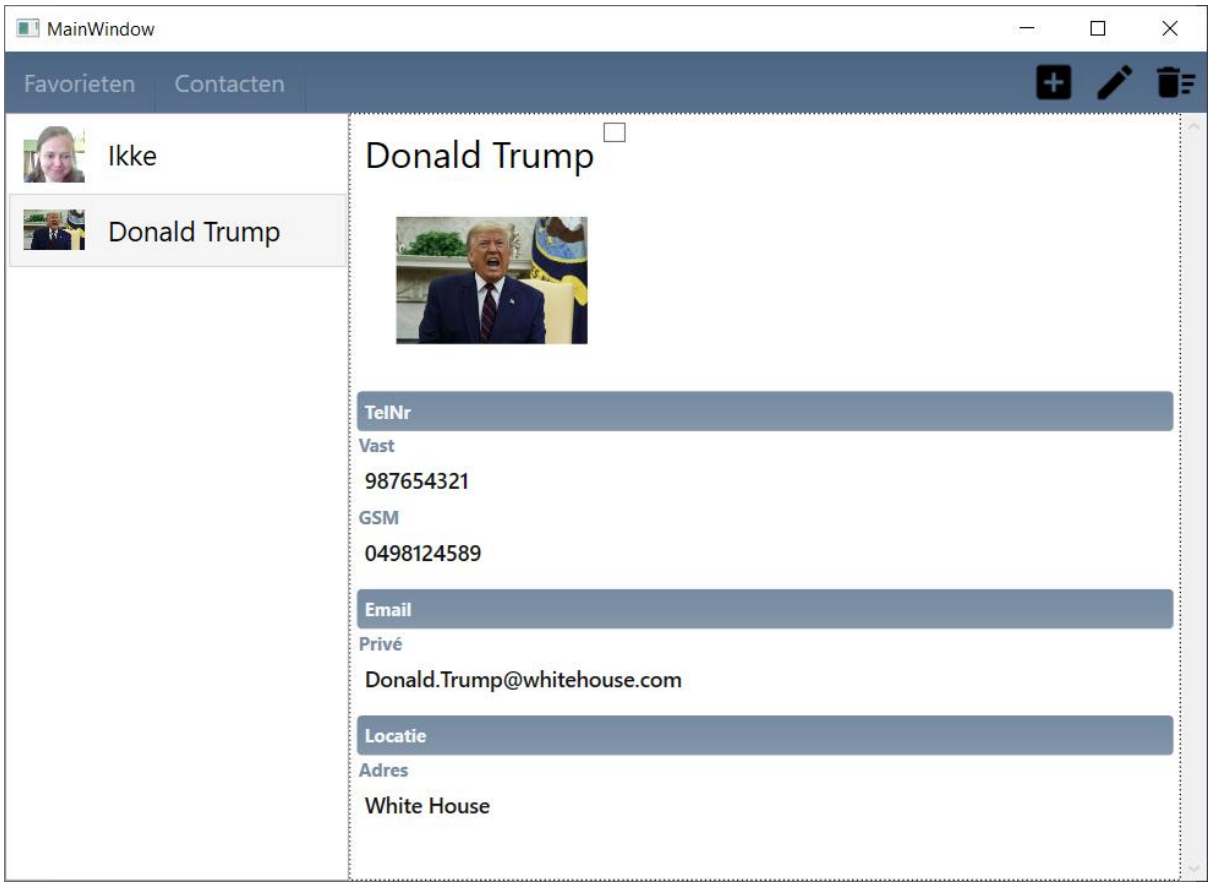
Stap voor stap Contacten WPF .NET Core MVVM app maken

In deze stap-voor-stap cursus leer je een eenvoudige MVVM WPF Core 3.1 applicatie maken, met gebruik van de belangrijkste controls en functionaliteiten, zoals CRUD operaties

De app houdt gegevens van contactpersonen bij en laat eveneens toe om een lijst van contactpersonen te tonen, filteren en gegevens toe te voegen, wijzigen en te verwijderen



1) Maak een nieuw WPF .Net Core 3.1 App in VS 2019 met naam `Contacten1`

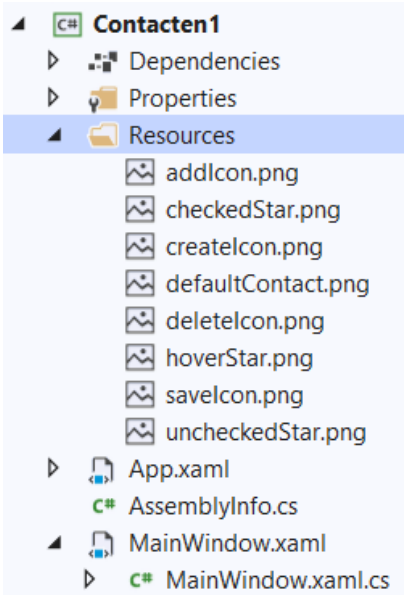
2) Open App.xaml set Brush resources in <Application.Resources>:

```
<Application x:Class="Contacten1.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Contacten1"
             StartupUri="MainWindow.xaml">
    <Application.Resources>

        <!--<DataTemplate DataType="{x:Type vm:BoekViewModel}">
            <views:BoekView/>
        </DataTemplate>-->

        <BitmapImage x:Key="DefaultContactImage"  UriSource="pack://siteoforigin:,,,/Resources/defaultContact.png"/>

        <!--<helper:BoolToVisibilityConverter x:Key="IsEditConverter"/>
        <helper:NullToVisibilityConverter x:Key="SelectedContactConverter"/>-->
```

```
<!--Brushes (Palette from: https://flatuicolors.com/palette/de)-->
<SolidColorBrush x:Key="BlueHorizon" Color="#FF4B6584"/>
<SolidColorBrush x:Key="BlueGrey" Color="#FF778CA3"/>

<SolidColorBrush x:Key="Innuendo" Color="#FFA5b1C2"/>
<SolidColorBrush x:Key="TwinkleBlue" Color="#FFD1D8E0"/>

<SolidColorBrush x:Key="GloomyPurple" Color="#FF8854D0"/>
<SolidColorBrush x:Key="LighterPurple" Color="#FFA55EEA"/>

<SolidColorBrush x:Key="RoyalBlue" Color="#FF3867D6"/>
<SolidColorBrush x:Key="NTSC" Color="#FF4B7BEC"/>

<SolidColorBrush x:Key="BoyZone" Color="#FF2D98DA"/>
<SolidColorBrush x:Key="HighBlue" Color="#FF45AAF2"/>

<SolidColorBrush x:Key="AlgalFuel" Color="#FF20BF6B"/>
<SolidColorBrush x:Key="ReptileGreen" Color="#FF26DE81"/>

<SolidColorBrush x:Key="TurquioseTopaz" Color="#FF0FB9B1"/>
<SolidColorBrush x:Key="MaximumBlueGreen" Color="#FF2BCBBA"/>

<SolidColorBrush x:Key="AlmostBlack" Color="#FF131313"/>

<!--Gradient Brushes-->
<LinearGradientBrush x:Key="BlueGreyGradient" EndPoint="0,1">
    <GradientStop Color="#FF4B6584" Offset="0.0" />
    <GradientStop Color="#FF526B89" Offset="0.5" />
    <GradientStop Color="#FF59718F" Offset="1.0" />
</LinearGradientBrush>

<LinearGradientBrush x:Key="LightBlueGreyGradient" EndPoint="0,1">
    <GradientStop Color="#FF778CA3" Offset="0.0" />
    <GradientStop Color="#FF7B8FA4" Offset="0.5" />
    <GradientStop Color="#FF8798AA" Offset="1.0" />
</LinearGradientBrush>

<!-- Controls -->
<Style x:Key="MenuTabButton" TargetType="Button">
    <Setter Property="Background" Value="{StaticResource BlueGreyGradient}"/>
    <Setter Property="Foreground" Value="{StaticResource Innuendo}"/>
    <Setter Property="BorderBrush" Value="{StaticResource BlueHorizon}"/>
    <Setter Property="BorderThickness" Value="0,0,1,0"/>
    <Setter Property="FontSize" Value="16"/>
</Style>

<Style x:Key="MenuIconButton" TargetType="Button">
    <Setter Property="Background" Value="{StaticResource BlueGreyGradient}"/>
    <Setter Property="Foreground" Value="{StaticResource MaximumBlueGreen}"/>
    <Setter Property="Padding" Value="5"/>
    <Setter Property="BorderThickness" Value="0"/>
</Style>

<Style x:Key="DetailLabel" TargetType="Label">
    <Setter Property="Foreground" Value="{StaticResource BlueGrey}"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Padding" Value="1"/>
</Style>

<Style x:Key="ContentLabel" TargetType="Label">
    <Setter Property="Foreground" Value="{StaticResource AlmostBlack}"/>
    <Setter Property="FontWeight" Value="DemiBold"/>
    <Setter Property="FontSize" Value="14"/>
</Style>

<Style x:Key="DetailBorder" TargetType="Border">
    <Setter Property="CornerRadius" Value="3"/>
    <Setter Property="Background" Value="{StaticResource LightBlueGreyGradient}"/>
</Style>

<Style x:Key="EditBox" TargetType="{x:Type TextBoxBase}">
    <Setter Property="SnapsToDevicePixels" Value="True"/>
    <Setter Property="OverridesDefaultStyle" Value="True"/>
    <Setter Property="KeyboardNavigation.TabNavigation" Value="None"/>
    <Setter Property="FocusVisualStyle" Value="{x:Null}"/>
    <Setter Property="MinWidth" Value="120"/>
    <Setter Property="MinHeight" Value="20"/>
    <Setter Property="AllowDrop" Value="true"/>
    <Setter Property="Template">
        <Setter.Value>
            <!-- What are control templates? Visit: https://dzone.com/articles/control-templates-wpf -->
            <ControlTemplate TargetType="{x:Type TextBoxBase}">
                <Border Name="Border" CornerRadius="3" Padding="2"
                        Background="White"
                        BorderBrush="{StaticResource BlueHorizon}"
                        BorderThickness="1" >

                    <!-- What is this? Visit: http://paulstovell.com/blog/wpf-part-names -->
                    <ScrollViewer Margin="0" x:Name="PART_ContentHost"/>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsEnabled" Value="False">
                        <Setter TargetName="Border" Property="Background" Value="#FFa5b1c2"/>
                        <Setter TargetName="Border" Property="BorderBrush" Value="#FF69717C"/>
                        <Setter Property="Foreground" Value="#FFd1d8e0"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
```

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

            <Style x:Key="favoriteCheckbox" TargetType="{x:Type CheckBox}">
                <Setter Property="Template">
                    <Setter.Value>
                        <ControlTemplate TargetType="{x:Type CheckBox}">
                            <StackPanel Orientation="Horizontal">
                                <Image x:Name="checkboxImage" Source="pack://siteoforigin:,,,/Resources/uncheckedStar.png"
        Width="32"/>
                                <ContentPresenter/>
                            </StackPanel>

                            <ControlTemplate.Triggers>

                                <Trigger Property="IsChecked" Value="True">
                                    <Setter TargetName="checkboxImage" Property="Source"
        Value="pack://siteoforigin:,,,/Resources/checkedStar.png"/>
                                </Trigger>

                                <MultiTrigger>
                                    <MultiTrigger.Conditions>
                                        <Condition Property="IsMouseOver" Value="True"/>
                                        <Condition Property="IsChecked" Value="False"/>
                                    </MultiTrigger.Conditions>
                                    <Setter TargetName="checkboxImage" Property="Source"
        Value="pack://siteoforigin:,,,/Resources/hoverStar.png"/>
                                </MultiTrigger>

                            </ControlTemplate.Triggers>

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

        </Application.Resources>
</Application>
```

3) Kopieer de Resources folder met images (van Github) onder je projectfolder :



3) Kies in Menu Project/Properties/Links Resources klikken/ Klik naar images en kies Add Existing File (kies meerdere images in Resource dir)

4) Open MainWindow.xaml. Verwijder de <Grid> </Grid> en zet in de plaats een <DockPanel> met 2 grids erin

