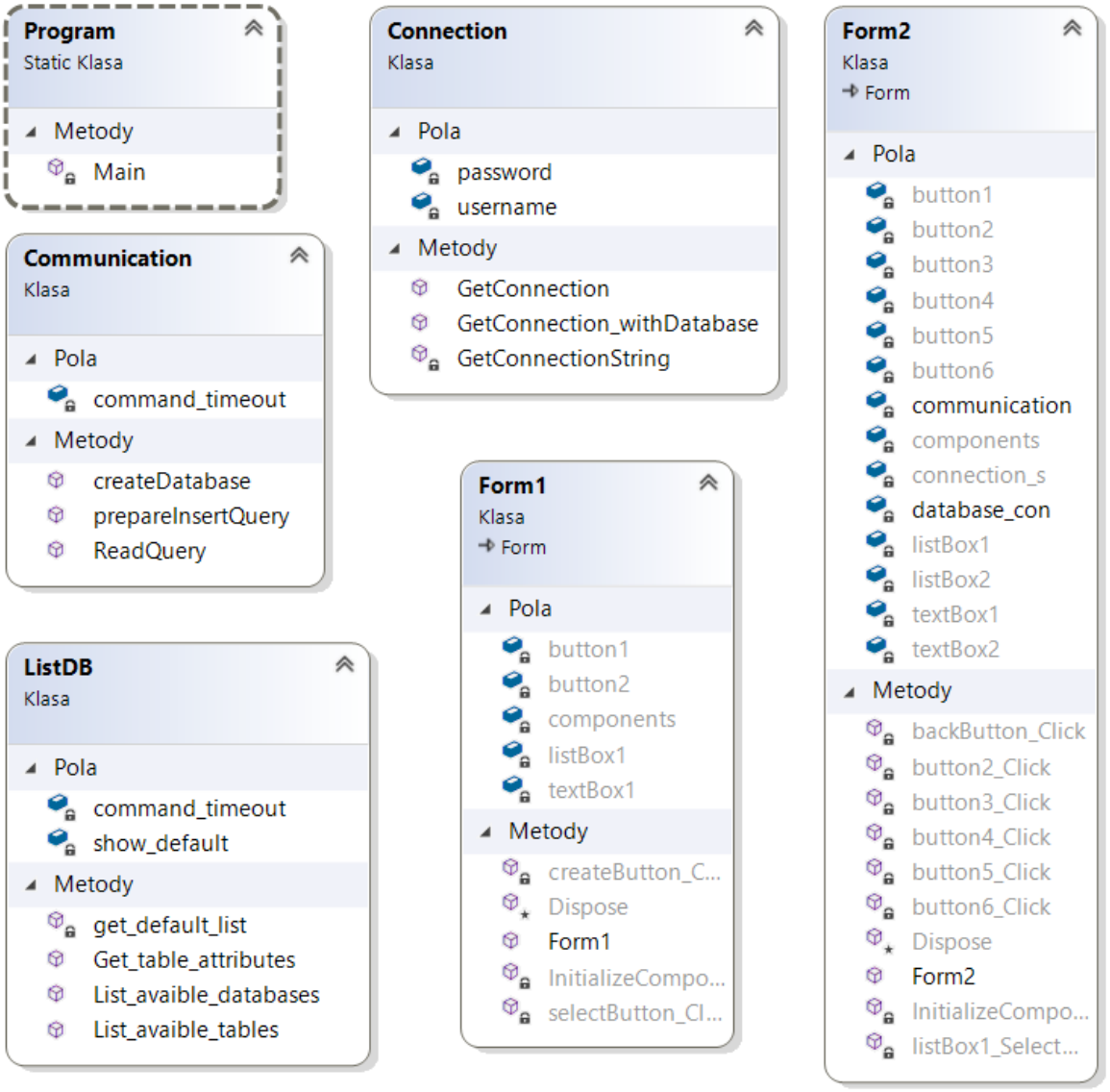


Rys. 6.Zdjęcie głównego okna programu razem z adnotacjami w postaci cyfr

1. Lista dostępnych tabel w bazie danych
2. Atrybuty wybranej tabeli razem z typem danych w nawiasach
3. Okno, w którym wpisujemy zapytania SQL
4. Okno, w którym wyświetlają się wyniki naszych zapytań
5. Po naciśnięciu wrócimy do wyboru bazy danych
6. Po kliknięciu wykonane zostaną wszystkie polecenia z okna trzeciego
7. Po kliknięciu wykonane zostaną wszystkie zaznaczone przez nas zapytania
8. Po kliknięciu w oknie numer 3 otrzymamy szablon polecenia INSERT
9. Po kliknięciu w oknie numer 3 otrzymamy szablon polecenia SELECT
10. Po kliknięciu w oknie numer 3 otrzymamy szablon polecenia DELETE

