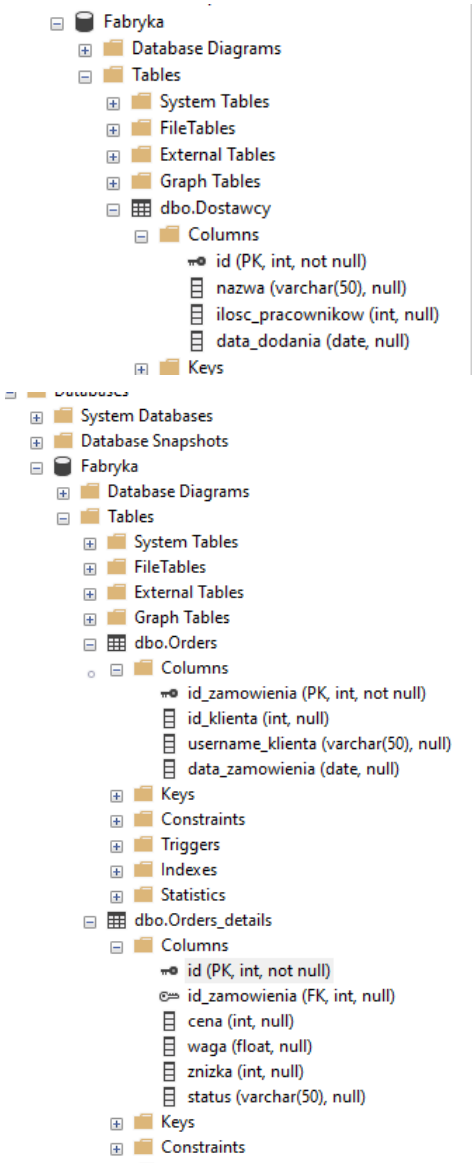
Zespół Szkół Elektrycznych w Gorzowie Wlkp.		
Informatyka		
Nr i temat ćwiczenia Praktyczne wprowadzanie do Przetwarzania transakcyjnego. Wszystkie zadania wykonywane będą z wykorzystaniem dostawcy danych ADO.NET – SqlClient (**System.Data.SqlClient**). Na tym etapie nie powinno się wykorzystywać do tego celu żadnego ORM-a.	Nr w d 22,23	Klasa 3PT5
	Miesiąc Czerwiec	Data oddania sprawozdania 14.06.2022
Wykonał ćwiczenie: Karol Tokarek, Jakub Tokarek	Ocena pkt.	Podpis

1.Cel ćwiczenia.

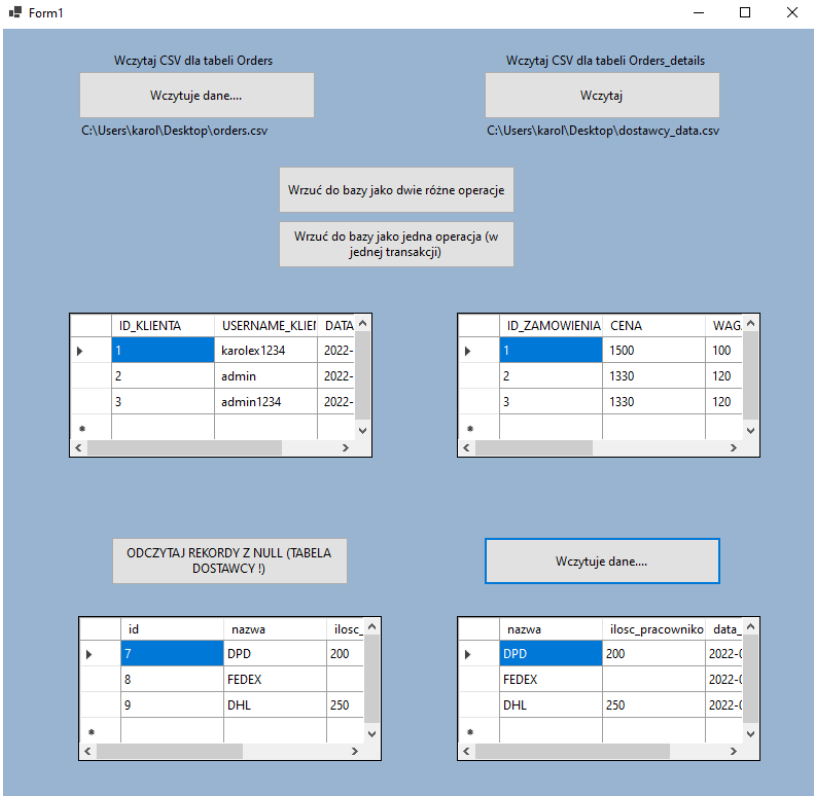
Zapoznanie się z podstawowymi zagadnieniami dotyczącymi łączenia się z bazą danych (Microsoft SQL Server) z poziomu aplikacji napisanej w C# (WinForms – Net.Framework, WPF, WinFormsc- .Net Core lub też winforms .Net 6), zapoznanie się z przetwarzanie transakcyjnym, pisanie zapytań w SQL Server.

