Order Manager Application / Aplikacja do zarządzania zamówieniami

Autorzy : Krzysztof Szymczyk i Przemysław Jasiaczyk

1. Wytyczne dotyczące obsługi i instalacji

Order Manager to aplikacja napisana w C# z użyciem Windows Forms, która służy do zarządzania zamówieniami oraz płatnościami w firmie. Aplikacja korzysta z pięciu tabel w bazie danych zarządzanej przy użyciu Microsoft SQL Server: Orders, Payments, OrderDetails, Customers i Products. Aplikacja umożliwia zarządzanie danymi w czterech głównych widokach.

W widoku Klienci użytkownicy mogą dodawać, edytować i usuwać informacje o klientach, co pozwala na utrzymanie aktualnych danych kontaktowych i adresowych.

Widok Produkty pozwala na zarządzanie asortymentem poprzez dodawanie nowych produktów, edytowanie informacji o istniejących produktach oraz usuwanie tych, które są już nieaktualne.

Widok Zamówienia umożliwia tworzenie nowych zamówień, przeglądanie szczegółów, dodawanie pozycji do zamówień, usuwanie zamówień oraz ich wysyłanie.

W widoku Płatności można tworzyć nowe płatności, wycofywać je w razie potrzeby oraz opłacać zrealizowane zamówienia, co ułatwia zarządzanie finansami i monitorowanie stanu płatności.

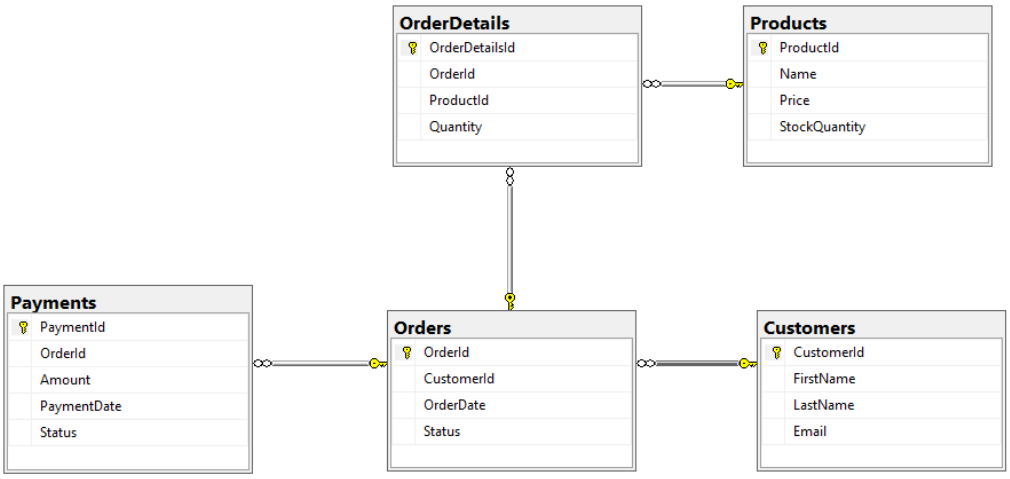
Wymagania:

1. VisualStudio 2022
2. .NET SDK 8.0
3. SQL Server 2019
4. SQL Server Management Studio (SSMS)

Instalacja:

1. Wczytaj skrypt scriptCreatingDB.sql aplikacji z folderu projektu OrderManagerApp\DataAccess\Data do SSMS.
   Alternatywnie można ręcznie utworzyć nową bazę danych i wczytać skrypt zawierający wyłącznie generowanie tabel: DBSchemaScript.sql
2. Skonfiguruj połączenie z bazą danych w pliku konfiguracyjnym App.config w projekcie OrderManagerApp (OrderManagerApp\OrderManagerApp\App.config). Należy w podmienić: data source = „nazwaServera” i database = „nazwaBazy’.
3. Skompiluj projekt i uruchom aplikację.

