Laboratorium 2 **Sprawozdanie z realizacji laboratorium**			
Temat: Praca z bazą danych SqlClient	**Nr Albumu:** 028487	**Grupa/zespół:** GL01	**Rok/semestr:** III / 6
Wykonał: Oleksii Hudzishevskyi	**Data wykonania:** 15/03/2023		**Data oddania:** 01/07/2023
	Ocena:		**Podpis prowadzącego:**

1. Spis treści

2. Cel ćwiczenia

Praktyczne wprowadzanie do Przetwarzania transakcyjnego. Wszystkie zadania w trakcie tego laboratorium wykonywane będą z wykorzystaniem dostawcy danych ADO.NET – SqlClient (System.Data.SqlClient). Na tym etapie studenci nie powinni wykorzystywać do tego celu żadnego ORM-a

3. Wymagania znajomości zagadnień

- Pisanie prostych aplikacji w C# lub innym obiektowym języku wysokiego poziomu
- Podstawowa znajomość SQL, umiejętność pisania zapytań do bazy danych
- Wskazana podstawowa znajomość języka angielskiego lub też umiejętność korzystania z narzędzi tłumaczenia on-line. Wynika to z faktu, że większość użytecznej i najbardziej aktualnej dokumentacji jest publikowana właśnie w języku angielskim.

4. Literatura, materiały dydaktyczne

- https://docs.microsoft.com/pl-pl/dotnet/csharp/
- https://www.sqlpedia.pl/
- https://www.mssqltips.com/sqlservertip/5771/querying-sql-server-tables-from-net/
- https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/
- https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-5.0
- https://docs.microsoft.com/pl-pl/dotnet/api/system.data.sqlclient.sqltransaction?view=dotnet-plat-ext-5.0
- https://www.webtrainingroom.com/adonet/transaction
- https://www.plukasiewicz.net/Artykuly/SQL_Transactions

5. Wiadomości teoretyczne.

Przydatne wskazówki:

- **Pobranie ID ostatnio dodanego do bazy danych rekordu**

W przypadku gdy mamy w bazie danych tabele dla której klucz główny (id) nadawane są przez bazę automatycznie i chcemy uzyskać dostęp do przydzielonego id czyli mamy włączone IDENTITY z opcją auto-increment, to możemy to zrealizować na przykład poprzez:

Na końcu CommandText w SqlCommand dopisujemy:

```
cmd.CommandText += " SELECT SCOPE_IDENTITY();";
```

teraz możemy wysłać polecenie do bazy ale wykorzystując do tego ExecuteScalar():

```
object value = cmd.ExecuteScalar();
if (value != DBNull.Value)
{
    string ResultId = value.ToString();
}
```

Oczywiście jeżeli chcemy pobrać wartość jako na przykład **long** to musimy zastosować odpowiednie rzutowanie.

Wywołując **ExecuteScalar** najbezpieczniej jest zastosowanie przypisania do **object**, wynika to z faktu, że zwrócone może być `DBNull` przy czym ważna uwaga to nie jest to samo co **null** w C#. Nie możemy bezpośrednio rzutować się pomiędzy tymi typami.

- **Zapisywanie do bazy danych rekordu zawierającego w danych pola null**

W sytuacji gdy chcemy zapisać do bazy danych jakikolwiek obiekt, który zawiera atrybuty nullable (dopuszczające null) lub też typu string musimy uwzględnić to przy budowaniu polecenia Insert (lub Update). Ponieważ w przypadku SqlClient zaleca się wykorzystanie parametryzowanych kwerend, to nie możemy przypisywać do parametru wartości null (jak już wspomniano nie jest to tym samym **null** z bazy danych). Przykład w jaki sposób można wykonać właściwe rzutowanie typów.

```
SqlCommand myCommand = new SqlCommand();
myCommand.CommandText = "INSERT INTO [InitialCatalog].[dbo].[Table] ( "
                      + "[ColumnName1], "
                      + "[ColumnName2], "
                      + "[ColumnName3] "
                      + " ) "
                      + "VALUES ( @val1, @val2, @val3 ) ; ";
```

```
myCommand.Parameters.AddWithValue("@val1", var1);
if (var2 == null) myCommand.Parameters.AddWithValue("@val2", DBNull.Value);
else myCommand.Parameters.AddWithValue("@val2", var2);
myCommand.Parameters.AddWithValue("@val3", var3);
```

Jak widzimy w przytoczonym przykładzie, zmienne var1 oraz var3 są zmiennymi niedopuszczającymi wartości **null** (wiemy że takowa nie wystąpi) więc przy bindowaniu parametrów można swobodnie wykorzystać `Parameters.AddWithValue` przy czym co istotne nie musimy wnikać jakiego dokładnie typu to będą dane (`int`, `string`, `DateTime`, `decimal` ….), nie ma znaczenia – środowisko sobie poradzi o ile nie będziemy mieli do czynienia z null. Dla tej sytuacji (zmienna var2) z przykładu widzimy uwarunkowanie bindowanej wartości: w przypadku null przekazujemy `DBNull.Value`, w pozostałych zaś przypadkach postępujemy tak samo jak w przypadku innych zmiennych z przykładu.

- **Odczytanie z bazy danych rekordu zawierającego w danych pola null**

W przypadku pobierania danych z obiektu Reader musimy postępować analogicznie, tzn. przed mapowaniem należy się upewnić czy zwracana dana ma wartość czy też jest to `DBNull` i w zależności od tego postępować różnie w trakcie odczytu. Przykład dla tego samego obiektu co powyżej:

```
                List<NaszTyp> results = new List<NaszTyp>();
            myCommand.CommandText = "SELECT "
                                    + "[ColumnName1], "
                                    + "[ColumnName2], "
                                    + "[ColumnName3] "
                                + "FROM [InitialCatalog].[dbo].[Table] "
                                 + "WHERE …… ";
                    try
                    {
                         myCommand.Connection.Open();
                         SqlDataReader    reader    =    myCommand.Execut-
eReader();
                         if (reader.HasRows)
                         {
```

```
                        while (reader.Read())
                        {

                       int var1 = (int)reader["ColumnName1"];
                           DateTime? Var2 = null;
                           if      (!reader.IsDBNull(reader.GetOrdi-
                   nal("ColumnName2")))
                           {
                              var2    =    (DateTime)reader["Column-
                   Name2"];
                           }
                           string  var3  =  ((string)reader["Column-
Name3"]).Trim();
                           result.Results.Add(new      NaszTyp(var1,
var2, var3));
                       }
                   }
                   reader.Close();
               }
               catch (SqlException e)
               {
                   … (Jakaś obsługa wyjątku, logowanie etc.)
               }
               finally
               {
                   if (myCommand.Connection != null) myCommand.Con-
nection.Close();
               }
```