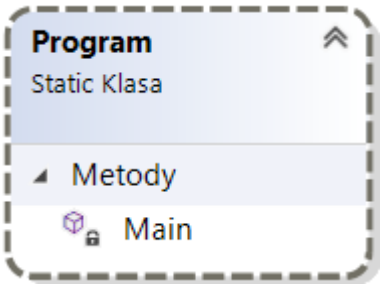
3. Specyfikacja Wewnętrzna

3.1 Zestawienie wszystkich napisanych klas:



Rys. 7.Zestawienie klas programu wygenerowane przez Microsoft Visual Studio

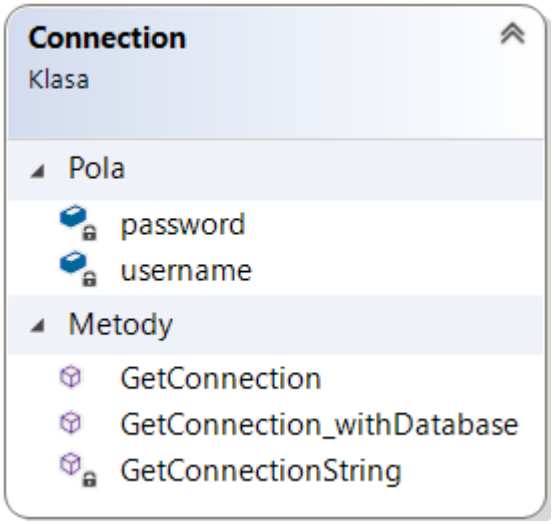
3.2 Szczegółowy opis ważniejszych metod i funkcjonalności klas



a)

Klasa **Program** to początek programu. Znajdziemy tam przede wszystkim metodę **main()** która inicjuje resztę programu. Po uruchomieniu aplikacji w metodzie main() wywołujemy klasę Connection, a następnie

zbieramy połączenie z naszą lokalną bazą danych i przekazujemy jej do tworzonego okna **Form1** tj. do reszty programu.



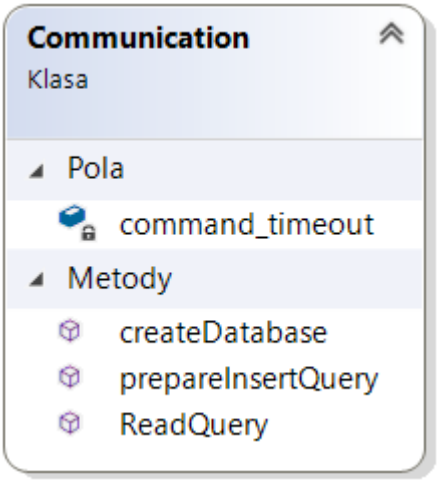
b)

Klasa **Connection** odpowiada za łączenie się ze określoną stroną na której znajduje się nasza baza danych. Domyślnie program jest napisany do łączenia się z bazą na localhostie, co sprawdzić możemy na linijce **connection_string** w metodzie **GetConnectionString()**. Jest to zapisany string ze stałym ip oraz numerem portu tj. *"datasource=127.0.0.1;port=3306;"*. Możemy również zmienić wartość zmiennej **username** i **password**, które domyślnie zakodowane są jako użytkownik root bez hasła.

GetConnection() jest używany w celu zdobycia połączenia właśnie ze stroną. Zwraca owe połączenie jako typ danej MySqlConnection z biblioteki MySql.Data.MySqlClient.

GetConnection_withDatabase(string database_name) to metoda która zwraca połączenie określone w **GetConnectionString()**, lecz dodatkowo na końcu naszego stringa dokleja przekazaną w zmiennej *database_name* nazwę bazy danych z którą chcemy się połączyć.

username - użytkownik jako który łączymy się ze stroną
password - hasło użytkownika jako który się łączymy
GetConnection - zwraca połączenie z określoną stroną
GetConnection_withDatabase - zwraca połączenie z określoną bazą danych
GetConnectionString - zwraca stringa, w której określona jest bazowa strona z którą się łączymy oraz jako kto jesteśmy połączeni