2. Diagram bazy



Opis relacji w bazie danych między tabelami

Tabela 1	Tabela 2	Typ relacji	Identyfikująca/Nieidentyfikująca	Obowiązkowa/Opcjonalna
Customers	Orders	1 do wielu	Nieidentyfikująca	Obowiązkowa po stronie Orders, opcjonalna po stronie Customers
Orders	OrderDetails	1 do wielu	Identyfikująca	Obowiązkowa po stronie OrderDetails, opcjonalna po stronie Orders
Products	OrderDetails	1 do wielu	Nieidentyfikująca	Obowiązkowa po stronie OrderDetails, opcjonalna po stronie Products
Orders	Payments	1 do 1	Identyfikująca	Obowiązkowa po stronie Payments, opcjonalna po stronie Orders

3. Klasy i enum

Klasa Product		
Nazwa	Typ	Wymagalność
ProductId	int	Not Null
Name	string	Not Null
Price	decimal	Not Null
StockQuantity	int	Not Null

Klasa Payment		
Nazwa	Typ	Wymagalność
PaymentId	int	Not Null
OrderId	int	Not Null
Amount	decimal	Not Null
PaymentDate	DateTime	Not Null
Status	PaymentStatus	
Paid	0	
Waiting	1	

Klasa Order		
Nazwa	Typ	Wymagalność
OrderId	int	Not Null
CustomerId	int	Not Null
OrderDate	DateTime	Not Null
Status	OrderStatus	Not Null
Processed	0	
Approved	1	
Delivered	2	

Klasa OrderDetails		
Nazwa	Typ	Wymagalność
OrderDetailsId	int	Not Null
OrderId	int	Not Null
ProductId	int	Not Null
Quantity	int	Not Null

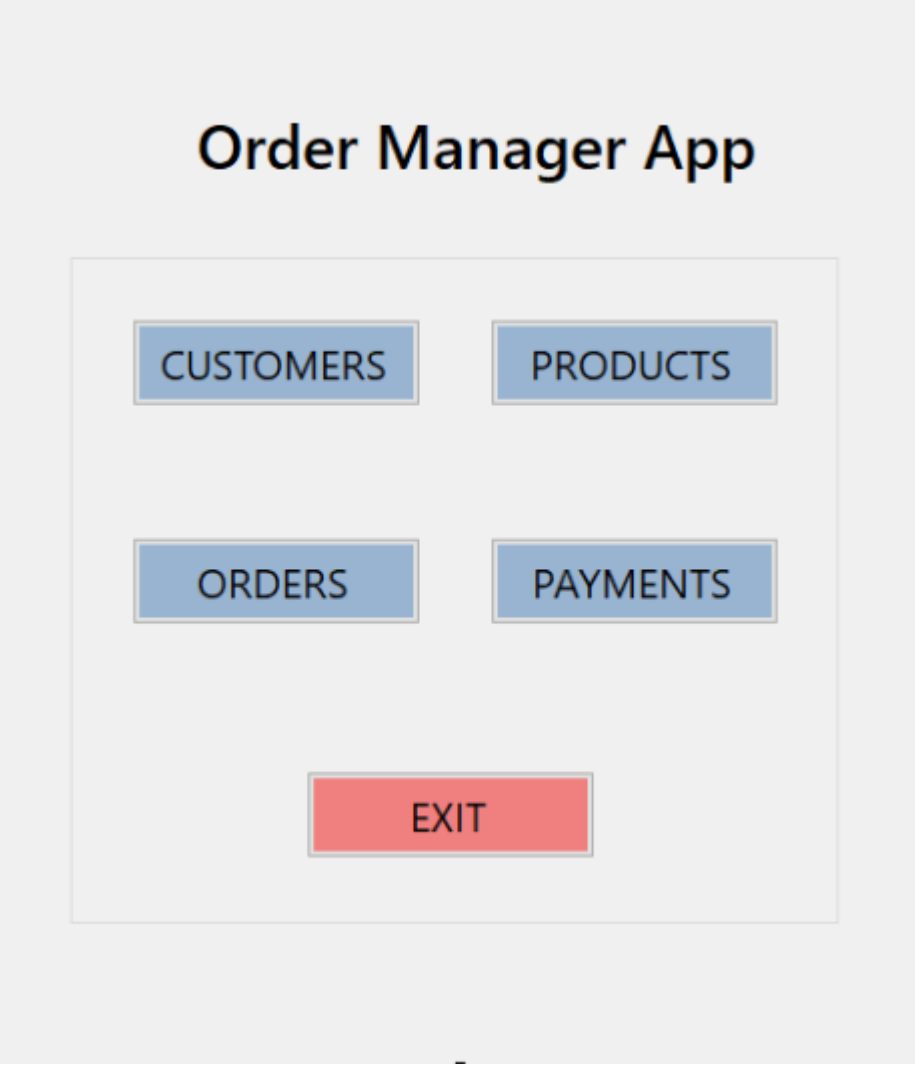
Klasa Customer		
Nazwa	Typ	Wymagalność
CustomerId	int	Not Null
FirstName	string	Not Null
LastName	string	Not Null
Email	string	Not Null