2. Opis / screeny:



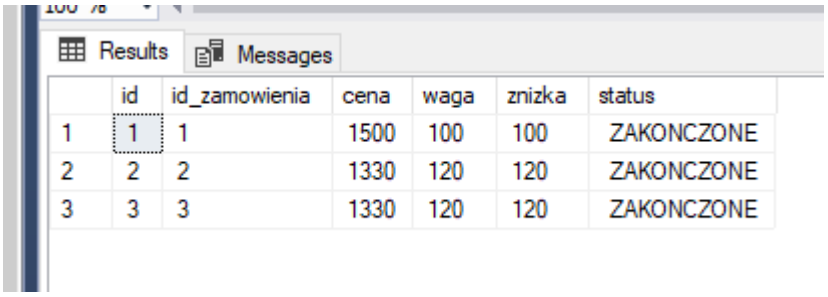
Wszystkie tabele, które zostały stworzone na potrzeby ćwiczenia ^.

Bieżący wygląd aplikacji.

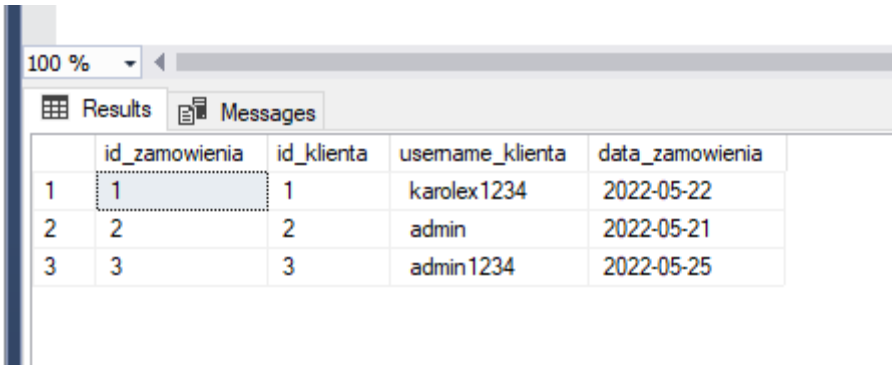


Efekt działania.

(Wszystkie pliki, które zostały użyte na potrzeby ćwiczenia dołączamy w archiwum.)


Results Messages

	id	id_zamowienia	cena	waga	znizka	status
1	1	1	1500	100	100	ZAKONCZONE
2	2	2	1330	120	120	ZAKONCZONE
3	3	3	1330	120	120	ZAKONCZONE


100 %

Results Messages

	id_zamowienia	id_klienta	username_klienta	data_zamowienia
1	1	1	karolex1234	2022-05-22
2	2	2	admin	2022-05-21
3	3	3	admin1234	2022-05-25

Test przy poprawnych plikach CSV(typy danych, ich długość itp. odpowiadają tym w bazie):

- Program poprawnie wczytuje dane do bazy jako dwie różne operacje (transakcje) jak i jako jedno operacja (transakcja)

Test przy niepoprawnych danych (za długi ciąg znaków w jednym z pól w .CSV):

