Cel:

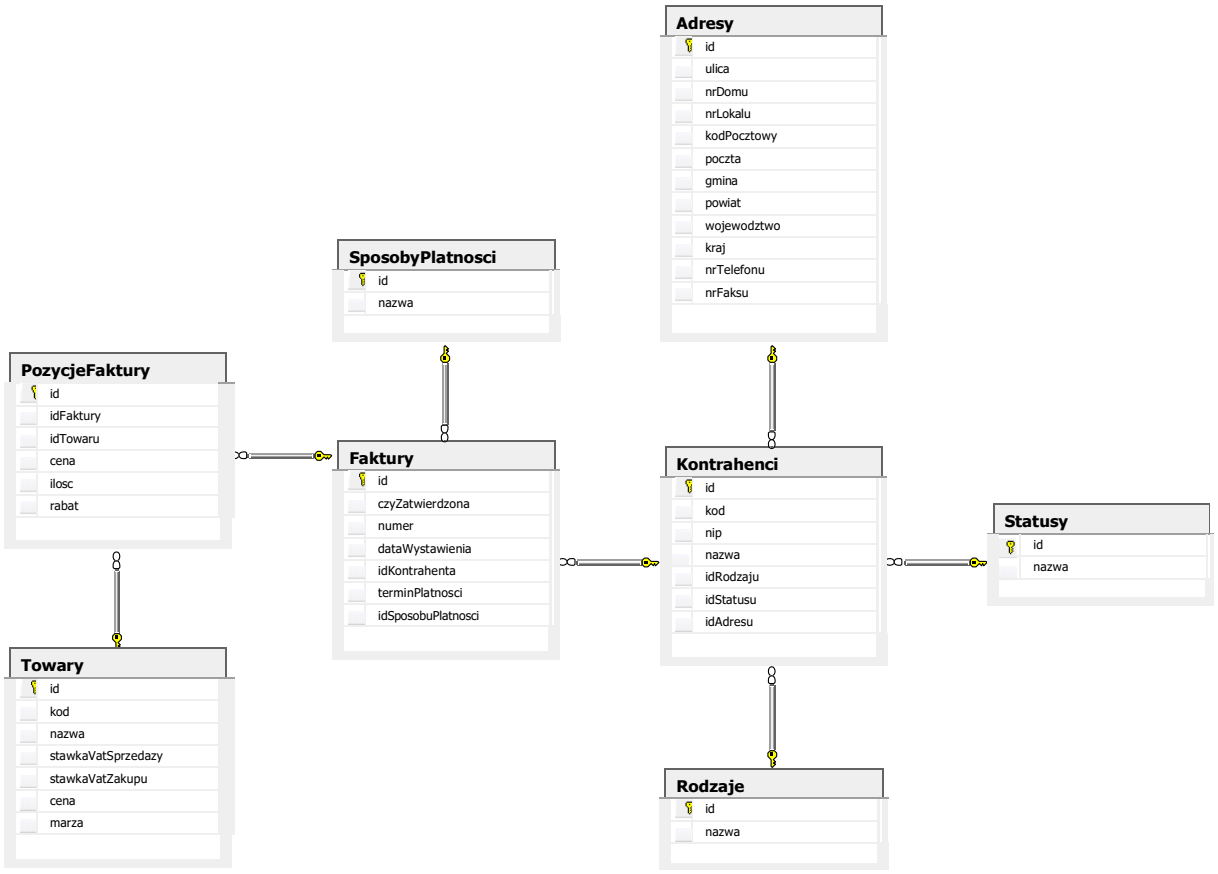
Przygotować aplikację do fakturowania według wzoru:

https://www.google.com/search?q=faktura+optima&rlz=1C1GCEU_plPL921PL921&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiCs5Ta15jsAhVrposKHeDyBM4Q_AUoAXoECAUQAw&biw=1536&bih=736&dpr=1.25

Entity Framework

Przygotowania

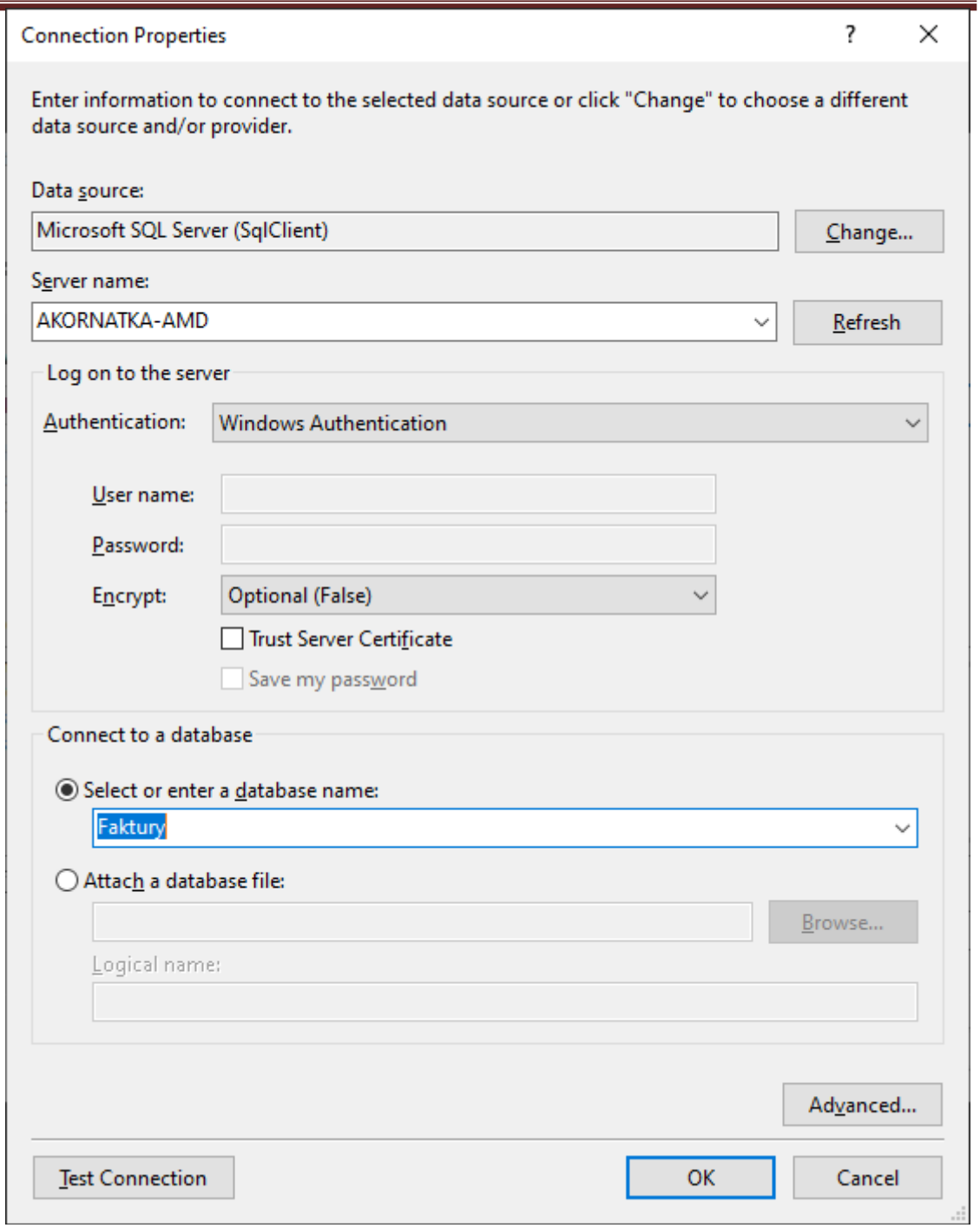
1. Przygotuj bazę danych ***Faktury*** według wzoru (z uwagi na ograniczony czas zajęć dokonano niewielkiego ograniczenia liczby pól tabel) :



2. Wypełnij tabele bazy danych przykładowymi danymi.

Generowanie modelu

1. Otwórz aplikację ***MVVMFirma*** z materiałów prowadzącego zajęcia.
2. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***Model***, a następnie wybierz ***Add->New Item->Zakładka Data-> ADO.NET Entity Data Model***. Model nazwij ***ModelFaktury***.
3. Wybierz **Next** i opcję **EF Designer from Database** i wybierz **Nex**t.
4. Wybierz opcję **New Connection**, wpisz nazwę swojego serwera i pozostałe ustawienia według wzoru:



5. Ustaw nazwę **Entities** na **FakturyEntities**, następnie **Next** i wszystkie tabele i **Finish**.

Tworzenie widoków dziedziczonych

1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***Views*** i wybierz ***Add->New Item->Zakładka WPF->Custom Control (WPF)***. ***Custom Control*** nazwij ***WszystkieViewBase***.
2. Wejdź do kodu właśnie utworzonej klasy i popraw jej dziedziczenie według wzoru (klasa powinna dziedziczyć po *UserControl*):

```
public class WszystkieViewBase : UserControl
```

3. Przy tworzeniu ***Custom Control*** oprócz klasy w katalogu głównym projektu tworzony jest folder ***Themes***, a w nim plik ***Generic.xaml***, który steruje wyglądem tego ***Custom Control-a***. Wejdź do kodu tego pliku i zmodyfikuj go według wzoru:

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presen
    tation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:local="clr-namespace:MVVMFirma.Views">
    <Style TargetType="{x:Type local:WszystkieViewBase}">
<Setter Property="Template">
          <Setter.Value>
             <ControlTemplate TargetType="{x:Type
             local:WszystkieViewBase}">
                    <Grid Margin="0,10,0,0">
                       <Grid.RowDefinitions>
                          <RowDefinition Height="34"/>
                          <RowDefinition Height="60"/>
                          <RowDefinition Height="*"/>
                       </Grid.RowDefinitions>
                       <ToolBar Grid.Row="0" Height="30"
                       Margin="0,2,0,2">
                          <ToggleButton Content="Dodaj" Width="70"
                          Height="30"/>
                          <ToggleButton Content="Modyfikuj" Width="70"
Height="30
"/>
                          <ToggleButton Content="Kasuj" Width="70"
                          Height="30"/>
                       <ToggleButton Content="Odswiez" Width="70"
                       Height="30"
Command="{Binding LoadCommand}"/>
                    </ToolBar>
                    <StackPanel Grid.Row="1" Background="Azure">
                       <Label Content="Sortowanie i filtorowanie"/>
                    </StackPanel>
                    <ContentPresenter Grid.Row="2" Margin="0,5,0,5" />
```

```
                </Grid>

            </ControlTemplate>
        </Setter.Value>
</Setter>
    </Style>
 </ResourceDictionary>
```

4. Następnie wejdź do katalogu ***Views***, naciśnij na strzałkę przy pliku ***WszystkieTowaryView.xaml*** i zmodyfikuj kod pliku ***WszystkieTowaryView.xaml.cs*** według wzoru:

```
public partial class WszystkieTowaryView : WszystkieViewBase
    {
public WszystkieTowaryView()
{
            InitializeComponent();
}


    }
```

5. Następnie wejdź do kodu pliku ***WszystkieTowaryView.xaml*** i zmodyfikuj ten kod według wzoru:

```
<local:WszystkieViewBase
            x:Class="MVVMFirma.Views.WszystkieTowaryView"
            xmlns="http://schemas.microsoft.com/winfx/2006/xam
            l/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/x
            aml" xmlns:local="clr-namespace:MVVMFirma.Views"
            Height="60
            0"
            Width="110
            0">
    <Grid>
<DataGrid AutoGenerateColumns="False" ItemsSource="{Binding
TowaryList}">
            <DataGrid.Columns>
                <DataGridTextColumn x:Name="id" Binding="{Binding
                Path=id}"
 Header="Nr"/
                <DataGridTextColumn x:Name="kod" Binding="{Binding
 >              Path=kod}"

 Header="Kod"  <DataGridTextColumn x:Name="nazwa" Binding="{Binding
                Path=nazwa}"
 />
                <DataGridTextColumn x:Name="stawkaVatSprzedazy"
 Header="Nazwa Binding="{Binding

 "/>


 Path=stawkaVatSprzedazy}" Header="Stawka Vat Sprzedazy"/>
                <DataGridTextColumn x:Name="stawkaVatZakupu"
```

```
Binding="{Binding Path=stawkaVatZakupu}" Header="Stawka Vat Zakupu"/>
            <DataGridTextColumn x:Name="cena" Binding="{Binding
Path=cena}" Header="Cena"/>
            <DataGridTextColumn x:Name="marza" Binding="{Binding
            Path=marza}"
Header="Marza"/>
        </DataGrid.Columns>
</DataGrid>
    </Grid>
</local:WszystkieViewBase>
```

Tworzenie WszystkieTowaryViewModel – wyświetlanie towarów

1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder

ViewModels i wybierz klasę ***WszystkieTowaryViewModel***.

Zmodyfikuj kod tej klasy według wzoru:

```
using
System.Linq;
using
System.Text;
using
System.Collections.ObjectModel
; using
MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using
System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    class WszystkieTowaryViewModel : WorkspaceViewModel
    {
#region Fields
//polaczenie z baza danych
private readonly FakturyEntities fakturyEntities;
private BaseCommand _LoadCommand;
//lista towarow zaladowana z bazy danych
private ObservableCollection<Towary> _TowaryList;
#endregion // Fields
#region Properties
public ICommand LoadCommand
{
          get
          {
             if (_LoadCommand == null)
             {
                _LoadCommand = new BaseCommand(() => load());
             }
             return _LoadCommand;
          }
}
public ObservableCollection<Towary> TowaryList
{
          ge           {
          t
          {
                       }
                                    }

          }
          se
          t
```

```
                aryList ==
                null) load();
i             return _TowaryList;
   f
   (
   _
   T          _TowaryList = value;
   o          OnPropertyChanged(() =>
   w          TowaryList);

#endregion //Properties
#region Constructor
public WszystkieTowaryViewModel()
{
          base.DisplayName = "Wszystkie towary";
```

```
            this.fakturyEntities = new FakturyEntities();
}
#endregion // Constructor
#region Helpers
private void load()
{
            TowaryList = new ObservableCollection<Towary>
               (from towar in fakturyEntities.Towary select towar);
}
#endregion
    }
 }
```

Tworzenie WszystkieViewModel – abstrakcyjne wyświetlanie

1. Zmień specyfikator dostępu do klasy ***BaseCommand*** (katalog ***Helper***) na public według wzoru:

   ```
   public class BaseCommand : ICommand
   ```

2. Do klasy ***BaseViewModel*** dodaj region ***Command*** według wzoru:

   ```
   #region Command
public delegate void CommandDelegate();
        protected BaseCommand GetCommand( BaseCommand baseCommand,
 CommandDelegate function )
{
            if (baseCommand == null)
            {
               baseCommand = new BaseCommand(() => function());
            }
            return baseCommand;
}
#endregion //Command
   ```

3. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***ViewModels*** i wybierz ***Add->Class***. Klasę nazwij ***WszystkieViewModel***, następnie utwórz ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using
```

```
System.Collections.ObjectModel
; using System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    public abstract class WszystkieViewModel<T> : WorkspaceViewModel
```

```
    {
#region Fields
//polaczenie z baza danych
protected readonly FakturyEntities fakturyEntities;
private BaseCommand _LoadCommand;
//lista towarow zaladowana z bazy danych
private ObservableCollection<T> _List;
#endregion // Fields
#region Properties
//public ICommand LoadCommand
//{
//          get
//          {
//             if (_LoadCommand == null)
//             {
//                _LoadCommand = new BaseCommand(() => load());
//             }
//             return _LoadCommand;
//          }
//}
public ICommand LoadCommand
{
          ge
          t
          {  return GetCommand(_LoadCommand,load);

          }
       }

public ObservableCollection<T> List
{
          ge
          t
          {  if (_List ==
                null)
                load();
             return _List;
          }
          se
          t
          {  _List = value;
             OnPropertyChanged(() =>
             List);
          }
       }


#endregion //Properties
#region Constructor
```

```
public WszystkieViewModel()
{
            this.fakturyEntities = new FakturyEntities();
}
#endregion // Constructor
#region Helpers
public abstract void load();
#endregion
    }
```

```
}
```

Modyfikacja WszystkieTowaryViewModel – dziedziczenie po klasie abstrakcyjnej

1. Edytuj klasę ***WszystkieTowaryViewModel*** i zmodyfikuj ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
System.Collections.ObjectModel
; using
MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using
System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    class WszystkieTowaryViewModel : WszystkieViewModel<Towary>
    {

#region Constructor
public WszystkieTowaryViewModel()
          : base()
{
          base.DisplayName = "Wszystkie towary";
}
#endregion // Constructor
#region Helpers
public override void load()
{
          List = new
            ObservableCollection<Towary> (
              from towar in
              fakturyEntities.Towary select
              towar
            );
}
#endregion
    }

}
```

2. Edytuj plik ***WszystkieTowaryView.xaml*** (katalog ***Views***) i zmień definicję ***DataGird*** (właściwość ***ItemsSource***) według wzoru:

<DataGrid AutoGenerateColumns="False" ItemsSource="{Binding List}">

Tworzenie widoków dodających rekord

1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***Views*** i wybierz ***Add->New Item->Zakładka WPF->Custom Control (WPF)***. ***Custom Control*** nazwij ***JedenViewBase***.
2. Wejdź do kodu właśnie utworzonej klasy i popraw jej dziedziczenie według wzoru (klasa powinna dziedziczyć po ***UserControl***):

```
public class JedenViewBase : UserControl
```

3. Przy tworzeniu kolejnego ***Custom Control*** w pliku ***Generic.xaml (z folderu Themes)***, utworzona zostanie kolejna sekcja sterująca wyglądem kolejnego ***Custom Control***. Wejdź do kodu tego pliku i zmodyfikuj jego sekcję

```
<Style TargetType="{x:Type local:JedenViewBase}">
```

według wzoru:

```
<Style TargetType="{x:Type local:JedenViewBase}">
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type
                local:JedenViewBase}">
                    <Grid Margin="0,10,0,0">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="34"/>
                            <RowDefinition Height="*"/>
                        </Grid.RowDefinitions>
                        <ToolBar Grid.Row="0" Height="30"
                        Margin="0,2,0,2">
                     <ToggleButton Content="Zapisz i zamknij"
Width="100" Height="30" Command="{Binding SaveCommand}"/>
                        </ToolBar>
                        <ContentPresenter Grid.Row="1"
                        Margin="0,5,0,5" />
                    </Grid>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

4. Następnie wejdź do katalogu ***Views***, naciśnij na strzałkę przy pliku ***NowyTowarView.xaml*** i zmodyfikuj kod pliku ***NowyTowarView.xaml.cs*** według wzoru:

```
public partial class NowyTowarView : JedenViewBase
    {
        public NowyTowarView()
        {
            InitializeComponent();
        }
```

5. Następnie wejdź do kodu pliku ***NowyTowarView.xaml*** i zmodyfikuj ten kod według wzoru:

```
<local:JedenViewBase x:Class="MVVMFirma.Views.NowyTowarView"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
            xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:local="clr-namespace:MVVMFirma.Views"
            mc:Ignorable="d"
            d:DesignHeight="600" d:DesignWidth="1100">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition Width="200"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
        </Grid.RowDefinitions>
        <Label Grid.Column="0" Grid.Row="0" Content="Kod"
        Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="0"
                                        Margin="0,8,0,8"
Width="50" HorizontalAlignment="Left" Text="{Binding
Path=Kod}"/>

        <Label Grid.Column="0" Grid.Row="1" Content="Nazwa"
        Margin="0,8,0,8" />
        <TextBox Grid.Column="1" Grid.Row="1"
                                        Margin="0,8,0,8"
Text="{Binding Path=Nazwa,
UpdateSourceTrigger=PropertyChanged}"/>

        <Label Grid.Column="0" Grid.Row="2" Content="Stawka Vat
Sprzedaży" Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="2"
                                        Margin="0,8,0,8"
Width="50" HorizontalAlignment="Left" Text="{Binding
Path=StawkaVatSprzedazy,UpdateSourceTrigger=PropertyChanged}"
/>

        <Label Grid.Column="0" Grid.Row="3" Content="Stawka Vat Zakupu"
```

```
Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="3"
                                        Margin="0,8,0,8"
Width="50" HorizontalAlignment="Left" Text="{Binding
Path=StawkaVatZakupu,
UpdateSourceTrigger=PropertyChanged}"/>

        <Label Grid.Column="0" Grid.Row="4" Content="Cena"
        Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="4"
                                        Margin="0,8,0,8"
Width="50" HorizontalAlignment="Left" Text="{Binding
Path=Cena, UpdateSourceTrigger=PropertyChanged}"/>

        <Label Grid.Column="0" Grid.Row="5" Content="Marża"
        Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="5"
                                        Margin="0,8,0,8"
Width="50" HorizontalAlignment="Left" Text="{Binding
Path=Marza, UpdateSourceTrigger=PropertyChanged}"/>
```

```
    </Grid>
</local:JedenViewBase>
```

Tworzenie NowyTowarViewModel – dodawanie towarów

1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***ViewModels*** i wybierz klasę ***NowyTowarViewModel***.