Enum OrderStatus : reprezentuje statusy zamówień dostępne w aplikacji : processed oznacza przyjęty, approved zatwierdzony a delivered dostarczony

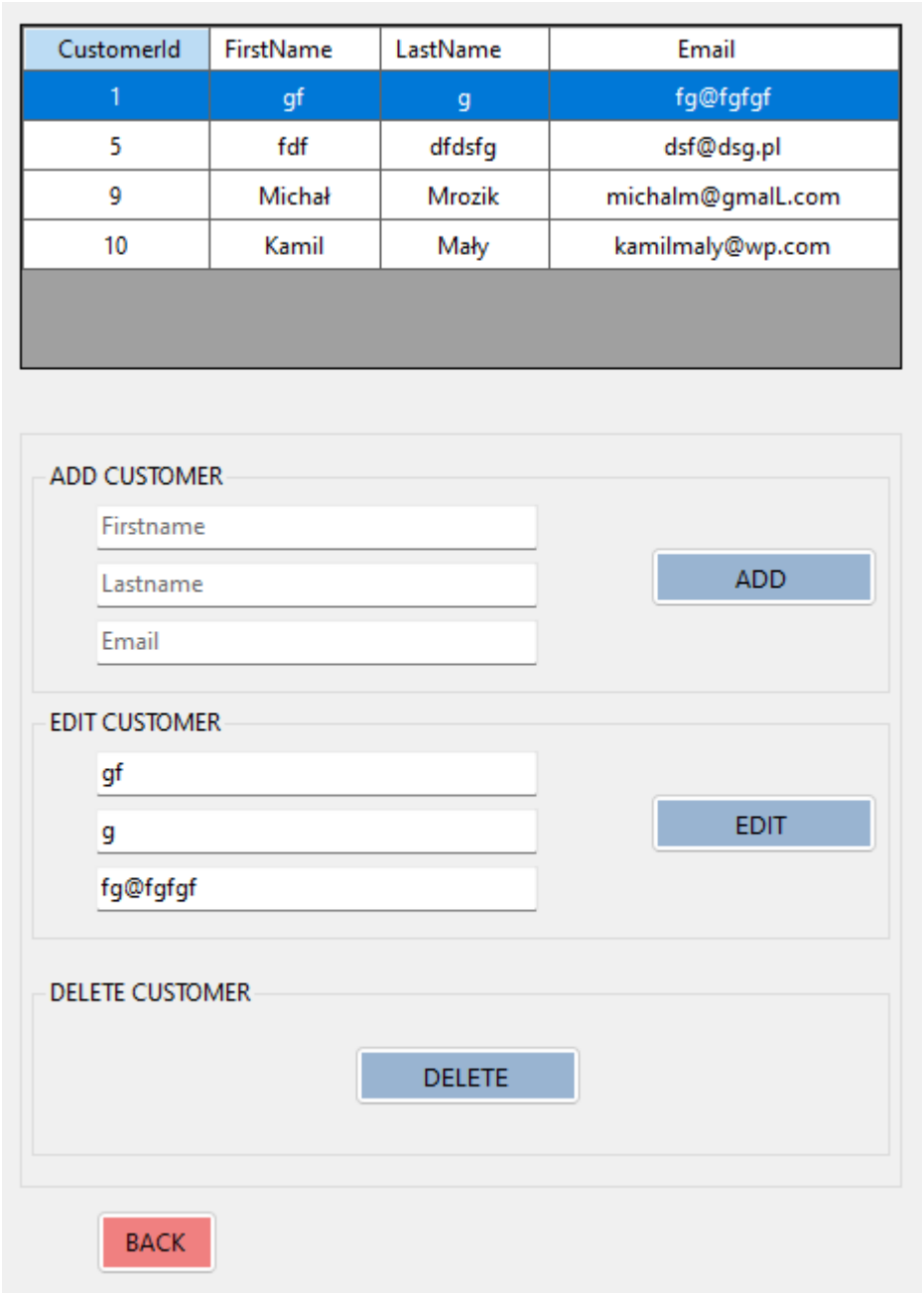
Enum PaymentStatus : zamówienie może być opłacone – Paid lub oczekujące – Waiting