```
<Window x:Class="Contacten1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Contacten1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
        <DockPanel>
        <!--Bovenste grid bevat Toolbar-->
        <Grid DockPanel.Dock="Top" Height="40" Background="{StaticResource BlueGreyGradient}">
        </Grid>
        <!--Inhoud-->
        <Grid DockPanel.Dock="Bottom">
        </Grid>
    </DockPanel>
</Window>
```

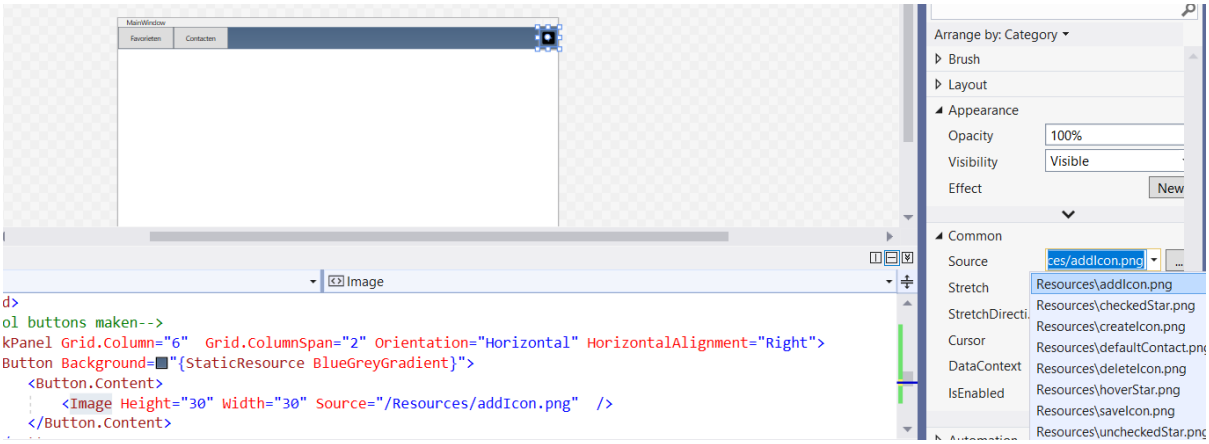
4) Open MainWindow.xaml. Verwijder de <Grid> </Grid> en zet in de plaats een <DockPanel> met 2 grids erin

5) In de Bovenste Grid (voor de toolbar) gaan we een aantal buttons toevoegen. In deze grid voegen we 8 kolommen toe (via ColumnDefinition tags) In kolom index 0 plaatsen we de button Favorieten, in de kolom met index 1 plaatsen we de button Contacten. In de Grid vanaf index 6 zetten we een Stackpanel die 2 kolommen overspant, Horizontale oriëntatie heeft en rechts gealigneerd is (`<StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right")..`)

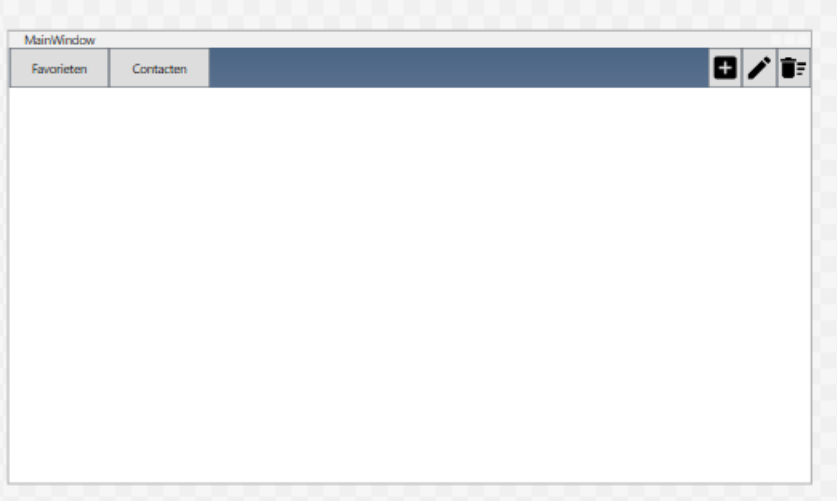
In deze Stackpanel plaatsen we een 3 Button met een Image om een contactpersoon toe te voegen, de wijzigen of de verwijderen.

```
<Window x:Class="Contacten1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Contacten1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <DockPanel>
        <!--Bovenste grid bevat Toolbar-->
        <Grid DockPanel.Dock="Top" Height="40" Background="{StaticResource BlueGreyGradient}">
            <!--8 kolommen-->
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>
            <Grid Grid.Column="0">
                <Button Content="Favorieten"/>
            </Grid>
            <Grid Grid.Column="1">
                <Button Content="Contacten"/>
            </Grid>
           <!--Tool buttons-->
            <StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right">
                <!--toevoegen-->
                <Button >
                    <Button.Content>
                        <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/addIcon.png"  />
                    </Button.Content>
                </Button>
                <!--Wijzigen -->
                <Button>
                    <Button.Content>
                        <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/createIcon.png"/>
                    </Button.Content>
                </Button>

                <!--Verwijderen-->
                <Button >
                    <Button.Content>
                        <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/deleteIcon.png"/>
                    </Button.Content>
                </Button>
            </StackPanel>
        </Grid>
        <!--Inhoud-->
        <Grid DockPanel.Dock="Bottom">
        </Grid>
    </DockPanel>
</Window>
```

Om de Source van de image in de vullen, druk op de F4 toets terwijl je de Image tag is geselecteerd. Je krijgt dan rechts het properties venster te zien. Je kan dan een image kiezen in de Source drop-down box :



Nu zie je in de designer-venster het volgende :



6) We willen de 3 rechtse buttons met een donkerblauwe achtergrond. Hiervoor is reeds een StaticResource `BlueGreyGradient` aangemaakt in de Application.Resources (in App.Xaml) met kleurdefinitie. We gebruiken dit om de achtergrondkleur in te stellen van de 3 buttons :

```
<Button Background="{StaticResource BlueGreyGradient}">
```

```
<!--Tool buttons-->
 <StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right">
     <!--toevoegen-->
     <Button Background="{StaticResource BlueGreyGradient}">
         <Button.Content>
             <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/addIcon.png"  />
         </Button.Content>
     </Button>
     <!--Wijzigen -->
     <Button Background="{StaticResource BlueGreyGradient}">
         <Button.Content>
             <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/createIcon.png"/>
         </Button.Content>
     </Button>

     <!--Verwijderen-->
     <Button Background="{StaticResource BlueGreyGradient}">
         <Button.Content>
             <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/deleteIcon.png"/>
         </Button.Content>
     </Button>
 </StackPanel>
```

7) De 3 buttons rechts hebben nu een donkerblauwe achtergrondkleur:

7) We plaatsen nu een ContentControl in de onderste grid van het dockpanel. Later gaan we dit verder invullen

.....

```
            </StackPanel>
        </Grid>
        <!--Inhoud-->
        <Grid DockPanel.Dock="Bottom">
            <ContentControl Content=""/>
        </Grid>
    </DockPanel>
</Window>
```

8) Voor de styling van de Buttons in de de toolbar, gaan we in de App.xaml een paar Style definities toevoegen: Een style voor de 2 buttons aan de linkerkant en een style voor de buttons aan de rechterkant
Open de App.xaml en plaats de volgende 2 style definities onderaan vóór </Application.Resources>

```
        <!--Button styles in toolbar -->
        <Style x:Key="MenuTabButton" TargetType="Button">
            <Setter Property="Background" Value="{StaticResource BlueGreyGradient}"/>
            <Setter Property="Foreground" Value="{StaticResource Innuendo}"/>
            <Setter Property="BorderBrush" Value="{StaticResource BlueHorizon}"/>
            <Setter Property="BorderThickness" Value="0,0,1,0"/>
            <Setter Property="FontSize" Value="16"/>
        </Style>
```

```
        <Style x:Key="MenuIconButton" TargetType="Button">
            <Setter Property="Background" Value="{StaticResource BlueGreyGradient}"/>
            <Setter Property="Foreground" Value="{StaticResource MaximumBlueGreen}"/>
            <Setter Property="Padding" Value="5"/>
            <Setter Property="BorderThickness" Value="0"/>
        </Style>

    </Application.Resources>
</Application>
```

9) Open nu terug MainWindow.xaml. We gaan nu voor de linker 2 buttons (Favorieten en Contacten) in de toolbar de style voor MenuTabButton toekennen:

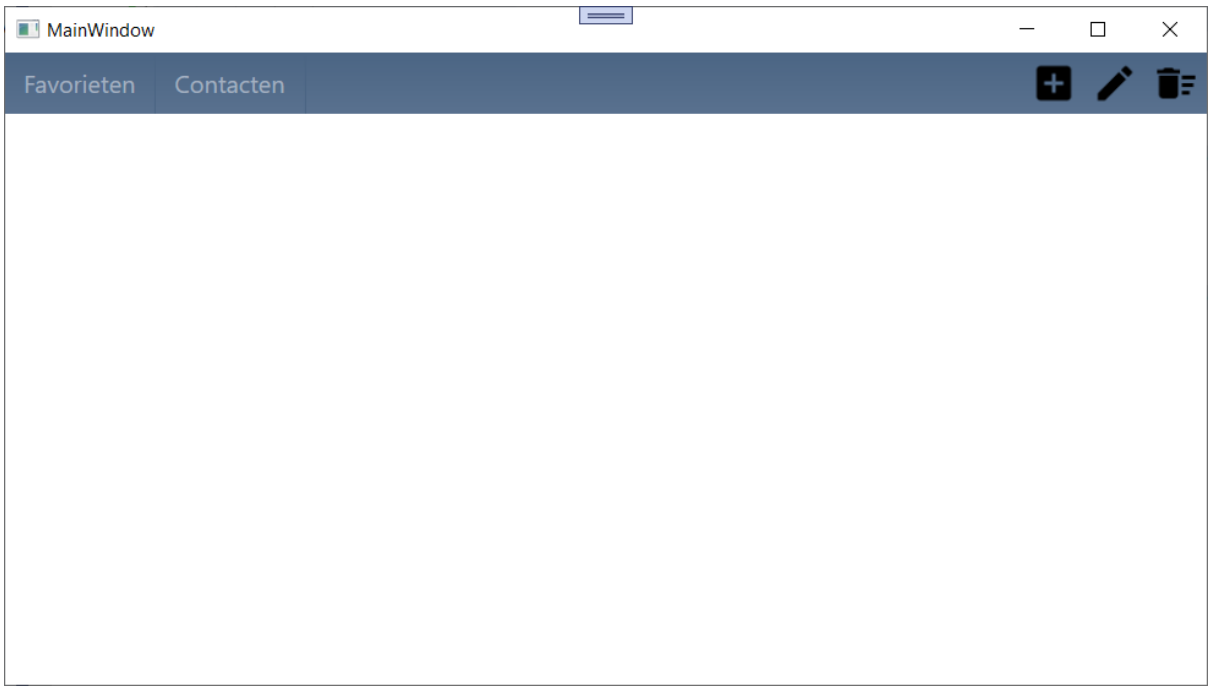
```
<!--Bovenste grid bevat Toolbar-->
<Grid DockPanel.Dock="Top" Height="40" Background="{StaticResource BlueGreyGradient}">
    <!--8 kolommen-->
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid Grid.Column="0">
        <Button  Style="{StaticResource }" Content="Favorieten"/>
    </Grid>
    <Grid Grid.Column="1">
        <Button Content="Contacten"/>
    </Grid>
   <!--Tool buttons-->
    <StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right">
        <!--toevoegen-->
```

MenuIconButton
MenuTabButton
ResourceKey

En voor de rechtse 3 buttons vervangen we de Backgrouncolor door de Style voor StaticResource MenuIconButton:

```
    </Grid.ColumnDefinitions>
    <Grid Grid.Column="0">
        <Button Style="{StaticResource MenuTabButton}" Content="Favorieten"/>
    </Grid>
    <Grid Grid.Column="1">
        <Button Style="{StaticResource MenuTabButton}" Content="Contacten"/>
    </Grid>
   <!--Tool buttons-->
    <StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right">
        <!--toevoegen-->
        <Button Style="{StaticResource MenuIconButton}">
            <Button.Content>
                <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/addIcon.png"  />
            </Button.Content>
        </Button>
        <!--Wijzigen -->
        <Button Style="{StaticResource MenuIconButton}">
            <Button.Content>
                <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/createIcon.png"/>
            </Button.Content>
        </Button>
        <!--Verwijderen-->
        <Button Style="{StaticResource MenuIconButton}">
            <Button.Content>
                <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/deleteIcon.png"/>
            </Button.Content>
        </Button>
    </StackPanel>
```

De 2 linker en 3 rechtse buttons worden nu mooier getoond:



(Source code tot nu Contacten1B op GitHub)

10) Maak nu voor MVVM de folders Models, Views en ViewModels aan in het project. Maak een nieuwe klasse ObservableObject aan onder de folder Utility met de volgende code :

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace Contacten1.Utilty
{
    public class ObservableObject : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }

        protected virtual bool OnPropertyChanged<T>(ref T backingField, T value,
[CallerMemberName] string propertyName = "")
        {
            if (EqualityComparer<T>.Default.Equals(backingField, value))
                return false;

            backingField = value;
            OnPropertyChanged(propertyName);
            return true;
        }
    }
}
```