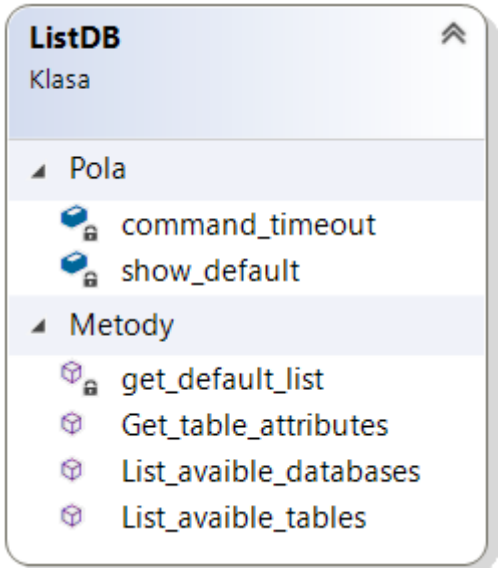
c)
Klasa **Communication** odpowiada za przekazywanie naszych własnych zapytań do bazy danych oraz obsłużenie odpowiedzi.

createDatabase - łączy się ze stroną do której określiliśmy połączenie w klasie **Connection**, a następnie tworzy bazę danych, której nazwę przekazujemy do tej właśnie metody

prepareInsertQuery - zwraca stringa z przygotowanym szablonem do zapytania typu INSERT dla wybranej przez nas tabeli.

ReadQuery - przekazuje nasze zapytanie do bazy danych, gdzie następnie jest ono wykonywane. Jeżeli zapytanie ma coś wyświetlić (np. SELECT) to metoda zwróci nam to w postaci stringa nadającego się do wyświetlenia w textBox.

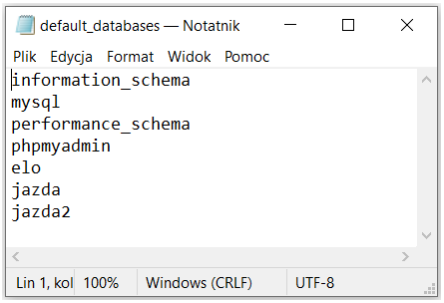
command_timeout - określa czas po którym program automatycznie się rozłączy z bazą danych z powodu braku odpowiedzi/reakcji



d)
Klasa **ListDB** służy do odbierania ze strony z którą jesteśmy połączeni listy baz danych, tabeli oraz informacji o nich.

command_timeout - określa czas po którym program automatycznie się rozłączy z bazą danych z powodu braku odpowiedzi/reakcji

show_default - zmienna typu bool, która decyduje o tym czy wyświetlą się (defaultowe) bazy danych określone przez nas w pliku tekstowym /DBMS/DBMS/data_files/default_databases.txt

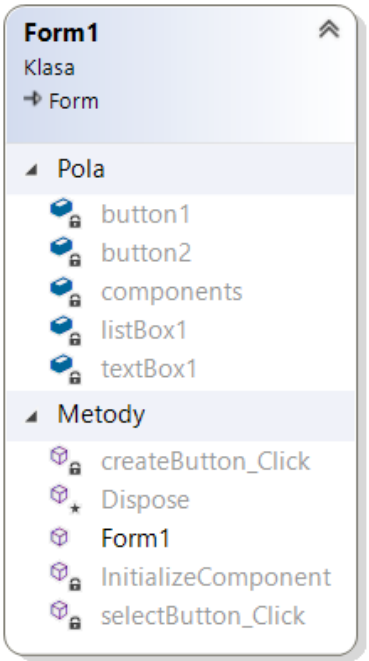


get_default_list - zwraca listę nazw baz danych, których nie chcemy wyświetlać. Ich nazwy są czytane z pliku tekstowego, do którego ścieżka jest określona przy opisie zmiennej `show_default`

Get_table_attributes - zwraca listę atrybutów każdej z kolumn w wybranej przez nas tabeli

List_avaible_databases - Łączy się ze stroną przez klasę **Connection**, a następnie zwraca listę dostępnych baz danych

List_avaible_tables - Łączy się ze stroną przez klasę **Connection**, a następnie zwraca listę dostępnych tabel z określonej przez nas bazy danych, której nazwa przekazywana jest w zmiennej



e) **Form1** to klasa w której określony jest wygląd pierwszego z naszych okien programu. W kodzie możemy znaleźć dokładny opis każdego z elementów (wymiary, kolor, itp.)

W dokumentacji opisane zostaną tylko funkcje każdego elementu oraz powiązane metody.