   Zmodyfikuj kod tej klasy według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.Entities;
using
System.Windows.Input;
using MVVMFirma.Helper;

namespace MVVMFirma.ViewModels
{
    class NowyTowarViewModel : WorkspaceViewModel
    {
        #region Fields
        private FakturyEntities
        fakturyEntities; private Towary
        towar;
        private BaseCommand _SaveCommand;
        #endregion // Fields

        #region Constructor
        public NowyTowarViewModel()
        {
            base.DisplayName = "Nowy Towar";
            fakturyEntities = new
            FakturyEntities(); towar = new
            Towary();
        }
        #endregion // Constructor

        #region
        Properties
        public string
        Kod
        {
            ge            t
```

```
        {

        }  return towar.kod;
        se
        t
        {
           if (value ==
              towar.kod)
              return;
           towar.kod = value;
        }  base.OnPropertyChanged(()=>Kod);
```

```
}
public string Nazwa
{
    ge
    t
    {  return towar.nazwa;

    }
    se
    t  if (value ==
    {     towar.nazwa)
          return;
       towar.nazwa = value;
       base.OnProperty Changed(() =>
       Nazwa);
    }
}

public int? StawkaVatSprzedazy
{
    ge
    t
    {  return towar.stawkaVatSprzedazy;

    }
    se
    t  if (value ==
    {     towar.stawkaVatSprzedazy)
          return;
       towar.stawkaVatSprzedazy = value;
       base.OnProperty Changed(() =>
       StawkaVatSprzedazy);
    }
}

public int? StawkaVatZakupu
{
    ge            }
    t                          }
    {

    }
    se
    t
    {
```

```
            if (value ==
return          towar.stawkaVatZakupu)
towar.stawka    return;
VatZakupu;  towar.stawkaVatZakupu = value;
            base.OnPropertyChanged(() =>
            StawkaVatZakupu);
        public double? Cena
        {
            ge
            t
            {  return towar.cena;

            }
            se
            t   if (value ==
            {       towar.cena)
                    return;
```

```
            towar.cena = value;
            base.OnPropertyChanged(() =>
            Cena);
        }
    }
    public double? Marza
    {
        ge
        t
        {  return towar.marza;

        }
        se
        t  if (value ==
        {     towar.marza)
              return;
            towar.marza = value;
            base.OnPropertyChanged(() =>
            Marza);
        }
    }


    #endregion //Properties

    #region Command
    public ICommand SaveCommand
    {
        ge
        t
        {  if (_SaveCommand == null)
            {
                _SaveCommand = new BaseCommand(() => saveAndClose());
            }
            return _SaveCommand;
        }
    }
    #endregion
    //Command
    #region Helpers
    public void
    Save()
    {
        fakturyEntities.Towary.Add(towar);
        fakturyEntities.SaveChanges();
    }
    private void saveAndClose()
    {
        Save();
```

```
            //dodaj tę funkcję do klasy BaseViewModel
            ShowMessageBoxInformation("Zapisano Nowy Towar");

            OnRequestClose();//w klasie WorkspaceViewModel zmien
specyfikator dostepu tej funkcji na public
        }
        #endregion
    }
}
```

Tworzenie JedenViewModel – abstrakcyjne dodawanie

1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***ViewModels*** i wybierz ***Add->Class***. Klasę nazwij ***DataBaseViewModel***, następnie utwórz ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using MVVMFirma.Model.Entities;

namespace MVVMFirma.ViewModels
{
    public class DataBaseViewModel:WorkspaceViewModel
    {

        #region Fields
        protected FakturyEntities
        fakturyEntities; #endregion // Fields
        #region Constructor
        public DataBaseViewModel()
        {
            this.fakturyEntities = new FakturyEntities();
        }
        #endregion // Constructor
    }
}
```

2. Zmodyfikuj kod klasy ***WszystkieViewModel*** według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using
System.Collections.ObjectModel
; using System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    public abstract class WszystkieViewModel<T> : DataBaseViewModel
```

```
    {
        #region Fields
        private BaseCommand _LoadCommand;
        //lista towarow zaladowana z bazy danych
        private ObservableCollection<T> _List;
        #endregion //
        Fields #region
        Properties
```

```
//public ICommand LoadCommand
//{
//  get
//  {
//      if (_LoadCommand == null)
//      {
//          _LoadCommand = new BaseCommand(() => load());
//      }
//      return _LoadCommand;
//  }
//}
//lub analogicznie jak
wyzej public ICommand
LoadCommand
{
    ge
    t
    {  return GetCommand(_LoadCommand,load);

    }
}

public ObservableCollection<T> List
{
    ge
    t
    {  if (_List ==
          null)
          load();
       return _List;
    }
    se
    t
    {  _List = value;
       OnPropertyChanged(() =>
       List);
    }
}


#endregion
//Properties
#region
Constructor
public WszystkieViewModel()

    {
    }
        : base()
```

```
        #endregion //
        Constructor #region
        Helpers
        public abstract void
        load(); #endregion
    }

}
```

3. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***ViewModels*** i wybierz ***Add->Class***. Klasę nazwij ***JedenViewModel***, następnie utwórz ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    public abstract class JedenViewModel<T> : DataBaseViewModel
    {
        #region Fields
        private BaseCommand _SaveCommand;
        protected T item;
        #endregion // Fields

        #region
        Constructor public
        JedenViewModel()
            : base()
        {
        }
        #endregion // Constructor

        #region Command
        public ICommand SaveCommand
        {
            ge
            t
            {  if (_SaveCommand == null)
                {
                    _SaveCommand = new BaseCommand(() => saveAndClose());
                }
                return _SaveCommand;
            }
        }
        //lub analogocznie jak wyżej
        //public ICommand SaveCommand
        //{
        //  get
        //  {
```

```
//      return GetCommand(_SaveCommand,saveAndClose);
//  }
//}
#endregion
//Command
#region Helpers
private void
save()
{
```

```
            //ShowMessageBox(item.GetType().Name);
            fakturyEntities.AddObject(item.GetType().Name ,item);
            fakturyEntities.SaveChanges();
        }
        private void saveAndClose()
        {
            save();
            ShowMessageBoxInformation("Zapisano Nowy Obiekt");//dodaj
tę funkcję do klasy BaseViewModel
            OnRequestClose();//w klasie WorkspaceViewModel zmien
specyfikator dostepu tej funkcji na public
        }
        #endregion

    }
}
```

Modyfikacja NowyTowarViewModel – dziedziczenie po klasie abstrakcyjnej

1. Edytuj klasę ***NowyTowarViewModel*** i zmodyfikuj ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.Entities;
using
System.Windows.Input;
using MVVMFirma.Helper;

namespace MVVMFirma.ViewModels
{
    class NowyTowarViewModel : JedenViewModel<Towary>
    {
        #region Constructor
        public NowyTowarViewModel()
            : base()
        {
            base.DisplayName =
            "Towar"; item = new
            Towary();
        }
        #endregion // Constructor

        #region
```

```
Properties
public string
Kod
{
    ge
    t
    {  return item.kod;

    }
    se
    t
    {
```

```
            if (value ==
                item.kod)
                return;
            item.kod = value;
            base.OnPropertyChanged(() =>
            Kod);
        }
    }
    public string Nazwa
    {
        ge
        t
        {  return item.nazwa;

        }
        se
        t   if (value ==
        {       item.nazwa)
                return;
            item.nazwa = value;
            base.OnPropertyChanged(() =>
            Nazwa);
        }
    }

    public int? StawkaVatSprzedazy
    {
        ge
        t
        {  return item.stawkaVatSprzedazy;

        }
        se
        t   if (value ==
        {       item.stawkaVatSprzedazy)
                return;
            item.stawkaVatSprzedazy = value;
            base.OnPropertyChanged(() =>
            StawkaVatSprzedazy);
        }
    }

    public int? StawkaVatZakupu
    {
        ge              set
        t               {
        {

        }
```

```
    }
}       return item.stawkaVatZakupu;


        if (value ==
           item.stawkaVatZakupu) return;
        item.stawkaVatZakupu = value;
        base.OnPropertyChanged(() =>
        StawkaVatZakupu);
public double? Cena
{
    get
    {
        return item.cena;
```

```
    }
    se
    t
    {  if (value ==
          item.cena)
          return;
       item.cena = value;
       base.OnPropertyChanged(() =>
    }  Cena);
}

public double? Marza
{
    ge
    t
    {  return item.marza;

    }
    se
    t  if (value ==
    {     item.marza)
          return;
       item.marza = value;
       base.OnPropertyChanged(() =>
       Marza);
    }
}
```

```
        #endregion //Properties
    }
}
```

Wyświetlanie tabeli z kluczem obcym

1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***Model*** i wybierz ***Add->New Folder***, folder nazwij ***EntitiesForView.***
2. Następnie naciśnij prawym klawiszem myszki na folder ***EntitiesForView*** i wybierz ***Add->Class***. Klasę nazwij ***FakturyForAllView*** i utwórz jej kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MVVMFirma.Model.EntitiesForView
{
    public class FakturyForAllView
    {
        #region
        Properties
        private int
        _Id; public
        int Id
        {
            ge
            t
            {  return _Id;

            }
            se
            t  if (_Id != value)
            {  {

                   _Id = value;
               }
            }
        }
        private String
        _Numer; public
        String Numer
        {
            ge            }
            t             set
            {             {
```

```
            if (_Numer != value)
return      {
_Numer;


                _Numer = value;
            }
        }
    }
    private DateTime?
    _DataWystawienia; public
    DateTime? DataWystawienia
    {
        get
        {
```

```
        return _DataWystawienia;
    }
    se
    t
    {   if (_DataWystawienia != value)
        {
            _DataWystawienia = value;
        }
    }
}
private String
_KontrahentNazwa; public
String KontrahentNazwa
{
    ge
    t
    {   return _KontrahentNazwa;

    }
    se
    t   if (_KontrahentNazwa != value)
    {   {

            _KontrahentNazwa = value;
        }
    }
}
private DateTime?
_TerminPlatnosci; public
DateTime? TerminPlatnosci
{
    ge
    t
    {   return _TerminPlatnosci;

    }
    se
    t   if (_TerminPlatnosci != value)
    {   {

            _TerminPlatnosci = value;
        }
    }
}
private String
_SposobPlatnosciNazwa; public
String SposobPlatnosciNazwa
{
    ge            t
```

```
{

} return _SposobPlatnosciNazwa;
se
t
{
    if (_SposobPlatnosciNazwa != value)
    {
        _SposobPlatnosciNazwa = value;
    }
```

```
            }
        }
        #endregion //Properties

    }
}
```

3. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder **View*Models*** i wybierz ***Add->Class***. Klasę nazwij ***WszystkieFakturyViewModel***, następnie utwórz ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using MVVMFirma.Model.Entities;
using
System.Collections.ObjectModel;
using
MVVMFirma.Model.EntitiesForView;

namespace MVVMFirma.ViewModels
{
    class WszystkieFakturyViewModel  :
    WszystkieViewModel<FakturyForAllView>
    {
        #region Constructor
        public WszystkieFakturyViewModel()
        :base()
        {
            base.DisplayName = "Wszystkie faktury";
        }
        #endregion //
        Constructor #region
        Helpers
        public override void load()
        {
            List = new
                ObservableCollection<FakturyForAllView
                >(
                    from faktura in
                    fakturyEntities.Faktury select new
                    FakturyForAllView
                    {
                        Id = faktura.id,
                        Numer = faktura.numer,
                        DataWystawienia = faktura.dataWystawienia,
```

```
                    KontrahentNazwa = faktura.Kontrahenci.nazwa,
                    TerminPlatnosci = faktura.terminPlatnosci,
                    SposobPlatnosciNazwa =
                    faktura.SposobyPlatnosci.nazwa
                }

            );
        }
        #endregion //Helpers
    }
}
```

4. Edytuj klasę ***MainWindowViewModel*** (z katalogu ***ViewModel***), a następnie uzupełnij jej funkcję ***CreateCommands*** według wzoru:

```
List<CommandViewModel> CreateCommands()
        {
            return new List<CommandViewModel>
            {
                new
                    CommandViewModel
                    ( "Wszystkie
                    Towary",
                    new BaseCommand(() => this.ShowAllTowar())),

                new
                    CommandViewMod
                    el( "Nowy
                    Towar",
                    new BaseCommand(() => this.CreateTowar())),

                new
                    CommandViewModel(
                    "Wszystkie
                    Faktury",
                    new BaseCommand(() => this.ShowAllFaktury()))

            };
        }
```

5. Edytuj klasę ***MainWindowViewModel*** (z katalogu ***ViewModel***), a następnie utwórz w niej dodatkową funkcję ***ShowAllFaktury*** według wzoru:

```
void ShowAllFaktury()
        {
            WszystkieFakturyViewModel workspace =
                this.Workspaces.FirstOrDefault(vm => vm is
                WszystkieFakturyViewModel) as WszystkieFakturyViewModel;

            if (workspace == null)
            {
                workspace = new WszystkieFakturyViewModel();
                this.Workspaces.Add(workspace);
            }

            this.SetActiveWorkspace(workspace);
        }
```

6. Naciśnij prawym klawiszem myszy na folder ***Views*** i wybierz ***Add->UserControl***.

UserControl nazwij ***WszystkieFakturyView***. Uzupełnij kod tego widoku według wzoru:

```
<local:WszystkieViewBase
        x:Class="MVVMFirma.Views.WszystkieFakturyView"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pr
        esentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
        compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/
        2008" xmlns:local="clr-namespace:MVVMFirma.Views"
        mc:Ignorable="d"
```

```
            d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <DataGrid AutoGenerateColumns="False" ItemsSource="{Binding
        List}">
            <DataGrid.Columns>
                <DataGridTextColumn x:Name="id" Binding="{Binding
                Path=Id}"
Header="Nr"/
                <DataGridTextColumn x:Name="numer" Binding="{Binding
>               Path=Numer}"

Header="Numer <DataGridTextColumn x:Name="dataWystawienia"
                Binding="{Binding
"/>


Path=DataWystawienia}" Header="Data Wystawienia"/>
                <DataGridTextColumn x:Name="kontrahenciNazwa"
Binding="{Binding Path=KontrahentNazwa}" Header="Nazwa Kontrahenta"/>
                <DataGridTextColumn x:Name="terminPlatnosci"
Binding="{Binding Path=TerminPlatnosci}" Header="Termin platnosci"/>
                <DataGridTextColumn x:Name="sposobuPlatnosciNazwa"
Binding="{Binding Path=SposobPlatnosciNazwa}" Header="Nazwa Sposobu
Platnosci"/>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</local:WszystkieViewBase>
```

7. Rozwiń widok ***WszystkieFakturyView*** i wejdź do kodu klasy ***WszystkieFakturyView.xaml.cs*** oraz zmodyfikuj dziedziczenie tej klasy według wzoru:

```
namespace MVVMFirma.Views
{
    /// <summary>
    /// Interaction logic for WszystkieFakturyView.xaml
    /// </summary>
    public partial class WszystkieFakturyView : WszystkieViewBase
    {
        public WszystkieFakturyView()
        {
            InitializeComponent();
        }
    }
}
```

8. Edytuj plik ***MainWindowResources.xaml*** (folder Views) i dodaj do niego kod według wzoru:

```
<DataTemplate DataType="{x:Type vm:WszystkieFakturyViewModel}">
        <vw:WszystkieFakturyView/>
    </DataTemplate>
```

Dodawanie rekordów do tabeli z kluczem obcym

1. Naciśnij prawym klawiszem myszki na folder ***EntitiesForView*** (katalog Model) i wybierz ***Add->Class***. Klasę nazwij ***ComboBoxKeyAndValue*** i utwórz jej kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MVVMFirma.Model.EntitiesForView
{
        public class ComboBoxKeyAndValue
        {
            public int Key { get;
            set; } public string Value
            { get; set; }
        }
}
```

2. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder **View*Models*** i wybierz ***Add->Class***. Klasę nazwij ***NowaFakturaViewModel***, następnie utwórz ją według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using MVVMFirma.Model.Entities;
using MVVMFirma.Model.EntitiesForView;

namespace MVVMFirma.ViewModels
{
    class NowaFakturaViewModel : JedenViewModel<Faktury>
    {
        #region Constructor
        public NowaFakturaViewModel()
```

```
    : base()
{
    base.DisplayName =
    "Faktura"; item = new
    Faktury();
}
#endregion // Constructor

#region
Properties
public string
Numer
{
    get
    {
        return item.numer;
```

```
    }
    se
    t
    {   if (value ==
            item.numer)
            return;
        item.numer = value;
        base.OnPropertyChanged(() =>
    }   Numer);
}


public DateTime? DataWystawienia
{
    ge
    t
    {   return item.dataWystawienia;

    }
    se
    t   if (value ==
    {       item.dataWystawienia)
            return;
        item.dataWystawienia = value;
        base.OnPropertyChanged(() =>
        DataWystawienia);
    }
}

public int? IdKontrahenta
{
    ge
    t
    {   return item.idKontrahenta;

    }
    se
    t   if (value ==
    {       item.idKontrahenta) return;
        item.idKontrahenta = value;
        base.OnPropertyChanged(() =>
        IdKontrahenta);

    }
}

public DateTime? TerminPlatnosci
{
    ge          t
```

```
    {

    } return item.terminPlatnosci;
    se
    t
    {
        if (value ==
            item.terminPlatnosci)
            return;
        item.terminPlatnosci = value;
    } base.OnPropertyChanged(() =>
}       TerminPlatnosci);

public int? IdSposobuPlatnosci
```

```
        {
            ge
            t
            {  return item.idSposobuPlatnosci;

            }
            se
            t  if (value ==
            {      item.idSposobuPlatnosci)
                   return;
               item.idSposobuPlatnosci = value;
               base.OnPropertyChanged(() =>
               IdSposobuPlatnosci);
            }
        }

        public IQueryable<SposobyPlatnosci> SposobyPlatnosciComboBoxItems
        {
            ge
            t
            {  return
                   (
                   from sposobPlatnosci in
                   fakturyEntities.SposobyPlatnosci select
                   sposobPlatnosci
                   ).ToList().AsQueryable();
            }
        }

        public IQueryable<ComboBoxKeyAndValue> KontrahenciComboBoxItems
        {
            ge
            t
            {  retur
                    n
                   (
                   from kontrahent in
                   fakturyEntities.Kontrahenci select new
                   ComboBoxKeyAndValue
                   {
                       Key = kontrahent.id,
kontrahent.n           Value = kontrahent.kod + " " + kontrahent.nazwa +
ip                     " " +

                   }
```

```
                ).ToList().AsQueryable();
            }
        }

        #endregion //Properties
        #region  Helpers
            public override void Save()
            {
               fakturyEntities.Faktury.Add(item);
               fakturyEntities.SaveChanges();
            }
        #endregion Heplpers
    }
}
```

3. Edytuj klasę ***MainWindowViewModel*** (z katalogu ***ViewModel***), a następnie utwórz w niej dodatkową funkcję ***CreateFaktura*** według wzoru:

```
void CreateFaktura()
{
            NowaFakturaViewModel workspace = new
            NowaFakturaViewModel();
            this.Workspaces.Add(workspace);
            this.SetActiveWorkspace(workspace);

}
```

4. Uzupełnij też funkcję ***CreateCommands()*** następującym kodem:

```
new CommandViewModel(
                "Nowa Faktura",
                new BaseCommand(() => this.CreateFaktura()))
```

5. Naciśnij prawym klawiszem myszy na folder ***Views*** i wybierz ***Add->UserControl***. ***UserControl*** nazwij ***NowaFakturaView***. Uzupełnij kod tego widoku według wzoru:

```
<local:JedenViewBase x:Class="MVVMFirma.Views.NowaFakturaView"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pr
            esentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:mc="http://schemas.openxmlformats.org/markup-
            compatibility/2006"
            xmlns:d="http://schemas.microsoft.com/expression/blend/
```

```
            2008" xmlns:local="clr-namespace:MVVMFirma.Views"
            mc:Ignorable="d"
            d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition Width="200"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
        </Grid.RowDefinitions>
        <Label Grid.Column="0" Grid.Row="0" Content="Numer"
        Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="0"
                                        Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" Text="{Binding
Path=Numer, UpdateSourceTrigger=PropertyChanged}"/>

        <Label Grid.Column="0" Grid.Row="1"
Content="DataWystawienia" Margin="0,8,0,8"/>
        <DatePicker Grid.Column="1" Grid.Row="1"
                                          Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" SelectedDate="{Binding
Path=DataWystawienia, UpdateSourceTrigger=PropertyChanged}"/>

        <Label Grid.Column="0" Grid.Row="2" Content="Kontrahent"
        Margin="0,8,0,8"/>
        <ComboBox Grid.Column="1" Grid.Row="2" Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" ItemsSource="{Binding
KontrahenciComboBoxItems}" DisplayMemberPath="Value"
SelectedValuePath="Key" SelectedValue="{Binding
Path=IdKontrahenta,Mode=TwoWay}"/>
        <Label Grid.Column="0" Grid.Row="3"
Content="TerminPlatnosci" Margin="0,8,0,8"/>
        <DatePicker Grid.Column="1" Grid.Row="3"
                                          Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" SelectedDate="{Binding
Path=TerminPlatnosci, UpdateSourceTrigger=PropertyChanged}" />

        <Label Grid.Column="0" Grid.Row="4" Content="Sposob
Platnosci" Margin="0,8,0,8"/>
        <ComboBox Grid.Column="1" Grid.Row="4" Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" ItemsSource="{Binding
SposobyPlatnosciComboBoxItems}" DisplayMemberPath="nazwa"
SelectedValuePath="id" SelectedValue="{Binding
Path=IdSposobuPlatnosci,Mode=TwoWay}"/>
```

```
    </Grid>
</local:JedenViewBase>
```

6. Rozwiń widok ***NowaFakturaView*** i wejdź do kodu klasy ***NowaFakturaView.xaml.cs*** oraz zmodyfikuj dziedziczenie tej klasy według wzoru:

```
namespace MVVMFirma.Views
{
    /// <summary>
    /// Interaction logic for NowaFakturaView.xaml
    /// </summary>
    public partial class NowaFakturaView : JedenViewBase
    {
        public NowaFakturaView()
        {
            InitializeComponent();
        }
    }
}
```

7. Edytuj plik ***MainWindowResources.xaml*** (folder ***Views***) i dodaj do niego kod według wzoru:

```
<DataTemplate DataType="{x:Type vm:NowaFakturaViewModel}">
        <vw:NowaFakturaView/>
    </DataTemplate>
```

8. Edytuj plik ***WszystkieFakturyView.xaml*** (folder ***Views***) i zmodyfikuj jego kod według wzoru:

```
<local:WszystkieViewBase
            x:Class="MVVMFirma.Views.WszystkieFakturyView"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
            xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:local="clr-namespace:MVVMFirma.Views"  mc:Ignorable="d"
    <Grid>  d:DesignHeight="300" d:DesignWidth="541">

        <DataGrid AutoGenerateColumns="False" ItemsSource="{Binding List}">
```

```
            <DataGrid.Columns>
                <DataGridTextColumn x:Name="id" Binding="{Binding
                Path=Id}"
Header="Nr"/
                <DataGridTextColumn x:Name="numer" Binding="{Binding
>               Path=Numer}"

Header="Numer <DataGridTemplateColumn x:Name="DataWystawienia"