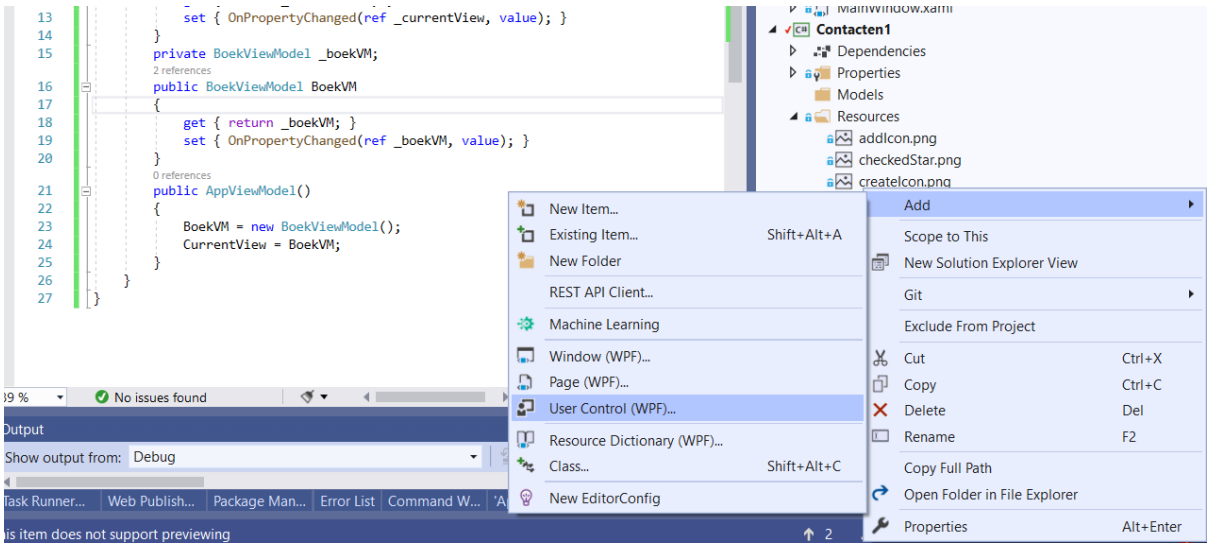
11) Maak klassen BoekViewModel en ContactenViewModel aan onder de folder ViewModels, die beiden afgeleide klassen zijn van `ObservableObject`

```
public class BoekViewModel : ObservableObject
{

}

public class ContactenViewModel : ObservableObject
{}
```

12) Maak nu een klasse AppViewModel aan onder de folder ViewModels, die ook een afgeleide klasse is van `ObservableObject`. Hierin maken we public properties om de huidige View bij te houden, een field en lees/schrijf- Property om een verwijzing naar het BoekViewModel bij te houden en een constructor die een BoekViewModel te initialiseren:

```
namespace Contacten1.ViewModels
{
    public class AppViewModel : ObservableObject
    {
        private object _currentView;
        public object CurrentView
        {
            get { return _currentView; }
            set { OnPropertyChanged(ref _currentView, value); }
        }
        private BoekViewModel _boekVM;
        public BoekViewModel BoekVM
        {
            get { return _boekVM; }
            set { OnPropertyChanged(ref _boekVM, value); }
        }
        public AppViewModel()
        {
            BoekVM = new BoekViewModel();
            CurrentView = BoekVM;
        }
    }
}
```

13) **Maak onder de folder Views een usercontrol met de naam BoekView.xaml (Rechtsklik op de folder Views/Add/User Control(WPF)).**



Zet de backgroundColor van de usercontrol BoekView op lichtblauw:

```
<UserControl x:Class="Contacten1.Views.BoekView"
             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:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800" Background="LightBlue">
    <Grid>

    </Grid>
</UserControl>
```

14) **Maak onder de folder Views een usercontrol met de naam ContactView.xaml (Rechtsklik op de folder Views/Add/User Control(WPF)). Voorlopig laten we de View leeg.**

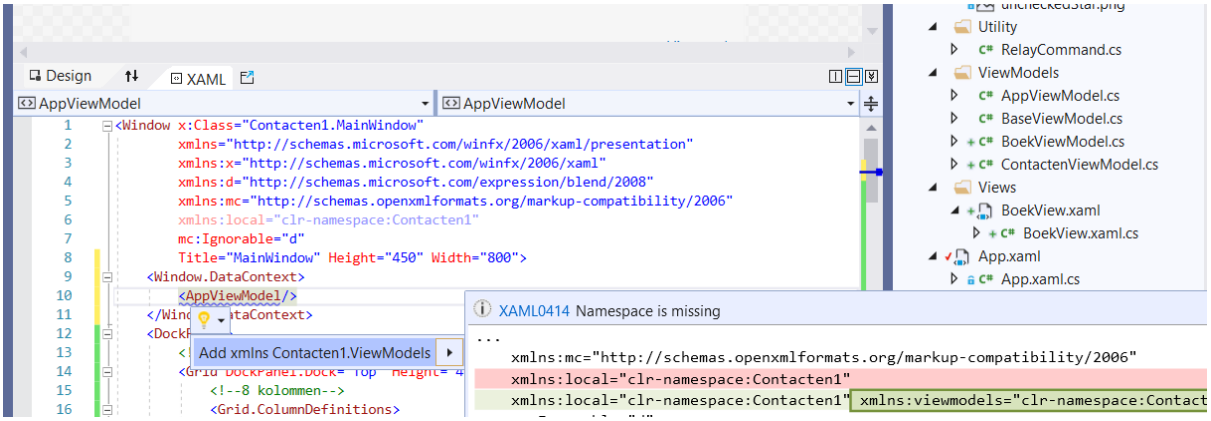
15) **Open App.xaml. We maken een DataTemplate voor het Type BoekViewModel voor View BoekView aan tussen de tags <Application.Resources> en </Application.Resources>**
Om de juiste namespace toe te voegen, voeg bovenaan de volgende namespace toe :

```
<Application x:Class="Contacten1.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Contacten1"
             xmlns:vm="clr-namespace:Contacten1.ViewModels"
             xmlns:views="clr-namespace:Contacten1.Views"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <DataTemplate DataType="{x:Type vm:BoekViewModel}">
            <views:BoekView/>
        </DataTemplate>
          ….

</Application.Resources>
```

16) **Open MainWindow.xaml. We gaan nu de DataContext van MainWindow en de BoekView binden (BoekView zal getoond worden in het onderste gedeelte van de Dockpanel).**
Open MainWindow.xaml en zet boven het Dockpanel tag de DataContext op AppViewModel. Bovenaan voeg je de namespace toe naar Contacten1.ViewModels:

```
<Window x:Class="Contacten1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Contacten1"
        xmlns:viewmodels="clr-namespace:Contacten1.ViewModels"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Window.DataContext>
        <viewmodels:AppViewModel/>
    </Window.DataContext>
    <DockPanel>
   ….
```

Tip : Om de namespace toe te voegen kan je ook op het lampje klikken en «Add xmlns Contacten1.ViewModels » selecteren
“

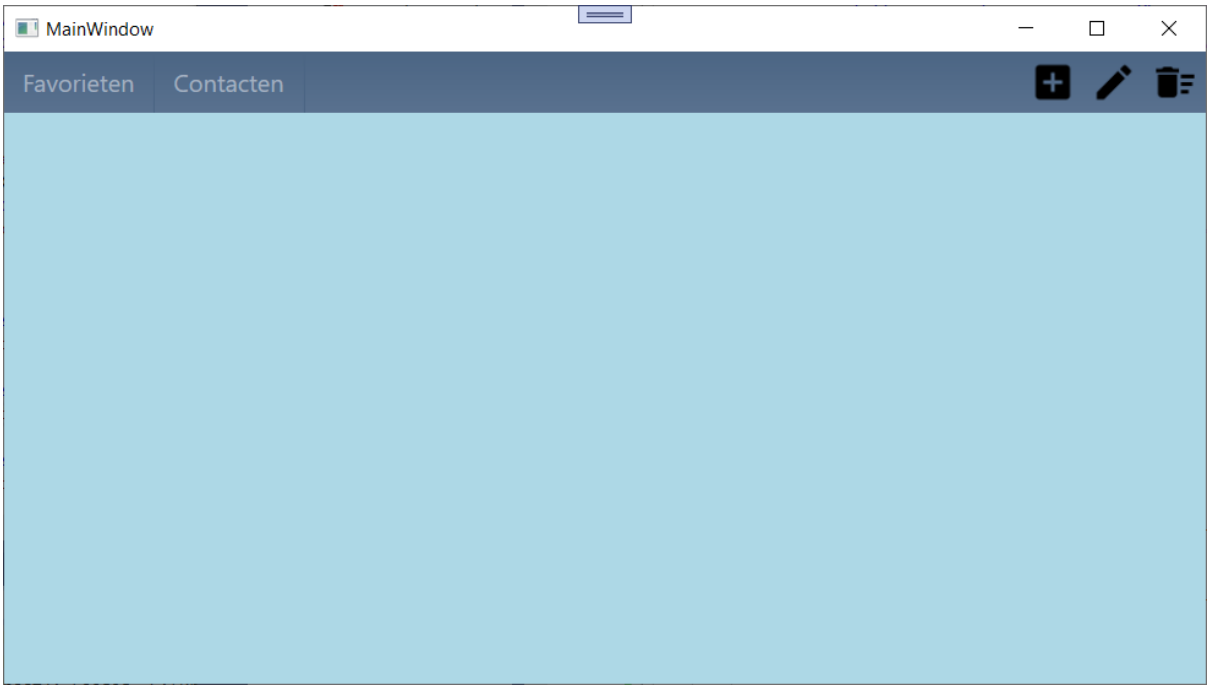


17) **Voeg onderaan in MainWindow.xaml aan de tag <ContentControl een binding voor de Content toe naar het Property CurrentView (van AppViewModel klasse):**

…

```
        <!--Inhoud-->
        <Grid DockPanel.Dock="Bottom">
            <ContentControl Content="{Binding CurrentView}"/>
        </Grid>
    </DockPanel>
</Window>
```

18) Run de app, als de binding goed is gelukt, krijg je het volgende scherm te zien:



(Code tot nu op Github : Contacten1C)

19) Maak onder de folder Models een klasse Contact aan, om gegevens van 1 contactpersoon bij te houden. Deze klasse is afgeleid van ObservableObject

```
using Contacten1.Utilty;

namespace Contacten1.Models
{
    public class Contact : ObservableObject
    {
        private string _naam;
        public string Naam
        {
            get { return _naam; }
            set { OnPropertyChanged(ref _naam, value); }
        }

        private string[] _telNr;
        public string[] TelNr
        {
            get { return _telNr; }
            set { OnPropertyChanged(ref _telNr, value); }
        }

        private string _email;
        public string Email
        {
            get { return _email; }
            set { OnPropertyChanged(ref _email, value); }
        }

        private string _locatie;
        public string Locatie
        {
```

```
            get { return _locatie; }
            set { OnPropertyChanged(ref _locatie, value); }
        }

        private bool _isFavoriet;
        public bool IsFavoriet
        {
            get { return _isFavoriet; }
            set { OnPropertyChanged(ref _isFavoriet, value); }
        }

        private string _imagePad;
        public string ImagePad
        {
            get { return _imagePad; }
            set { OnPropertyChanged(ref _imagePad, value); }
        }
    }


}
```

20) Open ContactenViewModel.cs en voeg de volgende code toe :

```
using Contacten1.Models;
using Contacten1.Utilty;
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace Contacten1.ViewModels
{
    public class ContactenViewModel : ObservableObject
    {
        private Contact _selectedContact;
        public Contact SelectedContact
        {
            get { return _selectedContact; }
            set { OnPropertyChanged(ref _selectedContact, value); }
        }

        public ObservableCollection<Contact> Contacten { get; private set; }

        public ContactenViewModel()
        {

        }

        public void ContactenLaden(IEnumerable<Contact> contacten)
        {
            Contacten = new ObservableCollection<Contact>(contacten);
            OnPropertyChanged(nameof(Contacten));
        }

    }
}
```

21) Open BoekViewModel.cs en voeg de volgende code toe :

```
-Een public Property ContactenVM met field met lees-en schrijftoegang om een
referentie bij te houden naar ContactenViewModel
-Een Constructor om ContactenVM te initialiseren
-2 methoden om AlleContacten te laten en om enkel de favoriete contacten te laden
```

```
using Contacten1.Utilty;


namespace Contacten1.ViewModels
{
    public class BoekViewModel : ObservableObject
    {
        private ContactenViewModel _contactenVM;
        public ContactenViewModel ContactenVM
        {
            get { return _contactenVM; }
            set { OnPropertyChanged(ref _contactenVM, value); }
        }


        public BoekViewModel()
        {
            ContactenVM = new ContactenViewModel();
        }

        private void LadenAlleContacten()
        {

        }

        private void LadenFavorieten()
        {

        }
    }
   }
```

De code tot nu op GitHub : Contacten1D

22) Voeg verder aan BoekViewModel.cs de volgende code toe :

Om de Command Acties op de Buttons «Favorieten » en «Contacten» te plaatsen, voeg het volgende toe :

2 properties van het type ICommand :

```
        public ICommand CommandLadenAlleContacten { get; private set; }
        public ICommand CommandLadenFavorieten { get; private set; }
```

2 lijnen toevoegen aan de Constructor om deze Properties te initialiseren,zodat te verwijzen naar de methoden LadenAlleContacten() en LadenFavorieten

```
            CommandLadenAlleContacten = new RelayCommand(LadenAlleContacten);
            CommandLadenFavorieten = new RelayCommand(LadenFavorieten);
```

```
using Contacten1.Utilty;
using System.Windows.Input;

namespace Contacten1.ViewModels
{
    public class BoekViewModel : ObservableObject
    {
        private ContactenViewModel _contactenVM;
        public ContactenViewModel ContactenVM
```

```
        {
            get { return _contactenVM; }
            set { OnPropertyChanged(ref _contactenVM, value); }
        }

        public ICommand CommandLadenAlleContacten { get; private set; }
        public ICommand CommandLadenFavorieten { get; private set; }
        public BoekViewModel()
        {
            ContactenVM = new ContactenViewModel();
            CommandLadenAlleContacten = new RelayCommand(LadenAlleContacten);
            CommandLadenFavorieten = new RelayCommand(LadenFavorieten);
        }

        private void LadenAlleContacten()
        {

        }

        private void LadenFavorieten()
        {

        }
    }
}
```

