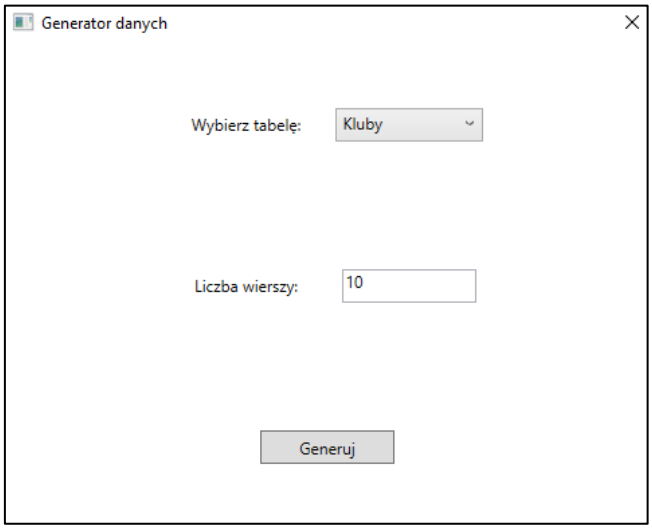
1. Opis środowiska programistycznego

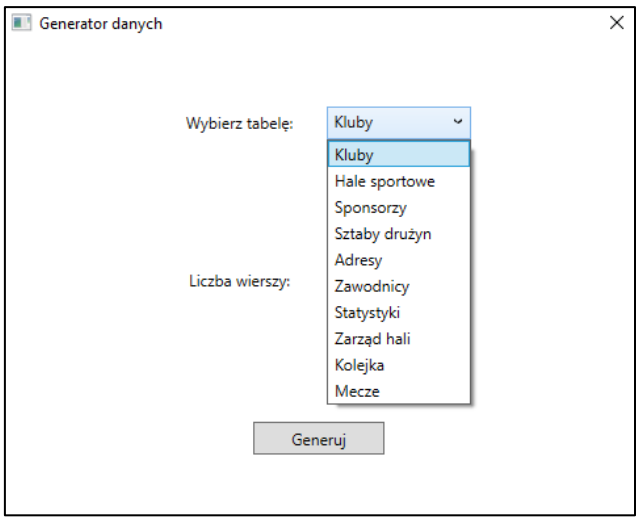
Generator danych został stworzony w środowisku Microsoft Visual Studio 2019, w technologii Windows Presentation Fundation (WPF) z wykorzystaniem języka programowania C#. Biblioteka (pakiet NuGET) wykorzystana do połączenia z bazą i manipulacji danymi z poziomu kodu źródłowego to *Oracle.ManagedDataAccess*.

2. Opis działania



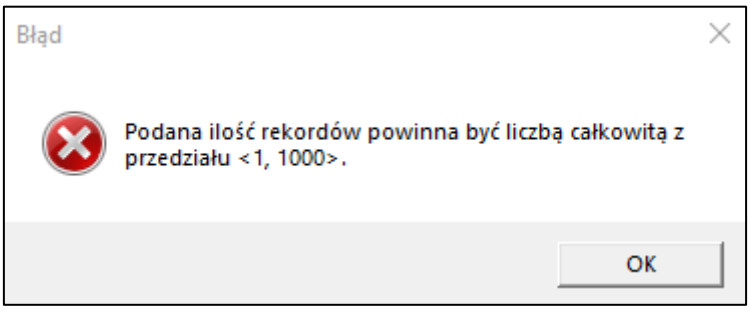
Zdjęcie 2.1 – GUI aplikacji

Zasada działania generatora polega na tym, iż w kontrolce ComboBox wybieramy tabelę, do której chcemy dodać rekordy. Następnie w kontrolce TextBox wpisujemy liczbę całkowitą z przedziału <1, 1000> dotyczącą ilości rekordów, które mają zostać wygenerowane. Wykonanie operacji zatwierdzamy przyciskiem „Generuj". Domyślnie po uruchomieniu aplikacjiwartość kontrolki ComboBox zostaje ustawiona na tabelę „Kluby", zaś ilość wierszy wynosi 10.