- Po kliknięciu „wrzuć jako dwie różne operacje” program poprawnie **wczytuje dane do bazy danych tylko z pierwszej (transakcji) (**czyli do jednej tabeli) oraz wyrzuca błąd dla drugiej tabeli (operacji):

```
ERROR !System.Data.SqlClient.SqlException (0x80131904): String or
binary data would be truncated in table 'Fabryka.dbo.Orders_details',
column 'status'. Truncated value: '
ZAKONCZONEDLUGIEZAKONCZONEDLUGIEZAKONCZONEDLUGIEZ'.
The statement has been terminated.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException
exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException
exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at
```

- Po kliknięciu „wrzuć jako jedna operacja (transakcja)” program **nie wczytuje żadnych danych do bazy danych (wnioski*)** oraz wyrzuca błąd:

```
ERROR !System.Data.SqlClient.SqlException (0x80131904): String or
binary data would be truncated in table 'Fabryka.dbo.Orders_details',
column 'status'. Truncated value: '
ZAKONCZONEDLUGIEZAKONCZONEDLUGIEZAKONCZONEDLUGIEZ'.
The statement has been terminated.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException
exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException
exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserS
tateObject stateObj, Boolean callerHasConnectionLock, Boolean
```

***Wnioski !:**

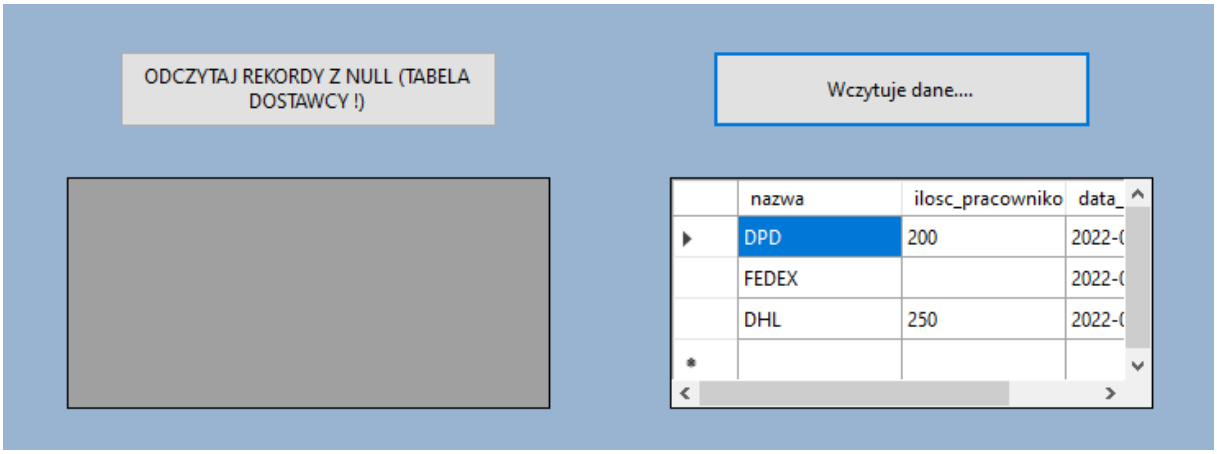
Jak mówi Wikipedia:

„*Transakcja – zbiór operacji na bazie danych, które stanowią w istocie pewną całość i jako takie powinny być wykonane wszystkie lub żadna z nich. Warunki jakie powinny spełniać transakcje bardziej szczegółowo opisują zasady ACID. Przykładem transakcji może być transakcja bankowa jaką jest przelew.”*

Podsumowując, rezultaty naszych operacji są jak najbardziej poprawne – w przypadku braku działania jednego zapytania cała transakcja jest „anulowana” i nie zostaje wykonane NIC. Podział na dwie różne transakcje spowodował że mogliśmy „ominąć” błąd i dane do jednej tabeli zostały przesłane.

(Z tej części aplikacji kod został umieszczony w archiwum ze względu na swoją objętość)

Program zapisujący i odczytujący rekordy które zawierające wartości liczbowe (integer) z dopuszczeniem null:



Po kliknięciu „zapisz dane….” Program zapisuje dane do bazy danych z wybranego pliku CSV do tabeli „Dostawcy” z dopuszczeniem „null”.

