

Sprawozdanie z Projektu na Programowanie IV

Temat projektu:

Ewidencja

Wykonał:

Szymon Murak

Gr.2a

Spis treści

1.Co należało wykonać	3
2.Tworzenie bazy danych	3
3.Tworzenie okna xaml.....	7
4.BaseViewModel oraz RelayCommand	10
5.Wprowadzanie danych.....	11
6.Wyświetlanie danych	15
7.Przykładowe działania na innych tabelach.....	19
• Dane Adresowe:	19
• Klienci:	20

1.Co należało wykonać

Do wykonania był kod tworzący bazę danych metodą Code First oraz strona xaml która umożliwi modyfikację oraz wyświetlanie danych z utworzonej bazy danych.

2.Tworzenie bazy danych

Pierwszym krokiem było utworzenie Context wraz z modelami tabel.

```
public class EwidencjaContext : DbContext
{
    Odwołania: 12
    public EwidencjaContext()
    {
    }

    Odwołania: 0
    public EwidencjaContext(DbContextOptions<EwidencjaContext> options)
        : base(options)
    {
    }

    Odwołania: 3
    public virtual DbSet<Produkty> Produkty { get; set; }
    Odwołania: 3
    public virtual DbSet<Zamówienia> Zamówienia { get; set; }
    Odwołania: 3
    public virtual DbSet<DaneAdresowe> DaneAdresowe { get; set; }
    Odwołania: 3
    public virtual DbSet<Klienci> Klienci { get; set; }
    Odwołania: 3
    public virtual DbSet<Pracownicy> Pracownicy { get; set; }
    Odwołania: 3
    public virtual DbSet<Faktury> Faktury { get; set; }

    Odwołania: 0
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=COMPHT;Database=Ewidencja;Trusted_Connection=True;");
    }
}
```

Context odpowiada za połączenie z bazą danych oraz mapowanie tabel.

```

Odwołania: 3
public partial class DaneAdresowe
{
    [Key]
    Odwołania: 3
    public int Id_Adresu { get; set; }
    [Required]
    Odwołania: 3
    public string Ulica { get; set; }
    [Required]
    Odwołania: 3
    public string Adres { get; set; }
    [Required]
    Odwołania: 3
    public string Miasto { get; set; }
    [Required]
    Odwołania: 3
    public string Kod_pocztowy { get; set; }
    [Required]
    Odwołania: 3
    public string Kraj { get; set; }
}

```

Model 1 Dane Adresowe

```

public class Faktury
{
    [Key]
    public int Id_Faktury { get; set; }
    [Required]
    public int Id_Klienta { get; set; }
    [Required]
    public int Id_Pracownika { get; set; }
    [Required]
    public int Id_Zamówienia { get; set; }
    [Required]
    Odwołania: 3
    public DateTime Termin_płatności { get; set; }
    [Required]
    Odwołania: 3
    public double Całkowita_kwota_do_zapłaty { get; set; }
    [Required]
    Odwołania: 3
    public DateTime Termin_wpłaty { get; set; }
}

```

Model 2 Faktury

```

public class Klienci
{
    [Key]

    public int Id_Klienta { get; set; }
    [Required]
    Odwołania: 3
    public string Imię_klienta { get; set; }
    [Required]
    Odwołania: 3
    public string Nazwisko_klienta { get; set; }
    [Required]
    Odwołania: 3
    public string Numer_telefonu_klienta { get; set; }
    Odwołania: 3
    public string? PESEL_klienta { get; set; }
    [Required]
    Odwołania: 3
    public string Email_klienta { get; set; }
    [Required]
    Odwołania: 3
    public int Id_Adresu { get; set; }
}

```

Model 3 Klienci

```

public class Pracownicy
{
    [Key]

    public int Id_Pracownika { get; set; }
    [Required]

    public string Imię_pracownika { get; set; }
    [Required]

    public string Nazwisko_pracownika { get; set; }
    [Required]

    public string Numer_telefonu_pracownika { get; set; }
    Odwołania: 3
    public string? PESEL_pracownika { get; set; }
    [Required]
    Odwołania: 3
    public string Email_pracownika { get; set; }
    [Required]
    Odwołania: 3
    public DateTime Data_zatrudnienia { get; set; }
    [Required]
    Odwołania: 3
    public DateTime Data_urodzenia { get; set; }
    [Required]
    Odwołania: 3
    public int Id_Adresu { get; set; }
}