- **Bardzo uproszczony przykład użycia transakcji w kodzie**

```
        using (SqlConnection connection = new SqlConnec-
        tion("Nasz ConnectionString"))
        {
            try
            {
                connection.Open();
                using (SqlTransaction trans = connection.Begin-
        Transaction())
                {
                    Tutaj nasze operacje na bazie danych(In-
        sert, Update, Delete);
                …
```

```
                …
                trans.Commit();
            }
        }
        catch (SqlException ex)
        {
            (Jakaś obsługa wyjątku, logowa-
    nie etc.)
        }
    }
```

6. Przebieg ćwiczenia

Stworzyć w bazie danych co najmniej 2 tabele (można sobie tę bazę przygotować z wykorzystaniem dowolnych narzędzi). Tabele mają zawierać jakieś ograniczenia, np. „minimum Value”. Następnie przygotować prosty program (aplikacja WinForms) pobierający dane od użytkownika (z formularza na ekranie) następnie wykonujący operacje zapisu tych danych do wskazanych tabel po naciśnięciu przycisku na ekranie. W pierwszych 2 częściach zadania wykonać to z wykorzystaniem ADO.NET – przykładowe snipety kodu były załączone do instrukcji laboratoryjnej. Uwaga, to są jedynie wycinki kodu (niekoniecznie kompletne) więc w razie problemów, wszelkie brakujące fragmenty omówimy na zajęciach.

W pierwszej części ćwiczenia zadanie wykonać jako dwie kolejne niepowiązane ze sobą operacje. W trakcie testów wprowadzić kolejno, zestaw danych zawierający prawidłowe wartości dla obu tabel, sprawdzić poprawność wykonania programu. Następnie podjąć próbę zapisania danych z zestawu zawierającego błędne dane, przy czym nieprawidłowość ta ma dotyczyć jednej z wybranych tabel. Sprawdzić zachowanie się programu oraz zmiany zachodzące w bazie danych po wywołaniu polecenia dodania nowych rekordów do tabel (w obu scenariuszach), spisać swoje uwagi oraz spostrzeżenia.

Po ukończeniu pierwszej części, należy zmodyfikować kod programu tak, aby operacje dodania wpisów do tych tabel zrealizowane zostały w obrębie jednej transakcji. Powtórzyć próby dodania danych analogicznie jak w poprzednim przypadku: zestaw prawidłowych danych, oraz zestaw zawierający błędy. Ponownie zweryfikować rezultat wykonywanych operacji w bazie danych, wyprowadzić i zanotować wnioski. Uzupełnić powyższe kody tak, by możliwym było zapisywanie rekordów zawierających null, zamiast wartości (oczywiście z uprzednią modyfikacją bazy danych, jeżeli będzie to konieczne). Mogą to być na przykład wartości liczbowe (integer) z dopuszczeniem null, czyli po stronie aplikacji są to typu nullable.

Zaprezentować wyniki działania kodu oraz omówić działanie. Jaka jest różnica pomiędzy NULL w aplikacji oraz NULL w bazie danych, co to w praktyce oznacza w trakcie pisania kodu dodającego takowe „wartości" do bazy danych.

W następnej części ćwiczenia postarać się odpowiedzieć na pytanie: co jest ekwiwalentem transakcji w przypadku ORM (EF 6). Przebudować kod z laboratorium tak by zrealizować oba powyższe punktu (niezależne oraz transakcyjne dodawanie danych do tabel) tak aby został on zrealizowany za pomocą EF 6. W razie wątpliwości bądź pytań, wszystkie niejasności mogą zostać wyjaśnione przez prowadzącego NA ZAJĘCIACH, osoby nieobecne będą musiały zmierzyć się z problemem bez dodatkowych wskazówek ze strony prowadzącego. Ale biorąc pod uwagę frekwencję na zajęciach, zakładam, że nikt z nieobecnych raczej problemów mieć nie będzie, bo wygląda na to, że dla wszystkich tematyka zajęć jest w pełni zrozumiała i nie jest wymagane dodatkowe omawianie tych problemów.

Uwaga, dla wszystkich części laboratorium dodać widok w aplikacji, wyświetlający wszystkie dane z tabel w stosownej kontrolce, DataGridView, przykład użycia kontrolki postaramy omówić się w trakcie zajęć.

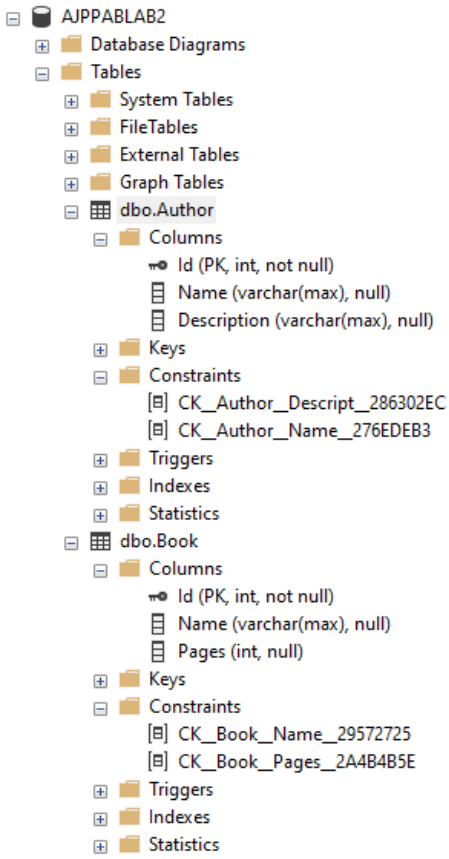
7. Opracowanie sprawozdania

Opracowanie sprawozdania jest podzielone na dwie części: ADO.NET i Entity Framework. W części **ADO.NET** wszystkie polecenia laboratoryjne zostały wykonane za pomocą standardowej biblioteki .NET'owej. W części **Entity Framework** wszystkie polecenia laboratoryjne zostały wykonane za pomocą Entity Framework'a wersji 7

7.1 ADO.NET

7.1.1 Niepowiązane transakcje

W bazie zostały utworzone 2 tabele (*Rysunek 1*), tabele te zawierają ograniczenia minimalnej i maksymalnej długości znaków i minimalnej i maksymalnej wartości. Ograniczenia te zostały wygenerowane za pomocą kwerendy SQL'owej pokazanej na *Snippet 1*