23) We maken nu dataservices aan die we kunnen aanroepen in LadenAlleContacten() en LadenFavorieten()

Eers maken we een interface IContactenDataService aan, omdat we gemakkelijk willen kunnen veranderen van soort DataService (bv eerst een MockDataService, dan een JsonDataService en ten slotte een SqlServerDataService)

Maak eerst een nieuwe folder Services aan in het project

Maak een interface `IContactDataService` onder de folder Services :

```
using Contacten1.Models;
using System.Collections.Generic;
namespace Contacten1.Services
{
    public interface IContactDataService
    {
        IEnumerable<Contact> GeefContacten();
        void Bewaar(IEnumerable<Contact> contacten);
    }
}
```

24) We starten met het aanmaken van de MockDataService klasse. Maak deze aan onder de folder Services. Deze klasse implementeert de interface `IContactDataService`

```
using Contacten1.Models;
using System;
using System.Collections.Generic;
using System.Text;

namespace Contacten1.Services
{
    public class MockDataService : IContactDataService
    {
        private IEnumerable<Contact> _contacten;

        public MockDataService()
        {
            _contacten = new List<Contact>()
            {
                new Contact
                {
                    Naam = "Jan Jansens",
                    TelNr = new string[]{"123456789","047202515" },
                    Email ="jan.jansens@personal.com",
                    Locatie = "Kerkstraat 2, 9000 Gent"
                },
                new Contact
                {
                    Naam = "Piet Pieters",
                    TelNr =new string[]{"987654321" ,"0498124589"},
                    Email= "Piet.pieters@personal.com",
                    Locatie = "Molenstraat 8, 8500 Brugge"
                }
            };
        }

        public void Bewaar(IEnumerable<Contact> contacten)
        {
            _contacten = contacten;
        }

        public IEnumerable<Contact> GeefContacten()
        {
            return _contacten;
        }

    }
}
```

25) We kunnen nu terug verder naar BoekViewModel.cs We willen vanuit de methoden LadenAllContacten() en LadenFavorieten() de MockDataService aanroepen
Voeg aan de klass BoekViewModel

- **een private field _dataService toe van het type IContactDataService**
- **Pas de constructor BoekViewModel aan zodat deze een parameter van het type IContactDataService heeft en initialiseer _dataService in de constructor**
- **Roep in de methoden LadenAllContacten() en LadenFavorieten() de MockDataService methode aan. In LadenFavorieten() maken we gebruik van Linq om een selectie uit de collectie te halen via .Where(lambda expressie). Hiervoor moet je de namespace using directive System.Linq bovenaan plaatsen.**

```
using Contacten1.Models;
using Contacten1.Services;
using Contacten1.Utilty;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Input;

namespace Contacten1.ViewModels
{

    public class BoekViewModel : ObservableObject
    {
        private IContactDataService _dataService;
        private ContactenViewModel _contactenVM;
        public ContactenViewModel ContactenVM
        {
            get { return _contactenVM; }
            set { OnPropertyChanged(ref _contactenVM, value); }
        }

        public ICommand CommandLadenAlleContacten { get; private set; }
        public ICommand CommandLadenFavorieten { get; private set; }
        public BoekViewModel(IContactDataService dataService)
        {
            _dataService = dataService;
            ContactenVM = new ContactenViewModel();
            CommandLadenAlleContacten = new RelayCommand(LadenAlleContacten);
            CommandLadenFavorieten = new RelayCommand(LadenFavorieten);
        }

        private void LadenAlleContacten()
        {
            IEnumerable<Contact> contacten = _dataService.GeefContacten();
            ContactenVM.ContactenLaden(contacten);
        }

        private void LadenFavorieten()
        {
            IEnumerable<Contact> favorieten = _dataService.GeefContacten().Where(c =>
c.IsFavoriet);
            ContactenVM.ContactenLaden(favorieten);
        }
    }
}
```

26) Open AppViewModel.cs We passen de aanroep naar de constructor van BoekViewModel aan, zodat we de MockDataService doorgeven als parameter.

```
using Contacten1.Services;
using Contacten1.Utilty;
using System;
using System.Collections.Generic;
using System.Text;

namespace Contacten1.ViewModels
{
    public class AppViewModel : ObservableObject
    {
        private object _currentView;
        public object CurrentView
        {
            get { return _currentView; }
            set { OnPropertyChanged(ref _currentView, value); }
        }
        private BoekViewModel _boekVM;
        public BoekViewModel BoekVM
        {
            get { return _boekVM; }
            set { OnPropertyChanged(ref _boekVM, value); }
        }
        public AppViewModel()
        {
            IContactDataService dataservice = new MockDataService();
            BoekVM = new BoekViewModel(dataservice);
            //BoekVM = new BoekViewModel();
            CurrentView = BoekVM;
        }
    }
}
```

(Code tot nu toe op GITHUB: Contacten1E)

27) Nu gaan we de layout verbeteren en aanvullen

Open App.xaml en maak een BitMapImage voor een default contact image :

```
<Application x:Class="Contacten1.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Contacten1"
             xmlns:vm="clr-namespace:Contacten1.ViewModels"
             xmlns:views="clr-namespace:Contacten1.Views"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <DataTemplate DataType="{x:Type vm:BoekViewModel}">
            <views:BoekView/>
        </DataTemplate>
        <BitmapImage x:Key="DefaultContactImage"
UriSource="pack://siteoforigin:,,,/Resources/defaultContact.png"/>
```

28) Maak een nieuwe usercontrol ContactInfoView.xaml onder de folder Views. Nu maken we de layout van deze View.

Maak 2 kolommen dvm ColumnDefinitions. De eerste kolom met breedte 50

In de kolom met index 0 plaatsen we een Image waarin we de de foto van de Contactpersoon tonen. We specifiëren de StaticResource DefaultContactImage als Fallback en TargetNullValue

In de kolom naast de image zetten we een stackpanel met een label die gebonden wordt aan de Property Naam van een Contact object. De FallBackValue (indien Niet Aanwezig is de tekst "N/A"

```
<UserControl x:Class="Contacten1.Views.ContactInfoView"
             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:Contacten1.Views"
             mc:Ignorable="d"
            d:DesignHeight="75" d:DesignWidth="250">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="50"/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>        <Image Grid.Column="0" Margin="5"
Source="{Binding ImagePad,
            FallbackValue={StaticResource
DefaultContactImage},TargetNullValue={StaticResource DefaultContactImage}}"/>
        <StackPanel Grid.Column="1" Margin="5" VerticalAlignment="Center">
            <Label FontSize="18" Content="{Binding Naam ,FallbackValue=N/A}"/>
        </StackPanel>
    </Grid>
</UserControl>
```

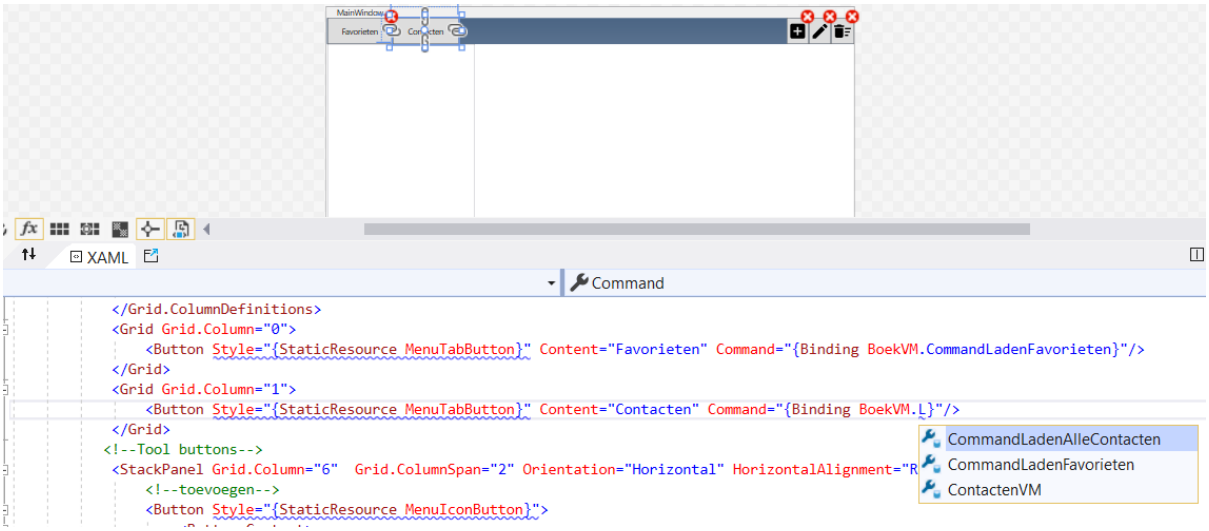
29) Open BoekView.xaml. We passen nu verder de layout aan.
Verwijder de Lichtblauwe achtergrondkleur van de usercontrol en zet de Background in de grid op White
Definieer 2 Kolommen via Grid.ColumnDefinitions, de eerste kolom heeft een breedte van 225
We tonen een ListView en definiëren een DataTemplate om de View ContactInfo refereren. We binden de ListView aan de public Property `ContactenVM` **van AppViewModel en daarvan de property** `Contacten.` **Het geselecteerde Item in de ListView wordt gebonden aan de Property** `ContactenVM.SelectedContact`

```
<UserControl x:Class="Contacten1.Views.BoekView"
             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:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid Background="White">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="225"/>
            <ColumnDefinition />
```

```
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="0">
            <ListView ItemsSource="{Binding ContactenVM.Contacten}" SelectedItem="{Binding
ContactenVM.SelectedContact}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <local:ContactInfoView/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </Grid>
        <Grid Grid.Column="1">

        </Grid>
    </Grid>
</UserControl>
```

30) Open MainWindow.xaml voorhet binden van de Commands van de buttons Contacten en Favorieten.

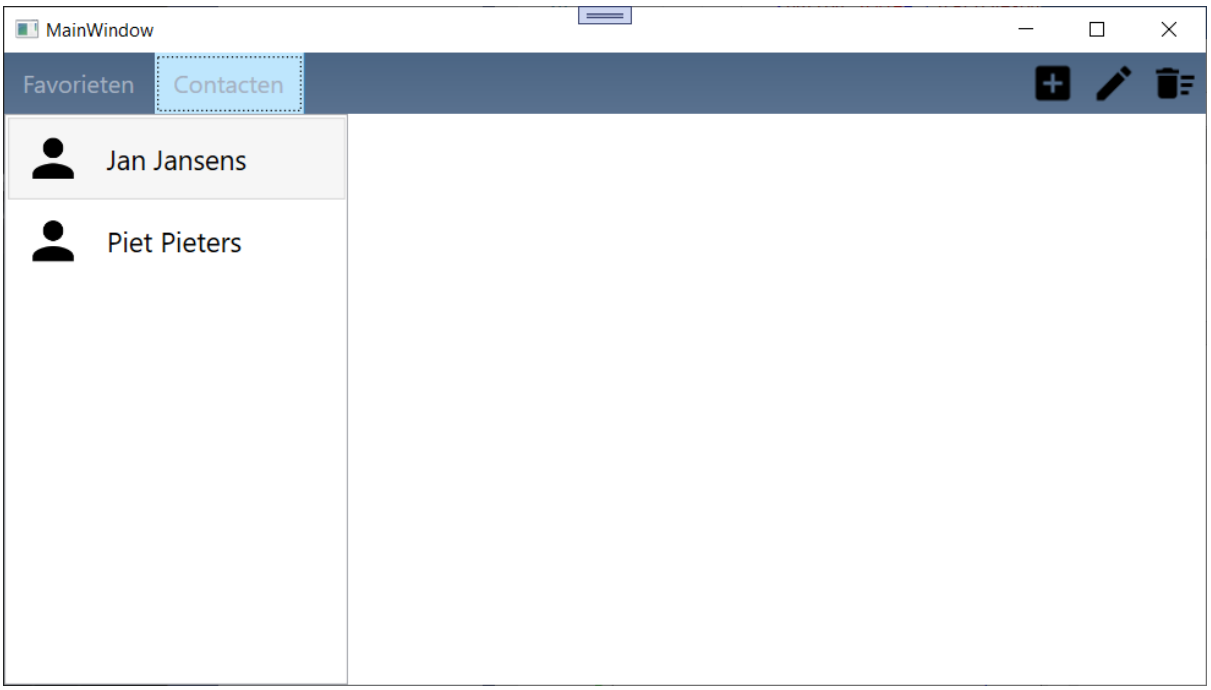
Om de acties bij het klikken op de buttons toe te voegen, moeten we de Command attributen van de buttons Binden :




```
<Window x:Class="Contacten1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Contacten1"
        xmlns:viewmodels="clr-namespace:Contacten1.ViewModels"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Window.DataContext>
        <viewmodels:AppViewModel/>
    </Window.DataContext>
    <DockPanel>
        <!--Bovenste grid bevat Toolbar-->
        <Grid DockPanel.Dock="Top" Height="40" Background="{StaticResource BlueGreyGradient}">
            <!--8 kolommen-->
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>
            <Grid Grid.Column="0">
                <Button Style="{StaticResource MenuTabButton}" Content="Favorieten" Command="{Binding
BoekVM.CommandLadenFavorieten}"/>
            </Grid>
```

```
            <Grid Grid.Column="1">
                <Button Style="{StaticResource MenuTabButton}" Content="Contacten" Command="{Binding
BoekVM.CommandLadenAlleContacten}"/>
            </Grid>
           <!--Tool buttons-->
            <StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right">
                <!--toevoegen-->
                <Button Style="{StaticResource MenuIconButton}">
                    <Button.Content>
                        <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/addIcon.png"  />
                    </Button.Content>
                </Button>
                <!--Wijzigen -->
                <Button Style="{StaticResource MenuIconButton}">
                    <Button.Content>
                        <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/createIcon.png"/>
                    </Button.Content>
                </Button>
                <!--Verwijderen-->
                <Button Style="{StaticResource MenuIconButton}">
                    <Button.Content>
                        <Image Height="30" Width="30" Source="pack://siteoforigin:,,,/Resources/deleteIcon.png"/>
                    </Button.Content>
                </Button>
            </StackPanel>
        </Grid>
        <!--Inhoud-->
        <Grid DockPanel.Dock="Bottom">
            <ContentControl Content="{Binding CurrentView}"/>
        </Grid>
    </DockPanel>
</Window>
```