```

Model 4 Pracownicy

```

public partial class Produkty
{
    [Key]
    public int Id_Projektu { get; set; }
    [Required]
    Odwołania: 3
    public string Nazwa_produkty { get; set; }
    [Required]
    Odwołania: 3
    public double Cena_jednostkowa { get; set; }
    [Required]
    Odwołania: 3
    public int Dostępna_ilość { get; set; }
}

```

Model 5 Produkty

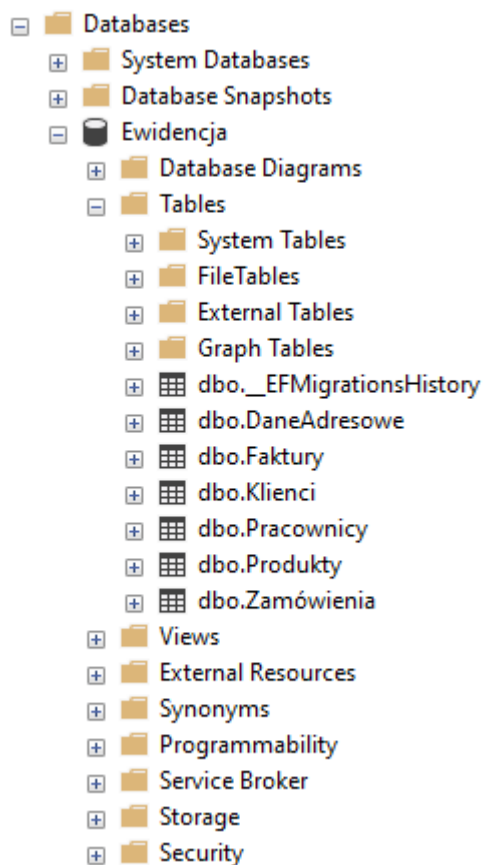
```

public class Zamówienia
{
    [Key]
    Odwołania: 3
    public int Id_Zamówienia { get; set; }
    [Required]
    Odwołania: 3
    public int Id_Projektu { get; set; }
    [Required]
    Odwołania: 3
    public double Cena_jednostkowa { get; set; }
    [Required]
    Odwołania: 3
    public int Ilość { get; set; }
}

```

Model 6 Zamówienia

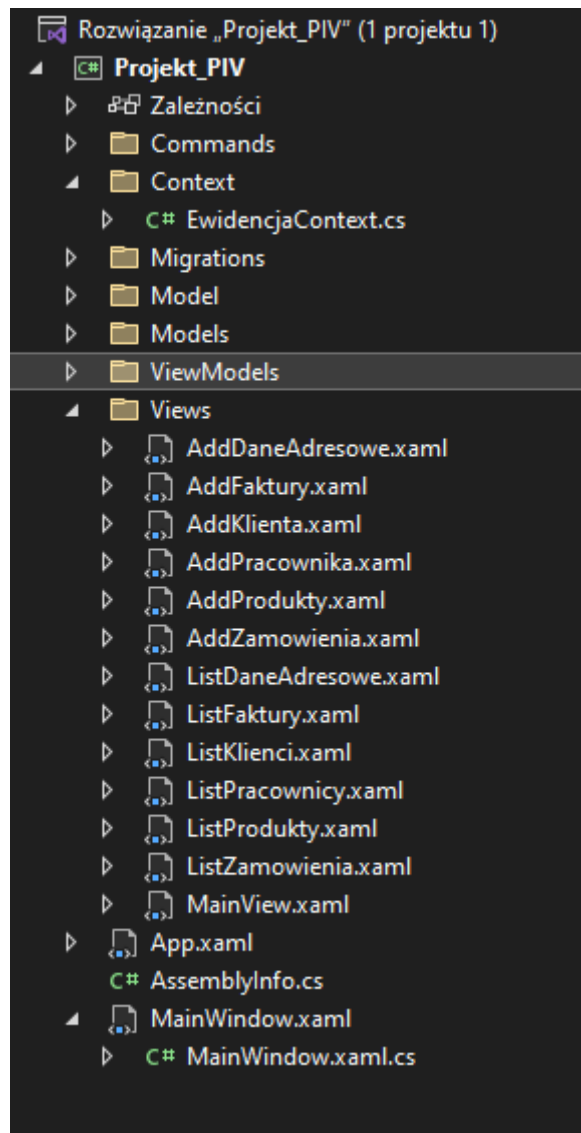
Po wykonaniu powyższych klas należało uruchomić Konsole Menadżera Pakietów i utworzyć migracje komendą Add-Migration „NazwaMigracji” a następnie update-database. Po prawidłowym wykonaniu tych komend baza danych została utworzona(w moim przypadku w Microsoft SQL Management Studio 18) .