4. Przewodnik po aplikacji

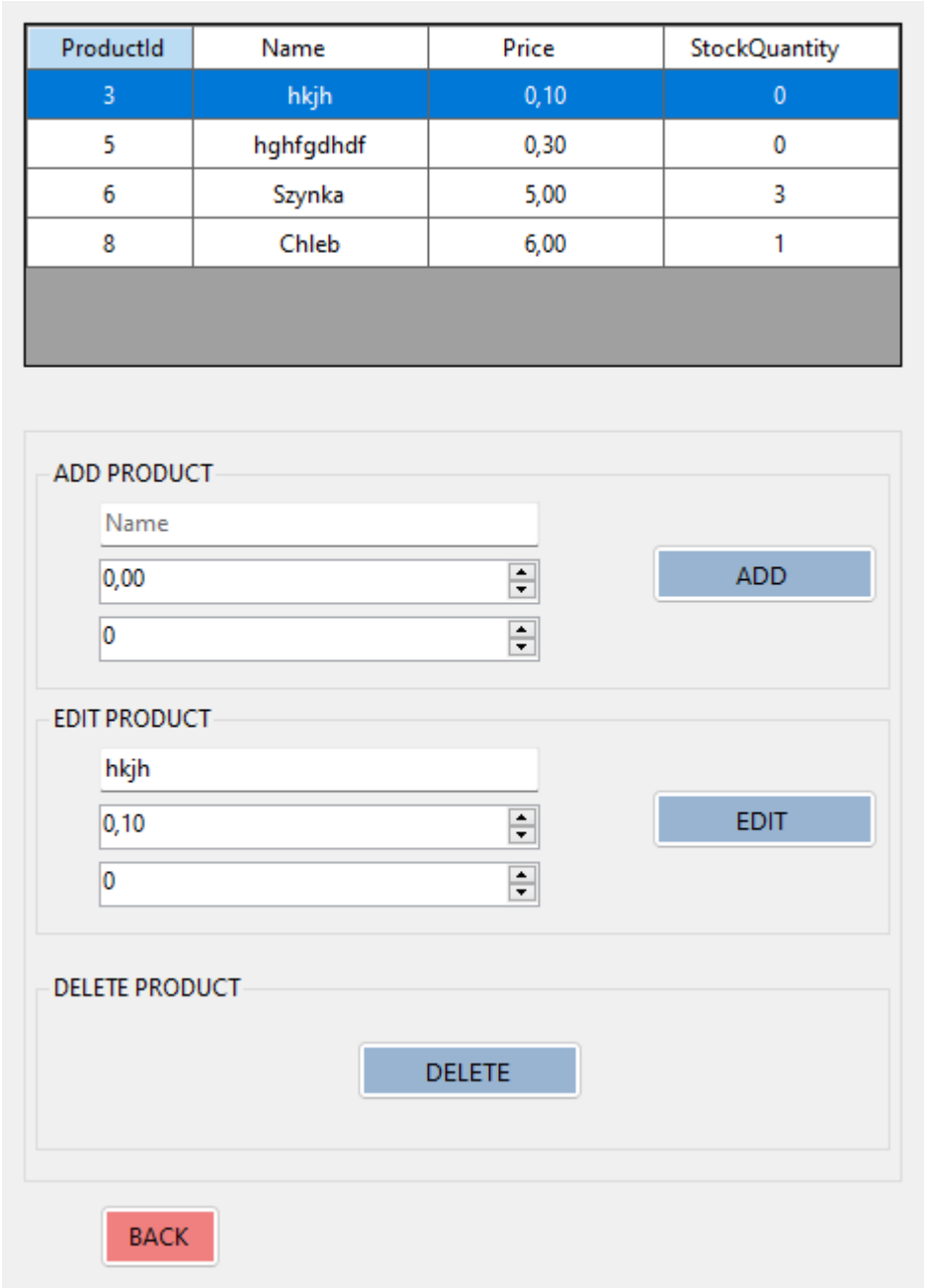
1. **MainForm** to główny widok aplikacji, który służy jako centralny punkt nawigacyjny. Użytkownik może wybierać spośród czterech głównych funkcji aplikacji: zarządzania klientami, produktami, zamówieniami i płatnościami. Każdy z tych obszarów jest reprezentowany przez odpowiednio oznaczone przyciski.