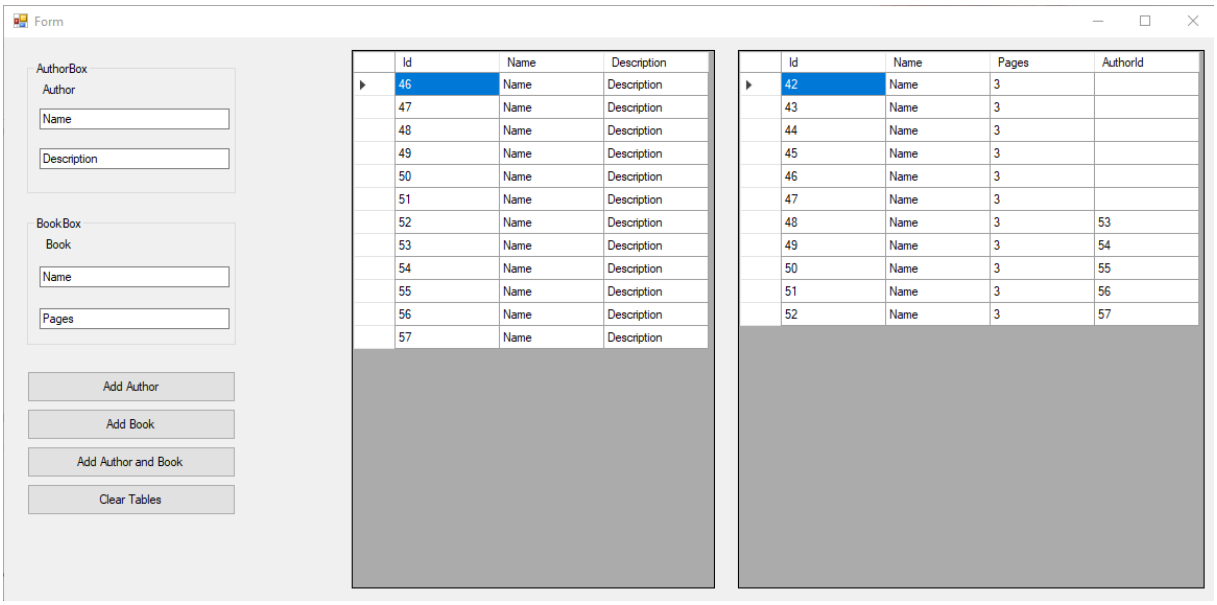


Rysunek 1. ADO.NET – Struktura bazy danych

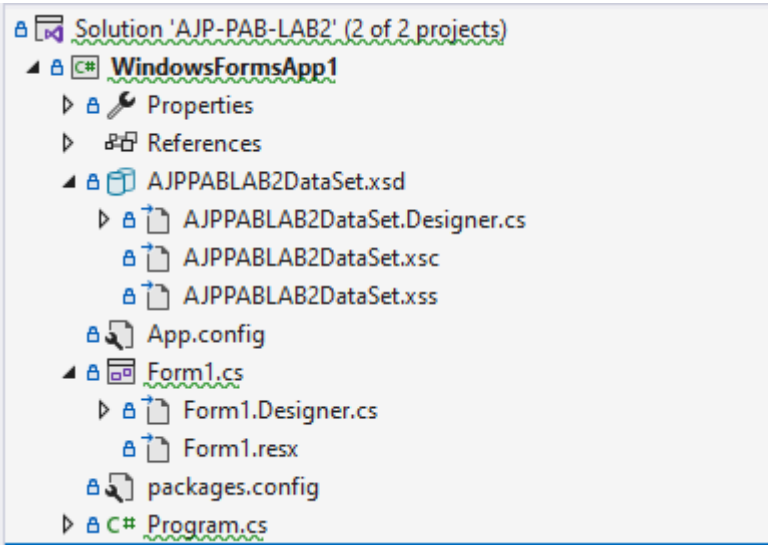
```
USE AJPPABLAB2
ALTER TABLE Author
ADD CHECK (LEN(Name) >= 1 AND LEN(Name) <= 20)

ALTER TABLE Author
ADD CHECK (LEN(Description) >= 1 AND LEN(Description) <= 100)

ALTER TABLE Book
ADD CHECK (LEN(Name) >= 1 AND LEN(Name) <= 40)

ALTER TABLE Book
ADD CHECK (Pages >= 3 AND Pages <= 2000)
```

Snippet 1. ADO.NET – Ograniczenia w kolumnach

Do realizacji laboratorium został przygotowany prosty program typu Windows Forms App (.NET Framework) pobierający dane użytkownika z formularza. Ten widok zawiera takie kontrolki jak TextBox, Label, Button i DataGridView. Wygląd tego programu jest pokazany na *Rysunek 2* a struktura projektu na *Rysunek 3*



Rysunek 2. ADO.NET - Program Windows Forms



Rysunek 3. ADO.NET - Struktura projektu

Następnie do przycisków, które znajdują się w Form1 zostały przypisane metody, które realizują funkcjonalność dodawania nowych wierszy do zaprojektowanych wcześniej tabel na podstawie danych zawartych w formularzach. Kod realizujący te funkcjonalności przedstawiony jest na *Snippet 2* i *Snippet 3* i *Snippet 4*

```
private void authorButton_Click(object sender, EventArgs e)
{
    try
    {
        string sqlExpression = "INSERT INTO Author (Name, Description) Values (@Name, @De-
scription)";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand(sqlExpression, connection);
```

```
            string authorName = authorNameText.Text;
            string authorDescription = authorDescriptionText.Text;

            SqlParameter authorNameParameter = new SqlParameter("@Name", authorName);
            SqlParameter authorDescriptionParameter = new SqlParameter("@Description", au-
thorDescription);

            command.Parameters.Add(authorNameParameter);
            command.Parameters.Add(authorDescriptionParameter);

            command.ExecuteNonQuery();
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show($"Occurred error: {ex.Message}");
    }
    finally
    {
        this.authorTableAdapter.Fill(this.aJPPABLAB2DataSet.Author);
    }
}
```

Snippet 2. ADO.NET - Author Button (1)

```
private void bookButton_Click(object sender, EventArgs e)
{
    try
    {
        string sqlExpression = "INSERT INTO Book (Name, Pages) Values (@Name, @Pages)";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand(sqlExpression, connection);

            string bookName = bookNameText.Text;
            int bookPages;
            if(!Int32.TryParse(bookPagesText.Text, out bookPages)) bookPages = 3;

            SqlParameter bookNameParameter = new SqlParameter("@Name", bookName);
            SqlParameter bookPagesParameter = new SqlParameter("@Pages", bookPages);

            command.Parameters.Add(bookNameParameter);
            command.Parameters.Add(bookPagesParameter);

            command.ExecuteNonQuery();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Occurred error: {ex.Message}");
    }
    finally
    {
        this.bookTableAdapter.Fill(this.aJPPABLAB2DataSet.Book);
    }
}
```

Snippet 3. ADO.NET - Book Button (1)