Start de app. Als je op de Button Contacten klikt, krijg je Jan Jansens en Piet Pieters in de listview te zien



(Code tot nu toe op Github : Contacten1F)

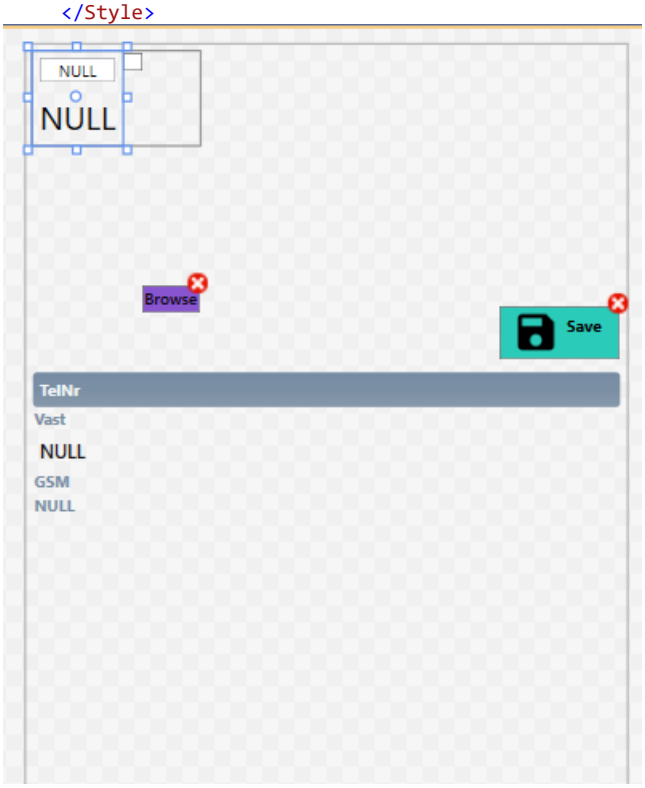
31) Maak onder de folder Views een nieuwe usercontrol ContactDetails.xaml 450 en plaats het volgende in de Grid :

```
<UserControl x:Class="Contacten1.Views.ContactDetails"
             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:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="1200" d:DesignWidth="450">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>

        <Grid Grid.Row="0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition Width="100"/>
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0" HorizontalAlignment="Left"
                        Margin="5">
                <StackPanel Orientation="Horizontal">
                    <StackPanel HorizontalAlignment="Center">

                        <TextBox Text="{Binding ElectedContact.Naam, FallbackValue=NULL, Mode=TwoWay}"
                                 VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
                                 Margin="5"/>

                        <Label Content="{Binding ElectedContact.Naam,FallbackValue=NULL}"
                               HorizontalAlignment="Center"
                               FontSize="24"/>
                    </StackPanel>
                    <CheckBox  IsChecked="{Binding SelectedContact.IsFavoriet}"  />
                </StackPanel>
                <Grid Height="125" Width="125">
                    <Image Height="125" Width="125"
                           Source="{Binding SelectedContact.ImagePad,
                        FallbackValue={StaticResource DefaultContactImage},
                        TargetNullValue={StaticResource DefaultContactImage}}"/>

                    <Button Style="{StaticResource MenuIconButton}" Background="{StaticResource GloomyPurple}"
                            VerticalAlignment="Bottom" HorizontalAlignment="Right"
                            Foreground="Black" FontWeight="DemiBold">
                        Browse
                    </Button>
                </Grid>
            </StackPanel>
            <StackPanel Grid.Column="1" VerticalAlignment="Bottom" Orientation="Horizontal">
                <Button Style="{StaticResource MenuIconButton}" Background="{StaticResource MaximumBlueGreen}"
Width="90" Height="40" Margin="5">
                    <Button.Content>
                        <StackPanel Orientation="Horizontal">
                            <Image Source="/Resources/saveIcon.png" />
                            <Label FontWeight="Bold">Save</Label>
                        </StackPanel>
                    </Button.Content>
                </Button>
            </StackPanel>
        </Grid>
       <!--Telefoon Details-->
        <StackPanel Grid.Row="1" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="TelNr"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Vast"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[0],FallbackValue=NULL}"/>

                    <Label Style="{StaticResource DetailLabel}"  Content="GSM"/>
                    <Label Style="{StaticResource DetailLabel}" Content="{Binding
SelectedContact.TelNr[1],FallbackValue=NULL}"/>
                </StackPanel>
            </StackPanel>
```

```
        </StackPanel>
    </Grid>
</UserControl>
```

32) Voeg de volgende Application.Resource styles toe aan App.xaml

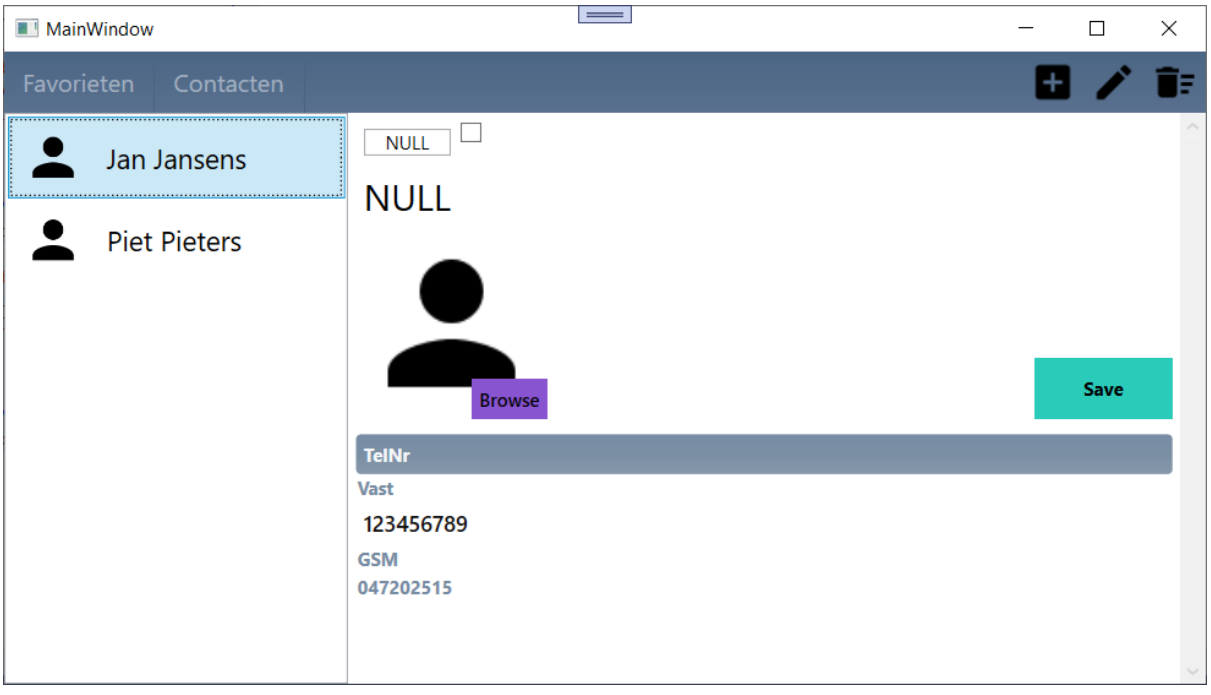
```
<Style x:Key="DetailLabel" TargetType="Label">
    <Setter Property="Foreground" Value="{StaticResource BlueGrey}"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Padding" Value="1"/>
</Style>

<Style x:Key="ContentLabel" TargetType="Label">
    <Setter Property="Foreground" Value="{StaticResource AlmostBlack}"/>
    <Setter Property="FontWeight" Value="DemiBold"/>
    <Setter Property="FontSize" Value="14"/>
</Style>

<Style x:Key="DetailBorder" TargetType="Border">
    <Setter Property="CornerRadius" Value="3"/>
    <Setter Property="Background" Value="{StaticResource LightBlueGreyGradient}"/>
</Style>
```



33) Open BoekView.xaml we voegen in de grid column=”1” een scrollviewer toe en daarin een ContentControl waar we de De ContactDetails usercontrol plaasten. We zetten eveneens de DataContext die we binden aan de property ContactenVM van AppViewModel klasse

```
<UserControl x:Class="Contacten1.Views.BoekView"
             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:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid Background="White">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="225"/>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="0">
            <ListView ItemsSource="{Binding ContactenVM.Contacten}"
SelectedItem="{Binding ContactenVM.SelectedContact}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <local:ContactInfoView/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </Grid>
        <Grid Grid.Column="1">
            <ScrollViewer>
                <ContentControl>
                    <ContentControl.Content>
                        <local:ContactDetails DataContext="{Binding ContactenVM}"/>
                    </ContentControl.Content>
                </ContentControl>
            </ScrollViewer>
        </Grid>
    </Grid>
</UserControl>
```

34) Start de app, aan de rechterkant zie je de ContactDetailsView. Wanneer je een selectie maakt uit de ListView, veranderen de Telefoonnummers



(code tot nu op Github : Contacten1G)

35) Open ContactDetails.xaml, voeg zoals de TelNr gegevens, ook controls toe voor de Email en Locatie.

```
<UserControl x:Class="Contacten1.Views.ContactDetails"
             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:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="1200" d:DesignWidth="450">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>

        <Grid Grid.Row="0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition Width="100"/>
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0" HorizontalAlignment="Left"
                        Margin="5">
                <StackPanel Orientation="Horizontal">
                    <StackPanel HorizontalAlignment="Center">

                        <TextBox Text="{Binding ElectedContact.Naam, FallbackValue=NULL, Mode=TwoWay}"
                                 VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
                                 Margin="5"/>

                        <Label Content="{Binding ElectedContact.Naam,FallbackValue=NULL}"
                               HorizontalAlignment="Center"
                               FontSize="24"/>
                    </StackPanel>
                    <CheckBox  IsChecked="{Binding SelectedContact.IsFavoriet}"  />
                </StackPanel>
                <Grid Height="125" Width="125">
                    <Image Height="125" Width="125"
                           Source="{Binding SelectedContact.ImagePad,
                        FallbackValue={StaticResource DefaultContactImage},
                        TargetNullValue={StaticResource DefaultContactImage}}"/>

                    <Button Style="{StaticResource MenuIconButton}" Background="{StaticResource GloomyPurple}"
                            VerticalAlignment="Bottom" HorizontalAlignment="Right"
                            Foreground="Black" FontWeight="DemiBold">
                        Browse
                    </Button>
                </Grid>
            </StackPanel>
            <StackPanel Grid.Column="1" VerticalAlignment="Bottom" Orientation="Horizontal">
                <Button Style="{StaticResource MenuIconButton}" Background="{StaticResource MaximumBlueGreen}"
Width="90" Height="40" Margin="5">
                    <Button.Content>
                        <StackPanel Orientation="Horizontal">
                            <Image Source="/Resources/saveIcon.png" />
                            <Label FontWeight="Bold">Save</Label>
                        </StackPanel>
                    </Button.Content>
                </Button>
            </StackPanel>
        </Grid>
       <!--Telefoon Details-->
        <StackPanel Grid.Row="1" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="TelNr"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Vast"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[0],FallbackValue=NULL}"/>

                    <Label Style="{StaticResource DetailLabel}"  Content="GSM"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[1],FallbackValue=NULL}"/>
                </StackPanel>
            </StackPanel>
```

```
        </StackPanel>
        <!--Email Details-->

        <StackPanel Grid.Row="2" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="Email"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Privé"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.Email,FallbackValue=NULL}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
<!--Locatie Details-->
```

```
        <StackPanel Grid.Row="3" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="Locatie"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Adres"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.Locatie,FallbackValue=NULL}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
    </Grid>
</UserControl>
```

36) Voeg eveneens een TextBox toe om toe te laten op de Naam van het contactPersoon aan te passen (zie gele marking in code boven). De textbox mag enkel zichtbaar worden wanneer de view in edit-mode is. We kunnen het tonen/verbergen van de Textbox instellen via een Converter klasse in WPF. Een Converter klasse moet de interfact `IValueConverter` Implementeren.