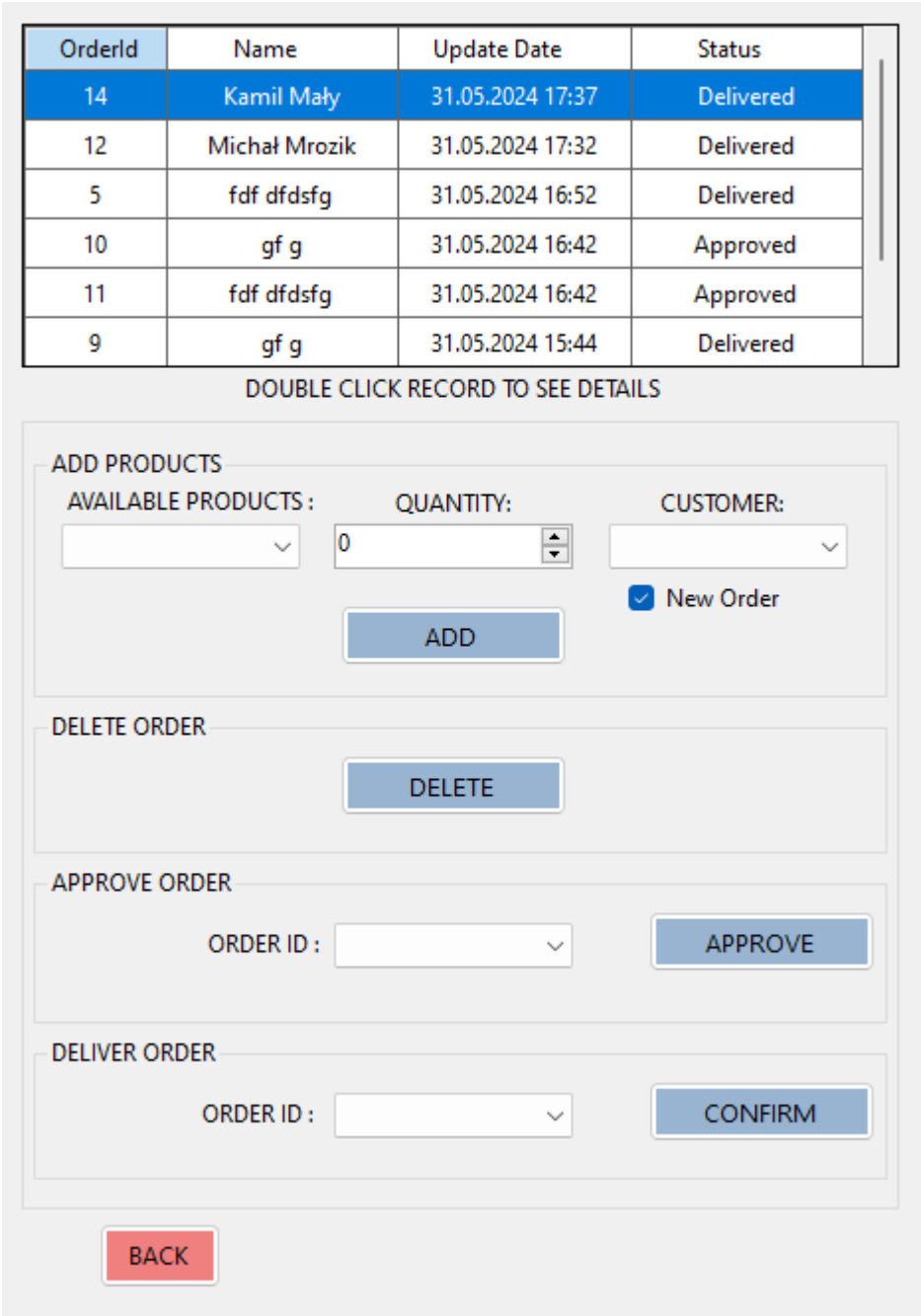
2. **CustomersForm** to widok dedykowany zarządzaniu danymi klientów. Umożliwia użytkownikowi dodawanie nowych klientów poprzez wprowadzenie imienia, nazwiska oraz adresu e-mail, a następnie kliknięcie przycisku "ADD". Widok ten pozwala także na edytowanie danych istniejących klientów, poprzez wprowadzenie zaktualizowanych informacji w odpowiednie pola i kliknięcie przycisku "EDIT". Dodatkowo, widok zawiera funkcję usuwania klientów za pomocą przycisku "DELETE". Na górze formularza znajduje się miejsce na wyświetlanie listy klientów. Przycisk "BACK" umożliwia powrót do głównego widoku aplikacji.