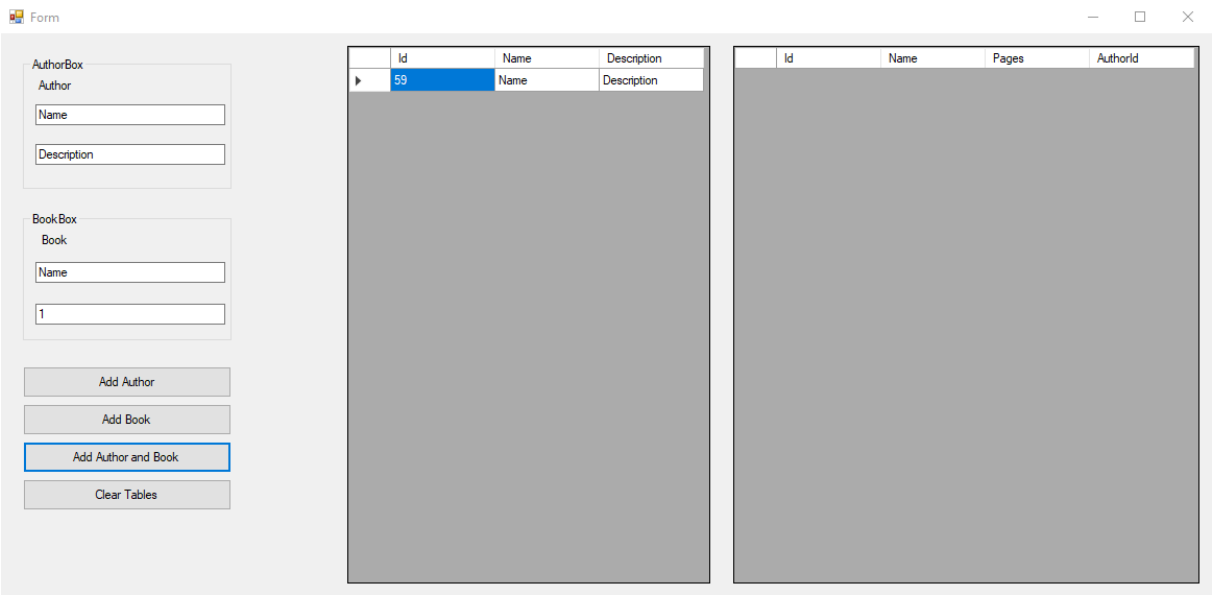
```
private void bothButton_Click(object sender, EventArgs e)
{
    int? authorId = null;
    try
    {
        string sqlExpression = "INSERT INTO Author (Name, Description) Values (@Name, @De-
scription); SELECT SCOPE_IDENTITY()";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand(sqlExpression, connection);

            string authorName = authorNameText.Text;
            string authorDescription = authorDescriptionText.Text;
```

```
            SqlParameter authorNameParameter = new SqlParameter("@Name", authorName);
            SqlParameter authorDescriptionParameter = new SqlParameter("@Description", au-
thorDescription);

            command.Parameters.Add(authorNameParameter);
            command.Parameters.Add(authorDescriptionParameter);
            authorId = Convert.ToInt32(command.ExecuteScalar());
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Occurred error: {ex.Message}");
    }
    finally
    {
        this.authorTableAdapter.Fill(this.aJPPABLAB2DataSet.Author);
    }

    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            string sqlExpression = "INSERT INTO Book (Name, Pages) Values (@Name, @Pages)";
            if (authorId != null)
            {
                sqlExpression = "INSERT INTO Book (Name, Pages, AuthorId) Val-
ues (@Name, @Pages, @AuthorId)";
            }

            connection.Open();
            SqlCommand command = new SqlCommand(sqlExpression, connection);

            string bookName = bookNameText.Text;
            int bookPages;
            if (!Int32.TryParse(bookPagesText.Text, out bookPages)) bookPages = 3;

            SqlParameter bookNameParameter = new SqlParameter("@Name", bookName);
            SqlParameter bookPagesParameter = new SqlParameter("@Pages", bookPages);
            command.Parameters.Add(bookNameParameter);
            command.Parameters.Add(bookPagesParameter);

            if (authorId != null)
            {
                SqlParameter authorIdParameter = new SqlParameter("@AuthorId", authorId);
                command.Parameters.Add(authorIdParameter);
            }

            command.ExecuteNonQuery();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Occurred error: {ex.Message}");
    }
    finally
    {
        this.bookTableAdapter.Fill(this.aJPPABLAB2DataSet.Book);
    }
}
```

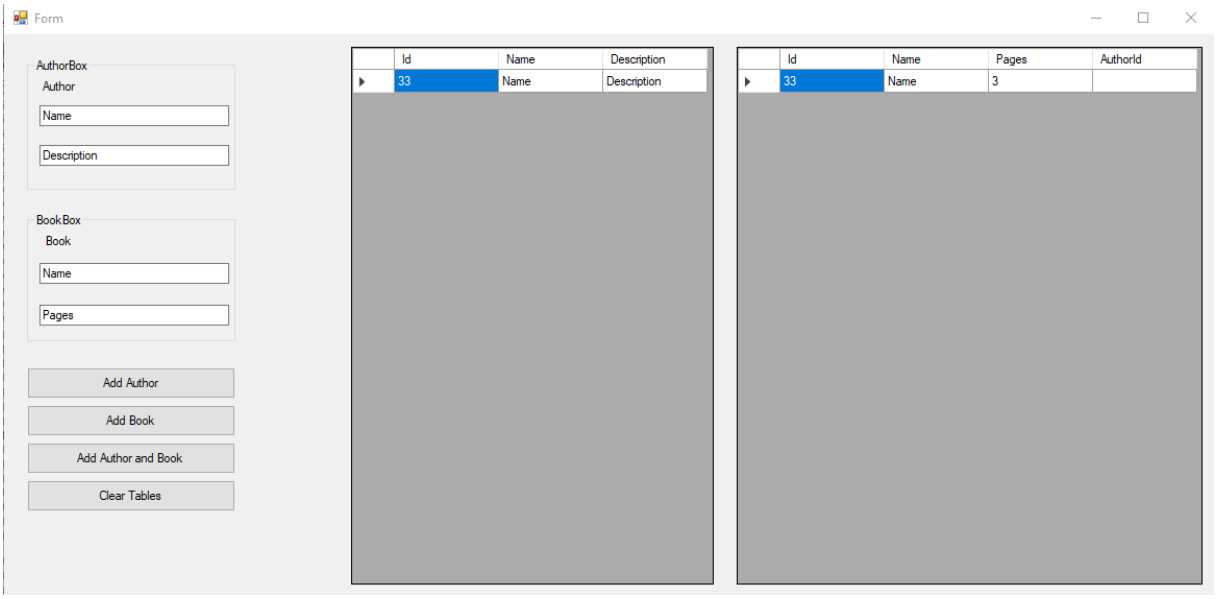
Snippet 4. ADO.NET - Author and Book Button (1)

Po kliknięciu przycisku dodającego autora dane zaczytywane z górnego formularza i zapisywane do tabeli w bazie danych, dane z tej tabeli wyświetlane na DataGridView, który znajduje się po lewej stronie. Po kliknięciu przycisku dodającego książkę dane zaczytywane z formularza znajdującego w kontrolce grupowej **Book Box** i zapisywane do tabeli w bazie danych, dane z tej tabeli wyświetlane na DataGridView, który znajduje się po prawej stronie.