Maak een nieuwe folder Helpers en hieronder een nieuwe class bestand Converters.cs Plaats hierin de volgende code

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

namespace Contacten1.Helpers
{
    public class BoolToVisibilityConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            var boolValue = (bool)value;

            if (boolValue)
                return Visibility.Visible;

            return Visibility.Collapsed;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }

    public class NullToVisibilityConverter : IValueConverter
    {
```

```
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value != null)
                return Visibility.Visible;

            return Visibility.Collapsed;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

37) Open App.xaml en plaats de 2 Converters in Application.Resources :

Voeg eveneens bovenaan de namespace toe van de Converters klassen:

```
xmlns:helper="clr-namespace:Contacten1.Helpers"
             StartupUri="MainWindow.xaml">

        <helper:BoolToVisibilityConverter x:Key="IsEditConverter"/>
        <helper:NullToVisibilityConverter x:Key="SelectedContactConverter"/>
```

38) Open ContactenViewModel.cs. We voegen nu properties toe die bijhouden of de view in Editmode is of in DisplayMode. We voegen eveneens een Property EditCommand toe, en initialiseren deze in de constructer. zodat er de view in Editmode kan worden gezet (via bool Property IsInEditMode op true te zetten) wanneer er op de Editbutton wordt geklikt

```
using Contacten1.Models;
using Contacten1.Utilty;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace Contacten1.ViewModels
{
    public class ContactenViewModel : ObservableObject
    {
        private Contact _selectedContact;
        public Contact SelectedContact
        {
            get { return _selectedContact; }
            set { OnPropertyChanged(ref _selectedContact, value); }
        }
        private bool _isInEditMode;
        public bool IsInEditMode
        {
            get { return _isInEditMode; }
            set {
                OnPropertyChanged(ref _isInEditMode, value);
                OnPropertyChanged(nameof(IsInDisplayMode));
            }
        }
        public bool IsInDisplayMode
        {
            get { return !_isInEditMode; }
        }
```

```
        public ObservableCollection<Contact> Contacten { get; private set; }
        public ICommand EditCommand {get; private set;}
        public ContactenViewModel()
        {
            EditCommand = new RelayCommand(Edit, CanEdit);
        }
        private bool CanEdit()
        {
            if (SelectedContact == null)
                return false;
            return !IsInEditMode;
        }
        private void Edit()
        {
            IsInEditMode = true;
        }
        public void ContactenLaden(IEnumerable<Contact> contacten)
        {
            Contacten = new ObservableCollection<Contact>(contacten);
            OnPropertyChanged(nameof(Contacten));
        }

    }
}
```

39) Nu moeten we nog de VisibilityConverters en de EditCommand binden op de nodige controls. Open ContactDetails.xaml en voeg aan de TextBox bovenaan (om de Naam van contactpersoon te wijzigen) de volgende Binding.
In Editmode willen we de Textbox tonen en de Label verbergen, wanneer in DisplayMode, tonen we de Label en verbergen we de Textbox:

```
<StackPanel Orientation="Horizontal">
    <StackPanel HorizontalAlignment="Center">

        <TextBox Text="{Binding ElectedContact.Naam, FallbackValue=NULL, Mode=TwoWay}"
                 Visibility="{Binding IsInEditMode,Converter={StaticResource IsEditConverter}}"
                 VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
                 Margin="5"/>

        <Label Content="{Binding ElectedContact.Naam,FallbackValue=NULL}"
               Visibility="{Binding IsInDisplayMode,Converter={StaticResource IsEditConverter}}"
               HorizontalAlignment="Center"
               FontSize="24"/>
    </StackPanel>
       <CheckBox  IsChecked="{Binding SelectedContact.IsFavoriet}"  />
```

40) De Browse button voor de image van de contactpersoon willen we ook enkel tonen in EditMode. We voeg dus eveneens Visibility Binding toe voor de browse button

```
                <Button Style="{StaticResource MenuIconButton}"
Background="{StaticResource GloomyPurple}"
                        Visibility="{Binding IsInEditMode,Converter={StaticResource
IsEditConverter}}"
                        VerticalAlignment="Bottom" HorizontalAlignment="Right"
                        Foreground="Black" FontWeight="DemiBold">
                    Browse
                </Button>
```

(code tot nu op Github : Contacten1H)

41) Verder willen we eveneens op dezelfde manier TextBoxen en Labels tonen/verbergen voor de gegevens TelNr, Email en Locaties. Dit is eveneens in de ContactDetails.xaml toe te voegen :

```
<UserControl x:Class="Contacten1.Views.ContactDetails"
             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:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="1200" d:DesignWidth="450">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>

        <Grid Grid.Row="0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition Width="100"/>
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0" HorizontalAlignment="Left"
                        Margin="5">
                <StackPanel Orientation="Horizontal">
                    <StackPanel HorizontalAlignment="Center">

                        <TextBox Text="{Binding ElectedContact.Naam, FallbackValue=NULL, Mode=TwoWay}"
                                 Visibility="{Binding IsInEditMode,Converter={StaticResource
IsEditConverter}}"
                                 VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
                                 Margin="5"/>

                        <Label Content="{Binding ElectedContact.Naam,FallbackValue=NULL}"
                               Visibility="{Binding IsInDisplayMode,Converter={StaticResource
IsEditConverter}}"
                               HorizontalAlignment="Center"
                               FontSize="24"/>
                    </StackPanel>
                    <CheckBox  IsChecked="{Binding SelectedContact.IsFavoriet}"  />
                </StackPanel>
                <Grid Height="125" Width="125">
                    <Image Height="125" Width="125"
                           Source="{Binding SelectedContact.ImagePad,
                        FallbackValue={StaticResource DefaultContactImage},
                        TargetNullValue={StaticResource DefaultContactImage}}"/>

                    <Button Style="{StaticResource MenuIconButton}" Background="{StaticResource
GloomyPurple}"
                            Visibility="{Binding IsInEditMode,Converter={StaticResource
IsEditConverter}}"
                           VerticalAlignment="Bottom" HorizontalAlignment="Right"
                           Foreground="Black" FontWeight="DemiBold">
                        Browse
                    </Button>
                </Grid>
            </StackPanel>
            <StackPanel Grid.Column="1" VerticalAlignment="Bottom" Orientation="Horizontal">
                <Button Style="{StaticResource MenuIconButton}" Background="{StaticResource
MaximumBlueGreen}" Width="90" Height="40" Margin="5">
                    <Button.Content>
                        <StackPanel Orientation="Horizontal">
                            <Image Source="/Resources/saveIcon.png" />
                            <Label FontWeight="Bold">Save</Label>
                        </StackPanel>
                    </Button.Content>
```

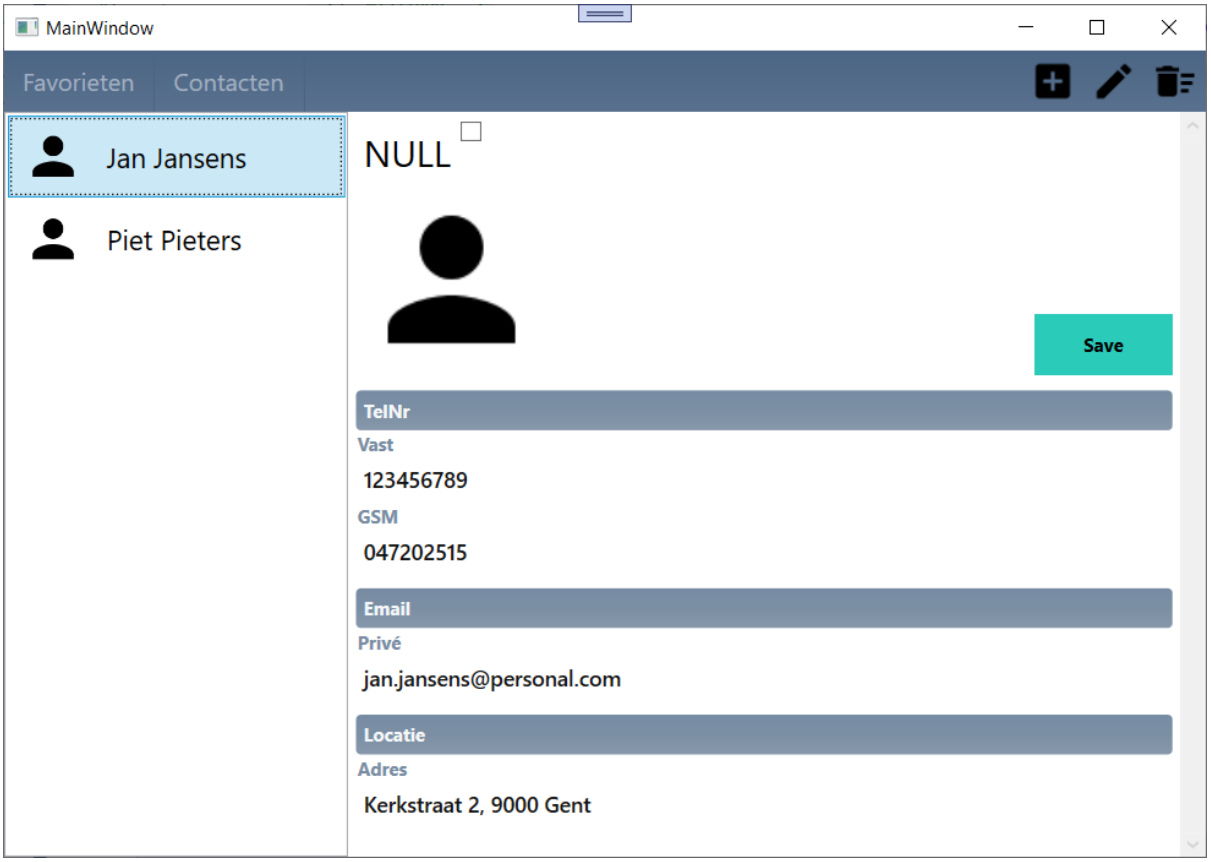
```
                </Button>
            </StackPanel>
        </Grid>
       <!--Telefoon Details-->
        <StackPanel Grid.Row="1" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="TelNr"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Vast"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[0],FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode, Converter={StaticResource
IsEditConverter}}"/>

                    <TextBox  Text="{Binding SelectedContact.TelNr[0], FallbackValue=N/A, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>

                    <Label Style="{StaticResource DetailLabel}"  Content="GSM"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[1],FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode, Converter={StaticResource
IsEditConverter}}"/>
                    <TextBox  Text="{Binding SelectedContact.TelNr[1], FallbackValue=N/A, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
        <!--Email Details-->
        <StackPanel Grid.Row="2" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="Email"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Privé"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.Email,FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode, Converter={StaticResource
IsEditConverter}}"/>
                    <TextBox  Text="{Binding SelectedContact.Email, FallbackValue=N/A, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
        <!--Locatie Details-->
        <StackPanel Grid.Row="3" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="Locatie"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Adres"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.Locatie,FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode, Converter={StaticResource
IsEditConverter}}"/>
                    <TextBox  Text="{Binding SelectedContact.Locatie, FallbackValue=N/A, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
    </Grid>
</UserControl>
```

42) Nu moet de EditCommand nog gebonden worden aan de EditButton rechtsboven in de toolbar. Open hiervoor MainWindow.xaml en pas de Editbutton aan:

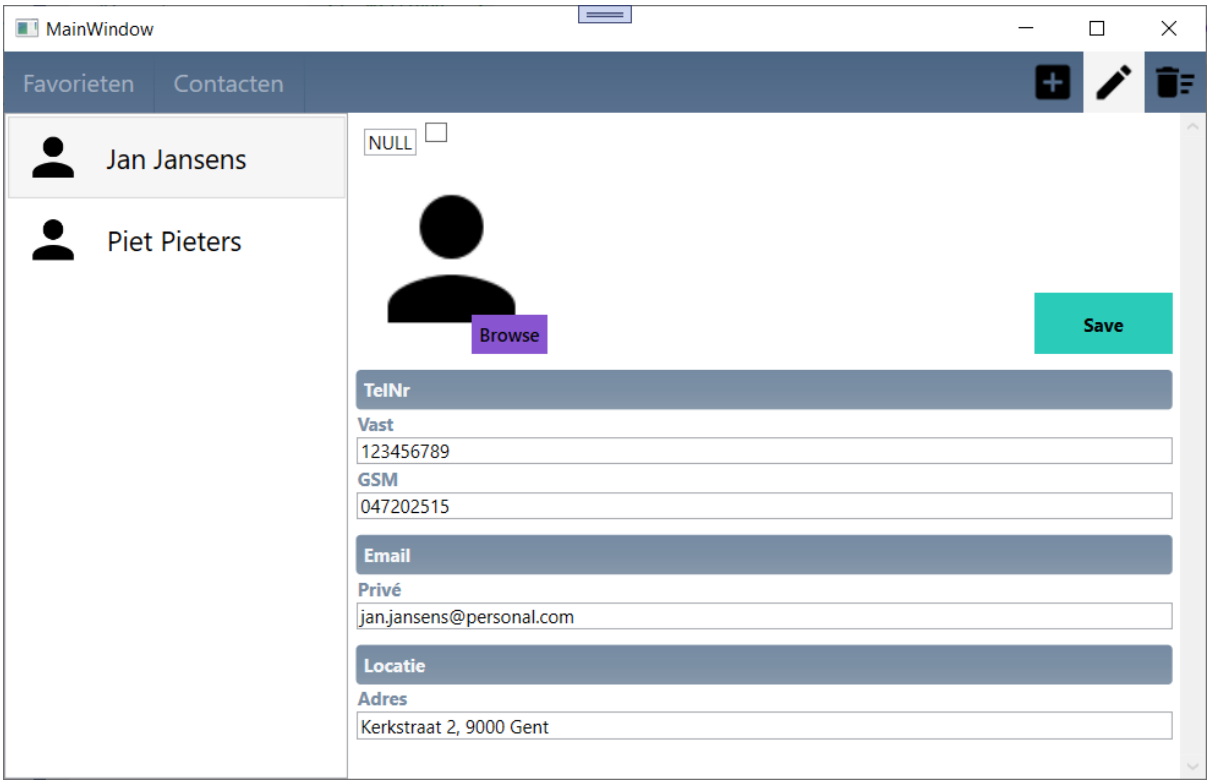
```
  <!--Wijzigen -->
                <Button Style="{StaticResource MenuIconButton}"  Command="{Binding
BoekVM.ContactenVM.EditCommand}">
                    <Button.Content>
                        <Image Height="30" Width="30"
Source="pack://siteoforigin:,,,/Resources/createIcon.png"/>
                    </Button.Content>
                </Button>
```