7 Utworzona Ewidencja

3. Tworzenie okna xaml

Po stworzeniu bazy danych utworzyłem główne okno xaml oraz jego podstrony opowiadające odpowiadające danym akcją.



8 Główne okno i strony

Główne okno zawiera usytuowane na górze strony menu pozwalające na nawigację pomiędzy stronami oraz kolor tła.


```

<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="*" />
    <RowDefinition Height="auto" />
  </Grid.RowDefinitions>
  <Menu DockPanel.Dock="Top" Background="■LightSkyBlue">
    <MenuItem Header="Wyświetl">
      <MenuItem Header="Fakturę" Click="ListFaktury_Click" />
      <MenuItem Header="Zamówienia" Click="ListZamowienia_Click" />
      <MenuItem Header="Klienta" Click="ListKlienci_Click" />
      <MenuItem Header="Pracownika" Click="ListPracownicy_Click" />
      <MenuItem Header="Produkt" Click="ListProdukty_Click" />
      <MenuItem Header="Dane Adresowe" Click="ListDaneAdresowe_Click" />
    </MenuItem>
    <MenuItem Header="Dodaj">
      <MenuItem Header="Fakturę" Click="AddFaktury_Click" />
      <MenuItem Header="Zamówienia" Click="AddZamowienia_Click" />
      <MenuItem Header="Klienta" Click="AddKlienta_Click" />
      <MenuItem Header="Pracownika" Click="AddPracownika_Click" />
      <MenuItem Header="Produkt" Click="AddProdukty_Click" />
      <MenuItem Header="Dane Adresowe" Click="AddDaneAdresowe_Click" />
    </MenuItem>
  </Menu>
  <Frame x:Name="CurrentForm" Margin="0,20,0,0" NavigationUIVisibility="Hidden" Loaded="CurrentForm_Loaded" />
</Grid>

```

9 Okno główne

Strona główna, która jest domyślnie wczytywana zawiera 3 etykiety.

```

<Grid.RowDefinitions>
  <RowDefinition Height="*" />
  <RowDefinition Height="auto" />
</Grid.RowDefinitions>

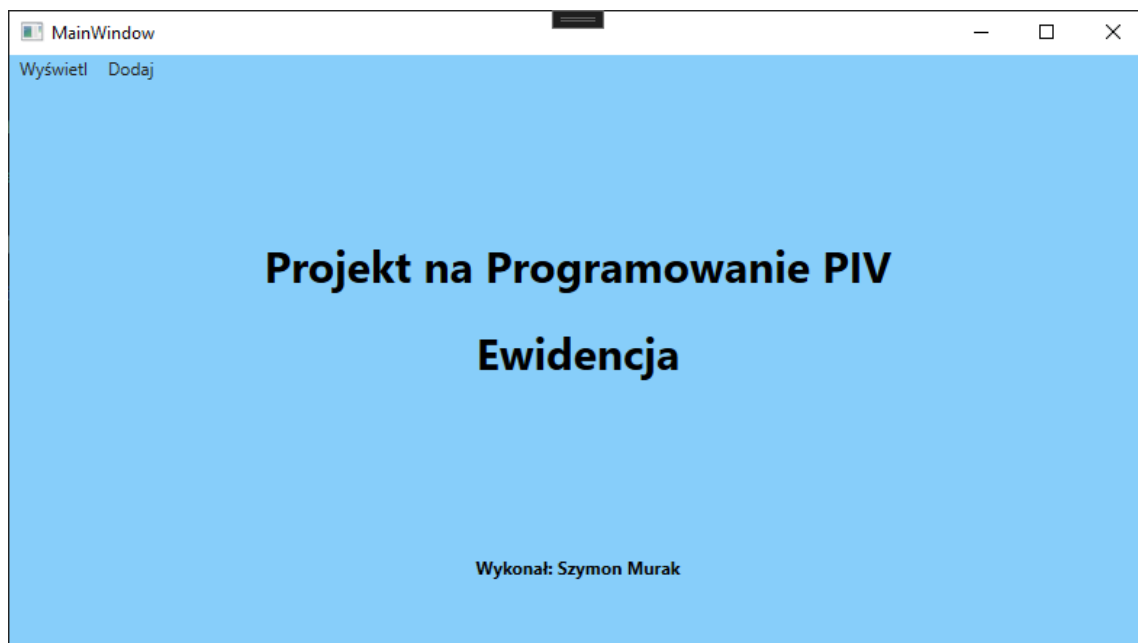
<ContentControl Content="{Binding SelectedViewModel}" />

<Label Content="Projekt na Programowanie PIV" HorizontalAlignment="Center" Margin="0,100,0,0" VerticalAlignment="Top" FontWeight="Bold" FontSize="30" />
<Label Content="Ewidencja" HorizontalAlignment="Center" Margin="0,160,0,0" VerticalAlignment="Top" FontWeight="Bold" FontSize="30" />
<Label Content="Wykonał: Szymon Murak" HorizontalAlignment="Center" Margin="0,320,0,0" VerticalAlignment="Top" FontWeight="Bold" />