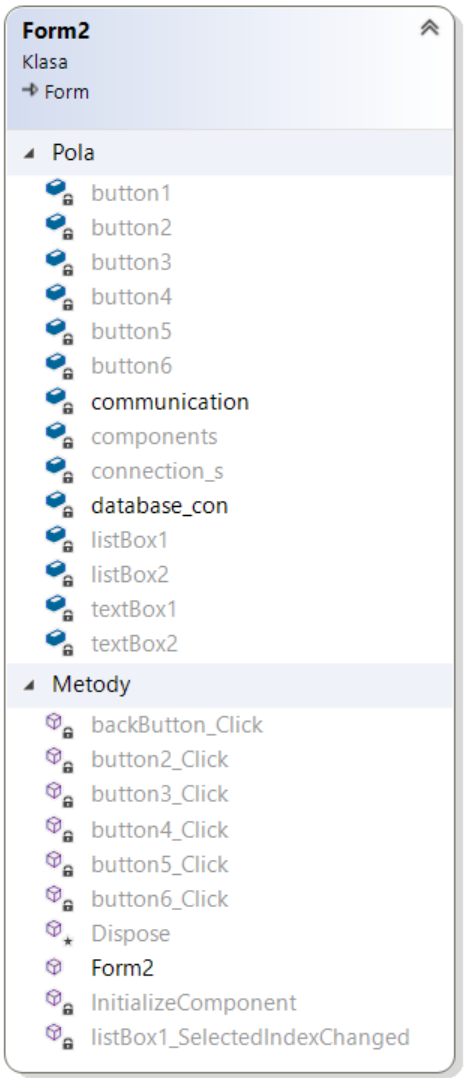
button1 - przycisk odpowiadający za potwierdzenie wyboru bazy danych.

Powiązany jest z metodą selectButton_Click(), która sprawdza czy wybrana baza danych jest prawidłowa, czy można się z nią połączyć. W przypadku powodzenia tworzy nam ona nowe okno z klasy **Form2** oraz zamyka obecne okno **Form1**. W przypadku niepowodzenia otrzymamy stosowny komunikat o błędzie.

button2 - przycisk odpowiadający za wysłanie nazwy bazy danej którą chcemy stworzyć do metody powiązanej createButton_Click(). Metoda ta łączy się ze stroną i przekazuje zapytanie SQL w formie "CREATE DATABASE *nazwa*", gdzie *nazwa* jest podawana przez użytkownika.

listBox1 - miejsce, w którym wyświetlane są dostępne bazy danych otrzymane z metody List_avaible_databases()

textBox1 - miejsce, w którym możemy wpisać nazwę bazy danych, którą chcemy utworzyć



f)
Form2 to klasa w której określony jest wygląd naszego drugiego okienka w programie. W kodzie możemy znaleźć dokładny opis każdego z elementów (wymiary, kolor, itp.)

W dokumentacji opisane zostaną tylko funkcje każdego elementu oraz powiązane metody.

button1 - po kliknięciu tego przycisku zostanie wywołana metoda backButton_Click(), która zamknie nasze aktualne okno i otworzy nowe okno **Form1**

button2 - po kliknięciu tego przycisku zostanie wywołana metoda button2_Click(), która zbierze cały

tekst z elementu textBox1 i wyśle go jako zapytanie na serwer SQL, z którym jesteśmy połączeniu.

button3 - po kliknięciu tego przycisku zostanie wywołana metoda button3_Click(), która zbierze cały zaznaczony przez nas tekst z elementu textBox1 i wyśle go jako zapytanie na serwer SQL, z którym jesteśmy połączeniu.

button4 - po kliknięciu tego przycisku zostanie wywołana metoda button4_Click(), która wyświetli w elemencie textBox1 przykładowy szablon zapytania DELETE do tabeli, którą zaznaczymy w listBox1

button5 - po kliknięciu tego przycisku zostanie wywołana metoda button5_Click(), która wyświetli w elemencie textBox1 przykładowy szablon zapytania SELECT do tabeli, którą zaznaczymy w listBox1

button6 - po kliknięciu tego przycisku zostanie wywołana metoda button6_Click(), która wyświetli w elemencie textBox1 przykładowy szablon zapytania INSERT do tabeli, którą zaznaczymy w listBox1. W środku metody w celu uzyskania informacji o wybranej tabeli najpierw połączymy się do wybranej bazy danych, a następnie w metodzie prepareInsertQuery z klasy **Communication** zostaną zwrócone interesujące nas atrybuty kolumn

listBox1 - miejsce w którym wyświetlane są dostępne tabele z wybranej bazy danych

listBox2 - miejsce w którym wyświetlane są kolumny i ich typy z wybranej tabeli z listBox1

textBox1 - miejsce w którym możemy wpisywać zapytania, które chcemy wysłać do bazy danych

textBox2 - miejsce w którym wyświetlane są wyniki zapytań