Start de app:

In Displaymode zijn de TextBoxes onzichtbaar :



In EditMode zijn de TextBoxes zichtbaar :



43) We gaan eveneens in ContactDetails.xaml de Grid Visibility binden aan de 2de Converter. Indien het geselecteerdecontact null is (Property SelectedContact ==null, wordt de grid collapsed, anders wordt ze getoond

Code tot nu op GitHub : Contacten1I

44) We gaan nu de Save command en functionaliteit voor een contact toevoegen. Open ContactenViewModel.cs en voeg een SaveCommand Property toe en bijhorende methoden :
Pas de constructor aan zodat de dataservice als parameter kan worden doorgegeven
Voeg een private field `IContactDataService _dataservice;` **toe en initialiseer deze in de constructor. Maak een Save() methode die de dataservice.Bewaar() methode aanroept en de IsEditMode Property of false zet.**

```
using Contacten1.Models;
using Contacten1.Services;
using Contacten1.Utilty;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace Contacten1.ViewModels
{
    public class ContactenViewModel : ObservableObject
    {
```

```
        IContactDataService _dataservice;
        private Contact _selectedContact;
        public Contact SelectedContact
        {
            get { return _selectedContact; }
            set { OnPropertyChanged(ref _selectedContact, value); }
        }
        private bool _isInEditMode;
        public bool IsInEditMode
        {
            get { return _isInEditMode; }
            set {
                OnPropertyChanged(ref _isInEditMode, value);
                OnPropertyChanged(nameof(IsInDisplayMode));
            }
        }
        public bool IsInDisplayMode
        {
            get { return !_isInEditMode; }
        }
        public ObservableCollection<Contact> Contacten { get; private set; }
        public ICommand EditCommand {get; private set;}
        public ICommand SaveCommand { get; private set; }
        public ContactenViewModel(IContactDataService dataservice)
        {
            _dataservice = dataservice;
            EditCommand = new RelayCommand(Edit, CanEdit);
            SaveCommand = new RelayCommand(Save, IsInEdit);
        }
        private bool IsInEdit()
        {
            return IsInEditMode;
        }
        private void Save()
        {
            _dataservice.Bewaar(Contacten);
            IsInEditMode = false;
            OnPropertyChanged(nameof(SelectedContact));
        }
        private bool CanEdit()
        {
            if (SelectedContact == null)
                return false;
            return !IsInEditMode;
        }
        private void Edit()
        {
            IsInEditMode = true;
        }
        public void ContactenLaden(IEnumerable<Contact> contacten)
        {
            Contacten = new ObservableCollection<Contact>(contacten);
            OnPropertyChanged(nameof(Contacten));
        }
    }
}
```

45) Open BoekViewModel.cs en pas de aanroep van de constructor van `ContactenViewModel` aan, zodat de dataservice wordt doorgegeven :

```
using Contacten1.Models;
using Contacten1.Services;
using Contacten1.Utilty;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Input;

namespace Contacten1.ViewModels
{
    public class BoekViewModel : ObservableObject
    {
        private IContactDataService _dataService;
        private ContactenViewModel _contactenVM;
        public ContactenViewModel ContactenVM
        {
            get { return _contactenVM; }
            set { OnPropertyChanged(ref _contactenVM, value); }
        }
        public ICommand CommandLadenAlleContacten { get; private set; }
        public ICommand CommandLadenFavorieten { get; private set; }
        public BoekViewModel(IContactDataService dataService)
        {
            _dataService = dataService;
            ContactenVM = new ContactenViewModel(_dataService);
            CommandLadenAlleContacten = new RelayCommand(LadenAlleContacten);
            CommandLadenFavorieten = new RelayCommand(LadenFavorieten);
        }
        private void LadenAlleContacten()
        {
            IEnumerable<Contact> contacten = _dataService.GeefContacten();
            ContactenVM.ContactenLaden(contacten);
        }
        private void LadenFavorieten()
        {
            IEnumerable<Contact> favorieten = _dataService.GeefContacten().Where(c =>
c.IsFavoriet);
            ContactenVM.ContactenLaden(favorieten);
        }
    }
}
```

46) Open ContactDetails.xaml, zet de Visibility converter en bind de SaveCommand op de Save Button

```
            <StackPanel Grid.Column="1" VerticalAlignment="Bottom"
Orientation="Horizontal">
                <Button Style="{StaticResource MenuIconButton}"
                        Command="{Binding SaveCommand}"
                           Visibility="{Binding
                    IsInEditMode,Converter={StaticResource IsEditConverter}}"
Background="{StaticResource MaximumBlueGreen}" Width="90" Height="40" Margin="5">
                    <Button.Content>
                        <StackPanel Orientation="Horizontal">
                            <Image Source="/Resources/saveIcon.png" />
                            <Label FontWeight="Bold">Save</Label>
                        </StackPanel>
                    </Button.Content>
                </Button>
            </StackPanel>
```

Code tot nu toe op GitHub : Contacten1J

47) We gaan een dialogwindow toevoegen om een foto te selecteren. Maak een nieuwe interface IDialogService onder de folder Services:

```
namespace Contacten1.Services
{
    public  interface IDialogService
    {
        string OpenFile(string filter);
    }
}
```

Voeg een nieuwe klasse toe onder de folder Service met naam WindowDialogService die de IDialogService implementeert.

```
using Microsoft.Win32;
namespace Contacten1.Services
{
    public class WindowDialogService : IDialogService
    {
        public string OpenFile(string filter)
        {
            var dialog = new OpenFileDialog();

            if (dialog.ShowDialog() == true)
            {
                return dialog.FileName;
            }

            return null;
        }
    }
}
```

48) Open BoekViewModel.cs en voeg een private field IDialogService _dialogService. Pas de constructor aan zodat de dialogService wordt doorgegeven als parameter. Initialiseer de _dialogService field in de constructor en geef de dataservice door bij de aanroep van de constructor van `ContactenViewModel`:

```
using Contacten1.Models;
using Contacten1.Services;
using Contacten1.Utilty;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Input;

namespace Contacten1.ViewModels
{
    public class BoekViewModel : ObservableObject
    {
        private IContactDataService _dataService;
        private IDialogService _dialogService;

        private ContactenViewModel _contactenVM;
        public ContactenViewModel ContactenVM
        {
            get { return _contactenVM; }
            set { OnPropertyChanged(ref _contactenVM, value); }
        }
```

```
        public ICommand CommandLadenAlleContacten { get; private set; }
        public ICommand CommandLadenFavorieten { get; private set; }
        public BoekViewModel(IContactDataService dataService, IDialogService
dialogService)
        {
            _dialogService = dialogService;
            _dataService = dataService;
            ContactenVM = new ContactenViewModel(_dataService, _dialogService);
            CommandLadenAlleContacten = new RelayCommand(LadenAlleContacten);
            CommandLadenFavorieten = new RelayCommand(LadenFavorieten);
        }

        private void LadenAlleContacten()
        {
            IEnumerable<Contact> contacten = _dataService.GeefContacten();
            ContactenVM.ContactenLaden(contacten);
        }

        private void LadenFavorieten()
        {
            IEnumerable<Contact> favorieten = _dataService.GeefContacten().Where(c
=> c.IsFavoriet);
            ContactenVM.ContactenLaden(favorieten);
        }
    }
}
```

49) Open ContactenViewModel.cs en pas op dezelfde manier de klasse aan met een nieuwe private field _dataService en een constructor met een 2de parameter zodat de _dataService kan worden geïnitialiseerd.

```
using Contacten1.Models;
using Contacten1.Services;
using Contacten1.Utilty;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace Contacten1.ViewModels
{
    public class ContactenViewModel : ObservableObject
    {
        private IDialogService _dialogService;
        private IContactDataService _dataservice;
        private Contact _selectedContact;
        public Contact SelectedContact
        {
            get { return _selectedContact; }
            set { OnPropertyChanged(ref _selectedContact, value); }
        }
        private bool _isInEditMode;
        public bool IsInEditMode
        {
            get { return _isInEditMode; }
            set {
                OnPropertyChanged(ref _isInEditMode, value);
                OnPropertyChanged(nameof(IsInDisplayMode));
            }
        }
```

```
        public bool IsInDisplayMode
        {
            get { return !_isInEditMode; }
        }
        public ObservableCollection<Contact> Contacten { get; private set; }
        public ICommand EditCommand {get; private set;}
        public ICommand SaveCommand { get; private set; }
        public ICommand UpdateCommand { get; private set; }
        public ICommand BrowseImageCommand { get; private set; }
        public ContactenViewModel(IContactDataService dataservice, IDialogService
dialogService)
        {
            _dialogService = dialogService;
            _dataservice = dataservice;
            EditCommand = new RelayCommand(Edit, CanEdit);
            SaveCommand = new RelayCommand(Save, IsInEdit);
            UpdateCommand = new RelayCommand(Update);
            BrowseImageCommand = new RelayCommand(BrowseImage, IsInEdit);
        }

        private void Update()
        {
            _dataservice.Bewaar(Contacten);
        }

        private void BrowseImage()
        {
            var filePath = _dialogService.OpenFile("Image
files|*.bmp;*.jpg;*.jpeg;*.png|All files");
            SelectedContact.ImagePad = filePath;
        }

        private bool IsInEdit()
        {
            return IsInEditMode;
        }
        private void Save()
        {
            _dataservice.Bewaar(Contacten);
            IsInEditMode = false;
            OnPropertyChanged(nameof(SelectedContact));
        }
        private bool CanEdit()
        {
            if (SelectedContact == null)
                return false;
            return !IsInEditMode;
        }
        private void Edit()
        {
            IsInEditMode = true;
        }
        public void ContactenLaden(IEnumerable<Contact> contacten)
        {
            Contacten = new ObservableCollection<Contact>(contacten);
            OnPropertyChanged(nameof(Contacten));
        }
    }
}
```

50) Open AppViewModel.cs en pas de code in de constructor aan om de dialogService object door te geven aan een BoekViewModel object.

```
using Contacten1.Services;
using Contacten1.Utilty;
using System;
using System.Collections.Generic;
using System.Text;

namespace Contacten1.ViewModels
{
    public class AppViewModel : ObservableObject
    {
        private object _currentView;
        public object CurrentView
        {
            get { return _currentView; }
            set { OnPropertyChanged(ref _currentView, value); }
        }
        private BoekViewModel _boekVM;
        public BoekViewModel BoekVM
        {
            get { return _boekVM; }
            set { OnPropertyChanged(ref _boekVM, value); }
        }
        public AppViewModel()
        {
            IContactDataService dataService = new MockDataService();
            IDialogService dialogService = new WindowDialogService();
            BoekVM = new BoekViewModel(dataService, dialogService);
            //BoekVM = new BoekViewModel(dataservice);
            //BoekVM = new BoekViewModel();
            CurrentView = BoekVM;
        }
    }
  }
```

51) Open ContactDetails.xaml en bind de button Browse Command aan de BrowseImageCommand property

```
                    <Button Style="{StaticResource MenuIconButton}"
Background="{StaticResource GloomyPurple}"
                             Visibility="{Binding
IsInEditMode,Converter={StaticResource IsEditConverter}}"
                            Command="{Binding BrowseImageCommand}"
                            VerticalAlignment="Bottom" HorizontalAlignment="Right"
                            Foreground="Black" FontWeight="DemiBold">
                        Browse
                    </Button>
```