```

10 Strona główna

Strona główna prezentuje się tak:



4.BaseViewModel oraz RelayCommand

BaseViewModel jest bazowym modelem w którym zawarta jest klasa OnPropertyChanged z której korzystają inne view modele.

```
Odwołania: 13
public class BaseViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler? PropertyChanged;

    Odwołania: 37
    public void OnPropertyChanged([CallerMemberName] string property = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(property));
    }
}
```

11 BaseViewModel

RelayCommand zawiera następującą instrukcje:

```
Odwołania: 13
internal class RelayCommand : ICommand
{
    private Action<object> execute;
    private Func<object, bool> canExecute;

    public event EventHandler CanExecuteChanged
    {
        add { CommandManager.RequerySuggested += value; }
        remove { CommandManager.RequerySuggested -= value; }
    }

    Odwołania: 12
    public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
    {
        this.execute = execute;
        this.canExecute = canExecute;
    }

    Odwołania: 0
    public bool CanExecute(object parameter)
    {
        return this.canExecute == null || this.canExecute(parameter);
    }

    Odwołania: 0
    public void Execute(object parameter)
    {
        this.execute(parameter);
    }
}
```

12 RelayCommand

5.Wprowadzanie danych

Wprowadzanie danych odbywa się za pomocą:

- strony xaml,
- kody c# tej strony,
- modelu przechowującego dane,
- View Modelu.

Kod pozwalający na wprowadzanie danych do tabeli Produkty:

- strona xaml:

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>

    <Label Grid.Column="0" Grid.Row="1" Content="Nazwa Produktu" HorizontalAlignment="Left" Margin="4" VerticalAlignment="Top"/>
    <Label Grid.Column="0" Grid.Row="2" Content="Cena Jednostkowa" HorizontalAlignment="Left" Margin="4" VerticalAlignment="Top"/>
    <Label Grid.Column="0" Grid.Row="3" Content="Dostępna Ilość" HorizontalAlignment="Left" Margin="4" VerticalAlignment="Top"/>

    <TextBox x:Name="NazwaProduktu" Grid.Column="1" Grid.Row="1" HorizontalAlignment="Left" Margin="4" Text="{Binding Nazwa_produktu, UpdateSourceTrigger=PropertyChanged}" Width="200"/>
    <TextBox x:Name="CJ" Grid.Column="1" Grid.Row="2" HorizontalAlignment="Left" Margin="4" Text="{Binding Cena_jednostkowa, UpdateSourceTrigger=PropertyChanged}" Width="200"/>
    <TextBox x:Name="DostepnaIlosc" Grid.Column="1" Grid.Row="3" HorizontalAlignment="Left" Margin="4" Text="{Binding Dostepna_ilość , UpdateSourceTrigger=PropertyChanged}" Width="200"/>
    <Button Grid.Column="1" Grid.Row="4" Margin="4" Content="Dodaj produkt" Command="{Binding AddProduktyClick}" HorizontalAlignment="Left" VerticalAlignment="Top"/>
</Grid>
```

13 AddProdukty.xaml

- Kod c#:

```

Odwołania: 3
public partial class AddProdukty : Page
{
    private readonly AddProduktyViewModel _viewModel;
    Odwołania: 0
    private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)
    {
        Regex regex = new Regex("[^0-9]+");
        e.Handled = regex.IsMatch(e.Text);
    }

    1 odwołanie
    public AddProdukty()
    {
        InitializeComponent();

        double CJResult = 0;
        if (double.TryParse(CJ.Text, out CJResult));
        int DosIlosResult = 0;
        if (int.TryParse(DostepnaIlosc.Text, out DosIlosResult));

        var model = new Model.AddProdukty()
        {
            Nazwa_produktu = NazwaProduktu.Text,
            Cena_jednostkowa = CJResult,
            Dostepna_ilość = DosIlosResult,
        };

        _viewModel = new AddProduktyViewModel(model);
        DataContext = _viewModel;
    }
}

```

14 AddProdukty.xaml.cs

- Model przechowujący dane:

```

Odwołania: 4
public class AddProdukty
{
    Odwołania: 6
    public string Nazwa_produktu { get; set; }
    Odwołania: 6
    public double Cena_jednostkowa { get; set; }
    Odwołania: 6
    public int Dostepna_ilość { get; set; }
}

```

15AddProdukty

- View Model:

Odwołania: 3

```
public class AddProduktyViewModel : BaseViewModel  
{
```

```
    private Model.AddProdukty _produkty;
```

Odwołania: 3

```
    public string Nazwa_produktu
```

```
{
```

```
    get { return _produkty.Nazwa_produktu; }
```

```
    set
```

```
{
```

```
        if (_produkty.Nazwa_produktu != value)
```

```
{
```

```
            _produkty.Nazwa_produktu = value;
```

```
            OnPropertyChanged(nameof(Nazwa_produktu));
```

```
        }
```

```
    }
```

```
}
```

Odwołania: 3

```
    public double Cena_jednostkowa
```

```
{
```

```
    get { return _produkty.Cena_jednostkowa; }
```

```
    set
```

```
{
```

```
        if (_produkty.Cena_jednostkowa != value)
```

```
{
```

```
            _produkty.Cena_jednostkowa = value;
```

```
            OnPropertyChanged(nameof(Cena_jednostkowa));
```

```
        }
```

```
    }
```

```
}
```

Odwołania: 3

```
    public int Dostępna_ilość
```

```
{
```

```
    get { return _produkty.Dostępna_ilość; }
```

```
    set
```

```
{
```

```
        if (_produkty.Dostępna_ilość != value)
```

```
{
```

```
            _produkty.Dostępna_ilość = value;
```

```
            OnPropertyChanged(nameof(Dostępna_ilość));
```

```
        }
```

```
    }
```

```
}
```

16AddProduktyViewModel1

```

    }
}

1 odwołanie
public AddProduktyViewModel(Model.AddProdukty produkty)
{
    _produkty = new Model.AddProdukty
    {
        Cena_jednostkowa = produkty.Cena_jednostkowa,
        Dostępna_ilość = produkty.Dostępna_ilość,
        Nazwa_produktu = produkty.Nazwa_produktu,
    };
    AddProduktyClick = new RelayCommand(x => DisplayMessage(), x => this.IsValid);
}

1 odwołanie
public bool IsValid { get => !string.IsNullOrEmpty(Nazwa_produktu) && Cena_jednostkowa != 0 && Dostępna_ilość != 0; }

1 odwołanie
private void DisplayMessage()
{
    using (var context = new EwidencjaContext())
    {
        context.Produkty.Add(new Produkty {
            Nazwa_produktu = Nazwa_produktu,
            Cena_jednostkowa = Cena_jednostkowa,
            Dostępna_ilość = Dostępna_ilość });
        context.SaveChanges();
    }
    MessageBox.Show($"Produkt została dodana do bazy.", "Info", MessageBoxButton.OK, MessageBoxImage.Information);
}

