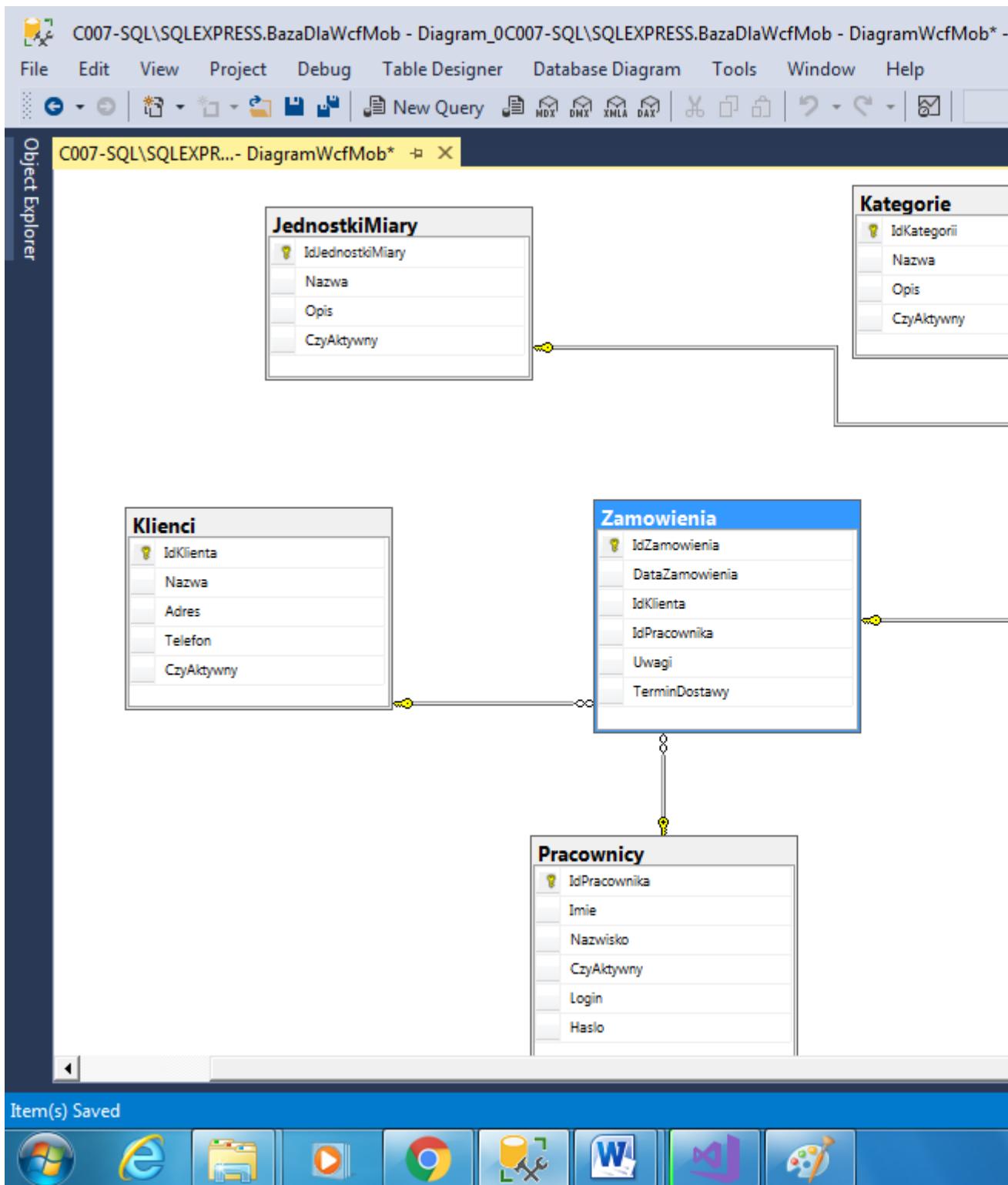


Creating application – WCF

Introduction

1. Open **MS Visual Studio**.
 2. **File->New->Project->Blanc Solution-> Name: SolutionOrders**
 3. **PPM SolutionOrders->Add->New->Project->Mobile App (Xamarin Forms)**.
Name your project: **AppMobileOrders (Flyout)**.
 4. In **MS SQL Server** Create database just like on image:



WCF

1. PPM **SolutionOrder** i Add->New Project->WCF->**WCF Service Application**.
Name WCF **WcfOrder**.
2. PPM **WcfOrder** i Add->New Folder->**Model**.

3. PPM Model i Add->New Folder->Entities.
 4. PPM Entities i Add->New Item-> (Left side) Data ->(right) ADO.NET Entity
- Data Model.** Name of model: **ModelOrder**.
5. Generate Model of created database – just like on desktop application
 6. Change name **IService1** to **IOrderService**.
 7. Change name **Service1** to **OrderService**.
 8. Edit **IOrderService** and change your code like on template:

```
[ServiceContract]
public interface IOrderService
{
    [OperationContract]
    List<ItemForView> GetItems();

    [OperationContract]
    List<ItemForView> GetItemSortByName();
}

[DataContract]
public class ItemForView
{
    [DataMember]
    public int IdItem { get; set; }
    [DataMember]
    public string Name { get; set; }
    [DataMember]
    public string Description { get; set; }
    [DataMember]
    public string CategoryName { get; set; }
    [DataMember]
    public decimal? Price { get; set; }
    [DataMember]
    public decimal? Quantity { get; set; }
    [DataMember]
    public string FotoUrl { get; set; }
    [DataMember]
    public string UnitOfMeasurementName { get; set; }
    [DataMember]
    public string Code { get; set; }
    public bool IsActive { get; set; }

    public ItemForView() { }
    public ItemForView(Item item)
    {
        IdItem = item.IdItem;
        Name = item.Name;
        Description = item.Description;
        CategoryName = item.Category.Name;
        Price = item.Price;
        Quantity = item.Quantity;
        FotoUrl = item.FotoUrl;
        UnitOfMeasurementName = item.UnitOfMeasurement.Name;
        Code = item.Code;
        IsActive = item.IsActive;
    }
}
```

9. Edit **OrderService**:

```
public class OrderService : IOrderService
{
    public List<ItemForView> GetItems()
    {
        var db = new OrderEntities();
        var query = from item in db.Item select item;

        return query.ToList()
            .Select(item => new ItemForView(item))
            .ToList();
    }

    public List<ItemForView> GetItemSortByName()
    {
        var db = new OrderEntities();
        var query = from item in db.Item select item;
        query = query.OrderBy(item => item.Name);

        return query.ToList()
            .Select(item => new ItemForView(item))
            .ToList();
    }
}
```

10. Edit with **XML(text) Editor** file **OrderService.svc** and change your code like in the example:

```
<%@ ServiceHost Language="C#" Debug="true" Service="WcfOrder.OrderService"
CodeBehind="OrderService.svc.cs" %>
```

11. With **Management Studio** add new records to database.

12. Select in solution explorer **OrderService.svc** and run the debug

Wprowadzenie do budowania interfejsów

Przykładowe aplikacje

1. Otwórz **MS Visual Studio**.
2. Wybierz rodzaj aplikacji **Mobile App (Xamarin.Forms)**. Filtrowanie przez C#, ALL platforms, Mobile.
3. Projekt nazwij: **InterfejsTest**.
4. Następnie wybierz **Blank**.
5. Zmodyfikuj kod **MainPage.xaml** i/lub **MainPage.xaml.cs** według wzoru:

Wersja 1

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
```

```
    <Label Text="Jakub Jaskulski"
        VerticalOptions="Center"
        HorizontalOptions="Center"
        TextColor="Red"
        BackgroundColor="Blue"
    />
```

```
</ContentPage>
```

Wersja 2

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
```

```
</ContentPage>
```

```
public partial class MainPage : ContentPage
{
    public MainPage()
    {
        Content = new Label
        {
            VerticalOptions = LayoutOptions.Center,
            Text = "Sterowanie z kodu"

        };
        Padding = new Thickness(50, 5, 5, 5);
    }
}
```

Wersja 3

```
public partial class MainPage : ContentPage
{
    public MainPage()
    {
        Content = new Label
        {
            Text = "Label 1",
            HorizontalOptions = LayoutOptions.Center,
            VerticalOptions = LayoutOptions.Center,
            BackgroundColor = Color.Red,
            TextColor = Color.Yellow
        };
    }
}
```

Wersja 4

```
public partial class MainPage : ContentPage
```

```

{
    public MainPage()
    {
        BackgroundColor = Color.Red;
        Content = new Frame
        {
            BackgroundColor = Color.Blue,
            HorizontalOptions = LayoutOptions.Center,
            VerticalOptions = LayoutOptions.Center,
            Content = new Label
            {
                Text = "Jakub Jaskulski",
                FontSize = Device.GetNamedSize(NamedSize.Large,
typeof(Label)),
                FontAttributes = FontAttributes.Italic,
                TextColor = Color.Green
            }
        };
    }
}

```

Wersja 5

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
    <Frame
        BackgroundColor="Yellow"
        HorizontalOptions="Center"
        VerticalOptions="Center"
    >
        <Frame.Content>
            <Label Text="Artur" />
        </Frame.Content>
    </Frame>
</ContentPage>

public partial class MainPage : ContentPage
{
    public MainPage()
    {
        InitializeComponent();
    }
}

```

Wersja 6

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
    <StackLayout>

```

```

<Frame BackgroundColor="Azure">
    <StackLayout Orientation="Horizontal">
        <BoxView Color="Red" />
        <Label Text="Red" VerticalOptions="Center"/>
    </StackLayout>

</Frame>
<Frame BackgroundColor="LightBlue">
    <StackLayout Orientation="Horizontal">
        <BoxView Color="Green" />
        <Label Text="Green" VerticalOptions="Center"/>
    </StackLayout>
</Frame>
<Frame BackgroundColor="LightGreen">
    <StackLayout Orientation="Horizontal">
        <BoxView Color="Blue" />
        <Label Text="Blue" VerticalOptions="Center"/>
    </StackLayout>
</Frame>
</StackLayout>

</ContentPage>
```

Wersja 7

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
    <Label VerticalOptions="Center">
        <Label.FormattedText>
            <FormattedString>
                <Span Text="To jest tekst " />
                <Span Text="bold" FontAttributes="Bold" />
                <Span Text=" i " />
                <Span Text="italic" FontAttributes="Italic" />
                <Span Text=" i " />
                <Span Text="duży" FontSize="Large" />
                <Span Text=" tekst." />
            </FormattedString>
        </Label.FormattedText>
    </Label>
</ContentPage>
```

Wersja 8

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
    <StackLayout>
        <Label
            x:Name="LabelCzas"
            FontSize="Large"
            HorizontalOptions="Center"
        />
        <Label
            x:Name="LabelData"
            FontSize="Large"
            HorizontalOptions="Center"
        />
    </StackLayout>
</ContentPage>
```

```

public partial class MainPage : ContentPage
{
    public MainPage()
    {
        InitializeComponent();
        Device.StartTimer(TimeSpan.FromSeconds(1), OnTimerTick);
    }
    bool OnTimerTick()
    {
        DateTime dt = DateTime.Now;
        LabelCzas.Text = dt.ToString("T");
        LabelData.Text = dt.ToString("D");
        return true;
    }
}

```

Wersja 9

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="InterfejsTest.MainPage">
    <OnPlatform x:TypeArguments="View">
        <OnPlatform.iOS>
            <Label
                Text="Witaj w IOS"
                HorizontalOptions="Center"
                VerticalOptions="Center"
            />
        </OnPlatform.iOS>
        <OnPlatform.Android>
            <Label
                Text="Witaj w Android"
                HorizontalOptions="Center"
                VerticalOptions="Center"
            />
        </OnPlatform.Android>
    </OnPlatform>
</ContentPage>

public partial class MainPage : ContentPage
{
    public MainPage()
    {
        InitializeComponent();
    }
}

```

Wersja 10

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

```

```

        x:Class="InterfejsTest.MainPage">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="100" />
        <RowDefinition Height="2*" />
        <RowDefinition Height="1*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label
        Text="Label 1"
        Grid.Row="0" Grid.Column="0"
        FontSize="Large"
        HorizontalOptions="End"
    />
    <Label
        Text="Label 2"
        Grid.Row="0" Grid.Column="1"
        FontSize="Small"
        HorizontalOptions="End"
        VerticalOptions="End" />
    <Label
        BackgroundColor="Gray"
        Grid.Row="1"
        Grid.Column="0"
        Grid.ColumnSpan="2">
    </Label>
    <BoxView
        Color="Green"
        Grid.Row="2"
        Grid.Column="0" />
    <BoxView
        Color="Red"
        Grid.Row="2"
        Grid.Column="1"
        Grid.RowSpan="2" />
    <BoxView
        Color="Blue"
        Opacity="0.5"
        Grid.Row="3"
        Grid.Column="0"
        Grid.ColumnSpan="2" />
</Grid>

</ContentPage>

```

Operacje na kolekcjach

1. Poniższe operacje będą wykonywane na projekcie **AppMobileOrders**.
2. PPM na **Models i Add->Class**. Klasę nazwij **Client**. Utwórz kod tej klasy według wzoru:

```
public class Client
{
    public int IdClient { get; set; }
    public string Name { get; set; }
    public string Adres { get; set; }
    public string PhoneNumber { get; set; }
}
```

3. Edytuj **Services->IDataStore** według wzoru:

```
public interface IDataStore<T>
{
    Task<bool> AddItemAsync(T item);
    Task<bool> UpdateItemAsync(T item);
    Task<bool> DeleteItemAsync(int id);
    Task<T> GetItemAsync(int id);
    Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh = false);
}
```

4. Zmień pozostałe klasy, tak aby używały **int Id** (string zamień na **int** przy polach id).

5. Edytuj **ViewModels->BaseViewModel** i przenieś (Cut) linię:

```
public           IDataStore<Item>           DataStore      =>
DependencyService.Get<IDataStore<Item>>();
do klasy ItemsViewModel.
```

Przekopiuj tę linię również do klasy NewItemViewModel oraz ItemDetailViewModel.