Przykładowe działanie dodania autora jest pokazane na *Rysunek 4* i dodanie nowej książki pokazane na *Rysunek 5*

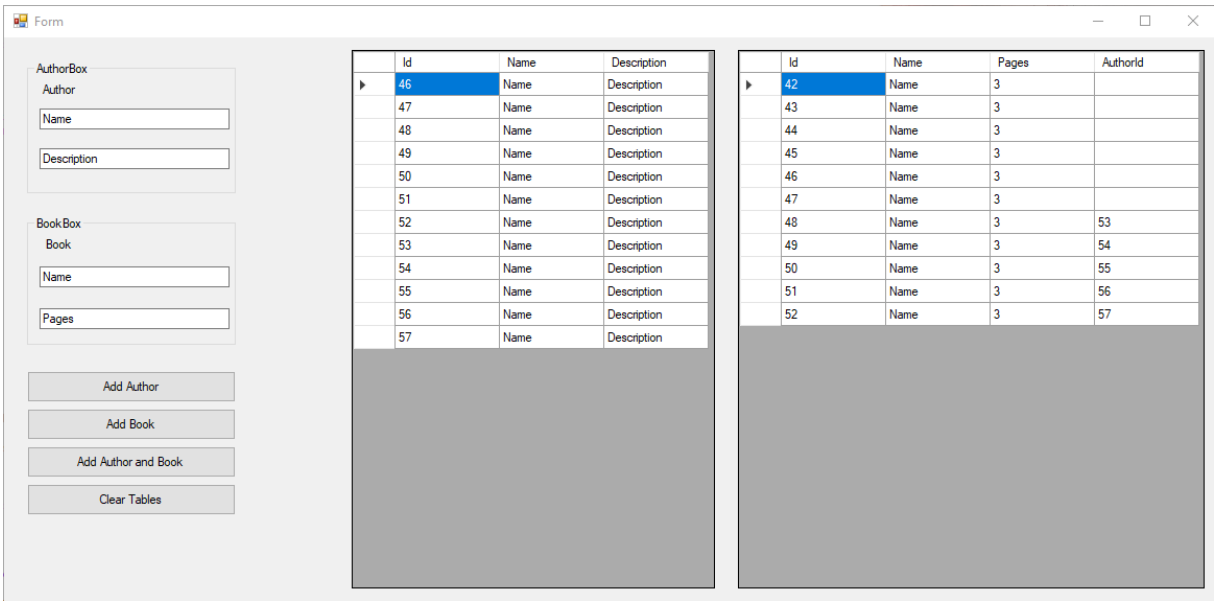


Rysunek 4. ADO.NET - Dodanie autora (1)



Rysunek 5. ADO.NET - Dodanie książki (1)

Po kliknięciu przycisku dodającego i autora i książkę na raz dane zaczytywane z obu formularzy na raz i zapisywane do bazy danych bez pomocy transakcji. Przy wysłaniu błędnych danych jeden z zapisów nie powiedzie się sukcesem i dane nie zostaną zapisane do bazy danych, lecz drugi dobry zapis będzie znajdować się w bazie danych. Przykładowe działanie tego kodu jest pokazane na *Rysunek 6*



Rysunek 6. ADO.NET - Zapis podwójny (1)

7.1.2 Powiązane transakcje

Następnie metoda dodająca dwa wiersze na raz zmodyfikowana w taki sposób, aby te operacje dodania wpisów do tabel realizowane były w obrębie jednej transakcji. Zmodyfikowany kod jest pokazany na *Snippet 5*

```
private void bothButton_Click(object sender, EventArgs e)
{
    // With transaction
    int? authorId = null;
    string sqlExpression = "INSERT INTO Author (Name, Description) Values (@Name, @Descrip-
tion); SELECT SCOPE_IDENTITY()";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        SqlTransaction transaction = connection.BeginTransaction();
        SqlCommand command = new SqlCommand(sqlExpression, connection);
        command.Transaction = transaction;

        try
        {
            string authorName = authorNameText.Text;
            string authorDescription = authorDescriptionText.Text;

            SqlParameter authorNameParameter = new SqlParameter("@Name", authorName);
            SqlParameter authorDescriptionParameter = new SqlParameter("@Description", au-
thorDescription);

            command.Parameters.Add(authorNameParameter);
            command.Parameters.Add(authorDescriptionParameter);
            command.CommandTimeout = 200;
            authorId = Convert.ToInt32(command.ExecuteScalar());

            sqlExpression = "INSERT INTO Book (Name, Pages) Values (@Name, @Pages)";
            if (authorId != null)
            {
                sqlExpression = "INSERT INTO Book (Name, Pages, AuthorId) Val-
ues (@Name, @Pages, @AuthorId)";
            }

            command.CommandText = sqlExpression;
```

```
            command.Parameters.Clear();

            string bookName = bookNameText.Text;
            int bookPages;
            if (!Int32.TryParse(bookPagesText.Text, out bookPages)) bookPages = 3;

            SqlParameter bookNameParameter = new SqlParameter("@Name", bookName);
            SqlParameter bookPagesParameter = new SqlParameter("@Pages", bookPages);
            command.Parameters.Add(bookNameParameter);
            command.Parameters.Add(bookPagesParameter);

            if (authorId != null)
            {
                SqlParameter authorIdParameter = new SqlParameter("@AuthorId", authorId);
                command.Parameters.Add(authorIdParameter);
            }

            command.ExecuteNonQuery();
            transaction.Commit();
            this.authorTableAdapter.Fill(this.aJPPABLAB2DataSet.Author);
            this.bookTableAdapter.Fill(this.aJPPABLAB2DataSet.Book);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Occurred error: {ex.Message}");
            transaction.Rollback();
            this.authorTableAdapter.Fill(this.aJPPABLAB2DataSet.Author);
            this.bookTableAdapter.Fill(this.aJPPABLAB2DataSet.Book);
        }
    }
}
```

Snippet 5. ADO.NET - Author and Book Button (2)