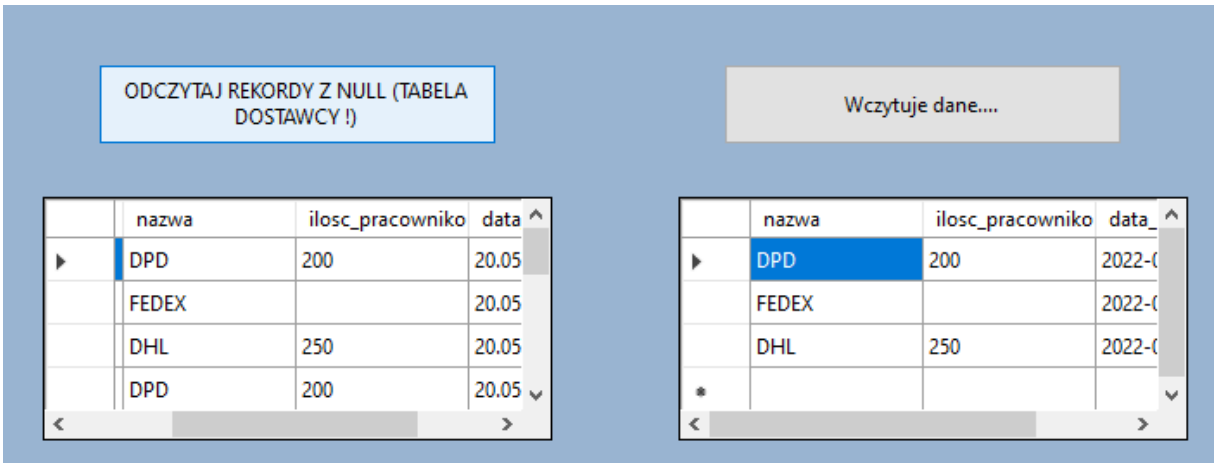
	id	nazwa	ilosc_pracownikow	data_dodania
1	7	DPD	200	2022-05-20
2	8	FEDEX	NULL	2022-05-20
3	9	DHL	250	2022-05-20

Jak widzimy program działa poprawnie, a tutaj mamy kod odpowiedzialny za to:

```
private void button4_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    dialog.ShowDialog();
    int ImportedRecord = 0, inValidItem = 0;
    string SourceURl = "";
    label4.Text = dialog.FileName;


    DataTable dt = new DataTable();
    try
    {
        string[] lines = System.IO.File.ReadAllLines(dialog.FileName);
        button4.Text = "Wczytuje dane....";

        if (lines.Length > 0)
        {
            addByWholeDataSqlClient(lines);
            //addByWholeDataSqlClient(lines);

            ///for wszystkie linie ///
            //TWORZENEI NAGLOWKA W TABELCE
            string firstLine = lines[0];
            string[] headerLabels = firstLine.Split(';');
            foreach (string headerWord in headerLabels)
            {
                dt.Columns.Add(new DataColumn(headerWord));
            }
            ///dla WYPELNIENIA DANYCH
            for (int i = 1; i < lines.Length; i++) //lines.Length
            {

                ////////// PO JEDNYM REKORDZIE //////////////////
                // addByOneByRecordbySqlClient(lines[i]);

                //addByOneByRecordbySqlBulkCopy(lines[i]);

                string[] dataWords = lines[i].Split(';');
                DataRow dr = dt.NewRow();
                int columnIndex = 0;
                foreach (string headerWord in headerLabels)
                {
                    dr[headerWord] = dataWords[columnIndex++];
                    // Debug.WriteLine(dataWords[c]);



                }
                dt.Rows.Add(dr);
            }
        }
        if (dt.Rows.Count > 0)
        {
            dataGridView3.DataSource = dt;

        }

        ///ZAPIS DO BAZY ////////////////////////////////////////////////////////////////////////////////
        ///
        //MessageBox.Show("Suma rekordu po rekordzie: " + sumka);
        //MessageBox.Show("Srednia rekordu: " + (float)(sumka / lines.Length));
    }
    catch (Exception)
    {
        MessageBox.Show("error, zla sciezka lub plik / niepoprawny plik");
    }
}
```

```
private void addByWholeDataSqlClient(string[] wholedata)  ///dziala
{

    Stopwatch stopwatch = new Stopwatch();


    string sqlconn = "Server= localhost; Database= Fabryka; Integrated Security = SSPI";
    // sqlconn = ConfigurationManager.ConnectionStrings["SqlCom"].ConnectionString;
    stopwatch.Start();

    try
    {
        con = new SqlConnection(sqlconn);
        // string[] dataWords = one_record_in_csv.Split(';');
        con.Open();

        for (int i = 1; i < wholedata.Length; i++)
        {
            string wierszyk = wholedata[i];
            string[] dataWords = wierszyk.Split(';');

            string sql = "INSERT INTO Dostawcy VALUES(@nazwa, @ilosc_pracownikow, @data_dodan
            comm = new SqlCommand(sql, con);
            comm.Parameters.AddWithValue("@nazwa", dataWords[0]);
            if (dataWords[1] == null || dataWords[1] == "")
            {
                comm.Parameters.AddWithValue("@ilosc_pracownikow", DBNull.Value);
            }
            else
            {
                comm.Parameters.AddWithValue("@ilosc_pracownikow", dataWords[1]);

            }
            comm.Parameters.AddWithValue("@data_dodania", dataWords[2]);
            comm.ExecuteNonQuery();
            Console.WriteLine("Records Inserted Successfully");

            //  comm.Dispose();
        }

    }
    catch (SqlException ex)

    {
        MessageBox.Show("Can not  ! " + ex.ToString());
    }
    finally
    {
        //MessageBox.Show("koniec polaczenia");

        con.Close();

    }
}

}
```