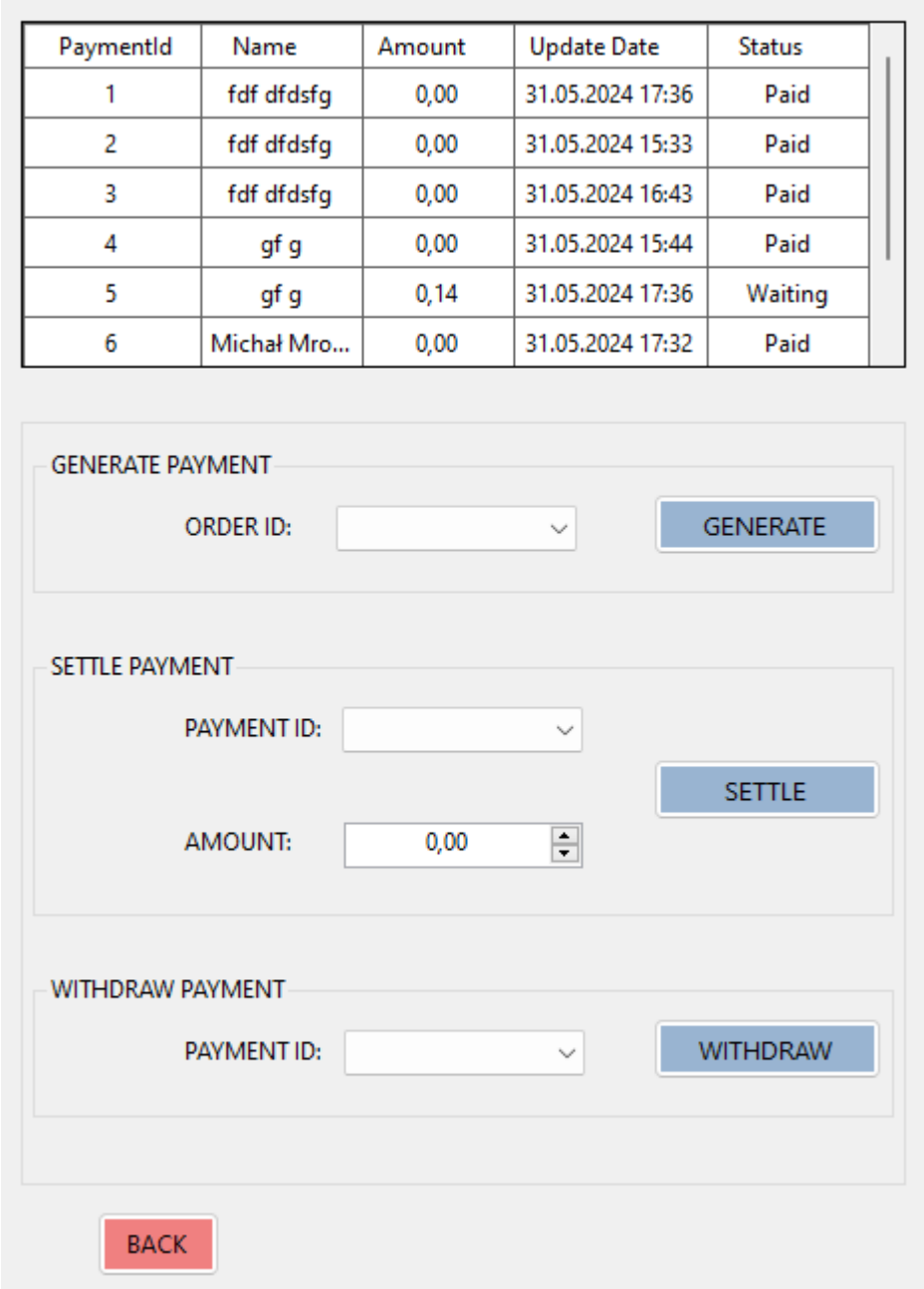
3. **ProductsForm** to widok w aplikacji Order Manager przeznaczony do zarządzania produktami. Umożliwia użytkownikowi dodawanie nowych produktów poprzez wprowadzenie odpowiednich informacji o produkcie, a następnie kliknięcie przycisku "ADD". Widok ten pozwala także na edytowanie danych istniejących produktów poprzez wprowadzenie zaktualizowanych informacji i kliknięcie przycisku "EDIT". Dodatkowo, użytkownik może usuwać produkty z bazy danych za pomocą przycisku "DELETE".