"/>
```

```
Header="DataWystawienia">
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <DatePicker
SelectedDate="{Binding Path=DataWystawienia}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                <DataGridTextColumn x:Name="kontrahenciNazwa"
Binding="{Binding Path=KontrahentNazwa}" Header="Nazwa Kontrahenta" />
                <DataGridTemplateColumn x:Name="TerminPlatnosci"
Header="TerminPlatnosci">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <DatePicker
SelectedDate="{Binding Path=TerminPlatnosci}"/>
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                <DataGridTextColumn x:Name="sposobuPlatnosciNazwa"
Binding="{Binding Path=SposobPlatnosciNazwa}" Header="Nazwa Sposobu
Platnosci"/>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</local:WszystkieViewBase>
```

MVVM light Messenger – wywołanie okna

1. W ***SolutionExplorer*** naciśnij prawym klawiszem myszki na cały projekt i wybierz ***Manage NuGet Packages***. Wybierz zakładkę Browse i wyszukaj ***MvvmLight***. Zainstaluj tę bibliotekę.
2. Edytuj plik ***WszystkieViewModel*** (katalog ***ViewModels***). Zmodyfikuj jego kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using MVVMFirma.Model.Entities;


using MVVMFirma.Helper;
using
System.Collections.ObjectModel
; using System.Windows.Input;
using GalaSoft.MvvmLight.Messaging;
```

```
namespace MVVMFirma.ViewModels
{
    public abstract class WszystkieViewModel<T> : DataBaseViewModel
    {
        #region Fields
        private BaseCommand
        _LoadCommand; private
        BaseCommand _AddCommand;
        //lista towarow zaladowana z bazy
        danych private ObservableCollection<T>
        _List; #endregion // Fields
        #region Properties
        //public ICommand LoadCommand
        //{
        //  get
        //  {
        //      if (_LoadCommand == null)
        //      {
        //          _LoadCommand = new BaseCommand(() => load());
        //      }
        //      return _LoadCommand;
        //  }
        //}
        //lub analogicznie jak
        wyzej  public ICommand
        LoadCommand
        {
            ge
            t
            {  return GetCommand(_LoadCommand,load);

            }
        }

        public ICommand AddCommand
        {
            ge
            t
            {  if (_AddCommand null)
               ==
               {

                   _AddCommand = new BaseCommand(() => Add());
               }
               return _AddCommand;
            }
        }

        public ObservableCollection<T> List
```

```
        {
            get
            {
                if (_List ==
                    null)
                    load();
                return _List;

            }
            se
            t   _List = value;
            {   OnPropertyChanged(() =>
                List);

            }
        }


        #endregion
        //Properties
        #region
        Constructor
        public WszystkieViewModel()
            : base()
        {
        }

        #endregion //
        Constructor #region
        Helpers
        public abstract void
        load(); private void
        Add()
        {
            Messenger.Default.Send(DisplayName + " Add");
        }
        #endregion
    }

}
```

3. Edytuj plik ***MainWindowViewModel*** (katalog ***ViewModels***). W klasie ***MainWindowViewModel*** zmodyfikuj kod funkcji ***CreateCommands()*** według wzoru:

```
List<CommandViewModel> CreateCommands()
        {
            Messenger.Default.Register<string>(this, open);
            return new List<CommandViewModel>
```

```
        {
            new
                CommandViewModel
                ( "Wszystkie
                Towary",
                new BaseCommand(() => this.ShowAllTowar())),

            new
                CommandViewMod
                el( "Nowy
                Towar",
                new BaseCommand(() => this.CreateTowar())),

            new
                CommandViewModel(
                "Wszystkie
                Faktury",
                new BaseCommand(() => this.ShowAllFaktury())),

            new
                CommandViewMod
                el( "Nowa
                Faktura",
                new BaseCommand(() => this.CreateFaktura())),


            new
                CommandViewMode
                l( "Nowa
                Faktura 2",
                new BaseCommand(() => this.CreateFaktura2()))

        };
    }
```

4. W klasie ***MainWindowViewModel*** dodaj do regionu ***Helpers*** funkcję ***open(...)*** według wzoru:

```
private void open(string name)
        {
            if (name == "Wszystkie faktury Add")
                CreateFaktura();

            if (name == "Wszystkie towary Add")
                CreateTowar();
        }
```

5. W pliku ***Generic.xaml*** (katalog **Themes**) wejdź do

```
<ControlTemplate TargetType="{x:Type local:WszystkieViewBase}">
```

i zmodyfikuj kod przycisku *Dodaj* według wzoru:

```
<ToggleButton Content="Dodaj" Width="70" Height="30" Command="{Binding AddCommand}"/>
```

Zadanie do samodzielnego wykonania

Wykonaj ***UserControl*** do wyświetlanie wszystkich kontrahentów (wyświetlane pola to: Id, Kod, Nip, Nazwa, RodzajNazwa, StatusNazwa). Odpowiednio nazwij klasy i widoki: ***KontrahenciForAllView***, ***WszyscyKontrahenciViewModel***, ***WszyscyKontrahenciView***.

MVVM light Messenger – wypełnianie klucza obcego przez okno modalne

1. Edytuj plik ***NowaFakturaView*** (katalog ***View***). Zmodyfikuj jego kod według wzoru:

```
<local:JedenViewBase x:Class="MVVMFirma.Views.NowaFakturaView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:local="clr-namespace:MVVMFirma.Views"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition Width="200"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
        </Grid.RowDefinitions>
```

```
        <Label Grid.Column="0" Grid.Row="0" Content="Numer"
        Margin="0,8,0,8"/>
        <TextBox Grid.Column="1" Grid.Row="0"
                                            Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" Text="{Binding
Path=Numer, UpdateSourceTrigger=PropertyChanged}"/>


        <Label Grid.Column="0" Grid.Row="1"
Content="DataWystawienia" Margin="0,8,0,8"/>
        <DatePicker Grid.Column="1" Grid.Row="1"
                                              Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" SelectedDate="{Binding
Path=DataWystawienia, UpdateSourceTrigger=PropertyChanged}"/>

        <Label Grid.Column="0" Grid.Row="2" Content="Kontrahent"
        Margin="0,8,0,8"/>
        <Label Grid.Column="1" Grid.Row="2" Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" Background="White"
        Content="{Binding Path=KontrahentDane,
UpdateSourceTrigger=PropertyChanged}" />
        <Button Grid.Column="1" Grid.Row="2"Margin="120,8,0,8"
Width="50" HorizontalAlignment="Left" Content="..."
Command="{Binding ShowKontrahenci}"/>

        <Label Grid.Column="0" Grid.Row="3"
Content="TerminPlatnosci" Margin="0,8,0,8"/>
        <DatePicker Grid.Column="1" Grid.Row="3"
                                              Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" SelectedDate="{Binding
Path=TerminPlatnosci, UpdateSourceTrigger=PropertyChanged}" />
```

```
        <Label Grid.Column="0" Grid.Row="4" Content="Sposob
Platnosci" Margin="0,8,0,8"/>
        <ComboBox Grid.Column="1" Grid.Row="4" Margin="0,8,0,8"
Width="120" HorizontalAlignment="Left" ItemsSource="{Binding
SposobyPlatnosciComboBoxItems}" DisplayMemberPath="nazwa"
SelectedValuePath="id" SelectedValue="{Binding
Path=IdSposobuPlatnosci,Mode=TwoWay}"/>

    </Grid>
</local:JedenViewBase>
```

2. Edytuj plik ***NowaFakturaViewModel*** (katalog ***ViewModel***). Zmodyfikuj kod klasy ***NowaFakturaViewModel*** według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using MVVMFirma.Model.Entities;
using
MVVMFirma.Model.EntitiesForView;
using MVVMFirma.Helper;
using
GalaSoft.MvvmLight.Messaging;
using System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    class NowaFakturaViewModel : JedenViewModel<Faktury>
    {
        #region Fields
        private BaseCommand _ShowKontrahenci;
        #endregion //Fields

        #region Constructor
        public NowaFakturaViewModel()
            : base()
        {
            base.DisplayName =
            "Faktura"; item = new
            Faktury();
            Messenger.Default.Register<KontrahenciForAllView>(this,
getWybranyKontrahent);
        }
        #endregion // Constructor
```

```
        #region Command
        public ICommand ShowKontrahenci
        {
            get
            {
                if (_ShowKontrahenci == null)
                {
                    _ShowKontrahenci = new
BaseCommand(() =>
Messenger.Default.Send("Kontrahenci.Show"));
```

```
            return
                _ShowKontrahenci;
            }
        }

        #region
        Properties
        public string
        Numer
        {
            ge
            t
            {  return item.numer;

            }
            se
            t  if (value ==
            {      item.numer)
                   return;
               item.numer = value;
               base.OnPropertyChanged(() =>
               Numer);
            }
        }


        public DateTime? DataWystawienia
        {
            ge
            t
            {  return item.dataWystawienia;

            }
            se
            t  if (value ==
            {      item.dataWystawienia)
                   return;
               item.dataWystawienia = value;
               base.OnPropertyChanged(() =>
               DataWystawienia);
            }
        }

        private string
        _KontrahentDane; public
        string KontrahentDane
        {
            ge            {
            t
```

```
    }
    se
    t  return _KontrahentDane;
    {


       if (value ==
          _KontrahentDane)
    }     return;
}     _KontrahentDane = value;
      base.OnPropertyChanged(() =>
      _KontrahentDane);

public int? IdKontrahenta
```

```
{
    get
    {  return item.idKontrahenta;

    }
    set
    {  if (value ==
          item.idKontrahenta) return;
       item.idKontrahenta = value;
       base.OnPropertyChanged(() =>
       IdKontrahenta);

    }
}

public DateTime? TerminPlatnosci
{
    get
    {  return item.terminPlatnosci;

    }
    set
    {  if (value ==
          item.terminPlatnosci)
          return;
       item.terminPlatnosci = value;
       base.OnPropertyChanged(() =>
       TerminPlatnosci);
    }
}

public int? IdSposobuPlatnosci
{
    get
    {  return item.idSposobuPlatnosci;

    }
    set
    {  if (value ==
          item.idSposobuPlatnosci)
          return;
       item.idSposobuPlatnosci = value;
       base.OnPropertyChanged(() =>
       IdSposobuPlatnosci);
    }
}
```

```
public IQueryable<SposobyPlatnosci> SposobyPlatnosciComboBoxItems
{
    ge
    t
    {  return
           from sposobPlatnosci in
           fakturyEntities.SposobyPlatnosci select
           sposobPlatnosci;
    }
}

public IQueryable<ComboBoxKeyAndValue> KontrahenciComboBoxItems
{
    get
```

```
            {
                return
                    from kontrahent in
                    fakturyEntities.Kontrahenci select
                    new ComboBoxKeyAndValue
                    {
                        Key = kontrahent.id,
                        Value = kontrahent.kod + " " + kontrahent.nazwa +
kontrahent.             " " +
nip
                    };
            }
        }


        #endregion //Properties

        #region Helpers
        private void getWybranyKontrahent(KontrahenciForAllView
kontrahent)
        {
            IdKontrahenta = kontrahent.Id;
            KontrahentDane = kontrahent.Kod + " " +
kontrahent.Nazwa + " " + kontrahent.Nip;

        }
        #endregion //Helpers

    }
}
```

3. W klasie ***MainWindowViewModel*** uzupełnij funkcję ***open(…)*** według wzoru:

```
private void open(string name)
        {
            if (name == "Wszystkie faktury Add")
                CreateFaktura();

            if (name == "Wszystkie towary Add")
                CreateTowar();

            if (name == "Kontrahenci Show")
                ShowAllKontrahenci();
        }
```

4. Edytuj plik **WszyscyKontrahenciView** (katalog ***Views***). Zmodyfikuj kod tego widoku według wzoru:

```
<local:WszystkieViewBase
            x:Class="MVVMFirma.Views.WszyscyKontrahenciView"
```

```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:local="clr-namespace:MVVMFirma.Views"
        mc:Ignorable="d"
        d:DesignHeight="300" d:DesignWidth="300">
<Grid>
    <Grid>
```