1 odwołanie
public ICommand AddProduktyClick
{
    get;
    private set;
}
}

```

17AddProduktyViewModel2

Wynik końcowy:

The screenshot shows a Windows application window titled 'MainWindow'. The window has a light blue background. At the top left, there are two buttons: 'Wyświetl' and 'Dodaj'. Below these, there are three text input fields. The first field is labeled 'Nazwa Produktu' and contains the text 'Banan'. The second field is labeled 'Cena Jednostkowa' and contains the number '5'. The third field is labeled 'Dostępna Ilość' and contains the number '300'. Below the third field is a button labeled 'Dodaj produkt'.

18Dodanie produktu

select * from Produkty

100 %

Results Messages

	Id_Produktu	Nazwa_produktu	Cena_jednostkowa	Dostępna_ilość
1	2003	Banan	5	300

19 Produkt SQL

6. Wyświetlanie danych

Wyświetlanie danych wymaga tych samych elementów co dodawanie.

Kod pozwalający na podgląd danych z tabeli Produkty:

- Strona xaml:

```
<Label Grid.Column="0" Grid.Row="1" Margin="4,0,0,0" Content="ID Faktury: " HorizontalAlignment="Right"/>
<Label Grid.Column="1" Grid.Row="1" Margin="250,0,0,0" Content="0 - wyświetla wszystkie produkty" HorizontalAlignment="Left"/>

<TextBox x:Name="Id_Produktu" PreviewTextInput="NumberValidationTextBox" Grid.Column="1" Grid.Row="1" Margin="4" Text="{Binding Id_Produktu, UpdateSourceTrigger=PropertyChanged}" HorizontalAlignment="Left" Width="200"/>

<Button Grid.Column="1" Grid.Row="4" Margin="4" Content="Szukaj produkt" Command="{Binding ListProduktyClick}" HorizontalAlignment="Left" VerticalAlignment="Top"/>

<ListView x:Name="Produkty" Margin="10,99,10,-205" Grid.Row="4" ItemsSource="{Binding FindProdukty}" Grid.ColumnSpan="2" RenderTransformOrigin="1,0" ScrollViewer.VerticalScrollBarVisibility="Visible">
    <ListView.View>
        <GridView AllowsColumnReorder="True" ColumnHeaderToolTip="Produkty">
            <GridViewColumn DisplayMemberBinding="{Binding Path=Id_Produktu, UpdateSourceTrigger=PropertyChanged}" Header="ID" Width="auto"/>
            <GridViewColumn DisplayMemberBinding="{Binding Path=Nazwa_produktu, UpdateSourceTrigger=PropertyChanged}" Header="Nazwa" Width="auto"/>
            <GridViewColumn DisplayMemberBinding="{Binding Path=Cena_jednostkowa, UpdateSourceTrigger=PropertyChanged}" Header="Cena Jednostkowa" Width="auto"/>
            <GridViewColumn DisplayMemberBinding="{Binding Path=Dostępna_ilość, UpdateSourceTrigger=PropertyChanged}" Header="Ilość" Width="auto"/>
        </GridView>
    </ListView.View>
</ListView>
```

20 ListProdukty.xaml

- Kod c#:

```

public partial class ListProdukty : Page
{
    private readonly ListProduktyViewModel _viewModel;
    1 odwołanie
    private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)
    {
        Regex regex = new Regex("[^0-9]+");
        e.Handled = regex.IsMatch(e.Text);
    }
    1 odwołanie
    public ListProdukty()
    {
        InitializeComponent();

        int result = 0;
        if (int.TryParse(Id_Projektu.Text, out result)) ;

        var model = new ListProduktyModel
        {
            Id_Projektu = result
        };

        _viewModel = new ListProduktyViewModel(model);
        DataContext = _viewModel;
    }
}

```

21 ListProdukty.xaml.cs

- Model przechowujący dane:

```

Odwołania: 7
public class ListProduktyModel
{
    Odwołania: 6
    public int Id_Projektu { get; set; }
    Odwołania: 2
    public string Nazwa_produkta { get; set; }
    Odwołania: 2
    public double Cena_jednostkowa { get; set; }
    Odwołania: 2
    public int Dostępna_ilość { get; set; }
}

```

22 ListProduktyModel

- View Model:


```

Odwolania: 3
public class ListProduktyViewModel : BaseViewModel
{
    private Model.ListProduktyModel _listProdukty;

    Odwołania: 5
    public int Id_Produktu
    {
        get { return _listProdukty.Id_Produktu; }
        set
        {
            if (_listProdukty.Id_Produktu != value)
            {
                _listProdukty.Id_Produktu = value;
                OnPropertyChanged(nameof(Id_Produktu));
            }
        }
    }

    1 odwołanie
    public ListProduktyViewModel(Model.ListProduktyModel listProduktyModel)
    {
        _listProdukty = listProduktyModel;