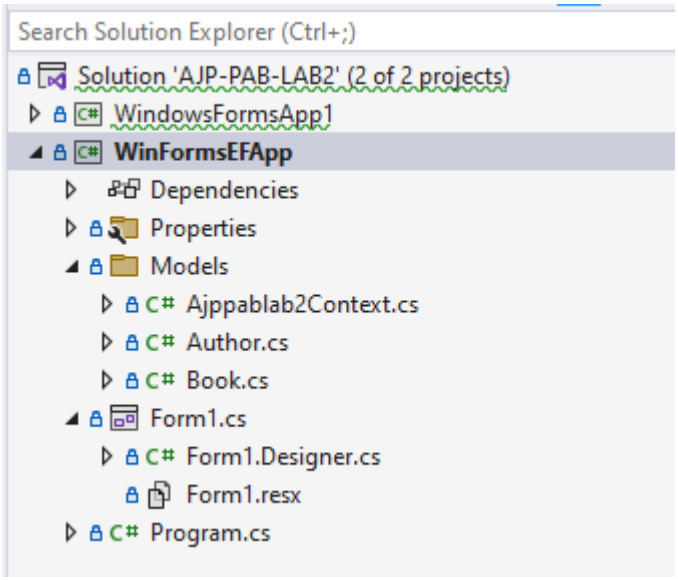
7.2 Entity Framework

Ekwiwalentem transakcji w przypadku EF jest komenda `BeginTransaction()` i komenda wycofania transakcji `Rollback()`. Przykładowe sposób wykorzystania tych komend jest pokazany na *Snippet 6. EF - Wykorzystanie transakcji*

```
private void Test()
{
    using (var dbContext = new MyDbContext())
    {
        using (var transaction = dbContext.Database.BeginTransaction())
        {
            try
            {
                // Wykonaj operacje bazodanowe (dodawanie, usuwanie, modyfikowanie rekordów)
                // ...

                // Zatwierdź transakcję
                dbContext.SaveChanges();
                transaction.Commit();
            }
            catch (Exception ex)
            {
                // Obsłuż wyjątek lub błąd
                // ...

                // Wycofaj transakcję
                transaction.Rollback();
            }
        }
    }
}
```

Snippet 6. EF - Wykorzystanie transakcji

Do realizacji tej części został utworzony nowy projekt Windows Forms z .NET 7. Struktura tego projektu pokazana jest na Rysunek 7. Kontekst i klasy tabel bazy danych zostały wygenerowane za pomocą polecenia `Scaffold-DbContext`.



Rysunek 7. EF - Struktura projektu

Następnie do przycisków, które znajdują się w Form1 zostały przypisane metody, które realizują funkcjonalność dodawania nowych wierszy do zaprojektowanych wcześniej tabel na podstawie danych zawartych w formularzach. Kod realizujący te funkcjonalności za pomocą EF7 przedstawiony jest na *Snippet 7* i *Snippet 8* i *Snippet 9*

```
private void authorButton_Click(object sender, EventArgs e)
{
    using (Ajppablab2Context db = new())
    {
        try
        {
            string authorName = authorNameText.Text;
            string authorDescription = authorDescriptionText.Text;

            db.Authors.Add(new() { Name = authorName, Description = authorDescription });
            db.SaveChanges();

            db.Authors.Load();
            this.authorBindingSource.DataSource = db.Authors.Local.ToBindingList();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Occured erorr {ex.Message}");
        }
    }
}
```

Snippet 7. EF - Author Button

```
private void bookButton_Click(object sender, EventArgs e)
{
    using (Ajppablab2Context db = new())
    {
        try
```

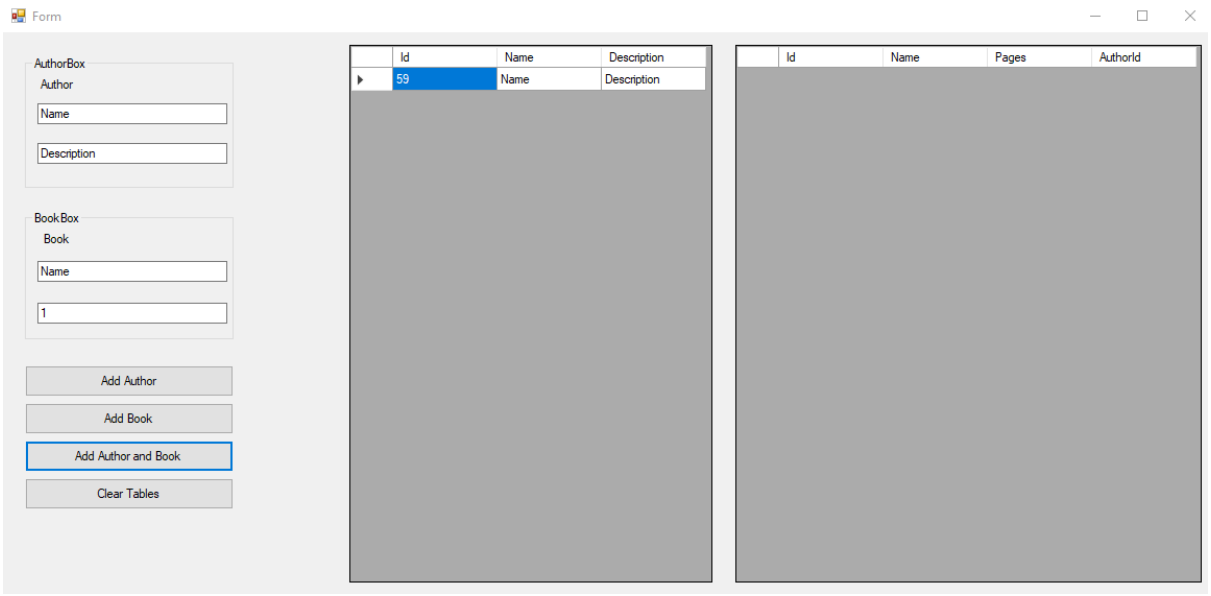
```
        {
            string bookName = bookNameText.Text;
            int bookPages;
            if (!Int32.TryParse(bookPagesText.Text, out bookPages)) bookPages = 3;

            db.Books.Add(new() { Name = bookName, Pages = bookPages });
            db.SaveChanges();

            db.Books.Load();
            this.bookBindingSource.DataSource = db.Books.Local.ToBindingList();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Occured erorr {ex.Message}");
        }
    }
}
```