Jak widać mamy zabezpieczenie przed typem null.



Po kliknięciu „odczytaj rekordy…" także poprawnie nam je odczytuje.

```
private void button3_Click(object sender, EventArgs e)
{

    string sql = "SELECT * FROM Dostawcy";

    using (var connection = new SqlConnection("Server= localhost; Database= Fabryka; Integrated Security = SSPI"))
    using (var command = new SqlCommand(sql, connection))
    using (var adapter = new SqlDataAdapter(command))
    {
        connection.Open();
        var myTable = new DataTable();
        adapter.Fill(myTable);
        dataGridView4.DataSource = myTable;
    }

    SqlConnection conn2 = new SqlConnection("Server= localhost; Database= Fabryka; Integrated Security = SSPI");
    using (conn2)
    {
        SqlCommand command = new SqlCommand(
          "SELECT nazwa, ilosc_pracownikow, data_dodania FROM Dostawcy",
          conn2);
        conn2.Open();

        SqlDataReader reader = command.ExecuteReader();

        if (reader.HasRows)
        {
            while (reader.Read())
            {
                string var1 = (string)reader["nazwa"];
                int? var2 = null;
                if (!reader.IsDBNull(reader.GetOrdinal("ilosc_pracownikow")))
                {
                    var2 = (int)reader["ilosc_pracownikow"];
                }
                DateTime var3 = ((DateTime)reader["data_dodania"]);
                Debug.WriteLine(var1 + ", " + var2 + ", " + var3);
            }
        }
        else
        {
            Console.WriteLine("No rows found.");
        }
        reader.Close();
    }

    ///oraz w debugu jak to wyglada:

}
```

Najpierw jest wczytanie do DataGridView ale jak widzimy SqlDataAdapter w połączenie z DataGridView nie potrzebuje zabezpieczeń przed nullable - pozostaje puste pole. (czyli null) Następnie jeśli chcemy odczytać dane inaczej (w moim przypadkiem do Debuga) - mamy zabezpieczenie przed typem nullable

3. Wnioski (ogólne):

Znamy zagadnienia dotyczące łączenia się z bazą danych (Microsoft SQL Server) z poziomu aplikacji napisanej w C# (WinForms). Operujemy przy wykorzystaniu dostawcy danych ADO.NET – SqlClient (**System.Data.SqlClient**). Umiemy odczytywać i zapisywać rekordy z dopuszczeniem null. Wiemy jak działają transakcje (w SQL).

Potrafimy też odczytywać plik CSV i wczytywać jego zawartość do bazy danych jak i do programu.

*Dodatkowo: potrafimy posługiwać się i zarządzać bazami, tworzyć je, dodawać tabele i relacje, wykonywać zapytania ITP. w Microsoft SQL Management Studio.