        ListProduktyClick = new RelayCommand(x => DisplayMessage(), x => this.IsValid);
    }

    1 odwołanie
    public bool IsValid { get => Id_Produktu >= 0; }

    Odwołania: 2
    public ObservableCollection<Model.ListProduktyModel> FindProdukty { get; } = new ObservableCollection<Model.ListProduktyModel>();
    1 odwołanie
    private void DisplayMessage()
    {
        using (var context = new EwidencjaContext())
        {
            if (Id_Produktu > 0)
            {
                var fakt = context.Produkty
                    .Where(x => x.Id_Produktu == Id_Produktu)
                    .ToList();
            }
        }
    }
}

```

23 ListProduktyViewModel

```

        foreach (var item in fakt)
        {
            FindProdukty.Add(new Model.ListProduktyModel()
            {
                Id_Projektu = item.Id_Projektu,
                Nazwa_projektu = item.Nazwa_projektu,
                Cena_jednostkowa = item.Cena_jednostkowa,
                Dostępna_ilość = item.Dostępna_ilość,
            });
        }
    }

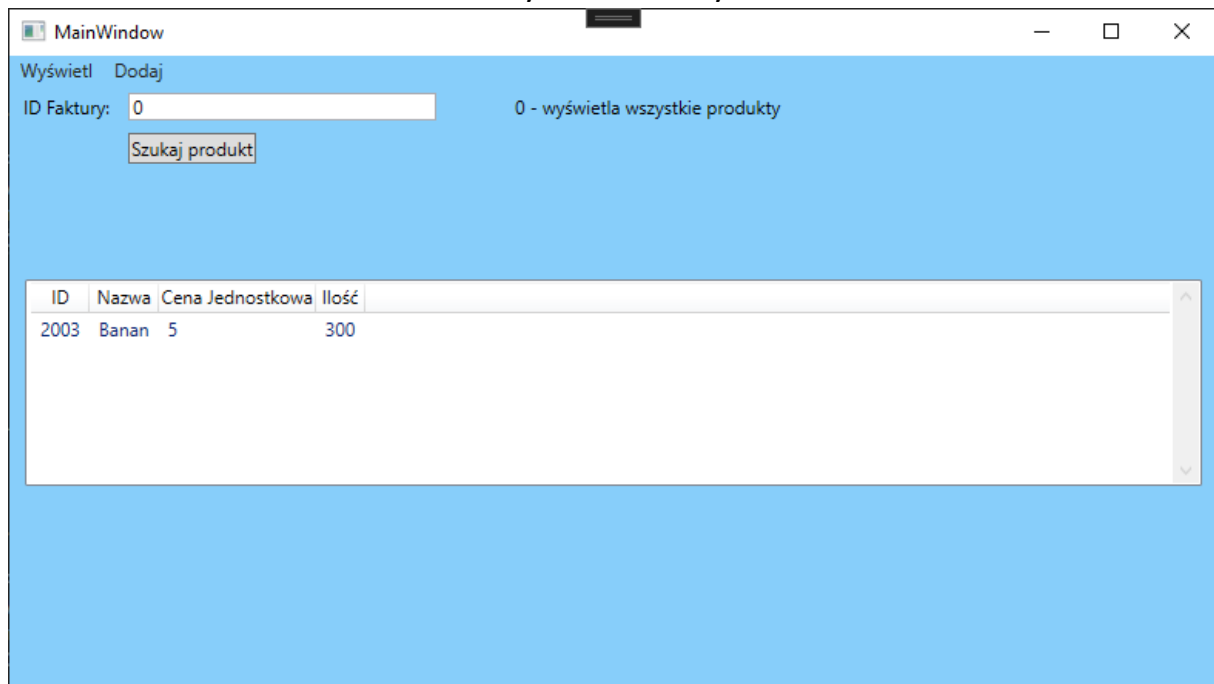
    else if (Id_Projektu == 0)
    {
        var fakt = context.Produkty.ToList();