4. **OrderForm** to widok, który umożliwia zarządzanie zamówieniami. Użytkownik może dodawać produkty do zamówienia, określając ich ilość. Formularz oferuje także opcję usunięcia całego zamówienia za pomocą odpowiedniego przycisku "DELETE". Po zakończeniu kompletowania zamówienia, użytkownik może zatwierdzić zamówienie klikając przycisk "APPROVE". Dodatkowo, widok zawiera funkcję zamówienia dostawy dla całkowicie opłaconych już zamówień. Dzięki podwójnemu kliknięciu wybranego wiersza tabeli dataGridView użytkownik będzie miał możliwość zobaczenia szczegółów danego zamówienia.

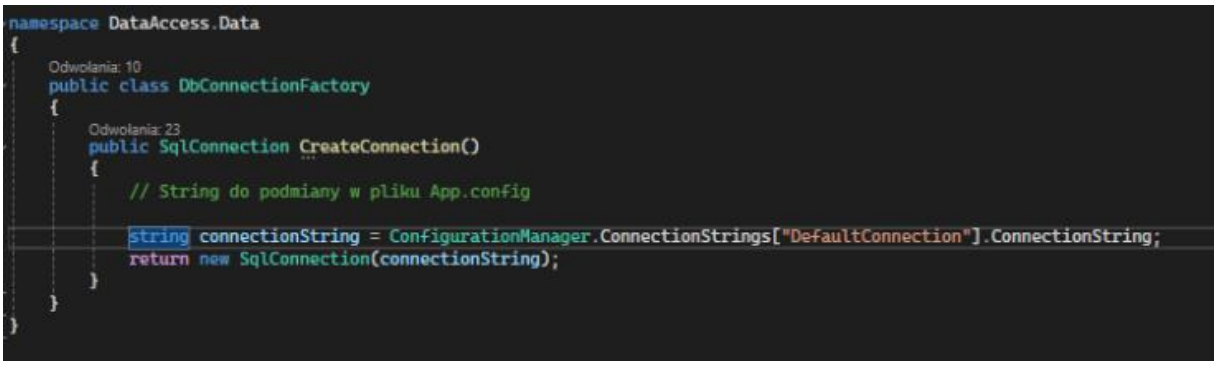


5. **PaymentsForm** to widok w aplikacji przeznaczony do zarządzania płatnościami. Umożliwia użytkownikowi generowanie nowych płatności dla zamówień oraz opłacanie istniejących płatności poprzez wprowadzenie odpowiedniej kwoty w polu NumericUpDown i kliknięcie przycisku "PAY". Dodatkowo, użytkownik może wycofać płatność, korzystając z przycisku "WITHDRAW".



5. Opis wykorzystanych komponentów bazodanowych

DbConnectionFactory: Jest to klasa pomocnicza, która zapewnia mechanizm tworzenia połączenia z bazą danych. Jest używana w całej wszystkich repozytoriach w projekcie DataAccess do utworzenia obiektu połączenia, który będzie używany do komunikacji z bazą danych.



```
namespace DataAccess.Data
{
    Odwołania: 10
    public class DbConnectionFactory
    {
        Odwołania: 23
        public SqlConnection CreateConnection()
        {
            // String do podmiany w pliku App.config

            string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;
            return new SqlConnection(connectionString);
        }
    }
}
```

SqlConnection: Ten komponent reprezentuje połączenie z bazą danych SQL Server. Jest wykorzystywany przez DbConnectionFactory do utworzenia połączenia z bazą danych przed wykonaniem każdego zapytania w projekcie.

SqlCommand: Jest to obiekt, który reprezentuje polecenie SQL, które ma zostać wykonane w bazie danych i znajduje się w każdej metodzie związanej z manipulacją danymi w bazie.

SqlDataAdapter: Ten komponent jest wykorzystywany do przesyłania danych między bazą danych a DataSet. W metodzie GetAllCustomers używasz SqlDataAdapter, aby wykonać zapytanie SQL na połączeniu i wypełnić dane klientów z bazy danych do DataSet.