6. PPM na Services i Add->Class. Klasę nazwij **ClientDataStore**. Utwórz kod tej klasy według wzoru:

```
using AppMobileOrders.Models;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace AppMobileOrders.Services
{
    public class ClientDataStore : IDataStore<Client>
    {
        readonly List<Client> items;

        public ClientDataStore()
        {
            items = new List<Client>()
            {
                new Client { IdClient = 1, Name = "Client 1", Adres="New Sącz 1",PhoneNumber="1" },
                new Client { IdClient = 2, Name = "Client 2", Adres="New Sącz 2",PhoneNumber="2" },
                new Client { IdClient = 3, Name = "Client 3", Adres="New Sącz 3",PhoneNumber="3" }
            };
        }

        public async Task<bool> AddItemAsync(Client item)
        {
            items.Add(item);
            return await Task.FromResult(true);
        }

        public async Task<bool> UpdateItemAsync(Client item)
        {
            var oldItem = items.Where((Client arg) => arg.IdClient == item.IdClient).FirstOrDefault();
            items.Remove(oldItem);
```

```
        items.Add(item);
    return await Task.FromResult(true);
}

public async Task<bool> DeleteItemAsync(int id)
{
    var oldItem = items.Where((Client arg) => arg.IdClient == id).FirstOrDefault();
    items.Remove(oldItem);
    return await Task.FromResult(true);
}

public async Task<Client> GetItemAsync(int id)
{
    return await Task.FromResult(items.FirstOrDefault(s => s.IdClient == id));
}

public async Task<IEnumerable<Client>> GetItemsAsync(bool forceRefresh = false)
{
    return await Task.FromResult(items);
}
```

7. PPM na ViewModels i Add->Class. Klasę nazwij **ClientViewModel**. Utwórz kod tej klasy według wzoru:

```
using AppMobileOrders.Models;
using AppMobileOrders.Services;
using AppMobileOrders.Views;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace AppMobileOrders.ViewModels
{
    public class ClientViewModel : BaseViewModel
    {
        public IDataProvider<Client> DataStore =>
            DependencyService.Get<IDataProvider<Client>>();

        private Client _selectedItem;
        public ObservableCollection<Client> Items { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<Client> ItemTapped { get; }

        public ClientViewModel()
        {
            Title = "Client";
            Items = new ObservableCollection<Client>();
            LoadItemsCommand = new Command(async () => await ExecuteLoadItemsCommand());
            ItemTapped = new Command<Client>(OnItemSelected);
            AddItemCommand = new Command(OnAddItem);
        }

        async Task ExecuteLoadItemsCommand()
        {
```

```

IsBusy = true;
try
{
    Items.Clear();
    var items = await DataStore.GetItemsAsync(true);
    foreach (var item in items)
    {
        Items.Add(item);
    }
}
catch (Exception ex)
{
    Debug.WriteLine(ex);
}
finally
{
    IsBusy = false;
}
}

public void OnAppearing()
{
    IsBusy = true;
    SelectedItem = null;
}

public Client SelectedItem
{
    get => _selectedItem;
    set
    {
        SetProperty(ref _selectedItem, value);
        OnItemSelected(value);
    }
}

private async void OnAddItem(object obj)

```

```
{  
    // await Shell.Current.GoToAsync(nameof(NewClientPage));  
}  
  
async void OnItemSelected(Client item)  
{  
    if (item == null)  
        return;  
    await  
    Shell.Current.GoToAsync($"{nameof(ItemDetailPage)}?{nameof(ItemDetai  
IViewModel.ItemId)}={item.IdClient}");  
}  
}  
}
```

8. PPM na Views i Add->NewItem->ContentPage (bez C#) Widok nazwij

ClientPage. Utwórz kod tego widoku według wzoru:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AppMobileOrders.Views.ClientPage"
    Title="{Binding Title}"
    xmlns:local="clr-namespace:AppMobileOrders.ViewModels"
    xmlns:model="clr-namespace:AppMobileOrders.Models"
    x:Name="BrowseItemsPage"
    >
<ContentPage.ToolbarItems>
    <ToolbarItem Text="Add" Command="{Binding AddItemCommand}" />
</ContentPage.ToolbarItems>
<RefreshView x:DataType="local:ClientViewModel" Command="{Binding LoadItemsCommand}" IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
    <CollectionView x:Name="ItemsListView"
        ItemsSource="{Binding Items}"
       SelectionMode="None">
        <CollectionView.ItemTemplate>
            <DataTemplate>
                <StackLayout Padding="10" x:DataType="model:Client">
                    <Label Text="{Binding Name}"
                        Style="{DynamicResource ListItemTextStyle}"
                        FontSize="16" />
                    <Label Text="{Binding Adres}"
                        Style="{DynamicResource ListItemDetailTextStyle}"
                        FontSize="13" />
                    <Label Text="{Binding PhoneNumber}"
                        Style="{DynamicResource ListItemDetailTextStyle}"
                        FontSize="13" />
                <StackLayout.GestureRecognizers>
                    <TapGestureRecognizer
                        NumberOfTapsRequired="1"
                        Command="{Binding AncestorType={x:Type local:ClientViewModel}}, Path=ItemTapped"
                        CommandParameter="{Binding .}">
                
```

```
        </TapGestureRecognizer>
    </StackLayout.GestureRecognizers>
</StackLayout>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</RefreshView>
</ContentPage>
```

9. Edytuj **Views->KlienciPage.xaml->KlienciPage.xaml.cs** i zmień jej kod według wzoru:

```
using AppMobileOrders.ViewModels;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace AppMobileOrders.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ClientPage : ContentPage
    {
        ClientViewModel _viewModel;

        public ClientPage()
        {
            InitializeComponent();

            BindingContext = _viewModel = new ClientViewModel();
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            _viewModel.OnAppearing();
        }
    }
}
```

10. Edytuj plik **AppShell.xaml** i zmień jego sekcję **FlyoutItem** według wzoru:

```
<FlyoutItem FlyoutDisplayOptions="AsMultipleItems">
    <ShellContent Title="About" Icon="tab_about.png" Route="AboutPage"
ContentTemplate="{DataTemplate local:AboutPage}" />
    <ShellContent Title="Browse" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:ItemsPage}" />
    <ShellContent Title="Client" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:ClientPage}" />
</FlyoutItem>
```

11. Edytuj plik **App.xaml->App.xaml.cs** i zmień jego konstruktor według wzoru:

```
public partial class App : Application
{
    public App()
    {
        InitializeComponent();
        DependencyService.Register<MockDataStore>();
        DependencyService.Register<ClientDataStore>(); // Zmiana
        MainPage = new AppShell();
    }
}
```

12. PPM na **ViewModels i Add->Class**. Klasę nazwij **NewClientViewModel**. Utwórz kod tej klasy według wzoru:

```
using AppMobileOrders.Models;
using AppMobileOrders.Services;
using System;
using Xamarin.Forms;

namespace AppMobileOrders.ViewModels
{
    public class NewClientViewModel : BaseViewModel
    {
        public IDataStore<Client> DataStore =>
DependencyService.Get<IDataStore<Client>>();
        private int idClient;
        private string name;
        private string adres;
        private string phoneNumber;
        public NewClientViewModel()
        {
            SaveCommand = new Command(OnSave, ValidateSave);
            CancelCommand = new Command(OnCancel);
            this.PropertyChanged +=
                (_, __) => SaveCommand.ChangeCanExecute();
        }
        private bool ValidateSave()
        {
```

```

        return IdClient > 0
            && !String.IsNullOrWhiteSpace(Name);
    }
    public int IdClient
    {
        get => idClient;
        set => SetProperty(ref idClient, value);
    }
    public string Name
    {
        get => name;
        set => SetProperty(ref name, value);
    }
    public string Adres
    {
        get => adres;
        set => SetProperty(ref adres, value);
    }
    public string PhoneNumber
    {
        get => phoneNumber;
        set => SetProperty(ref phoneNumber, value);
    }
    public Command SaveCommand { get; }
    public Command CancelCommand { get; }
    private async void OnCancel()
    {
        await Shell.Current.GoToAsync("..");
    }
    private async void OnSave()
    {
        Client newItem = new Client()
        {
            IdClient = IdClient,
            Name = Name,
            Adres = adres,
            PhoneNumber = PhoneNumber
        };
        await DataStore.AddItemAsync(newItem);
        await Shell.Current.GoToAsync(..);
    }
}

```

13.PPM na Views i Add->NewItem->ContentPage (bez C#) Widok nazwij

NewClientPage. Utwórz kod tego widoku według wzoru:

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AppMobileOrders.Views.NewClientPage"
    Shell.PresentationMode="ModalAnimated"
    Title="New Client"
    >
    <ContentPage.Content>
        <StackLayout Spacing="3" Padding="15">

```

```

<Label Text="Id" FontSize="Medium" />
<Entry Text="{Binding IdClient, Mode=TwoWay}" FontSize="Medium" />
<Label Text="Name" FontSize="Medium" />
<Editor Text="{Binding Name, Mode=TwoWay}" FontSize="Medium"/>
<Label Text="Adres" FontSize="Medium" />
<Editor Text="{Binding Adres, Mode=TwoWay}" FontSize="Medium"/>
<Label Text="PhoneNumber" FontSize="Medium" />
<Editor Text="{Binding PhoneNumber, Mode=TwoWay}" FontSize="Medium"/>
<StackLayout Orientation="Horizontal">
    <Button Text="Anuluj" Command="{Binding CancelCommand}" HorizontalOptions="FillAndExpand"></Button>
    <Button Text="Zapisz" Command="{Binding SaveCommand}" HorizontalOptions="FillAndExpand"></Button>
</StackLayout>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

14. Edytuj **Views->NowyKlientPage.xaml->NewClientPage.xaml.cs** i zmień jej kod według wzoru:

```

using AppMobileOrders.Models;
using AppMobileOrders.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace AppMobileOrders.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class NewClientPage : ContentPage
    {
        public Client Item { get; set; }

        public NewClientPage()
        {
            InitializeComponent();
            BindingContext = new NewClientViewModel();
        }
    }

```

```
    }  
}
```

- 15.** Edytuj plik **AppShell.xaml->AppShell.xaml.cs** i zmień jego konstruktor według wzoru:

```
public partial class AppShell : Xamarin.Forms.Shell  
{  
    public AppShell()  
    {  
        InitializeComponent();  
        Routing.RegisterRoute(nameof(ItemDetailPage), typeof(ItemDetailPage));  
        Routing.RegisterRoute(nameof(NewItemPage), typeof(NewItemPage));  
        Routing.RegisterRoute(nameof(NewClientPage), typeof(NewClientPage));  
    }  
}
```

Abstrakcja

1. Poniższe operacje będą wykonywane na projekcie **AppMobileOrders**.
2. PPM na **Services** i **Add -> Class** o nazwie **ItemDataStore**. Utwórz kod tej klasy według wzoru:

```
using System.Collections.Generic;  
using System.Threading.Tasks;
```

```
namespace AppMobileOrders.Services  
{  
    public abstract class ItemDataStore<T> : IDataStore<T>  
    {  
        public List<T> items;  
  
        public ItemDataStore()  
        {  
        }  
        public async Task<bool> AddItemAsync(T item)  
        {  
            items.Add(item);  
        }  
    }  
}
```

```

        return await Task.FromResult(true);
    }

    public abstract T Find(T item);

    public abstract T Find(int id);

    public async Task<bool> UpdateItemAsync(T item)
    {
        var oldItem = Find(item);
        items.Remove(oldItem);
        items.Add(item);

        return await Task.FromResult(true);
    }

    public async Task<bool> DeleteItemAsync(int id)
    {
        var oldItem = Find(id);
        items.Remove(oldItem);

        return await Task.FromResult(true);
    }

    public async Task<T> GetItemAsync(int id)
    {
        return await Task.FromResult(Find(id));
    }

    public async Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh =
false)
    {
        return await Task.FromResult(items);
    }
}

```

3. Zmodyfikuj klasę **ClientDataStore** według wzoru:

```

public class ClientDataStore : ItemDataStore<Client>
{
    public ClientDataStore()

```

```

        : base()
    {
        items = new List<Client>()
        {
            new Client{IdClient=1,Name="Client 1", Adres="Zielona 1",
PhoneNumber="1"},
            new Client{IdClient=2,Name="Client 2", Adres="Zielona 2",
PhoneNumber="2"},
            new Client{IdClient=3,Name="Client 3", Adres="Zielona 3",
PhoneNumber="3"},
            new Client{IdClient=4,Name="Client 4", Adres="Zielona 4",
PhoneNumber="4"},
        };
    }
    public override Client Find(Client item)
    {
        return items.Where((Client arg) => arg.IdClient ==
item.IdClient).FirstOrDefault();
    }
    public override Client Find(int id)
    {
        return items.Where((Client arg) => arg.IdClient == id).FirstOrDefault();
    }
}

```

4. Zmodyfikuj klasę [MockDataStore](#) według wzoru:

```

public class MockDataStore : ItemDataStore<Item>
{
    public MockDataStore()
    {
        items = new List<Item>()
        {
            new Item { Id = 1, Text = "First item", Description="This is an item
description." },
            new Item { Id = 2, Text = "Second item", Description="This is an item
description." },
            new Item { Id = 3, Text = "Third item", Description="This is an item
description." },
            new Item { Id = 4, Text = "Fourth item", Description="This is an item
description." },
            new Item { Id = 5, Text = "Fifth item", Description="This is an item
description." },
        }
    }
}

```

```
        new Item { Id = 6, Text = "Sixth item", Description="This is an item
description." }
    };
}
public override Item Find(Item item)
{
    return items.Where((Item arg) => arg.Id == item.Id).FirstOrDefault();
}
public override Item Find(int id)
{
    return items.Where((Item arg) => arg.Id == id).FirstOrDefault();
}
}
```

5. PPM na **ViewModels** i Add -> Class o nazwie **AItemsViewModel**. Utwórz kod tej klasy według wzoru:

```
using AppMobileOrders.Services;
using System;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace AppMobileOrders.ViewModels
{
    public abstract class AItemsViewModel<T> : BaseViewModel
    {
        public IDataStore<T> DataStore => DependencyService.Get<IDataStore<T>>();
        private T _selectedItem;
        public ObservableCollection<T> Items { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<T> ItemTapped { get; }

        public AItemsViewModel(string title)
        {
            Title = title;
            Items = new ObservableCollection<T>();
            LoadItemsCommand = new Command(async () => await ExecuteLoadItemsCommand());
            ItemTapped = new Command<T>(OnItemSelected);
            AddItemCommand = new Command(OnAddItem);
        }

        async Task ExecuteLoadItemsCommand()
        {
            IsBusy = true;
            try
            {
                Items.Clear();
                var items = await DataStore.GetItemsAsync(true);
                foreach (var item in items)
                {
                    Items.Add(item);
                }
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex);
            }
            finally
            {
                IsBusy = false;
            }
        }
        public void OnAppearing()
        {
            IsBusy = true;
            SelectedItem = default(T);
        }

        public T SelectedItem
```

```
{  
    get => _selectedItem;  
    set  
    {  
        SetProperty(ref _selectedItem, value);  
        OnItemSelected(value);  
    }  
}  
public abstract void GoToAddPage();  
public async void OnAddItem(object obj)  
{  
    GoToAddPage();  
}  
  
async void OnItemSelected(T item)  
{  
    if (item == null)  
        return;  
    //await  
Shell.Current.GoToAsync($"{nameof(ClientDetailPage)}?{nameof(ClientDetailViewModel.Id)}={item.IdClient}");  
}  
}
```

6. Zmodyfikuj klasę **ClientViewModel** według wzoru:

```
public class ClientViewModel : AItemsViewModel<Client>
{
    public ClientViewModel()
        :base("Client")
    {
    }
    public override void GoToAddPage()
    {
        Shell.Current.GoToAsync(nameof(NewClientPage));
    }
}
```

7. Zmodyfikuj klasę **ItemsViewModel** według wzoru:

```
public class ItemsViewModel : AItemsViewModel<Item>
{
    public ItemsViewModel()
        : base("Items")
    {

    }
    public override void GoToAddPage()
    {
        Shell.Current.GoToAsync(nameof(NewItemPage));
    }
}
```

8. PPM na **ViewModels** i Add -> Class o nazwie **ANewItemViewModel**. Utwórz kod tej klasy według wzoru:

```
using AppMobileOrders.Services;
using Xamarin.Forms;

namespace AppMobileOrders.ViewModels
{
    public abstract class ANewItemViewModel<T> : BaseViewModel
    {
        public IDataStore<T> DataStore =>
            DependencyService.Get<IDataStore<T>>();

        public ANewItemViewModel()
        {
            SaveCommand = new Command(OnSave, ValidateSave);
            CancelCommand = new Command(OnCancel);
            this.PropertyChanged +=
                (_, __) => SaveCommand.ChangeCanExecute();
        }

        public abstract bool ValidateSave();
        public Command SaveCommand { get; }
        public Command CancelCommand { get; }
        private async void OnCancel()
        {
            // This will pop the current page off the navigation stack
            await Shell.Current.GoToAsync(..");
        }

        public abstract T SetItem();
        private async void OnSave()
        {
            await DataStore.AddItemAsync(SetItem());
            // This will pop the current page off the navigation stack
            await Shell.Current.GoToAsync(..");
        }
    }
}
```

9. Zmodyfikuj klasę **NewItemViewModel** według wzoru:

```
public class NewItemViewModel : ANewItemViewModel<Item>
{
    private string text;
    private string description;
    public NewItemViewModel()
        :base()
    {

    }

    public override bool ValidateSave()
    {
        return !String.IsNullOrWhiteSpace(text)
            && !String.IsNullOrWhiteSpace(description);
    }

    public string Text
    {
        get => text;
        set => SetProperty(ref text, value);
    }
    public string Description
    {
        get => description;
        set => SetProperty(ref description, value);
    }
    public override Item SetItem()
    {
        Item newItem = new Item()
        {
            Id = 1,
            Text = Text,
            Description = Description
        };
        return newItem;
    }
}
```

```
}
```

10. Zmodyfikuj klasę **NewClientViewModel** według wzoru:

```
public class NewClientViewModel : ANewItemViewModel<Client>
{
    private string name;
    private string adres;
    private string phoneNumber;
    public NewClientViewModel()
        : base()
    {
    }
    public override bool ValidateSave()
    {
        return !String.IsNullOrWhiteSpace(name);
    }
    public string Name
    {
        get => name;
        set => SetProperty(ref name, value);
    }
    public string Adres
    {
        get => adres;
        set => SetProperty(ref adres, value);
    }
    public string PhoneNumber
    {
        get => phoneNumber;
        set => SetProperty(ref phoneNumber, value);
    }
    public override Client SetItem()
    {
        Client newItem = new Client()
        {
            IdClient = 1,
            Name = Name,
            Adres = adres,
            PhoneNumber = PhoneNumber
        };
        return newItem;
    }
}
```

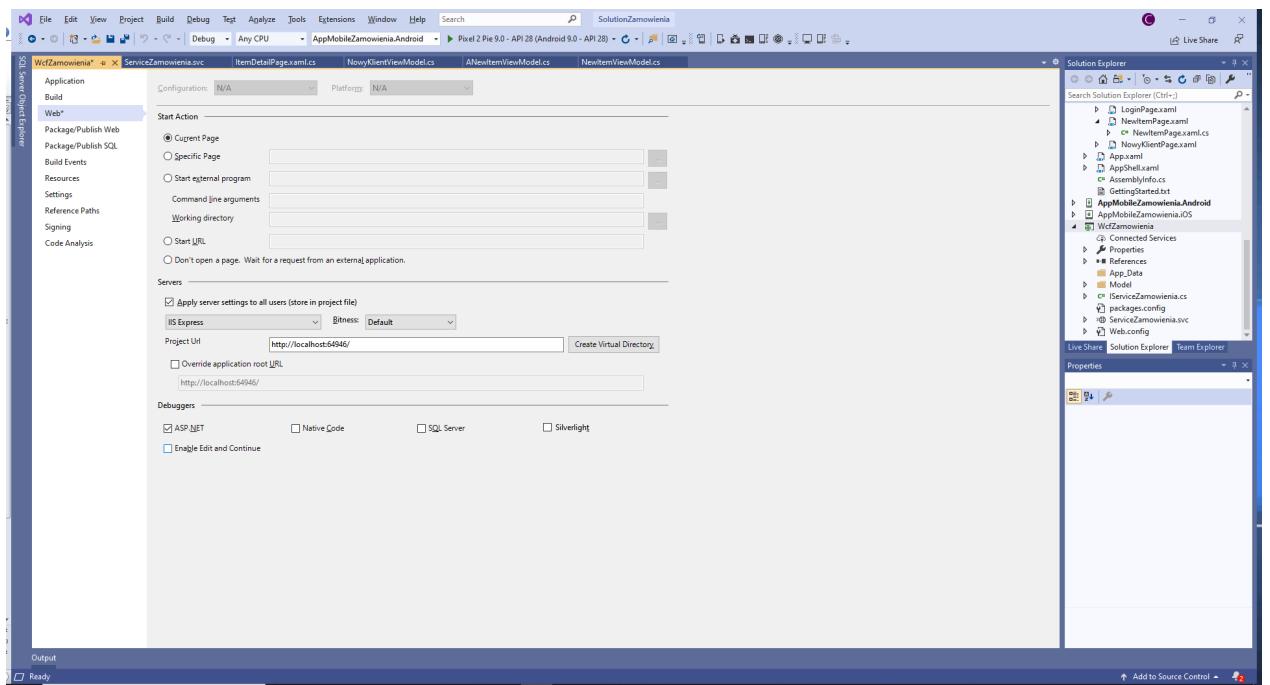
Łączenie aplikacji z WCF

- Uzupełnij wszystkie tabele bazy danych **Orders**, którą utworzyliśmy na pierwszych zajęciach.
- W projekcie **WCFOrder** naciśnij prawym klawiszem myszy na plik OrderService.svc i otwórz go za pomocą edytora HTML. Zmień jego zawartość według wzoru:

```
<%@ ServiceHost Language="C#" Debug="true" Service="WCFOrder" .OrderService" CodeBehind="OrderService.svc.cs" %>
```

Zapisz zmiany.

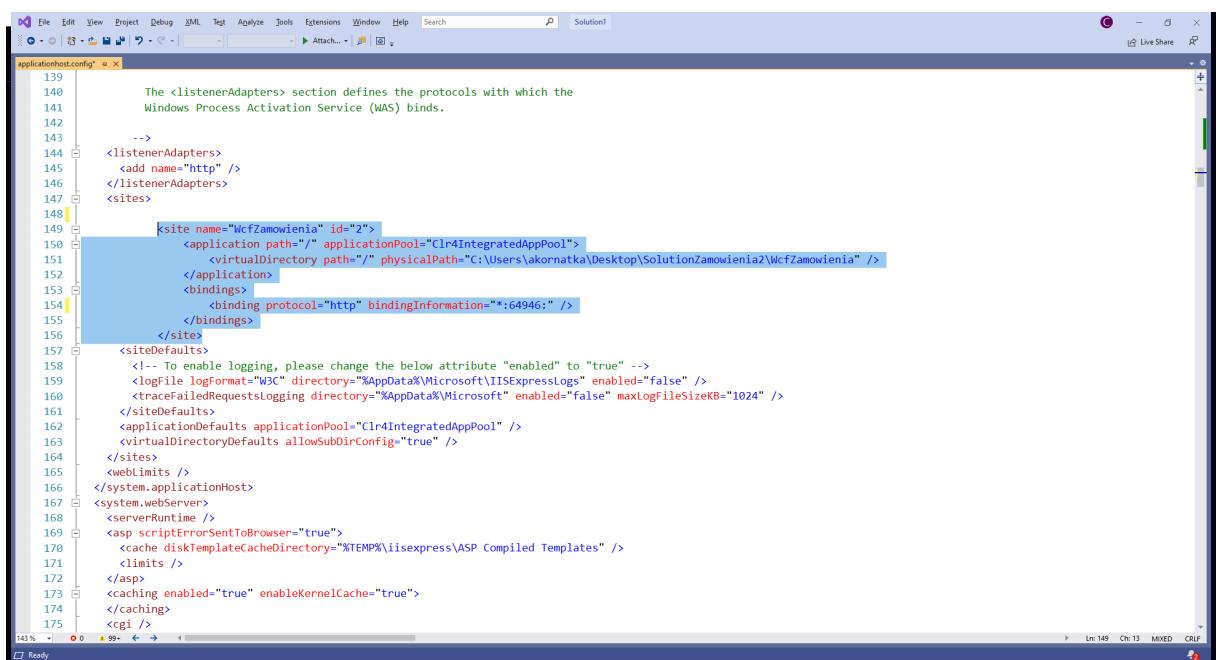
3. PPM na cały projekt **WCFOrders**->wybierz opcję **Properties**->wybierz zakładkę **Web**->odznacz opcję „**Enable Edit and Continue**”.



4. Wyłącz **Visual Studio**.

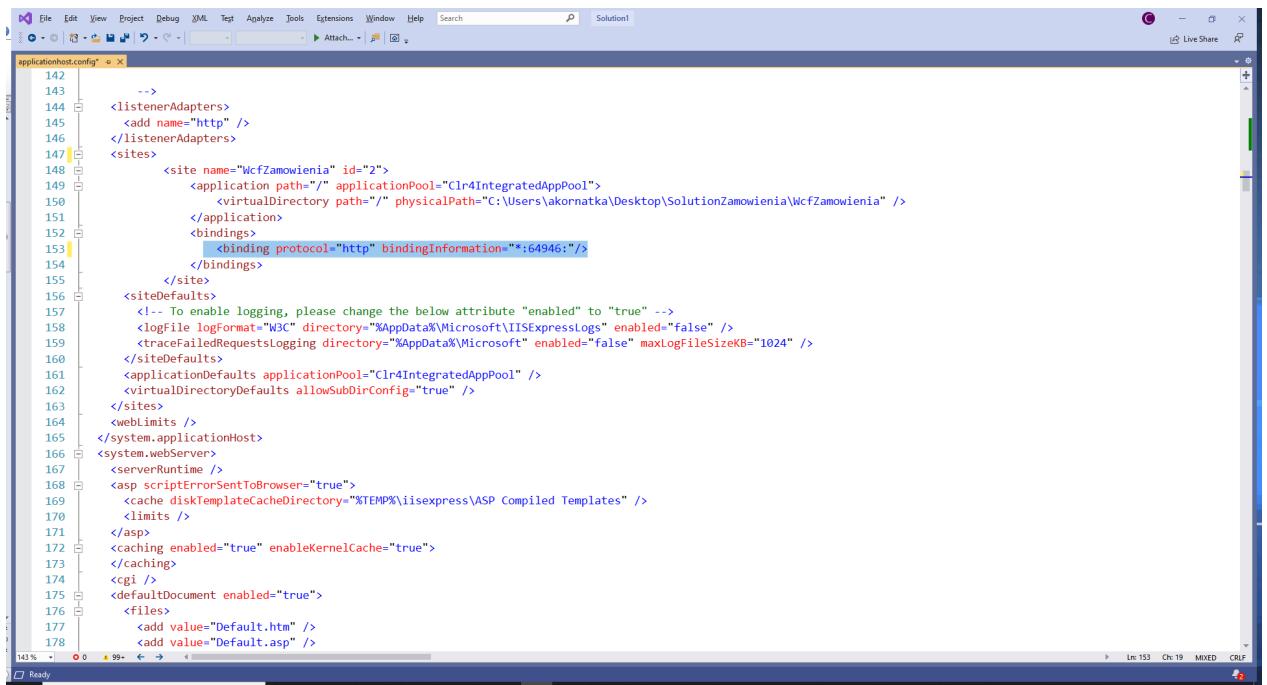
5. Odszukaj plik ...**WcfOrder\vs\WcfOrder\config\applicationhost.config**

6. W powyższym pliku powinna pozostać tylko jedna sekcja `<site>` w `<sites>`.



7. Następnie zmodyfikuj sekcję **binding protocol** według wzoru:

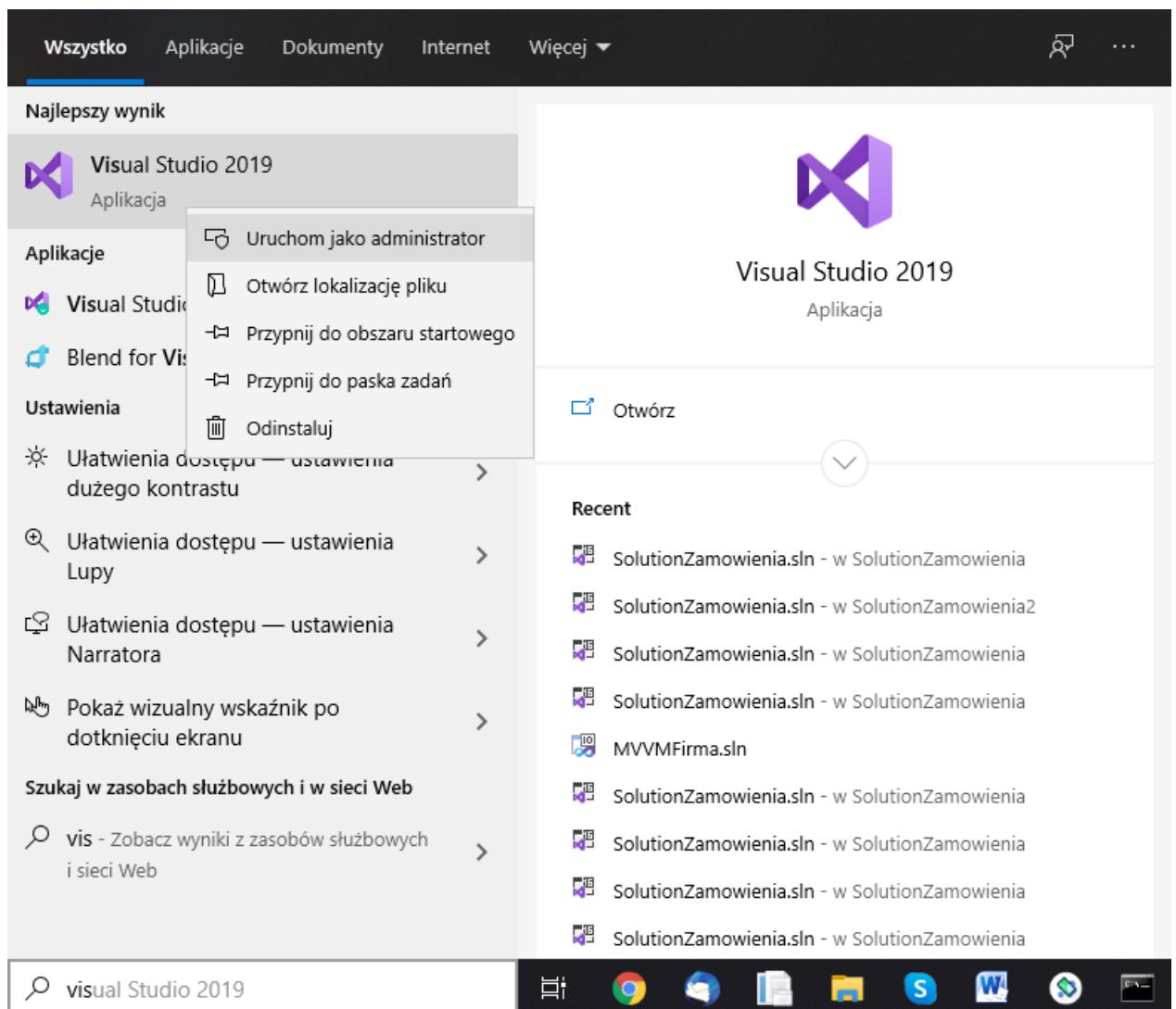
```
<binding protocol="http" bindingInformation="*:64946:/">
```

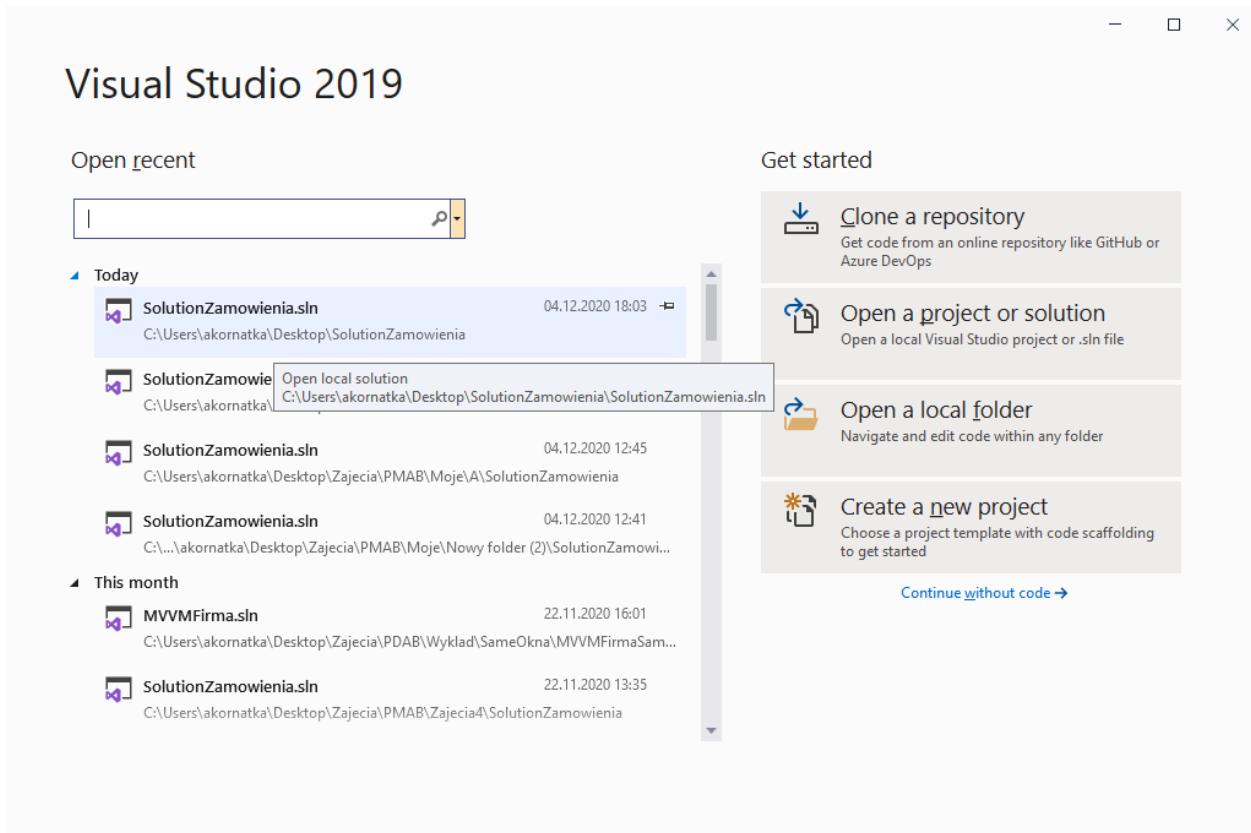


```
142
143    -->
144    <listenerAdapters>
145        <add name="http" />
146    </listenerAdapters>
147    <sites>
148        <site name="WcfZamowienia" id="2">
149            <application path="/" applicationPool="Clr4IntegratedAppPool">
150                <virtualDirectory path="/" physicalPath="C:\Users\akornatka\Desktop\SolutionZamowienia\WcfZamowienia" />
151            </application>
152            <bindings>
153                <binding protocol="http" bindingInformation="*:64946;" />
154            </bindings>
155        </site>
156        <siteDefaults>
157            <!-- To enable logging, please change the below attribute "enabled" to "true" -->
158            <logfile logFormat="W3C" directory="%AppData%\Microsoft\IISExpressLogs" enabled="false" />
159            <traceFailedRequestsLogging directory="%AppData%\Microsoft" enabled="false" maxLogFileSizeKB="1024" />
160        </siteDefaults>
161        <applicationDefaults applicationPool="Clr4IntegratedAppPool" />
162        <virtualDirectoryDefaults allowSubDirConfig="true" />
163    </sites>
164    <webLimits />
165 </system.applicationHost>
166 <system.webServer>
167     <serverRuntime />
168     <asp scriptErrorSentToBrowser="true">
169         <cache diskTemplateCacheDirectory="%TEMP%\iisexpress\ASP Compiled Templates" />
170         <limits />
171     </asp>
172     <caching enabled="true" enableKernelCache="true">
173         <cgi />
174         <defaultDocument enabled="true">
175             <files>
176                 <add value="Default.htm" />
177                 <add value="Default.asp" />
178             </files>