```
            <DataGrid AutoGenerateColumns="True"
ItemsSource="{Binding List}" SelectedItem="{Binding
Path=WybranyKontrahent, Mode=TwoWay}"/>

        </Grid>

    </Grid>
</local:WszystkieViewBase>
```

5. Edytuj plik ***WszyscyKontrahenciViewModel*** (katalog ***ViewModels***). Dodaj do klasy ***WszyscyKontrahenciViewModel*** region ***Properties*** według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.EntitiesForView;
using
System.Collections.ObjectModel;
using
GalaSoft.MvvmLight.Messaging;

namespace MVVMFirma.ViewModels
{
    class WszyscyKontrahenciViewModel :
    WszystkieViewModel<KontrahenciForAllView>
    {
        #region Constructor
        public WszyscyKontrahenciViewModel()
            : base()
        {
            base.DisplayName = "Wszyscy Kontrahenci";
        }
        #endregion // Constructor

        #region
        Properties
        private KontrahenciForAllView
        _WybranyKontrahent;
        public KontrahenciForAllView WybranyKontrahent
```

```
        {
            get
            {
                return _WybranyKontrahent;
            }
            set
            {
                if (_WybranyKontrahent !=
                value)
                {
                    _WybranyKontrahent =
                    value;
                    ShowMessageBoxInformation("Kontrahent: "
                    +
_WybranyKontrahent.Nazwa + " " + _WybranyKontrahent.Kod);
                    Messenger.Default.Send(_WybranyKontrahent)
                    ;
                    OnRequestClose();
                }
            }
        }


        #endregion // Properties

        #region Helpers
        public override void load()
        {
            List = new
                ObservableCollection<KontrahenciForAllVie
                w> (
                    from kontrahent in
                    fakturyEntities.Kontrahenci select
                    new KontrahenciForAllView
                    {
                        Id = kontrahent.id,
                        Kod=kontrahent.kod,
                        Nip=kontrahent.nip,
                        Nazwa=kontrahent.nazwa,
                        RodzajNazwa=kontrahent.Rodzaje.
                        nazwa,
                        StatusNazwa=kontrahent.Statusy
                        .nazwa
                    }
                );
        }
        #endregion //Helpers
    }
}
```

Zadanie do samodzielnego wykonania

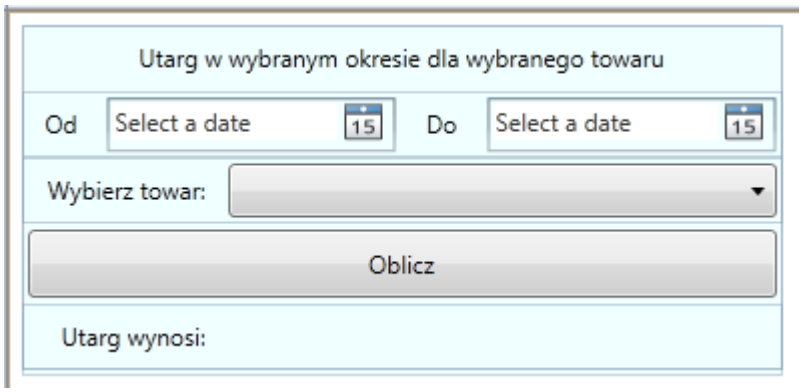
Wykonaj ***UserControl*** do dodawania ***pozycji Faktury*** (wypełniaj pole ***idTowaru*** przez przycisk z trzema kropkami). Odpowiednio nazwij klasy i widoki: ***NowaPozycjaFakturyViewModel***, ***NowaPozycjaFakturyView***. Ponadto uzupełnij klasę ***MainWindowViewModel*** o funkcję

```
void CreatePozycjeFaktury()
        {
            NowaPozycjaFakturyViewModel workspace = new
            NowaPozycjaFakturyViewModel(); this.Workspaces.Add(workspace);
            this.SetActiveWorkspace(workspace);
        }
```

oraz dopisz do funkcji ***open(string name)***, która znajduje się w tej klasie, następujący kod:

```
if (name == "Pozycje Faktury Add")
                CreatePozycjeFaktury();
```

Logika Biznesowa



1. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***Views*** i wybierz ***Add->User Control***. ***User Control*** nazwij ***RaportSprzedazyView*** i utwórz jej kod według wzoru:

```
<UserControl x:Class="MVVMFirma.Views.RaportSprzedazyView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pr
             esentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
             compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/
             2008" mc:Ignorable="d"
             d:DesignHeight="340" d:DesignWidth="400">
```

```
    <Grid>
        <StackPanel Margin="8" Background="Azure"  Height="175"
Width="380" VerticalAlignment="Top"
HorizontalAlignment="Left" >
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="33"/>
                    <RowDefinition Height="33"/>
                    <RowDefinition Height="33"/>
                    <RowDefinition Height="40"/>
                    <RowDefinition Height="33"/>
                </Grid.RowDefinitions>
                <Label Grid.Row="0" Content="Utarg w wybranym okresie dla
wybranego towaru" Margin="3" HorizontalContentAlignment="Center"/>
                <Grid Grid.Row="1">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="40"/>
                        <ColumnDefinition Width="*"/>
                        <ColumnDefinition Width="40"/>
                        <ColumnDefinition Width="*"/>
                    </Grid.ColumnDefinitions>
                    <Label Grid.Column="0" Content="Od" Margin="3"
HorizontalContentAlignment="Center"/>
                    <DatePicker Grid.Column="1" Margin="3"
                    SelectedDate="{Binding
DataOd}"/>
                    <Label Grid.Column="2" Content="Do" Margin="3"
HorizontalContentAlignment="Center"/>
```

```
                <DatePicker Grid.Column="3" Margin="3"
DataDo}          SelectedDate="{Binding
"/>
            </Grid>
            <Grid Grid.Row="2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="100"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="Wybierz towar:"
                Margin="3"
HorizontalContentAlignment="Right" />
                <ComboBox Grid.Column="1"   Margin="3"
ItemsSource="{Binding TowaryComboBoxItems}"
DisplayMemberPath="Value" SelectedValuePath="Key"
SelectedValue="{Binding Path=IdTowaru,Mode=TwoWay}"/>
            </Grid>
            <Button Grid.Row="3" Margin="3" Content="Oblicz"
Command="{Binding ObliczCommand}"/>
            <Grid Grid.Row="4">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="100"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="Utarg wynosi:"
Margin="3" HorizontalContentAlignment="Right" />
                <Label Grid.Column="1"
                                    Margin="3"
HorizontalContentAlignment="Left"
Content="{Binding Utarg}"          />
            </Grid>
        </Grid>
    </StackPanel>

    </Grid>
</UserControl>
```

2. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***Model***, a następnie wybierz ***Add->New Folder***. Folder nazwij ***BusinessLogic***.
3. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***BusinessLogic*** i wybierz ***Add->Class***. Klasę nazwij ***DatabaseClass*** i utwórz jej kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using MVVMFirma.Model.Entities;

namespace MVVMFirma.Model.BusinessLogic
{
    public class DatabaseClass
    {
        #region Fields
        protected FakturyEntities fakturyEntities;
```

```
        #endregion //Fields

        #region Constructor
        public DatabaseClass(FakturyEntities fakturyEntities)
        {
            this.fakturyEntities = fakturyEntities;
        }
        #endregion //Constructor
    }
}
```

4. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***BusinessLogic*** i wybierz ***Add->Class***. Klasę nazwij ***TowarB*** i utwórz jej kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.EntitiesForView;
using MVVMFirma.Model.Entities;

namespace MVVMFirma.Model.BusinessLogic
{
    public class TowarB:DatabaseClass
    {
        public TowarB(FakturyEntities fakturyEntities)
        :base(fakturyEntities)
        {
        }
        public IQueryable<ComboBoxKeyAndValue> GetTowaryComboBoxItems()
        {
            return (
                    from towar in
                    fakturyEntities.Towary select new
                    ComboBoxKeyAndValue
                    {
                        Key = towar.id,
                        Value = towar.nazwa + " " + towar.kod
                    }).ToList().AsQueryable();
        }
    }
}
```

5. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder

BusinessLogic i wybierz ***Add->Class***. Klasę nazwij ***UtargB*** i utwórz jej kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using MVVMFirma.Model.Entities;

namespace MVVMFirma.Model.BusinessLogic
{
    public class UtargB:DatabaseClass
    {
        public UtargB(FakturyEntities fakturyEntities)
        :base(fakturyEntities)
        {
        }
        public double? UtargOkresTowar(int idTowaru, DateTime odDaty,
        DateTime doDaty)
        {
            return
                (
                from pozycja in
                fakturyEntities.PozycjeFaktury where
                    pozycja.idTowaru == idTowaru &&
                    pozycja.Faktury.dataWystawienia >= odDaty &&
                    pozycja.Faktury.dataWystawienia <= doDaty
                select pozycja.cena * pozycja.ilosc
                ).Sum();

        }
    }
}
```

6. W okienku ***SolutionExplorer*** naciśnij prawym klawiszem myszki na folder ***ViewModel*** i wybierz ***Add->Class***. Klasę nazwij ***RaportSprzedazyViewModel*** i utwórz jej kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Helper;
using System.Windows.Input;
using
MVVMFirma.Model.EntitiesForView;
using MVVMFirma.Model.BusinessLogic;

namespace MVVMFirma.ViewModels
{
    public class RaportSprzedazyViewModel : DataBaseViewModel
    {
```

```
#region Fields and
Properties private
DateTime _DataOd; public
DateTime DataOd
{
    ge
    t
    {  return _DataOd;

    }
```

```
    se
    t
    {  if (value ==
           _DataOd)
           return;
       _DataOd = value;
       OnPropertyChanged(() =>
    }  DataOd);
}

private DateTime
_DataDo; public
DateTime DataDo
{
    ge
    t
    {  return _DataDo;

    }
    se
    t  if (value ==
    {      _DataDo)
           return;
       _DataDo = value;
       OnPropertyChanged(() =>
       DataDo);
    }
}

private int
_IdTowaru; public
int IdTowaru
{
    ge
    t
    {  return _IdTowaru;

    }
    se
    t  if (value ==
    {      _IdTowaru)
           return;
       _IdTowaru = value;
       OnPropertyChanged(() =>
       IdTowaru);
    }
}

public IQueryable<ComboBoxKeyAndValue> TowaryComboBoxItems
```

```
{
    ge
    t
    {  return new
       TowarB(fakturyEntities).GetTowaryComboBoxItems();
    }
}

private double?
_Utarg; public
double? Utarg
{
    ge
    t
    {  return _Utarg;

    }
    se
    t
```

```
        {
            if (value ==
                _Utarg)
                return;
            _Utarg = value;
            OnPropertyChanged(() =>
            Utarg);
        }
    }

    #endregion // Fields and Properties

    #region Constructor
    public RaportSprzedazyViewModel()
        :base()
    {
        base.DisplayName = "Raport
        Sprzedaży"; DataOd =
        DateTime.Now;
        DataDo =
        DateTime.Now;
        Utarg = 0;
    }
    #endregion // Constructor

    #region Command
    private BaseCommand
    _ObliczCommand; public ICommand
    ObliczCommand
    {
        ge
        t
        {  if (_ObliczCommand == null)
            {
                _ObliczCommand = new BaseCommand(() =>
                obliczUtargClick());
            }
            return _ObliczCommand;
        }
    }
    #endregion
    //Command #region
    Private Helpers

    private void obliczUtargClick()
    {
        Utarg = new
UtargB(fakturyEntities).UtargOkresTowar(IdTowaru,DataOd,DataDo);
    }
```

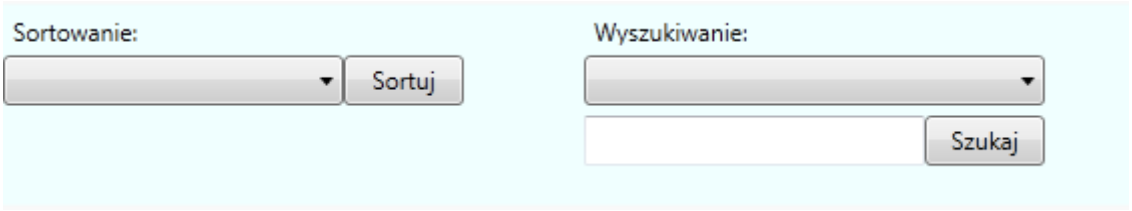
```
        #endregion //Private Helpers
    }
}
```

Zadanie do samodzielnego wykonania

Podłącz wykonany właśnie widok ***RaportSprzedazyView*** do okna głównego. W pliku ***MainWindowResources*** dokonaj skojarzenia ***RaportSprzedazyView*** z ***RaportSprzedazyViewModel***.