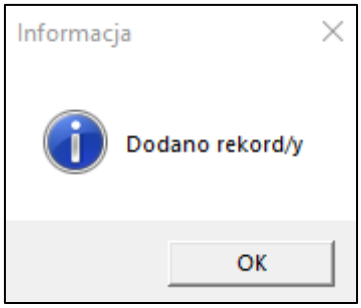
Zdjęcie 2.2 – Wszystkie wartości kontrolki ComboBox (tabele)

W przypadku wpisania liczby wykraczającej poza obsługiwany zakres zostanie wyświetlony odpowiedni komunikat.



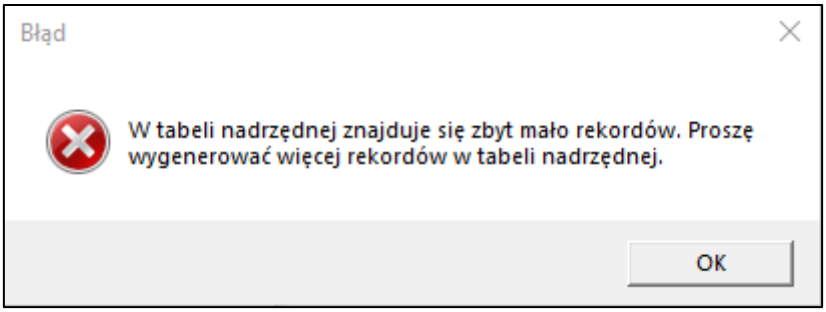
Zdjęcie 2.3 – Komunikat dotyczący liczby rekordów wykraczających poza obsługiwany zakres

Gdy operacja wstawiania rekordów do bazy danych się powiedzie, również zostanie wyświetlony komunikat.



Zdjęcie 2.4 – Komunikat dotyczący udanej operacji wstawiania rekordów do DB

W przypadku, gdy pomiędzy dwoma tabelami występuje relacja 1:1 (np. Kluby i Adresy – każdy klub ma jeden określony adres i każdy adres jest przyporządkowany jednemu konkretnemu klubowi) i w bazie będzie znajdować się 20 klubów, a w generatorze będziemy chcieli wygenerować 25 adresów to zostanie również wyświetlona stosowana informacja.



Zdjęcie 2.5 – Komunikat dotyczący niepoprawnej liczby rekordów w stosunku do danych znajdujących się w bazie danych

Dodatkowo wszystkie metody odpowiedzialne za wstawianie rekordów do określonej tabeli w bazie danych są asynchroniczne, aby nie blokować głównego wątku aplikacji przypadku dużej liczby rekordów.

```
public static async void InsertSponsorsToDatabaseAsync(int rows)
{
    List<string> commandsToTextFile = new List<string>();
    List<int> clubIds = new List<int>();
    clubIds.Clear();
    clubIds = GetIds("id_klubu", "Kluby");
    for (int i = 0; i < rows; i++)
    {
        var commandText = "insert into Sponsorzy (nazwa_sponsora, Kluby_id_klubu, wklad_pieniezny) values(:nazwa_sponsora, :Kluby_id_klubu, :wklad_pieniezny)";

        using (OracleConnection connection = new OracleConnection(SqlConnection.connectionString))
        {
            using (OracleCommand command = new OracleCommand(commandText, connection))
            {
                command.Parameters.Add(new OracleParameter("nazwa_sponsora", SD.sponsors[RandomElements.GetRandomNumber(0, SD.sponsors.Count-1)]));
                command.Parameters.Add(new OracleParameter("Kluby_id_klubu", clubIds[RandomElements.GetRandomNumber(0, clubIds.Count-1)]));
                command.Parameters.Add(new OracleParameter("wklad_pieniezny", RandomElements.GetRandomNumber(20000, 900000)));
                commandsToTextFile.Add("insert into Sponsorzy (nazwa_sponsora, Kluby_id_klubu, wklad_pieniezny) " +
                    "values(" + command.Parameters[0].Value.ToString() + ", " + command.Parameters[1].Value.ToString() + ", " + command.Parameters[2].Value.ToString() + ")");

                command.Connection.Open();
                await command.ExecuteNonQueryAsync();
                command.Connection.Close();
            }
        }
    }
    WriteCommandsToTextFile("Sponsorzy", commandsToTextFile);
    clubIds.Clear();
}
```

Zdjęcie 2.6 – Metoda asynchroniczna odpowiedzialna za wstawienie rekordów do tabeli Sponsorzy