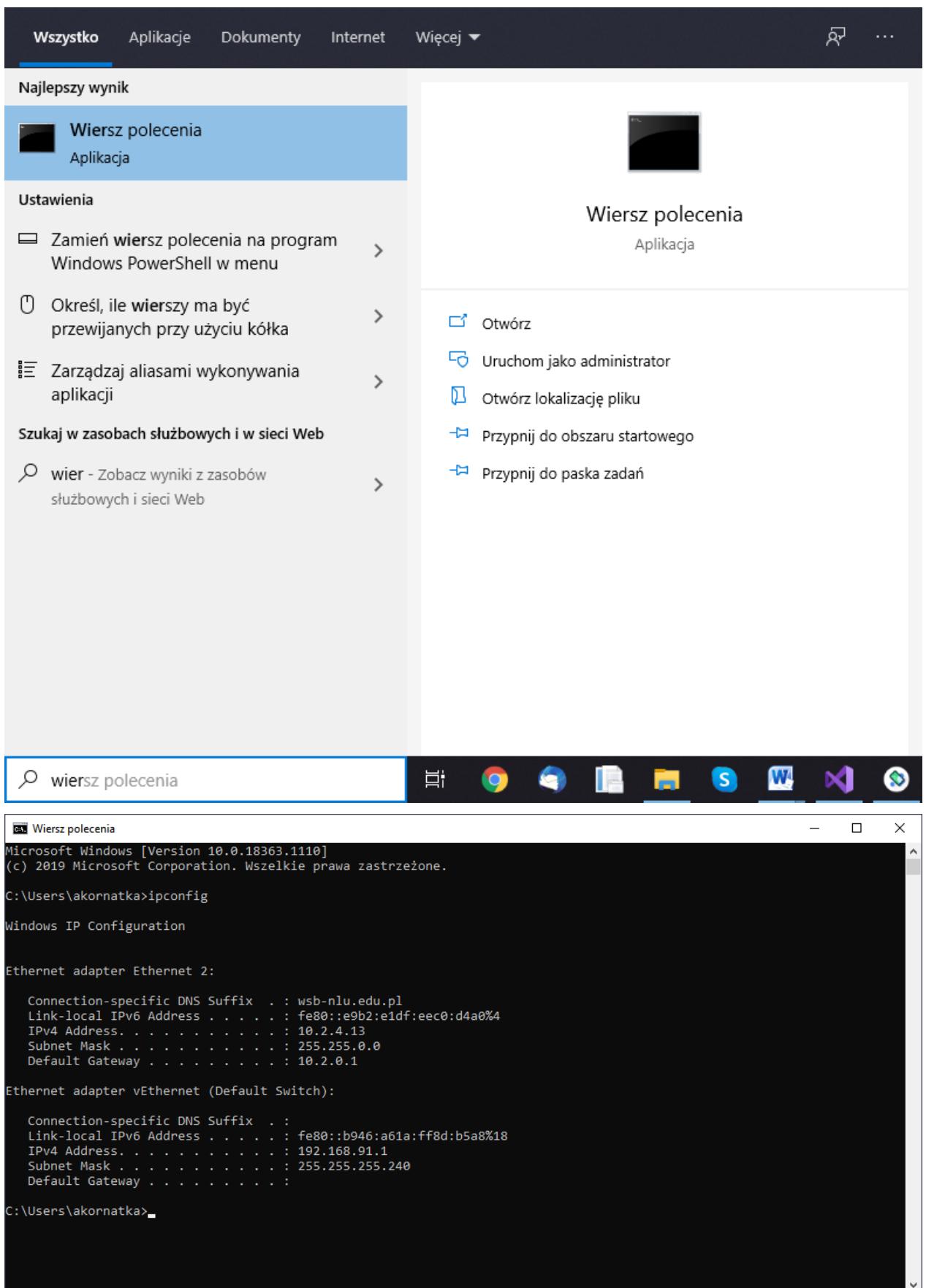
```

8. Uruchom **Visual Studio** z projektem **jako Administrator** (jeżeli wyskoczy błąd zamknij wszystkie zakładki Visual Studio i dokonaj komplikacji, ewentualnie uruchom ponownie Visual Studio).

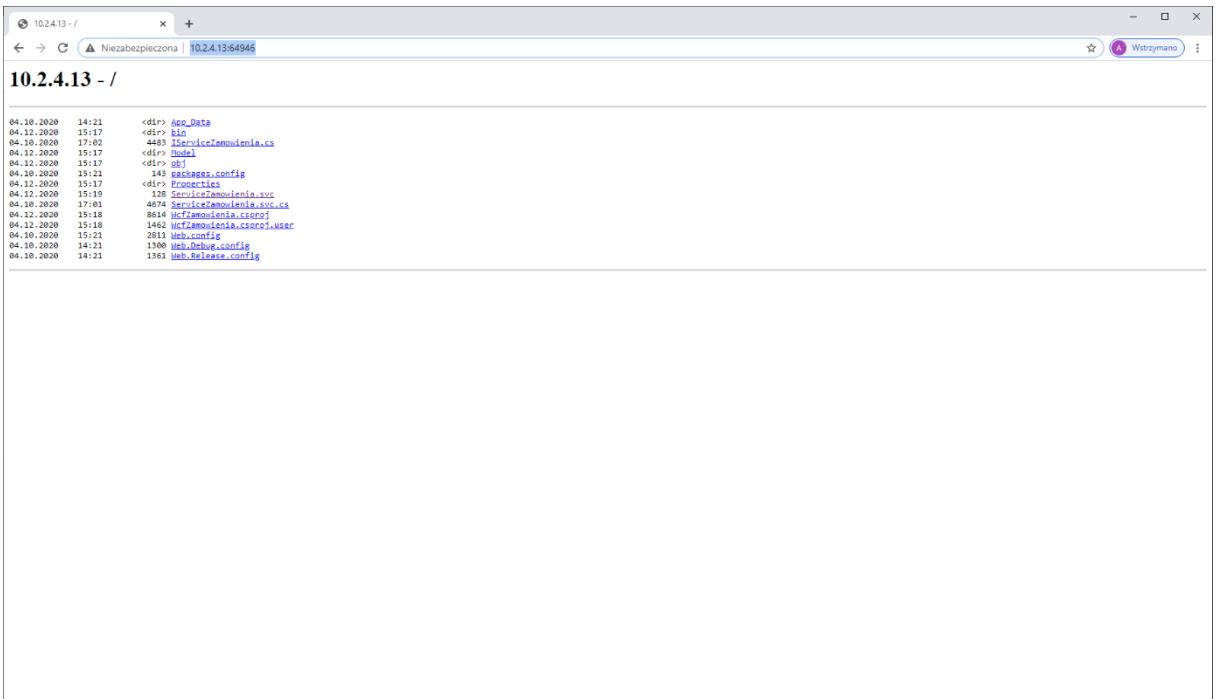




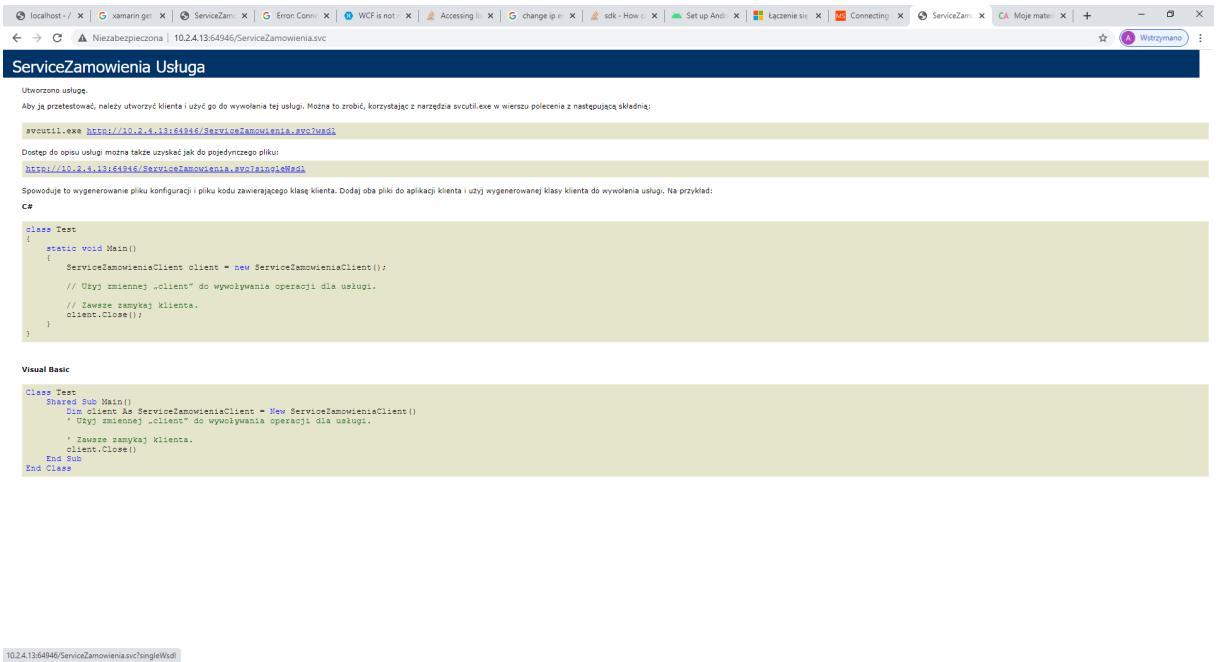
9. Oznacz jako projekt startowy **WCFOrder**. Dokonaj komplikacji WCF.
10. Oznacz jako projekt startowy **AppMobileOrders.Android**.
11. PPM na projekt **WCFOrder->Debug->Start New Instance**. Sprawdź **IP** swojego komputera i wejdź do **WCF na swoim IP**:



Popraw adres przeglądarki według wzoru: <http://10.2.4.13:64946/> (zamiast tekstu żółtego wpisz swoje IP).



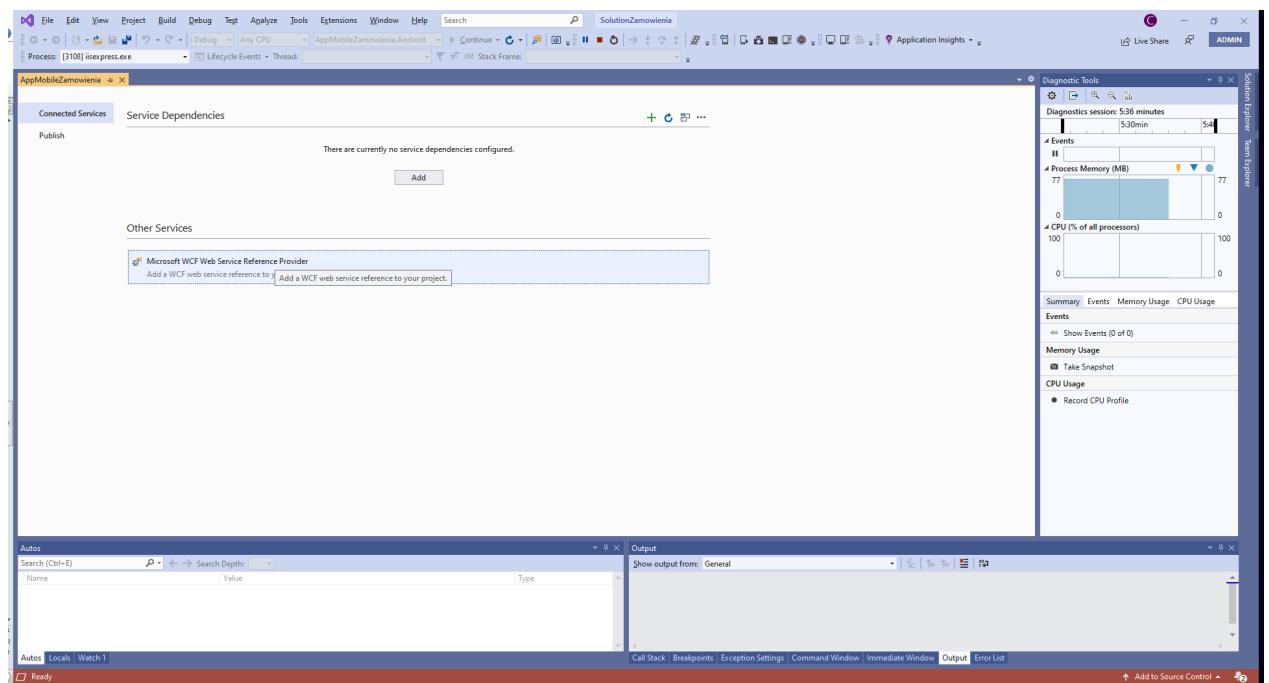
Zmodyfikuj adres według wzoru: <http://10.2.4.13:64946/OrderService.svc>



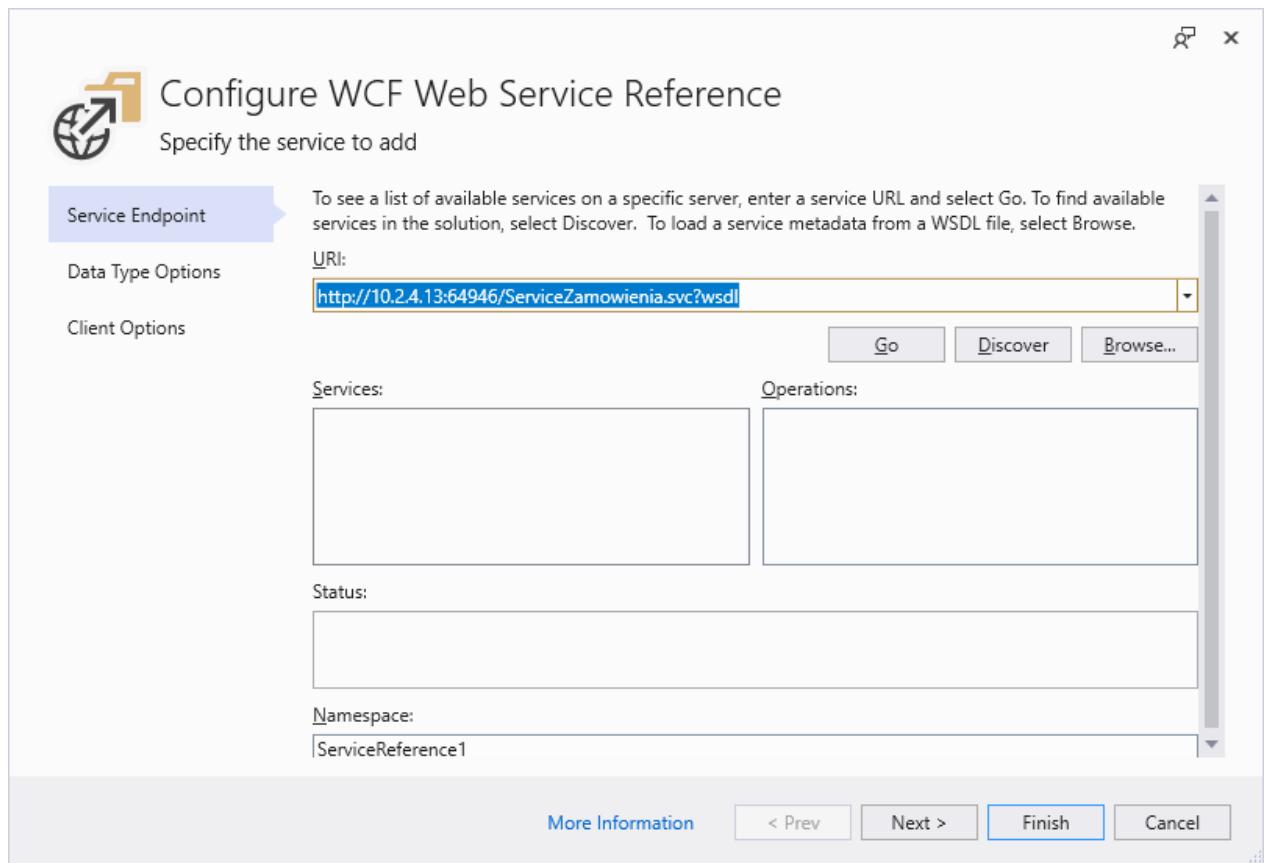
12. Włączaj WCF według powyższego sposobu zawsze kiedy używasz aplikacji mobilnej lub nie zamykaj WCF.

13. W prawym rogu wejdź ponownie do okna **Solution Explorer**.

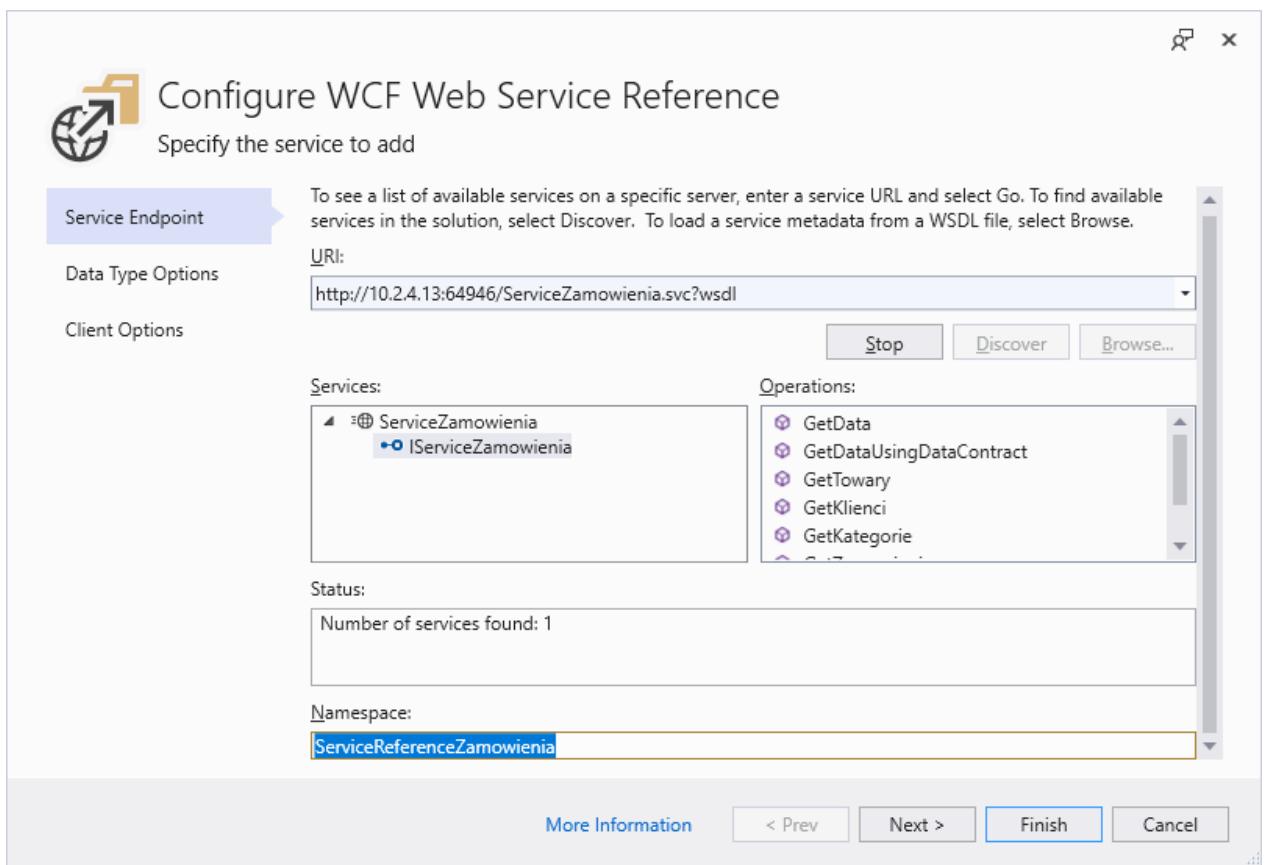
14. PPM na cały projekt **AppMobileOrders** -> **Add ->Service Reference->Microsoft WCF Web Service Reference Provider**.



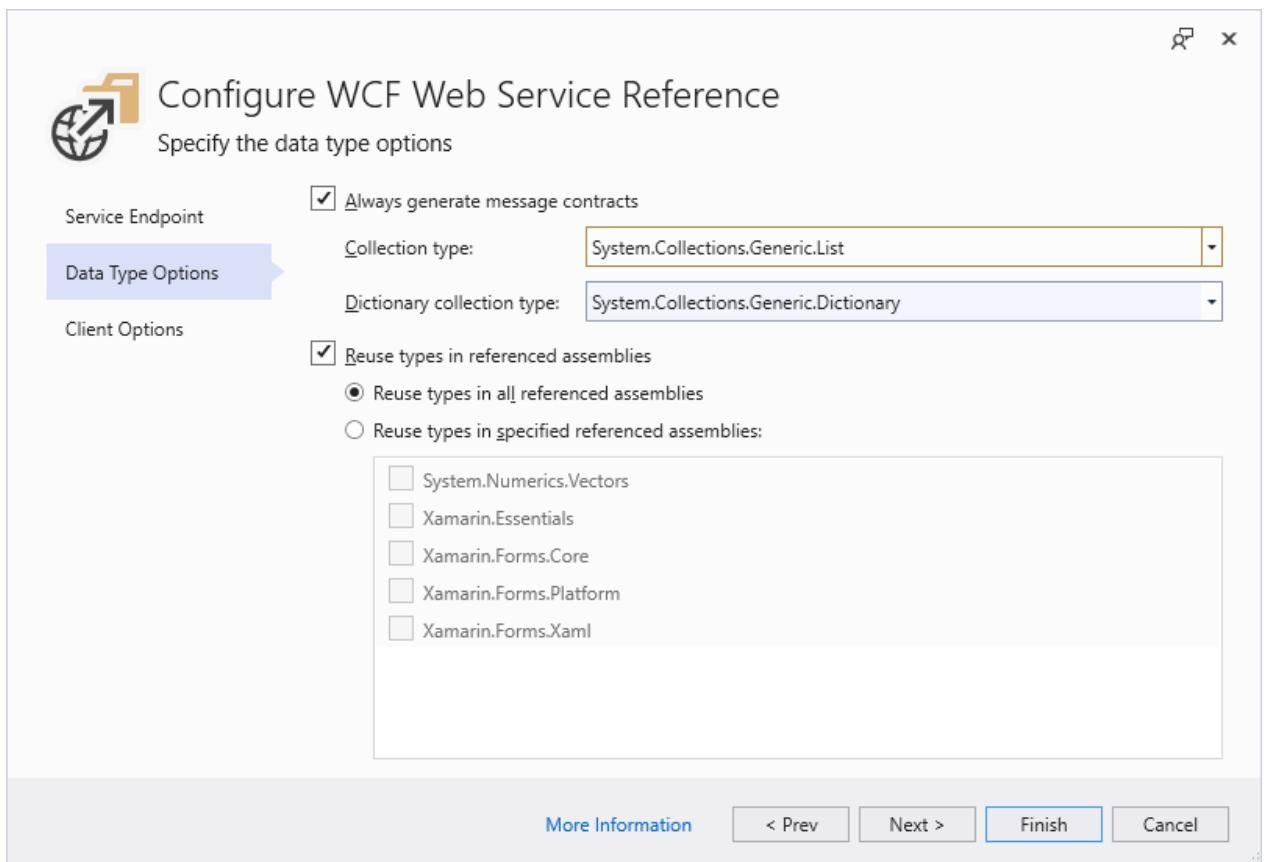
15. Wypełni pole URL według wzoru: <http://10.2.4.13:64946/OrderService.svc?wsdl> i naciśnij GO.



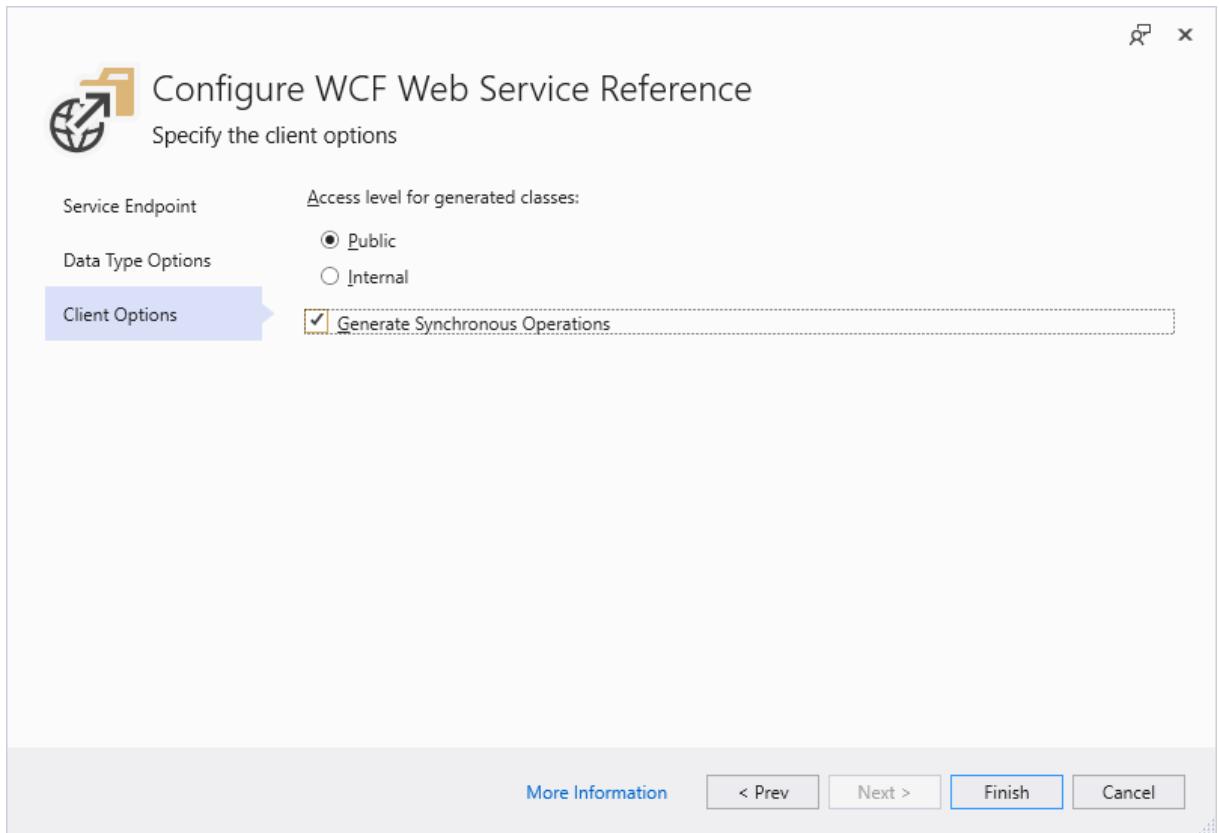
16. Service nazwij **ServiceReferenceOrders** i wybierz **Next**.



17. Wybierz opcję według wzoru:



18. Wybierz Next i zaznacz **Generate Synchronous Operations** i wciśnij Finish.



19. Zmodyfikuj plik **App.xaml.cs** według wzoru:

```
public partial class App : Application
{
    public App()
    {
        InitializeComponent();
        DependencyService.Register<MockDataStore>();
        DependencyService.Register<ClientDataStore>();

        DependencyService.Register<ServiceReferenceOrders.OrderServiceClient>();
        MainPage = new AppShell();
    }

    protected override void OnStart()
    {
    }

    protected override void OnSleep()
    
```

```

    {
}

protected override void OnResume()
{
}
}
}

```

20. Zmodyfikuj klasę Services->ClientDataStore według wzoru.

```

public class ClientDataStore : ItemDataStore<Client>
{
    public ClientDataStore()
    {
        var _OrdersServices = DependencyService.Get<ServiceReferenceOrders.IOrderService>();
        items = _OrdersServices.GetClients(null).GetClientsResult.Select(k=> new Client
        {
            IdClient = k.IdClient ,
            Name =k.Name ,
            Adres=k.Adres,
            PhoneNumber=k.PhoneNumber
        }).ToList();
        //items = new List<Client>()
        //{
        //    new Client{IdClient=1,Name="Client 1", Adres="Zielona 1",
        PhoneNumber="1"},
        //    new Client{IdClient=2,Name="Client 2", Adres="Zielona 2",
        PhoneNumber="2"},
        //    new Client{IdClient=3,Name="Client 3", Adres="Zielona 3",
        PhoneNumber="3"},
        //    new Client{IdClient=4,Name="Client 4", Adres="Zielona 4",
        PhoneNumber="4"},
        //}
    }

    public override Client Find(Client item)
    {

```

```

        return items.Where((Client arg) => arg.IdClient == item.IdClient).FirstOrDefault();
    }

    public override Client Find(int id)
    {
        return items.Where((Client arg) => arg.IdClient == id).FirstOrDefault();
    }

}

}

```

21. Wersja II: Zmodyfikuj klasę **Services->ClientDataStore** według wzoru.

```

public class ClientDataStore : ItemDataStore<Client>
{
    public ClientDataStore()
    {
        var _OrdersServices = DependencyService.Get<ServiceReferenceOrders.IOrderService>();
        //items = _OrdersServices.GetClients(null).GetClientsResult.Select(k =>
new Client
{
    //{
    // IdClient = k.IdClient,
    // Name = k.Name,
    // Adres = k.Adres,
    // PhoneNumber = k.PhoneNumber
}).ToList();

List<ClientForView> listaKFV =
_OrdersServices.GetClients(null).GetClientsResult.ToList();
        items = new List<Client>(); //to dodać
        foreach (var k in listaKFV)
        {
            items.Add(new Client
{
    IdClient = k.IdClient,
    Name = k.Name,
    Adres = k.Adres,

```

```

                PhoneNumber = k.PhoneNumber
            }
        );
    }

    //items =_zaowieniaServices.GetClients(null).GetClientsResult
    //items = new List<Client>()
    //{
    //    new Client{IdClient=1,Name="Client 1", Adres="Zielona 1",
    PhoneNumber="1"},
    //    new Client{IdClient=2,Name="Client 2", Adres="Zielona 2",
    PhoneNumber="2"},
    //    new Client{IdClient=3,Name="Client 3", Adres="Zielona 3",
    PhoneNumber="3"},
    //    new Client{IdClient=4,Name="Client 4", Adres="Zielona 4",
    PhoneNumber="4"},
    //}
}

public override Client Find(Client item)
{
    return items.Where((Client arg) => arg.IdClient ==
item.IdClient).FirstOrDefault();
}

public override Client Find(int id)
{
    return items.Where((Client arg) => arg.IdClient == id).FirstOrDefault();
}

}

```

22.Zmodyfikuj klasę **Services->ItemDataStore** według wzoru:

```

public abstract class ItemDataStore<T>: IDataStore<T>
{
    public IOrderService OrdersServices { get; set; }
    public List<T> items { get; set; }
    public ItemDataStore()
    {
        OrdersServices =
DependencyService.Get<ServiceReferenceOrders.IOrderService>();
    }
}

```

}

...

23.Zmodyfikuj klasę **Services->ClientDataStore** według wzoru:

```
public class ClientDataStore : ItemDataStore<Client>
{
    public ClientDataStore()
    {
        items = OrdersServices.GetClients(null).GetClientsResult.Select(k=> new
Client
{
    IdClient = k.IdClient,
    Name =k.Name,
    Adres=k.Adres,
    PhoneNumber=k.PhoneNumber
}).ToList();
    }
}
```

....

24.Sprawdź czy aplikacja działa również na WCF uruchomionym jako:

<http://localhost:64946/OrderService.svc>

25.Uruchom nowe okno **Visual Studio** jako administrator (nie zamkaj aktualnego).

26.Ustaw jako projekt startowy **WCFOrders**.

27.Przejdź do projektu **WcfOrders**.

28.Zmodyfikuj **IOrderService** dodając funkcje **AddClient** według wzoru:

```
[OperationContract]
void AddClient(Klienci Client);
```

29. Zmodyfikuj **OrderService** dodając funkcje **AddClient** według wzoru:

```
public void AddClient(Klienci Client)
{
    OrdersEntities db = new OrdersEntities();
    db.Klienci.Add(Client);
    db.SaveChanges();
}
```

30.Uruchom WCF.

31.Na pierwszym oknie **VisualStudio** PPM na **ServiceReferenceOrders** i wybierz **Update Microsoft WCF Web Service Provider**.

32.Edytuj **ItemDataStore** i **testowo** zmodyfikuj funkcje **AddItemAsync** według wzoru:

```

//oczywiście ta funkcja będzie poprawiona
//wersja I
public async Task<bool> AddItemAsync(T item)
{
    OrdersServices.AddClient ( new AddClientRequest(new Klienci { Name = "1",
Adres = "1",PhoneNumber="1",CzyAktywny=true }));
    items.Add(item);
    return await Task.FromResult(true);
}
//wersja II //tu koniec
public abstract void Add(T item);
public async Task<bool> AddItemAsync(T item)
{
    Add(item);
    items.Add(item);
    return await Task.FromResult(true);
}

```

33.Edytuj ClientDataStore i **testowo** dodaj funkcje **AddItem** według wzoru:

```

public override void Add(Client item)
{
    OrdersServices.AddClient(new AddClientRequest(new Klienci { Name =
item.Name,      Adres      =      item.Adres,      PhoneNumber=item.PhoneNumber,
CzyAktywny=true }));
}

```

34.Edytuj MockDataStore i **testowo** dodaj funkcje **AddItem** według wzoru:

```

public override void Add(Item item)
{
}

}

```

Operacje na tabelach z kluczem obcym

1. Wejdź do projektu **WcfOrder**.

2.Zmodyfikuj **IServiceOrder** dodając funkcję **AddOrder** według wzoru:

```

[OperationContract]
void AddOrder(Zamowienia Order);

```

3. Zmodyfikuj **ServiceOrder** dodając funkcję **AddOrder** według wzoru:

```

public void AddOrder(Zamowienia Order)

```

```
{  
    ZamowieniaEntities db = new ZamowieniaEntities();  
    db.Zamowienia.Add(Order);  
    db.SaveChanges();  
}
```

4. Uruchom WCF.
5. Poniższe operacje będą wykonywane na projekcie **AppMobileOrders**.
6. PPM na **Connected Services-> ServiceReferenceZamowienia** i wybierz **Update Microsoft WCF Web Service Provider**.
7. PPM na **Models i Add->Class**. Klasę nazwij **Order**. Utwórz kod tej klasy według wzoru:

```
public class Order  
{  
    public int IdOrder { get; set; }  
    public DateTime? DataZamowienia { get; set; }  
    public int IdClient { get; set; }  
    public string ClientDane { get; set; }  
    public int IdWorker { get; set; }  
    public string WorkerDane { get; set; }  
    public string Uwagi { get; set; }  
    public DateTime? TerminDostawy { get; set; }  
  
}
```

8. PPM na **Services i Add->Class**. Klasę nazwij **OrderDataStore**. Utwórz kod tej klasy według wzoru:

```
public class OrderDataStore : ItemDataStore<Order>  
{  
    public OrderDataStore()  
    {  
        items =  
zamowieniaServices.GetZamowienia(null).GetZamowieniaResult.Select(z => new  
Order  
{  
    IdOrder=z.IdOrder,  
    DataZamowienia=z.DataZamowienia,  
    ClientDane=z.ClientDane,  
    WorkerDane=z.WorkerDane,
```

```

        Uwagi=z.Uwagi,
        TerminDostawy=z.TerminDostawy,
    }).ToList();
}
public override void Add(Order item)
{
    zamowieniaServices.AddOrder(new AddOrderRequest(new Zamowienia {
        DataZamowienia=item.DataZamowienia,IdClient=item.IdClient,IdWorkera=item.I
        dWorkera,Uwagi=item.Uwagi,TerminDostawy=item.TerminDostawy}));
}

public override Order Find(Order item)
{
    return items.Where((Order arg) => arg.IdOrder ==
item.IdOrder).FirstOrDefault();
}
public override Order Find(int id)
{
    return items.Where((Order arg) => arg.IdOrder == id).FirstOrDefault();
}

}

```

9. PPM na **ViewModels i Add->Class**. Klasę nazwij **ZamowieniaViewModel**. Utwórz kod tej klasy według wzoru:

```

public class ZamowieniaViewModel : AItemsViewModel<Order>
{
    public ZamowieniaViewModel()
        : base()
    {
        Title = "Zamówienia";
    }
    public override void GoToAddPage()
    {
        Shell.Current.GoToAsync(nameof(NewOrderPage));
    }
}

```

10.PPM na Views i Add->NewItem->ContentPage (bez C#) Widok nazwij

ZamowieniaPage. Utwórz kod tego widoku według wzoru:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AppMobileOrders.Views.ZamowieniaPage"
    Title="{Binding Title}"
    xmlns:local="clr-namespace:AppMobileOrders.ViewModels"
    xmlns:model="clr-namespace:AppMobileOrders.Models"
    >
<ContentPage.ToolbarItems>
    <ToolbarItem Text="Dodaj" Command="{Binding AddItemCommand}" />
</ContentPage.ToolbarItems>

<RefreshView x:DataType="local:ZamowieniaViewModel" Command="{Binding LoadItemsCommand}" IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
    <CollectionView x:Name="ItemsListView"
        ItemsSource="{Binding Items}"
        SelectionMode="None">
        <CollectionView.ItemTemplate>
            <DataTemplate>
                <StackLayout Padding="10" x:DataType="model:Order">
                    <Label Text="{Binding ClientDane}"
                        LineBreakMode="NoWrap"
                        Style="{DynamicResource ListItemTextStyle}"
                        FontSize="16" />
                    <Label Text="{Binding DataZamowienia}"
                        LineBreakMode="NoWrap"
                        Style="{DynamicResource ListItemDetailTextStyle}"
                        FontSize="13" />
                    <Label Text="{Binding Uwagi}"
                        LineBreakMode="NoWrap"
                        Style="{DynamicResource ListItemDetailTextStyle}"
                        FontSize="13" />
                    <Label Text="{Binding TerminDostawy}"
                        LineBreakMode="NoWrap"
                        Style="{DynamicResource ListItemDetailTextStyle}">
                
```

```

        FontSize="13" />
<Label Text="{Binding WorkerDane}"
       LineBreakMode="NoWrap"
       Style="{DynamicResource ListItemDetailTextStyle}"
       FontSize="13" />
<StackLayout.GestureRecognizers>
    <TapGestureRecognizer
        NumberOfTapsRequired="1"
        Command="{Binding
                    Source={RelativeSource
AncestorType={x:Type local:ZamowieniaViewModel}}},
                    Path=ItemTapped}">
        CommandParameter="{Binding .}">
    </TapGestureRecognizer>
</StackLayout.GestureRecognizers>
</StackLayout>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</RefreshView>
</ContentPage>

```

11. Edytuj **Views->ZamowieniaPage.xaml->ZamowieniaPage.xaml.cs** i i zmień jej kod według wzoru:

```

[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class ZamowieniaPage : ContentPage
{
    ZamowieniaViewModel _viewModel;
    public ZamowieniaPage()
    {
        InitializeComponent();
        BindingContext = _viewModel = new ZamowieniaViewModel();
    }
    protected override void OnAppearing()
    {
        base.OnAppearing();
        _viewModel.OnAppearing();
    }
}

```

12. Edytuj plik **AppShell.xaml** i zmień jego sekcję **FlyoutItem** według wzoru:

```
<FlyoutItem FlyoutDisplayOptions="AsMultipleItems">
    <ShellContent Title="About" Icon="tab_about.png" Route="AboutPage"
ContentTemplate="{DataTemplate local:AboutPage}" />
    <ShellContent Title="Browse" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:ItemsPage}" />
    <ShellContent Title="Klienci" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:KlienciPage}" />
    <ShellContent Title="Zamowienia" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:ZamowieniaPage}" />
</FlyoutItem>
```

13. Edytuj plik **App.xaml->App.xaml.cs** i zmień jego konstruktor według wzoru:

```
public App()
{
    InitializeComponent();
    DependencyService.Register<MockDataStore>();
    DependencyService.Register<ClientDataStore>();
    DependencyService.Register<OrderDataStore>();

    DependencyService.Register<ServiceReferenceZamowienia.ServiceOrderClient>();
    MainPage = new AppShell();
}
```

14. PPM na **ViewModels i Add->Class**. Klasę nazwij **NewOrderViewModel**. Utwórz kod tej klasy według wzoru:

==Wersja 1 (testowa)==

```
public class NewOrderViewModel : ANewItemViewModel<Order>
{
    private DateTime? dataZamowienia;
    private int idClient;
    private string ClientDane;
    private int idWorkera;
    private string WorkerDane;
    private string uwagi;
    private DateTime? terminDostawy;
    public NewOrderViewModel()
        : base()
{}
```

```
        }

        public override bool ValidateSave()
        {
            return !String.IsNullOrEmpty(ClientDane);
        }

        public DateTime? DataZamowienia
        {
            get => dataZamowienia;
            set => SetProperty(ref dataZamowienia, value);
        }

        public int IdClient
        {
            get => idClient;
            set => SetProperty(ref idClient, value);
        }

        public string ClientDane
        {
            get => ClientDane;
            set => SetProperty(ref ClientDane, value);
        }

        public int IdWorkera
        {
            get => idWorkera;
            set => SetProperty(ref idWorkera, value);
        }

        public string WorkerDane
        {
            get => WorkerDane;
            set => SetProperty(ref WorkerDane, value);
        }

        public string Uwagi
        {
            get => uwagi;
            set => SetProperty(ref uwagi, value);
        }

        public DateTime? TerminDostawy
        {
```

```

        get => terminDostawy;
        set => SetProperty(ref terminDostawy, value);
    }
    public override Order SetItem()
    {
        Order newItem = new Order()
        {
            IdOrder = 1,
            DataZamowienia = DataZamowienia,
            TerminDostawy = TerminDostawy,
            Uwagi = Uwagi,
            ClientDane = "AAA",
            WorkerDane = "BBB",
            IdClient=IdClient,
            IdWorkera=IdWorkera
        };
        return newItem;
    }
}
==Wersja 2 ==

```

```

public class NewOrderViewModel : ANewItemViewModel<Order>
{
    private DateTime? dataZamowienia;
    private Client selectedClient;
    private string uwagi;
    private DateTime? terminDostawy;
    public List<Client> Klienci
    {
        get
        {
            return new ClientDataStore().items;
        }
    }
    public NewOrderViewModel()
        : base()
    {
        selectedClient = new Client();
    }
}

```

```

        DataZamowienia = DateTime.Now;
        TerminDostawy = DateTime.Now;
    }

    public override bool ValidateSave()
    {
        return !String.IsNullOrEmpty(selectedClient.Nazwa);
    }

    public DateTime? DataZamowienia
    {
        get => dataZamowienia;
        set => SetProperty(ref dataZamowienia, value);
    }

    public Client SelectedClient
    {
        get => selectedClient;
        set => SetProperty(ref selectedClient, value);
    }

    public string Uwagi
    {
        get => uwagi;
        set => SetProperty(ref uwagi, value);
    }

    public DateTime? TerminDostawy
    {
        get => terminDostawy;
        set => SetProperty(ref terminDostawy, value);
    }

    public override Order SetItem()
    {
        Order newItem = new Order()
        {
            IdOrder = 1,
            DataZamowienia = DataZamowienia,
            TerminDostawy = TerminDostawy,
            Uwagi = Uwagi,
            IdClient = SelectedClient.IdClient,
            ClientDane = SelectedClient.Nazwa,
        };
    }
}

```

```

        //Zadanie: Analogicznie wykonaj wybieranie Workera za pomocą Picker
        WorkerDane = "Jakub Jaskulski",
        IdWorkera=1
    };
    return newItem;
}
}

```

15.PPM na Views i Add->NewItem->ContentPage (bez C#) Widok nazwij **NewOrderPage**. Utwórz kod tego widoku według wzoru:

==Wersja 1 (testowa)==

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AppMobileOrders.Views.NewOrderPage"
    Shell.PresentationMode="ModalAnimated"
    Title="New Order"
    >
<ContentPage.Content>
    <StackLayout Spacing="3" Padding="15">
        <Label Text="Data Zamowienia" FontSize="Medium" />
        <Entry Text="{Binding DataZamowienia, Mode=TwoWay}"
            FontSize="Medium" />
        <Label Text="IdClient" FontSize="Medium" />
        <Editor Text="{Binding IdClient, Mode=TwoWay}"
            AutoSize="TextChanges" FontSize="Medium" Margin="0" />
        <Label Text="ClientDane" FontSize="Medium" />
        <Entry Text="{Binding ClientDane, Mode=TwoWay}" FontSize="Medium"
            />
        <Label Text="IdWorkera" FontSize="Medium" />
        <Entry Text="{Binding IdWorkera, Mode=TwoWay}" FontSize="Medium"
            />
        <Label Text="WorkerDane" FontSize="Medium" />
        <Entry Text="{Binding WorkerDane, Mode=TwoWay}"
            FontSize="Medium" />
        <Label Text="Uwagi" FontSize="Medium" />
        <Entry Text="{Binding Uwagi, Mode=TwoWay}" FontSize="Medium" />
    </StackLayout>
</ContentPage.Content>

```

```

<Label Text="Termin Dostawy" FontSize="Medium" />
<Entry      Text="{Binding      TerminDostawy,      Mode=TwoWay}"
FontSize="Medium" />
<StackLayout Orientation="Horizontal">
    <Button      Text="Anuluj"      Command="{Binding      CancelCommand}"
HorizontalOptions="FillAndExpand"></Button>
    <Button      Text="Zapisz"      Command="{Binding      SaveCommand}"
HorizontalOptions="FillAndExpand"></Button>
</StackLayout>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