Sortowanie i Filtrowanie

1. Edytuj plik ***Generic.xaml*** (folder ***Themes***) i zmodyfikuj jego sekcję `<ControlTemplate TargetType="{x:Type local:WszystkieViewBase}">` **według wzoru:**




```
<ControlTemplate TargetType="{x:Type local:WszystkieViewBase}">
                    <Grid Margin="0,10,0,0">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="34"/>
                            <!--Tu-->
                            <RowDefinition Height="100"/>
                            <RowDefinition Height="*"/>
                        </Grid.RowDefinitions>
                        <ToolBar Grid.Row="0" Height="30"
                        Margin="0,2,0,2">
                       <ToggleButton Content="Dodaj" Width="70"
Height="30" Command="{Binding AddCommand}"/>
                            <ToggleButton Content="Modyfikuj" Width="70"
Height="30"
/>
                            <ToggleButton Content="Kasuj" Width="70"
                            Height="30"/>
                       <ToggleButton Content="Odswiez" Width="70"
                       Height="30"
Command="{Binding LoadCommand}"/>
                    </ToolBar>
```

```
                    <StackPanel Grid.Row="1"
                    Background="Azure">
                        <Grid>
                            <Grid.RowDefinitions>
                                <RowDefinition
                                Height="25"/>
                    <!--Tu-->

                            </Grid.RowDefinitions>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="230"/>
                                <ColumnDefinition Width="60"/>
                                <ColumnDefinition Width="230"/>
                            </Grid.ColumnDefinitions>
                            <Label Grid.Row="0"
Grid.Column="0" Content="Sortowanie: "/>
                            <ComboBox Grid.Row="1"
Grid.Column="0" ItemsSource="{Binding
SortComboboxItems}" SelectedValue="{Binding
Path=SortField,Mode=TwoWay}" Margin="0,0,60,0"/>
                            <Button Content="Sortuj" Grid.Row="1"
Grid.Column="0" Command="{Binding SortCommand}"
                            Margin="170,0,0,0"/>
```

```
                                <Label Grid.Row="0"
Grid.Column="2" Content="Wyszukiwanie: "/>
                                <ComboBox Grid.Row="1"
Grid.Column="2" ItemsSource="{Binding FindComboboxItems}"
SelectedValue="{Binding Path=FindField,Mode=TwoWay}" />
                                <TextBox Grid.Row="2"
Grid.Column="2" Margin="0,5,60,0" Text="{Binding
FindTextBox}"/>
                                <Button Content="Szukaj" Grid.Row="3"
Grid.Column="2" Command="{Binding FindCommand}" Margin="170,5,0,0"/>


                            </Grid>
                        </StackPanel>
                        <ContentPresenter Grid.Row="2" Margin="0,5,0,5" />
                    </Grid>

                </ControlTemplate>
```

2. Edytuj plik ***WszystkieViewModel*** (folder ***ViewModel***) i zmodyfikuj jego kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using
System.Collections.ObjectModel;
using System.Windows.Input;
using GalaSoft.MvvmLight.Messaging;

namespace MVVMFirma.ViewModels
{
    public abstract class WszystkieViewModel<T> : DataBaseViewModel
    {
        #region Fields
        private BaseCommand
        _LoadCommand; private
        BaseCommand _AddCommand;
        //tu
        private BaseCommand
         SortCommand; private
        private ObservableCollection<T> _List;
        #endregion //
        Fields #region
```

```
Properties
//public ICommand LoadCommand
//{
//  get
//  {
//      if (_LoadCommand == null)
//      {
//          _LoadCommand = new BaseCommand(() => load());
//      }
```

```
//      return _LoadCommand;
//   }
//}
//lub analogicznie jak
wyzej public ICommand
LoadCommand
{
   ge
   t
   {  return GetCommand(_LoadCommand,load);

   }
}

public ICommand AddCommand
{
   ge
   t
   {  if (_AddCommand == null)
      {
         _AddCommand = new BaseCommand(() => Add());
      }
      return _AddCommand;
   }
}

public ObservableCollection<T> List
{
   ge
   t
   {  if (_List ==
         null)
         load();
      return _List;
   }
   se
   t
   {  _List = value;
      OnPropertyChanged(() =>
      List);
   }
}
```

```
public string SortField { get; set; } public List<string> SortComboboxItems
{
    get
    {
        return getComboboxSortList();
    }
}
public ICommand SortCommand
{
    get
    {
        if (_SortCommand == null)
        {
```

```
            return _SortCommand;
        }
    }
    public string FindField { get; set;
    } public string FindTextBox { get;
    set; } public List<string>
    FindComboboxItems
    {
        get
        {
            return getComboboxFindList();
        }
    }
    public ICommand FindCommand
    {
        get
        {
            if (_FindCommand == null)
            {
                _FindCommand = new BaseCommand(() =>
                find());
            }
    #endregion //Properties

    #region Constructor
    public WszystkieViewModel()
        : base()
    {
    }

    #endregion //
    Constructor #region
    Helpers
    //tu
    public abstract void sort();
    public abstract List<String> getComboboxSortList();

    public abstract void find();
    public abstract List<String> getComboboxFindList();

    public abstract void
    load(); private void
    Add()
    {
        Messenger.Default.Send(DisplayName + " Add");
    }
    #endregion
}

}
```

3. Edytuj plik ***WszystkieTowaryViewModel*** (folder ***ViewModel***) i zmodyfikuj jego kod według wzoru:

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using
System.Collections.ObjectModel;
using MVVMFirma.Model.Entities;
using MVVMFirma.Helper;
using
System.Windows.Input;

namespace MVVMFirma.ViewModels
{
    class WszystkieTowaryViewModel : WszystkieViewModel<Towary>
    {

        #region Constructor
        public WszystkieTowaryViewModel()
            : base()
        {
            base.DisplayName = "Wszystkie towary";
        }
        #endregion //
        Constructor #region
        Helpers
        public override List<string> getComboboxSortList()
        {
            return new List<string> { "nazwa", "cena", "kod" };
        }
        public override void sort()
        {

            if (SortField == "nazwa")
                List = new
ObservableCollection<Towary>(List.OrderBy(item =>
item.nazwa));
            if (SortField == "cena")
                List = new
ObservableCollection<Towary>(List.OrderBy(item => item.cena));
            if (SortField == "kod")
                List = new
ObservableCollection<Towary>(List.OrderBy(item => item.kod));

        }
        public override List<string> getComboboxFindList()
        {
            return new List<string> { "nazwa", "kod" };
        }
```

```
        public override void find()
        {
            if (FindField == "nazwa")
                List = new ObservableCollection<Towary>(List.Where(item =>
item.nazwa
!= null && item.nazwa.StartsWith(FindTextBox)));
            if (FindField == "kod")
                List = new ObservableCollection<Towary>(List.Where(item
=> item.kod != null && item.kod.StartsWith(FindTextBox)));
        }


        public override void load()
        {
            List = new
              ObservableCollection<Towary> (
                from towar in
                fakturyEntities.Towary select
                towar
              );
        }
        #endregion
    }

}
```