Snippet 8. EF - Book Button

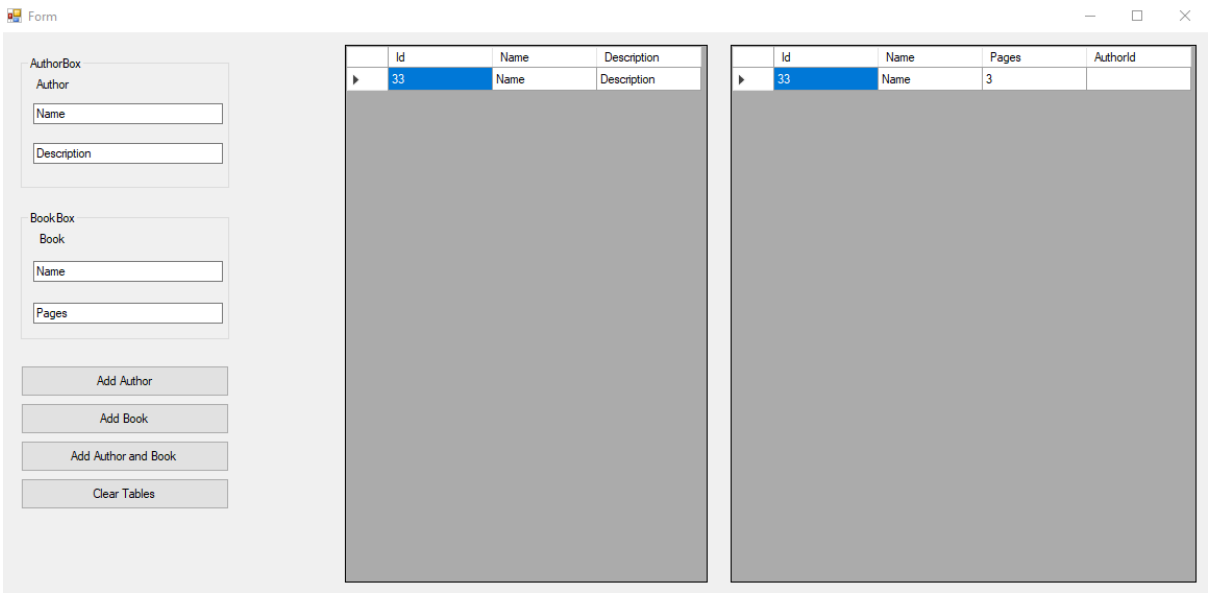
```
private void bothButton_Click(object sender, EventArgs e)
{
    // Without transaction
    using (AjppablabContext db = new())
    {
        try
        {

            string authorName = authorNameText.Text;
            string authorDescription = authorDescriptionText.Text;

            Author addedAuthor = new() { Name = authorName, Description = authorDescription };
            db.Authors.Add(addedAuthor);
            db.SaveChanges();

            db.Authors.Load();
            this.authorBindingSource.DataSource = db.Authors.Local.ToBindingList();

            string bookName = bookNameText.Text;
            int bookPages;
            if (!Int32.TryParse(bookPagesText.Text, out bookPages)) bookPages = 3;

            db.Books.Add(new() { Name = bookName, Pages = bookPages, AuthorId = addedAu-
thor.Id });
            db.SaveChanges();

            db.Books.Load();
            this.bookBindingSource.DataSource = db.Books.Local.ToBindingList();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Occured error: {ex.Message}");
        }
    }
}
```

Snippet 9. EF - Author and Book Button (1)

Po kliknięciu przycisku dodającego autora dane zaczytywane z górnego formularza i zapisywane do tabeli w bazie danych, dane z tej tabeli wyświetlane na DataGridView, który znajduje się po lewej stronie. Po kliknięciu przycisku dodającego książkę dane zaczytywane z formularza znajdującego w kontrolce grupowej Book Box i zapisywane do tabeli w bazie da-nych, dane z tej tabeli wyświetlane na DataGridView, który znajduje się po prawej stronie. Przykładowe działanie dodania autora jest pokazane na *Rysunek 8* i dodanie nowej książki pokazane na *Rysunek 9*



Rysunek 8. EF - Dodanie autora



Rysunek 9. EF - Dodanie książki

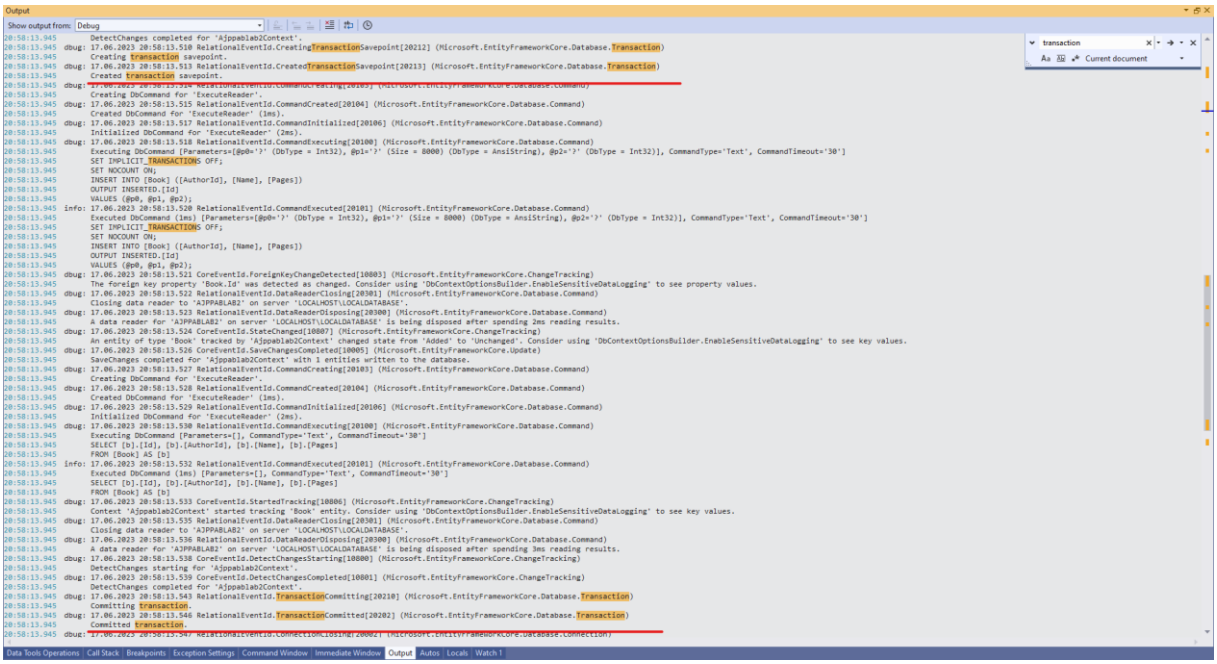
Następnie kod, który realizuje dodawania dwóch wierszy na raz został zmodyfikowany, żeby operacje dodawania działały w obrębie jednej transakcji. Zmodyfikowany kod jest pokazany na *Snippet 10*

```
private void bothButton_Click(object sender, EventArgs e)
{
    // With transaction
    using (AjppablabContext db = new())
    {
        using (IDbContextTransaction transaction = db.Database.BeginTransaction())
        {
            try
            {
                string authorName = authorNameText.Text;
                string authorDescription = authorDescriptionText.Text;
```