==Wersja 2 ==

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AppMobileOrders.Views.NewOrderPage"
    Shell.PresentationMode="ModalAnimated"
    Title="New Order"
    >
<ContentPage.Content>
    <StackLayout Spacing="3" Padding="15">
        <Label Text="Data Zamowienia" FontSize="Medium" />
        <DatePicker      Date="{Binding      DataZamowienia,      Mode=TwoWay}"
FontSize="Medium" />
        <Picker      Title="Wybierz Client"      ItemsSource="{Binding      Klienci}"
ItemDisplayBinding="{Binding Nazwa}"      SelectedItem="{Binding SelectedClient}"
FontSize="Medium" Margin="0" />
        <Label Text="Uwagi" FontSize="Medium" />
        <Entry Text="{Binding Uwagi, Mode=TwoWay}" FontSize="Medium" />
        <Label Text="Termin Dostawy" FontSize="Medium" />
        <DatePicker      Date="{Binding      TerminDostawy,      Mode=TwoWay}"
FontSize="Medium" />
        <StackLayout Orientation="Horizontal">
            <Button      Text="Anuluj"      Command="{Binding      CancelCommand}"
HorizontalOptions="FillAndExpand"></Button>

```

```

        <Button    Text="Zapisz"    Command="{Binding    SaveCommand}"
HorizontalOptions="FillAndExpand"></Button>
        </StackLayout>
    </StackLayout>
</ContentPage.Content>
</ContentPage>

```

16.Edytuj **Views->NewOrderPage.xaml->NewOrderPage.xaml.cs** i zmień jej kod według wzoru:

```

[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class NewOrderPage : ContentPage
{
    public Order Item { get; set; }
    public NewOrderPage()
    {
        InitializeComponent();
        BindingContext = new NewOrderViewModel();
    }
}

```

17.Edytuj plik **AppShell.xaml->AppShell.xaml.cs** i zmień jego konstruktor według wzoru:

```

public AppShell()
{
    InitializeComponent();
    Routing.RegisterRoute(nameof(ItemDetailPage), typeof(ItemDetailPage));
    Routing.RegisterRoute(nameof(NewItemPage), typeof(NewItemPage));
    Routing.RegisterRoute(nameof(NewClientPage), typeof(NewClientPage));
    Routing.RegisterRoute(nameof(NewOrderPage), typeof(NewOrderPage));
}

```

Logika biznesowa

1. Wejdź do projektu **WcfZamowienia**.

2.Zmodyfikuj **IServiceZamowienia** dodając funkcje:

```

[OperationContract]
decimal? WartoscZamowienPracownikaZDanegoDnia (int
idPracownika, DateTime data);
[OperationContract]

```

```

        decimal? WartoscZamowienWszystkichPracownikowZDanegoDnia(DateTime
data);
    [OperationContract]
    decimal? WartoscWszystkichZamowienPracownika(int idPracownika);

```

3.Zmodyfikuj **ServiceZamowienia** dodając funkcje według wzoru:

```

public decimal? WartoscZamowienPracownikaZDanegoDnia(int idPracownika,
DateTime data)
{
    ZamowieniaEntities db = new ZamowieniaEntities();
    return
    (
        from pozycja in db.PozycjeZamowienia
        where      pozycja.Zamowienia.DataZamowienia==data      &&
pozycja.Zamowienia.IdPracownika==idPracownika && pozycja.CzyAktywny==true
        select pozycja.Cena*pozycja.Ilosc
    ).Sum();
}
public decimal?
WartoscZamowienWszystkichPracownikowZDanegoDnia(DateTime data)
{
    ZamowieniaEntities db = new ZamowieniaEntities();
    return
    (
        from pozycja in db.PozycjeZamowienia
        where      pozycja.Zamowienia.DataZamowienia      ==      data      &&
pozycja.CzyAktywny == true
        select pozycja.Cena * pozycja.Ilosc
    ).Sum();
}
public decimal? WartoscWszystkichZamowienPracownika(int idPracownika)
{
    ZamowieniaEntities db = new ZamowieniaEntities();
    return
    (
        from pozycja in db.PozycjeZamowienia
        where      pozycja.Zamowienia.IdPracownika      ==      idPracownika      &&
pozycja.CzyAktywny == true
    )
}
```

```

        select pozycja.Cena * pozycja.Ilosc
    ).Sum();
}

4. Uruchom WCF.
5. Poniższe operacje będą wykonywane na projekcie AppMobileZamowienia.
6. PPM na Connected Services-> ServiceReferenceZamowienia i wybierz Update Microsoft WCF Web Service Provider.
7. PPM na Services i Add->Class. Klasę nazwij WartoscZamowienDataStore.  

Utwórz kod tej klasy według wzoru:  

public class WartoscZamowienDataStore
{
    public IServiceZamowienia zamowieniaServices { get; set; }
    public WartoscZamowienDataStore()
    {
        zamowieniaServices =
            DependencyService.Get<ServiceReferenceZamowienia.IServiceZamowienia>();
    }
    public decimal? WartoscZamowienPracownikaZDanegoDnia(int idPracownika,
    DateTime data)
    {
        return
            zamowieniaServices.WartoscZamowienPracownikaZDanegoDnia(new
            WartoscZamowienPracownikaZDanegoDniaRequest(idPracownika,data))
            .WartoscZamowienPracownikaZDanegoDniaResult;
    }
    public decimal?
            WartoscZamowienWszystkichPracownikowZDanegoDnia(DateTime data)
    {
        return
            zamowieniaServices.WartoscZamowienWszystkichPracownikowZDanegoDnia(new
            WartoscZamowienWszystkichPracownikowZDanegoDniaRequest(data))
            .WartoscZamowienWszystkichPracownikowZDanegoDniaResult;
    }
    public decimal? WartoscWszystkichZamowienPracownika(int idPracownika)
    {
        return zamowieniaServices.WartoscWszystkichZamowienPracownika(new
        WartoscWszystkichZamowienPracownikaRequest(idPracownika))
    }
}

```

```
        .WartoscWszystkichZamowienPracownikaResult;  
    }  
  
}
```

8. PPM na ViewModels i Add->Class. Klasę nazwij **WartoscZamowienViewModel**.

Utwórz kod tej klasy według wzoru:

```
public class WartoscZamowienViewModel : BaseViewModel  
{  
    private int idPracownika;  
    public int IdPracownika  
    {  
        get => idPracownika;  
        set => SetProperty(ref idPracownika, value);  
    }  
    private DateTime data;  
    public DateTime Data  
    {  
        get => data;  
        set => SetProperty(ref data, value);  
    }  
    private decimal? wartoscZamowienPracownikaZDanegoDnia;  
    public decimal? WartoscZamowienPracownikaZDanegoDnia  
    {  
        get => wartoscZamowienPracownikaZDanegoDnia;  
        set => SetProperty(ref wartoscZamowienPracownikaZDanegoDnia, value);  
    }  
    private decimal? wartoscZamowienWszystkichPracownikowZDanegoDnia;  
    public decimal? WartoscZamowienWszystkichPracownikowZDanegoDnia  
    {  
        get => wartoscZamowienWszystkichPracownikowZDanegoDnia;  
        set = > SetProperty(ref  
            wartoscZamowienWszystkichPracownikowZDanegoDnia, value);  
    }  
    private decimal? wartoscWszystkichZamowienPracownika;  
    public decimal? WartoscWszystkichZamowienPracownika  
    {  
        get => wartoscWszystkichZamowienPracownika;
```

```

        set => SetProperty(ref wartoscWszystkichZamowienPracownika, value);
    }
    public Command WartoscZamowienPracownikaZDanegoDniaCommand { get;
}
    public                                     Command
WartoscZamowienWszystkichPracownikowZDanegoDniaCommand { get; }
    public Command WartoscWszystkichZamowienPracownikaCommand { get; }
    public WartoscZamowienViewModel()
{
    IdPracownika = 1;
    Data = new DateTime(2021,1,16);
    WartoscZamowienPracownikaZDanegoDniaCommand      =      new
Command(OnWartoscZamowienPracownikaZDanegoDnia);
    WartoscZamowienWszystkichPracownikowZDanegoDniaCommand  =  new
Command(OnWartoscZamowienWszystkichPracownikowZDanegoDnia);
    WartoscWszystkichZamowienPracownikaCommand      =      new
Command(OnWartoscWszystkichZamowienPracownika);
    WartoscZamowienPracownikaZDanegoDnia = 0;
    WartoscZamowienWszystkichPracownikowZDanegoDnia = 0;
    WartoscWszystkichZamowienPracownika = 0;
}
private void OnWartoscZamowienPracownikaZDanegoDnia()
{
    WartoscZamowienPracownikaZDanegoDnia=new
WartoscZamowienDataStore().WartoscZamowienPracownikaZDanegoDnia(IdPraco
wnika, Data);
}
private void OnWartoscZamowienWszystkichPracownikowZDanegoDnia()
{
    WartoscZamowienWszystkichPracownikowZDanegoDnia=new
WartoscZamowienDataStore().WartoscZamowienWszystkichPracownikowZDanego
Dnia(Data);
}
private void OnWartoscWszystkichZamowienPracownika()
{

```

```
        WartoscWszystkichZamowienPracownika=new  
        WartoscZamowienDataStore().WartoscWszystkichZamowienPracownika(IdPracowni  
ka);  
    }  
  
}
```

9. PPM na **Views i Add->NewItem->ContentPage** (bez C#) Widok nazwij **WartoscZamowienPage**. Utwórz kod tego widoku według wzoru:

```
<?xml version="1.0" encoding="utf-8" ?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  
    x:Class="AppMobileZamowienia.Views.WartoscZamowienPage">  
    <ContentPage.Content>  
        <StackLayout>  
            <StackLayout BackgroundColor="LightBlue" Margin="4">  
                <DatePicker Date="{Binding Data, Mode=TwoWay}"  
                    FontSize="Medium" HorizontalOptions="Center" />  
            </StackLayout>  
            <StackLayout BackgroundColor="LightBlue" Margin="4">  
                <Label Text="Wartosc Zamowien Pracownika Z Danego Dnia: "  
                    HorizontalOptions="Center"/>  
                <Label Text="{Binding WartoscZamowienPracownikaZDanegoDnia,  
                    Mode=TwoWay}" HorizontalOptions="Center"/>  
                <Button Text="Wykonaj" Command="{Binding  
                    WartoscZamowienPracownikaZDanegoDniaCommand}"  
                    HorizontalOptions="Center" Margin="4"/>  
            </StackLayout>  
            <StackLayout BackgroundColor="LightBlue" Margin="4">  
                <Label Text="Wartosc Zamowien Wszystkich Pracownikow Z Danego  
                    Dnia: " HorizontalOptions="Center"/>  
                <Label Text="{Binding WartoscZamowienWszystkichPracownikowZDanegoDnia,  
                    Mode=TwoWay}" HorizontalOptions="Center"/>  
            </StackLayout>  
        </ContentPage.Content>  
    </ContentPage>
```

```

        <Button           Text="Wykonaj"           Command="{Binding
WartoscZamowienWszystkichPracownikowZDanegoDniaCommand}"
HorizontalOptions="Center" Margin="4"/>
    </StackLayout>

    <StackLayout BackgroundColor="LightBlue" Margin="4">
        <Label   Text="Wartosc Wszystkich Zamowien Pracownika:  "
HorizontalOptions="Center"/>
        <Label   Text="{Binding     WartoscWszystkichZamowienPracownika,
Mode=TwoWay}" HorizontalOptions="Center"/>
        <Button           Text="Wykonaj"           Command="{Binding
WartoscWszystkichZamowienPracownikaCommand}"   HorizontalOptions="Center"
Margin="4"/>
    </StackLayout>

</StackLayout>
</ContentPage.Content>
</ContentPage>
```

10.Edytuj

Views->WartoscZamowienPage.xaml-

>**WartoscZamowienPage.xaml.cs** i i zmień jej kod według wzoru:

```
[XamlCompilation(XamlCompilationOptions.Compile)]
```

```
public partial class WartoscZamowienPage : ContentPage
{
```

```
    public WartoscZamowienPage()
    {
        InitializeComponent();
        BindingContext = new WartoscZamowienViewModel();
    }
```

```
}
```

11.Edytuj plik **AppShell.xaml** i i zmień jego sekcję **FlyoutItem** według wzoru:

```

<FlyoutItem FlyoutDisplayOptions="AsMultipleItems">
    <ShellContent Title="About" Icon="tab_about.png" Route="AboutPage"
ContentTemplate="{DataTemplate local:AboutPage}" />
    <ShellContent           Title="Browse"           Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:ItemsPage}" />
```

```
<ShellContent Title="Klienci" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:KlienciPage}" />
<ShellContent Title="Zamowienia" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:ZamowieniaPage}" />
<ShellContent Title="Raporty" Icon="tab_feed.png"
ContentTemplate="{DataTemplate local:WartoscZamowienPage}" />
</FlyoutItem>
```

12.Edytuj plik **App.xaml->App.xaml.cs** i zmień jego konstruktor według wzoru:

```
public App()
{
    InitializeComponent();
    DependencyService.Register<MockDataStore>();
    DependencyService.Register<KlientDataStore>();
    DependencyService.Register<ZamowienieDataStore>();
    DependencyService.Register<WartoscZamowienDataStore>();
```

```
DependencyService.Register<ServiceReferenceZamowienia.ServiceZamowieniaClient>();
    MainPage = new AppShell();
}
```

13.Edytuj plik **AppShell.xaml->AppShell.xaml.cs** i zmień jego konstruktor według wzoru:

```
public AppShell()
{
    InitializeComponent();
    Routing.RegisterRoute(nameof(ItemDetailPage), typeof(ItemDetailPage));
    Routing.RegisterRoute(nameof(NewItemPage), typeof(NewItemPage));
    Routing.RegisterRoute(nameof(NowyKlientPage), typeof(NowyKlientPage));
    Routing.RegisterRoute(nameof(NoweZamowieniePage),
    typeof(NoweZamowieniePage));
    Routing.RegisterRoute(nameof(WartoscZamowienPage),
    typeof(WartoscZamowienPage));
}
```