4. Podobnie zmodyfikuj kod klas ***WszystkieFakuryViewModel*** i ***WszyscyKontrahenciViewModel***.

Przykładowe zapytania linq-u:

```
using MojeLinq.Model;


namespace MojeLinq

{

    public partial class Form1 : Form

    {

        public Form1()

        {

            InitializeComponent();

        }


        private void button1_Click(object sender, EventArgs e)
```

```
        {
            //ten obiekt daje dostep do bazy
            LinqEntities db=new LinqEntities();
            //pobieramy wszystkie kolumny pracownika
            dataGridView1.DataSource=db.Pracowniks.ToList();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //tylko wybrane kolumny
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    select new
                    {
                        pracownik.Imie,
                        pracownik.Nazwisko,
                        pracownik.Stanowisko
                    }
                ).ToList();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //tylko wybrane kolumny II wersja
            dataGridView1.DataSource =
                db.Pracowniks.Select(pracownik=>
                new {pracownik.Imie, pracownik.Nazwisko,
pracownik.Stanowisko} ).ToList();
```

```
        }

        private void button3_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //tylko wybrane rekordy
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    where pracownik.PlacaNetto>=7000
                    select new
                    {
                        pracownik.Imie,
                        pracownik.Nazwisko,
                        pracownik.PlacaNetto
                    }
                ).ToList();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //tylko wybrane rekordy
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    where pracownik.PlacaNetto >= 7000 &&
pracownik.PlacaNetto <= 10000
                    select new
                    {
                        pracownik.Imie,
```

```
                        pracownik.Nazwisko,
                        pracownik.PlacaNetto
                    }
                ).ToList();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            decimal a =Convert.ToDecimal(textBox1.Text); //do mina wsatwiamy
teskt z textBox1 zamieniony na double
            decimal b = Convert.ToDecimal(textBox2.Text);

            LinqEntities db = new LinqEntities();
            //tylko wybrane rekordy
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    where pracownik.PlacaNetto >= a && pracownik.PlacaNetto
<= b
                    select new
                    {
                        pracownik.Imie,
                        pracownik.Nazwisko,
                        pracownik.PlacaNetto
                    }
                ).ToList();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //tylko wybrane rekordy
```

```
            dataGridView1.DataSource = db.Pracowniks
                .Where(p=>p.PlacaNetto>=7000 &&
p.PlacaNetto<=10000).ToList();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            dataGridView1.DataSource = db.Pracowniks
                .Where(p => p.PlacaNetto >= 7000 && p.PlacaNetto <=
10000).Select(pracownik =>
                new { pracownik.Imie, pracownik.Nazwisko,
pracownik.PlacaNetto }).ToList();



        }

        private void button16_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //sortujemy po płacy netto, orderby - sortowanie
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    where pracownik.PlacaNetto >= 3000
                    orderby pracownik.PlacaNetto
                    select new
                    {
                        pracownik.Imie,
                        pracownik.Nazwisko,
                        pracownik.PlacaNetto
                    }
```

```
        ).ToList();
}

private void button15_Click(object sender, EventArgs e)
{
    LinqEntities db = new LinqEntities();
    //sortujemy po płacy netto malejąco
    dataGridView1.DataSource =
        (
            from pracownik in db.Pracowniks
            where pracownik.PlacaNetto >= 3000
            orderby pracownik.PlacaNetto descending
            select new
            {
                pracownik.Imie,
                pracownik.Nazwisko,
                pracownik.PlacaNetto
            }
        ).ToList();
}

private void button14_Click(object sender, EventArgs e)
{
    LinqEntities db = new LinqEntities();
    //sortujemy po płacy netto a w ramach tej samej placy po nazwisku
    dataGridView1.DataSource =
        (
            from pracownik in db.Pracowniks
            orderby pracownik.PlacaNetto, pracownik.Nazwisko
            select new
            {
```

```
                    pracownik.Imie,
                    pracownik.Nazwisko,
                    pracownik.PlacaNetto
                }
            ).ToList();
        }

        private void button13_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            dataGridView1.DataSource = db.Pracowniks
                .Where(p => p.PlacaNetto >= 5000 && p.PlacaNetto <= 30000)
                .OrderBy(p => p.PlacaNetto)
                .Select(pracownik =>
                new { pracownik.Imie, pracownik.Nazwisko,
pracownik.PlacaNetto }).ToList();
        }

        private void button12_Click(object sender, EventArgs e)
        {
            //wyswietlimy kazde satnowisko tylko raz
            LinqEntities db = new LinqEntities();
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    group pracownik by pracownik.Stanowisko
                ).ToList();
        }

        private void button11_Click(object sender, EventArgs e)
        {
```

```
            LinqEntities db = new LinqEntities();
            //wyswietlimy kazde satnowisko tylko raz tylko pry pomocy
distinct
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    select new
                    {
                        pracownik.Stanowisko
                    }
                ).Distinct().ToList();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            //dodajemy nową kolumnę obliczaną
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    select new
                    {
                        pracownik.Imie,
                        pracownik.Nazwisko,
                        pracownik.PlacaNetto,
                        Podatek= pracownik.PlacaNetto*20/100
                    }
                ).ToList();
        }

        private void button9_Click(object sender, EventArgs e)
```

```
{
    //zapytanie wyświetli sumę wszsytkich pensji pracowników
    LinqEntities db = new LinqEntities();
    label1.Text = "Suma wyplat: "+
        (
            from pracownik in db.Pracowniks
            select pracownik.PlacaNetto
        ).Sum().ToString();
}

private void button24_Click(object sender, EventArgs e)
{
    //zapytanie wyświetli minimalna place
    LinqEntities db = new LinqEntities();
    label1.Text = "Min wyplata: " +
        (
            from pracownik in db.Pracowniks
            select pracownik.PlacaNetto
        ).Min().ToString();
}

private void button23_Click(object sender, EventArgs e)
{
    //zapytanie wyświetli max place
    LinqEntities db = new LinqEntities();
    label1.Text = "Min wyplata: " +
        (
            from pracownik in db.Pracowniks
            select pracownik.PlacaNetto
        ).Max().ToString();
}
```

```
private void button22_Click(object sender, EventArgs e)
{
    //zapytanie wyświetli sumę wszsytkich pensji pracowników
    LinqEntities db = new LinqEntities();
    label1.Text = "Suma wyplat: " +
        (
            from pracownik in db.Pracowniks
            select new
            {
                pracownik.Imie,
                pracownik.Nazwisko,
                pracownik.PlacaNetto
            }
        ).Sum(p=>p.PlacaNetto).ToString();
}

private void button21_Click(object sender, EventArgs e)
{
    //zapytanie wyświetli srednią place
    LinqEntities db = new LinqEntities();
    label1.Text = "Srednia placa: " +
        (
            from pracownik in db.Pracowniks
            select pracownik.PlacaNetto

        ).Average().ToString();
}

private void button20_Click(object sender, EventArgs e)
{
```

```
    //zapytanie sprawdzi...
    LinqEntities db = new LinqEntities();
    var zapytanie1=
         (
            from pracownik in db.Pracowniks
            select pracownik.PlacaNetto

        ).Any(p => p >= 20000);
    if (zapytanie1 == true)
        label1.Text = "Jest płaca >=20000";
    else
        label1.Text = "Brak placy >=20000";

}

private void button19_Click(object sender, EventArgs e)
{
    //zapytanie sprawdzimy czy wszystkie place są wieksze od 7000
    LinqEntities db = new LinqEntities();
    var zapytanie1 =
         (
            from pracownik in db.Pracowniks
            select pracownik.PlacaNetto
        ).All(p => p >= 7000); //all - czy wszystkie
    if (zapytanie1 == true)
        label1.Text = "Wszystkie place >=7000";
    else
        label1.Text = "Jest placa <7000";
}

private void button18_Click(object sender, EventArgs e)
```

```
        {
            //pobieramy tylko 2 pierwsze rekordy
            LinqEntities db = new LinqEntities();
            //dodajemy nową kolumnę obliczaną
            dataGridView1.DataSource =
                (
                    from pracownik in db.Pracowniks
                    orderby pracownik.Nazwisko //sortowanie wzgledem nazwiska
                    select new
                    {
                        pracownik.Imie,
                        pracownik.Nazwisko,
                        pracownik.PlacaNetto,
                    }
                ).Take(2).ToList();
        }

        private void button17_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            label1.Text=db.Pracowniks.Average(p=>p.PlacaNetto).ToString();
        }

        private void button32_Click(object sender, EventArgs e)
        {
            //pobieramy pierwszego pracownika
            LinqEntities db = new LinqEntities();
            var pierwszy = db.Pracowniks.First();
            label1.Text ="Pierwszy pracownik: " +pierwszy.Imie + " " +
pierwszy.Nazwisko;
        }
```

```
        private void button31_Click(object sender, EventArgs e)
        {
            //pobieramy praownika o id ==1
            LinqEntities db = new LinqEntities();
            var pierwszy = db.Pracowniks.First(p=>p.IdPracownika==1);
            label1.Text = "Pracownik o id==1: " + pierwszy.Imie + " " +
pierwszy.Nazwisko;
        }

        private void button30_Click(object sender, EventArgs e)
        {
            //co zrobić jak takiego pracownika brak
            //warto zawsze FirstOrDefault
            //ta funkcja zwraca pracownika jak jest, a zwaraca null jak go
brak
            LinqEntities db = new LinqEntities();
            var pracownik = db.Pracowniks.FirstOrDefault(p => p.IdPracownika
== 100);
            if (pracownik == null)
                label1.Text = "Brak pracownika o Id==100";
            else
            label1.Text = "Pracownik o id==100: " + pracownik.Imie + " " +
pracownik.Nazwisko;
        }

        private void button29_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            var pracownik = db.Pracowniks.FirstOrDefault(p => p.PlacaNetto>=
20000);
            if (pracownik == null)
                label1.Text = "Brak pracownika o placy >=20000";
            else
```

```
        label1.Text = "Pracownik o placy >= 20 000: "
            + pracownik.Imie + " " + pracownik.Nazwisko;

}

private void button28_Click(object sender, EventArgs e)
{
    LinqEntities db = new LinqEntities();
    var pracownik =
        (
            from p in db.Pracowniks
            where p.PlacaNetto>=1000
            orderby p.PlacaNetto
            select p
        ).FirstOrDefault();
    if (pracownik == null)
        label1.Text = "Brak pracownika o placy >=20000";
    else
        label1.Text = "Pracownik o placy >= 20 000: "
            + pracownik.Imie + " " + pracownik.Nazwisko;
}

private void button27_Click(object sender, EventArgs e)
{
    //pobieramy tylko jeden rekord
    LinqEntities db = new LinqEntities();
    var pracownik = db.Pracowniks.Single(p => p.IdPracownika==1);
    label1.Text = "Pracownik o id==1: "
            + pracownik.Imie + " " + pracownik.Nazwisko;
}
```

```
private void button26_Click(object sender, EventArgs e)
{
    //zapytania zag....
    LinqEntities db = new LinqEntities();

    var zapytanie1 =
        (
           from pracownik in db.Pracowniks
           select pracownik
        );

    var zapytanie2=
        (
            from pracownik in zapytanie1
            where pracownik.PlacaNetto>=7000
            select pracownik
        );

    var zapytanie3 =
        (
            from pracownik in zapytanie2
            orderby pracownik.PlacaNetto
            select pracownik
        );

    var zapytanie4 =
       (
           from pracownik in zapytanie3
           select new
           {
               pracownik.Imie,
```

```
                pracownik.Nazwisko,
                pracownik.PlacaNetto
            }
        );

    dataGridView1.DataSource = zapytanie4.ToList();
}

private void button25_Click(object sender, EventArgs e)
{
    //Procedura dodawania rekordu/obiektu do kolekcji
    LinqEntities db = new LinqEntities();
    //1. Tworzymy pusty obiekt
    Pracownik nowy = new Pracownik();
    //2. Wypełniamy go danymi
    nowy.Imie = "Imie1";
    nowy.Nazwisko= "Nazwisko1";
    nowy.Stanowisko = "Stanowisko1";
    nowy.PlacaNetto = 1;
    //3. dodajemy go do lokalnej kolekcji
    db.Pracowniks.Add(nowy);
    //4. Wysłać zmiany do bazy danych - zapisać pracownika
    db.SaveChanges();

}

private void button40_Click(object sender, EventArgs e)
{
    //Procedura modyfikowania rekordu - obiektu
    LinqEntities db = new LinqEntities();
    //1. Szukamy rekordu/obiektu
```

```
            //var pracownik = db.Pracowniks.FirstOrDefault(p =>
p.IdPracownika == 6);
            //find dostaje id
            var pracownik = db.Pracowniks.Find(7);
            //2. Modyfikujemy wybrane dane rekordu
            pracownik.Nazwisko = "Nowe nazwisko";
            pracownik.PlacaNetto = 50000;
            //3. Zapisujemy zmiany do bazy danych
            db.SaveChanges();
        }

        private void button39_Click(object sender, EventArgs e)
        {
            //Procedura kasowania rekordu
            LinqEntities db = new LinqEntities();
            //1. Szukamy rkordu/obiektu do wykasowania
            var pracownik = db.Pracowniks.Find(7);
            //2. Kasujemy obiket
            db.Pracowniks.Remove(pracownik);
            //3. Zapisujemy zmiany w bazie
            db.SaveChanges();

        }

        private void button38_Click(object sender, EventArgs e)
        {
            //wysiwetlamy wszystkie płaty, ale tam jest klucz obcy
idPracownika
            LinqEntities db = new LinqEntities();
            dataGridView1.DataSource =
                (
                    from wyplata in db.Wyplata
```

```
                    select new
                    {
                        wyplata.IdWyplaty,
                        wyplata.IdPracownika,
                        wyplata.Kwota,
                        wyplata.DataWyplaty
                    }
                ).ToList();
            //ale wyswietlanie klucza obcego IdPracownka jest mało
funkcjonalne
        }

        private void button37_Click(object sender, EventArgs e)
        {
            LinqEntities db = new LinqEntities();
            dataGridView1.DataSource =
                (
                    from wyplata in db.Wyplata
                    select new
                    {
                        wyplata.IdWyplaty,
                        Imie=wyplata.IdPracownikaNavigation.Imie,
                        Nazwisko=wyplata.IdPracownikaNavigation.Nazwisko,
                        wyplata.Kwota,
                        wyplata.DataWyplaty
                    }
                ).ToList();
        }
    }
}
```