        foreach (var item in fakt)
        {
            FindProdukty.Add(new Model.ListProduktyModel()
            {
                Id_Projektu = item.Id_Projektu,
                Nazwa_projektu = item.Nazwa_projektu,
                Cena_jednostkowa = item.Cena_jednostkowa,
                Dostępna_ilość = item.Dostępna_ilość,
            });
        }
    }
}
1 odwołanie
public ICommand ListProduktyClick
{
    get;
    private set;
}
}

```

24 ListProduktyViewModel2

Wynik końcowy:



Wyświetl Dodaj

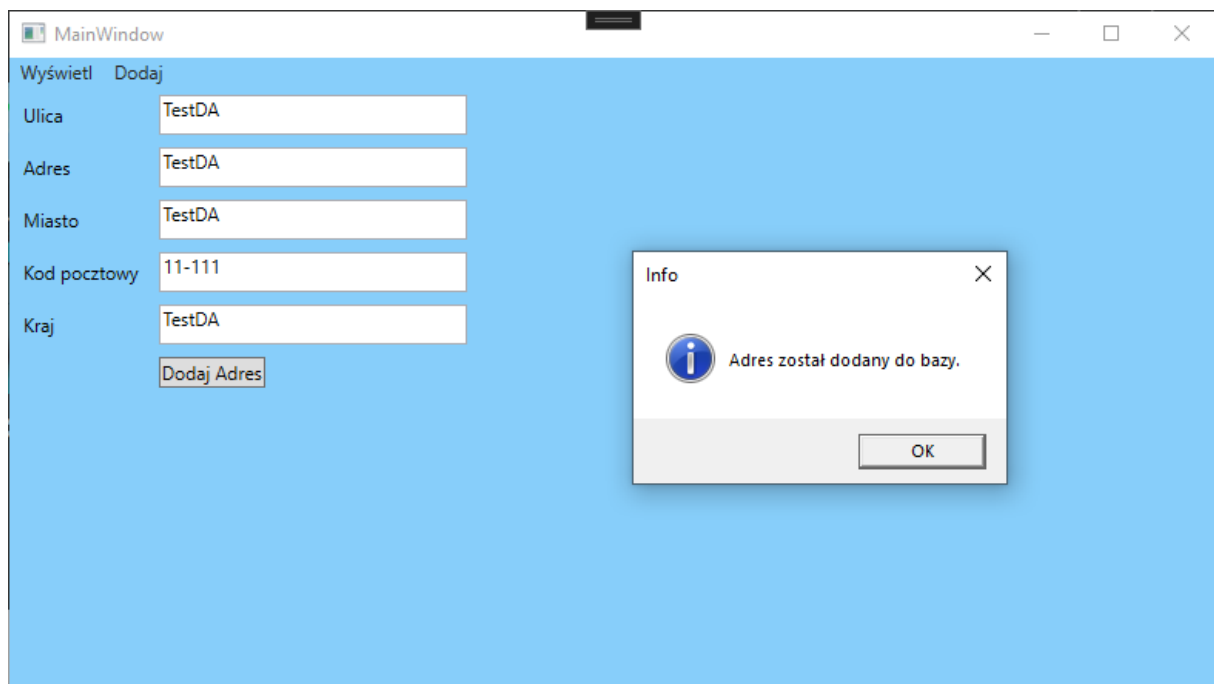
ID Faktury: 0 - wyświetla wszystkie produkty

ID	Nazwa	Cena Jednostkowa	Ilość
2003	Banan	5	300

25 Wyświetlanie produkty

7.Przykładowe działania na innych tabelach

- Dane Adresowe:



Wyświetl Dodaj

Ulica


Adres

Miasto

Kod pocztowy

Kraj

Info

 Adres został dodany do bazy.

26 Dodawanie Danych Adresowych

MainWindow

Wyświetl Dodaj

ID Adresu: 0 - wyświetla wszystkie dane adresowe

ID Adres	Ulica	Adres	Miasto	Kod pocztowy	Kraj
3	TestDA	TestDA	TestDA	11-111	TestDA

27 Wyświetlanie Danych Adresowych

- Klienci:

MainWindow

Wyświetl Dodaj

Imię klienta

Nazwisko klienta


Numer telefonu

Pesel

Email

Id Adresu

Info

 Klient została dodana do bazy.

28 Dodawanie klienta

MainWindow

Wyświetl Dodaj

ID Klienta: 0 - wyświetla wszystkich klientów

ID Klient	Imię klienta	Nazwisko klienta	Numer telefonu	Pesel	Email	Id Adresu
2	Test	Test	123456789	3333333333	Test@wp.pl	1

29 Wyświetlanie klienta