SqlDataAdapter: Ten komponent jest wykorzystywany do przesyłania danych między bazą danych a DataSet. Wykorzystywany we wszystkich metodach w repozytoriach. Przykładowo w metodzie GetAllCustomers w CustomerRepository SqlDataAdapter użyty jest, aby wykonać zapytanie SQL na połączeniu i wypełnić dane klientów z bazy danych do DataSet.

```
using (var command = new SqlCommand(sql, connection))
{
    SqlDataAdapter adapter = new SqlDataAdapter(command);

    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet, "Customers");
```

DataSet: Reprezentuje pamięć podręczną danych w formie zbioru tablic i relacji. Używany w zdecydowanej większości funkcji służących do zarządzania danymi w aplikacji. Przykładowo w metodzie GetAllCustomers w CustomerRepository użyty, aby przechować dane klientów pobrane z bazy danych przed ich przetworzeniem (screen powyżej).

DataRow: Reprezentuje wiersz danych w tabeli DataSet. Stosowany w zdecydowanej większości metod aplikacji wykonujących polecenia bazodanowe : SELECT, UPDATE, INSERT.

```
foreach (DataRow row in dataSet.Tables[0].Rows)
{
    int id = (int)row["ProductId"];
    string Name = (string)row["Name"];
    decimal Price = (decimal)row["Price"];
    int StockQuantity = (int)row["StockQuantity"];
```

DataTable: Reprezentuje kolekcję wierszy i kolumn, w której przechowywane są dane z bazy danych. Wykorzystywane w metodzie GetMissingOrderIdsInPayments w OrderRepository – tworzone są dwie tabele: dla wyników zapytania o id zamówień ze statusem „Sent” z tabeli Orders i dla wyników zapytań o id zamówień z tabeli Payments .

```
DataTable ordersTable = new DataTable();

using (var ordersCommand = new SqlCommand(ordersSql, connection))
{
    SqlDataAdapter ordersAdapter = new SqlDataAdapter(ordersCommand);
    ordersAdapter.Fill(ordersTable);
}

string paymentsSql = "SELECT OrderId FROM Payments";
DataTable paymentsTable = new DataTable();
using (var paymentsCommand = new SqlCommand(paymentsSql, connection))
{
    SqlDataAdapter paymentsAdapter = new SqlDataAdapter(paymentsCommand);
    paymentsAdapter.Fill(paymentsTable);
}
```

DataGridView : kontrolka w WindowsForms która pozwala na wyświetlanie i edytowanie tabel danych w formie siatki. W metodzie refreshDGV() kontrolka dataGridViewCustomers została wykorzystana do wyświetlania wszystkich klientów z bazy danych przy użyciu metody GetAllCustomers.

```
Odwołania: 4
private void refreshDGV()
{
    try
    {
        dataGridViewCustomers.DataSource = _customerService.GetAllCustomers();

        dataGridViewCustomers.Columns["CustomerId"].AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
        dataGridViewCustomers.Columns["FirstName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
        dataGridViewCustomers.Columns["LastName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

6. Zbiór reguł biznesowych

- Klient może zostać utworzony oraz jego dane mogą zostać zmodyfikowane tylko przy podaniu wszystkich możliwych informacji (imię, nazwisko, email),

- Klient może zostać usunięty tylko wtedy, gdy nie posiada jeszcze żadnych aktywnych zamówień,

- Płatność może zostać wygenerowana tylko dla zamówień posiadających status Approved – Zatwierdzone (są to zamówienia z kompletną listą produktów),

- Płatność za zamówienie może zostać częściowo/całkowicie opłacona tylko wtedy gdy zamówienie jest kompletne (nie posiada statusu Processed – Przetwarzane),

- Jednorazowo płatność nie może przewyższyć kwoty (Amount) potrzebnej do opłacenia zamówienia – nie ma możliwości nadpłaty,

- Anulować płatność można tylko dla tych zamówień, które nie zostały jeszcze dostarczone (nie posiadają statusu Delivered),

- Klient może usuwać tylko te zamówienia które mają status „Processed”,

- Klient nie może dodać do zamówienia większej ilości produktów niż jest w kolumnie StockQuantity w tabeli Products,

- Klient może wysłać tylko te zamówienia, które zostały całkowicie opłacone (odpowiadający rekord w tabeli Payments ma status: Paid),

- Dodanie produktu do zamówienia spowoduje zmniejszenie StockQuantity dla danego produktu w tabeli Products. Usunięcie zamówienia spowoduje przywrócenie stanu StockQuantity sprzed dokonania zamówienia dla każdego produktu.