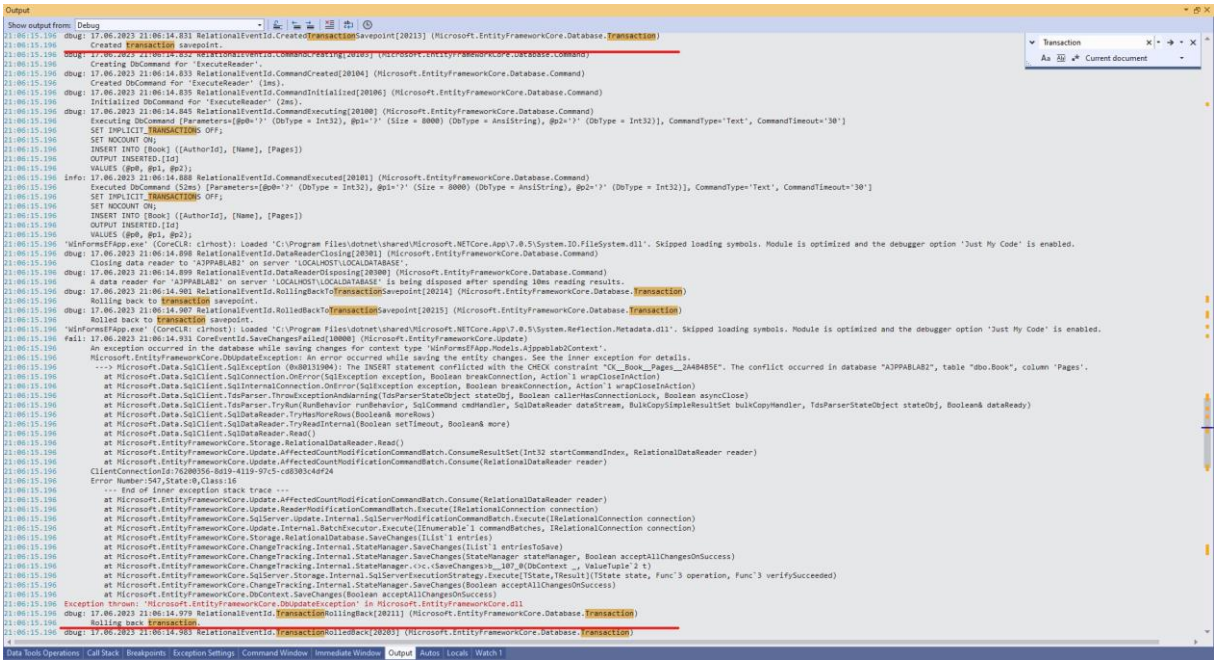
```
                Author addedAuthor = new() { Name = authorName, Description = authorDescrip-
tion };
                db.Authors.Add(addedAuthor);
                db.SaveChanges();

                db.Authors.Load();
                this.authorBindingSource.DataSource = db.Authors.Local.ToBindingList();

                string bookName = bookNameText.Text;
                int bookPages;
                if (!Int32.TryParse(bookPagesText.Text, out bookPages)) bookPages = 3;

                db.Books.Add(new() { Name = bookName, Pages = bookPages, AuthorId = addedAu-
thor.Id });
                db.SaveChanges();

                db.Books.Load();
                this.bookBindingSource.DataSource = db.Books.Local.ToBindingList();
                transaction.Commit();
            }
            catch (Exception ex)
            {
                transaction.Rollback();
                MessageBox.Show($"Occured error: {ex.Message}");

                db.ChangeTracker.Clear();

                db.Authors.Load();
                this.authorDataGridView.DataSource = db.Authors.Local.ToBindingList();

                db.Books.Load();
                this.bookDataGridView.DataSource = db.Books.Local.ToBindingList();
            }
        }
    }
}
```

Snippet 10. EF - Author and Book Button (2)

Efekt działania tych operacji w obrębie jednej transakcji można dobrze prześledzić w logach EF, na *Rysunek 10* jest pokazane jak jest tworzony tak zwany savepoint transakcji, wykonanie operacji i na końcu jest commit wykonanych operacji. Na *Rysunek 11* jest pokazane jak w trakcie został wyrzucony błąd związany z zapisem i transakcja została wycofana do zapisanego wcześniej savepoint'a



Rysunek 10. EF - Wykonanie transakcji (1)



Rysunek 11. EF - Wykonanie transakcji (2)

8. Wnioski

W trakcie wykonania tego laboratorium udało się teoretycznie i praktycznie wprowadzić się w taki temat jak przetwarzanie transakcyjne w ADO.NET i Entity Framework i przetestować różne sytuacje wynikające w przetwarzaniach transakcyjnych.

Podsumowując, transakcje jest to dobre narzędzie, kiedy musimy zapisać powiązane dane do bazy danych i chcemy mieć gwarancje, że te zależne dane zostaną zapisane razem.

9. Bibliografia

1. Źródła pomocnicze
    a. Dokumentacja Microsoft [https://learn.microsoft.com/], dostęp: 15.03.2023
    b. How to Use Constraints in SQL Server Like a Pro [https://www.c-sharpcorner.com/UploadFile/f0b2ed/constraints-in-sql-server/], dostęp 15.03.2023
    c. Constraints in SQL Server: SQL NOT NULL, UNIQUE and SQL PRIMARY KEY [https://www.sqlshack.com/commonly-used-sql-server-constraints-not-null-unique-primary-key/], dostęp: 27.03.2023
    d. Kolumny w DataGrid [https://wpf-tutorial.com/pl/89/kontrolka-datagrid/kolumny-w-datagrid/], dostęp: 30.03.2023
    e. Entity Frmework Tutorial - Transaction in Entity Framework [https://www.entityframeworktutorial.net/entityframework6/transaction-in-entity-framework.aspx], dostęp: 15.04.2023
2. Napotkane problem i ich rozwiązania
    a. How to trigger a button click in my code? [https://stackoverflow.com/questions/16792160/how-to-trigger-a-button-click-in-my-code], dostęp: 15.03.2023
    b. Execute Insert command and return inserted Id in Sql [https://stackoverflow.com/questions/18373461/execute-insert-command-and-return-inserted-id-in-sql], dostęp: 15.03.2023
    c. How to find an identity of the last inserted row in Entity Framework? [https://stackoverflow.com/questions/4068084/how-to-find-an-identity-of-the-last-inserted-row-in-entity-framework], dostęp 15.03.2023
    d. How to get an id of a saved entity in Entity Framework? [https://www.entityframeworktutorial.net/faq/how-to-get-id-of-saved-entity-in-entity-framework.aspx], dostęp: 15.03.2023

10. Spis ilustracji

11. Spis snippetów