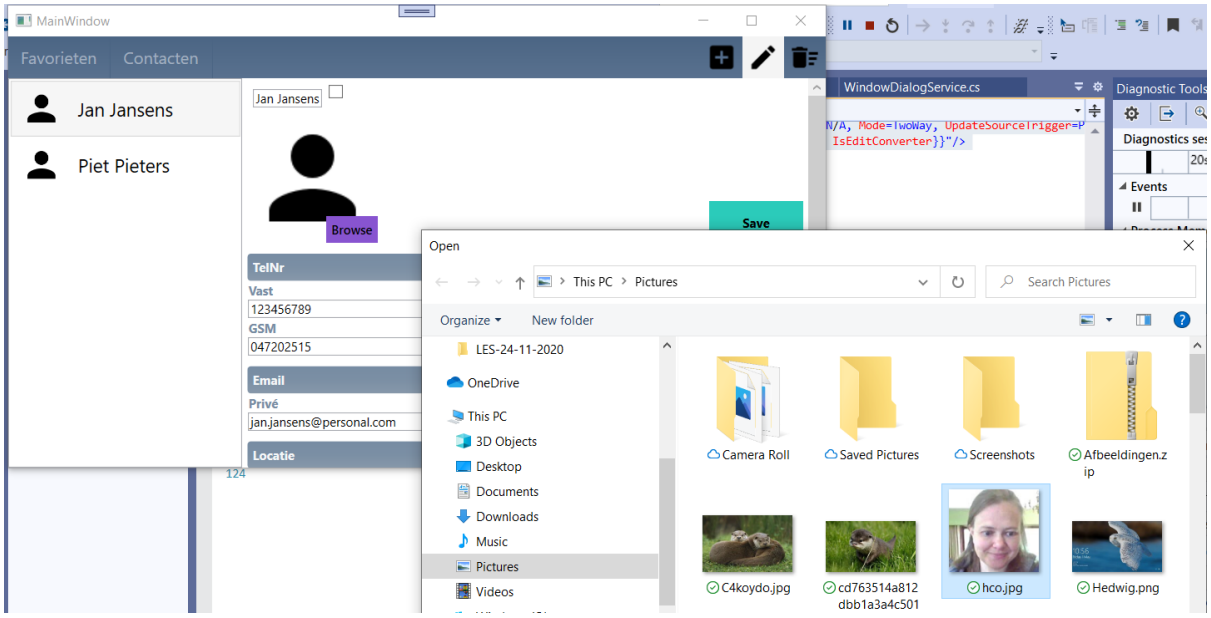
52) Volledige ContactDetails.xaml code:

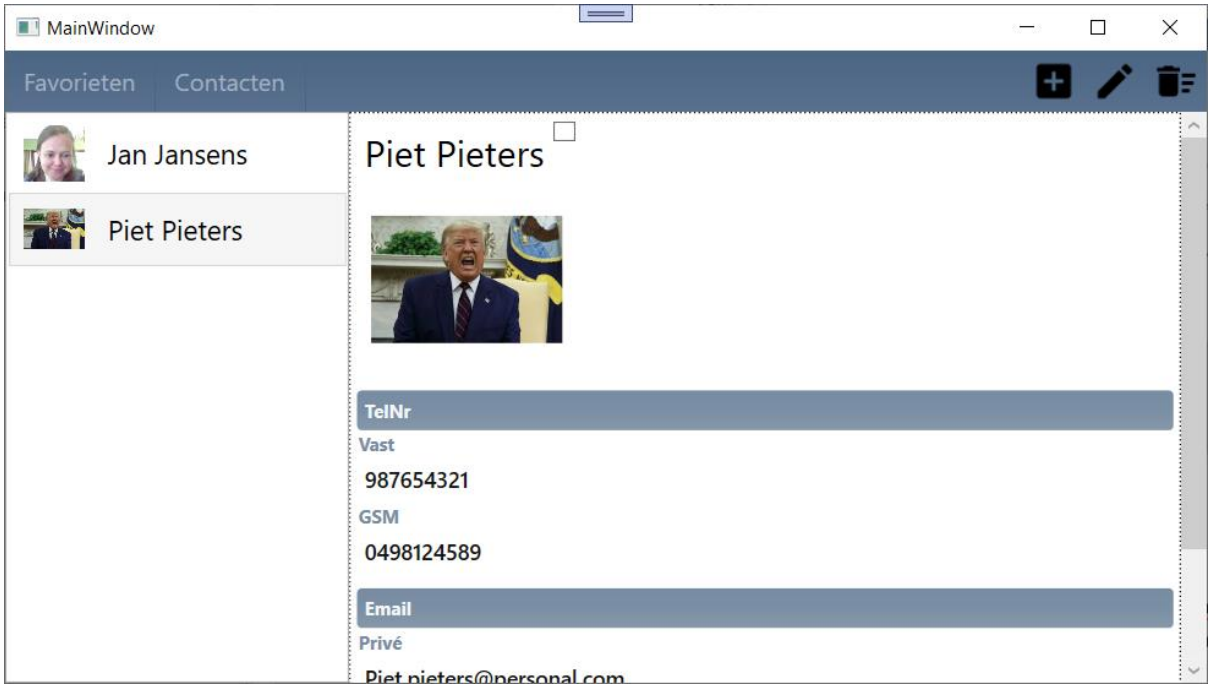
```
<UserControl x:Class="Contacten1.Views.ContactDetails"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Contacten1.Views"
             mc:Ignorable="d"
             d:DesignHeight="1200" d:DesignWidth="450">
```

```
    <Grid Visibility="{Binding SelectedContact, Converter={StaticResource
SelectedContactConverter}}">
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>

        <Grid Grid.Row="0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition Width="100"/>
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0" HorizontalAlignment="Left"
                        Margin="5">
                <StackPanel Orientation="Horizontal">
                    <StackPanel HorizontalAlignment="Center">

                        <TextBox Text="{Binding SelectedContact.Naam,
FallbackValue=NULL, Mode=TwoWay}"
                                 Visibility="{Binding
IsInEditMode,Converter={StaticResource IsEditConverter}}"
                                 VerticalContentAlignment="Center"
HorizontalContentAlignment="Center"
                                 Margin="5"/>

                        <Label Content="{Binding
SelectedContact.Naam,FallbackValue=N/A}"
                                Visibility="{Binding
IsInDisplayMode,Converter={StaticResource IsEditConverter}}"
                                HorizontalAlignment="Center"
                                FontSize="24"/>
                    </StackPanel>
                    <CheckBox  IsChecked="{Binding SelectedContact.IsFavoriet}"  />
                </StackPanel>
                <Grid Height="125" Width="125">
                    <Image Height="125" Width="125"
                           Source="{Binding SelectedContact.ImagePad,
                        FallbackValue={StaticResource DefaultContactImage},
                        TargetNullValue={StaticResource DefaultContactImage}}"/>

                    <Button Style="{StaticResource MenuIconButton}"
Background="{StaticResource GloomyPurple}"
                             Visibility="{Binding
IsInEditMode,Converter={StaticResource IsEditConverter}}"
                            Command="{Binding BrowseImageCommand}"
                            VerticalAlignment="Bottom" HorizontalAlignment="Right"
                            Foreground="Black" FontWeight="DemiBold">
                        Browse
                    </Button>
                </Grid>
            </StackPanel>
            <StackPanel Grid.Column="1" VerticalAlignment="Bottom"
Orientation="Horizontal">
                <Button Style="{StaticResource MenuIconButton}"   Visibility="{Binding
IsInEditMode,Converter={StaticResource IsEditConverter}}"
                        Command="{Binding SaveCommand}"
```

```
                        Background="{StaticResource MaximumBlueGreen}" Width="90"
Height="40" Margin="5">
                    <Button.Content>
                        <StackPanel Orientation="Horizontal">
                            <Image Source="/Resources/saveIcon.png" />
                            <Label FontWeight="Bold">Save</Label>
                        </StackPanel>
                    </Button.Content>
                </Button>
            </StackPanel>
         </Grid>
       <!--Telefoon Details-->
        <StackPanel Grid.Row="1" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="TelNr"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Vast"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[0],FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode,
Converter={StaticResource IsEditConverter}}"/>

                    <TextBox  Text="{Binding SelectedContact.TelNr[0],
FallbackValue=N/A, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>

                    <Label Style="{StaticResource DetailLabel}"  Content="GSM"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.TelNr[1],FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode,
Converter={StaticResource IsEditConverter}}"/>
                    <TextBox  Text="{Binding SelectedContact.TelNr[1],
FallbackValue=N/A, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
        <!--Email Details-->
        <StackPanel Grid.Row="2" Margin="5">
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9" Content="Email"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Privé"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.Email,FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode,
Converter={StaticResource IsEditConverter}}"/>
                    <TextBox  Text="{Binding SelectedContact.Email, FallbackValue=N/A,
Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
                      Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
        <!--Locatie Details-->
        <StackPanel Grid.Row="3" Margin="5">
```

```
            <StackPanel>
                <Border Style="{StaticResource DetailBorder}">
                    <Label FontWeight="Bold" Foreground="#FFF5F7F9"
Content="Locatie"/>
                </Border>
                <StackPanel>
                    <Label Style="{StaticResource DetailLabel}" Content="Adres"/>
                    <Label Style="{StaticResource ContentLabel}" Content="{Binding
SelectedContact.Locatie,FallbackValue=N/A}"
                           Visibility="{Binding IsInDisplayMode,
Converter={StaticResource IsEditConverter}}"/>
                    <TextBox  Text="{Binding SelectedContact.Locatie,
FallbackValue=N/A, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
                            Visibility="{Binding IsInEditMode, Converter={StaticResource
IsEditConverter}}"/>
                </StackPanel>
            </StackPanel>
        </StackPanel>
    </Grid>
</UserControl>
```





53) Code op Github: Contacten1K

54) Favorites Checkbox style met star-icoontje

Open DetailsView.xaml en zet een Style op de Checkbox

```
<CheckBox Style="{StaticResource favoriteCheckbox}" IsChecked="{Binding
SelectedContact.IsFavorite}"Command="{Binding UpdateCommand}"/>
```

55) Open App.xaml en set de favoriteCheckBox Style definitie in Application.Resources :

```
        <Style x:Key="favoriteCheckbox" TargetType="{x:Type CheckBox}">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type CheckBox}">
                        <StackPanel Orientation="Horizontal">
                            <Image x:Name="checkboxImage"
Source="pack://siteoforigin:,,,/Resources/uncheckedStar.png" Width="32"/>
                            <ContentPresenter/>
                        </StackPanel>

                        <ControlTemplate.Triggers>

                            <Trigger Property="IsChecked" Value="True">
                                <Setter TargetName="checkboxImage" Property="Source"
Value="pack://siteoforigin:,,,/Resources/checkedStar.png"/>
                            </Trigger>

                            <MultiTrigger>
                                <MultiTrigger.Conditions>
                                    <Condition Property="IsMouseOver" Value="True"/>
                                    <Condition Property="IsChecked" Value="False"/>
                                </MultiTrigger.Conditions>
                                <Setter TargetName="checkboxImage" Property="Source"
Value="pack://siteoforigin:,,,/Resources/hoverStar.png"/>
                            </MultiTrigger>

                        </ControlTemplate.Triggers>

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

56) Run de app en test de style op de checkbox

57) Code op Github: Contacten1L

58) Voeg nog de ICommand properties toe en bijhorende code voor RemoveCommand en AddCommand en leg de juiste bindings op de Command van de juiste Buttons

Open ContactenViewModel.cs en voeg de volgende public Auto-properties toe :

```
public ICommand AddCommand { get; private set; }
public ICommand DeleteCommand { get; private set; }
```

Pas de constructor ContactenViewModel aan zodat deze properties worden geïnitialiseerd :

```
        public ContactsViewModel(IContactDataService dataService, IDialogService
dialogService)
        {
            _dataService = dataService;
            _dialogService = dialogService;

            EditCommand = new RelayCommand(Edit, CanEdit);
            SaveCommand = new RelayCommand(Save, IsEdit);
            UpdateCommand = new RelayCommand(Update);
            BrowseImageCommand = new RelayCommand(BrowseImage, IsEdit);
            AddCommand = new RelayCommand(Add);
            DeleteCommand = new RelayCommand(Delete, CanDelete);
          }
```

Implementeer de methoden Add(), Delete() en CanDelete()

```
        private void Delete()
        {
            Contacten.Remove(SelectedContact);
            Save();
        }

        private bool CanDelete()
        {
            return SelectedContact == null ? false : true;
        }

        private void Add()
        {
            Contact nieuwContact = new Contact
            {
                Naam = "N/A",
                TelNr = new string[2],
                Email = "N/A",
                Locatie = "N/A"
            };

            Contacten.Add(nieuwContact);
            SelectedContact = nieuwContact;
          }
```

59) Open MainWindow.xaml en voeg een Command Binding toe op de Buttons Add en Remove Tool Buttons

```
          <!--Tool buttons-->
           <StackPanel Grid.Column="6"  Grid.ColumnSpan="2" Orientation="Horizontal"
HorizontalAlignment="Right">
               <!--toevoegen-->
               <Button Style="{StaticResource MenuIconButton}"
Command="{Binding BoekVM.ContactenVM.AddCommand}">
                   <Button.Content>
                       <Image Height="30" Width="30"
Source="pack://siteoforigin:,,,/Resources/addIcon.png"  />
```

```
                    </Button.Content>
                </Button>
                <!--Wijzigen -->
                <Button Style="{StaticResource MenuIconButton}"  Command="{Binding
BoekVM.ContactenVM.EditCommand}">
                    <Button.Content>
                        <Image Height="30" Width="30"
Source="pack://siteoforigin:,,,/Resources/createIcon.png"/>
                    </Button.Content>
                </Button>
                <!--Verwijderen-->
                <Button Style="{StaticResource MenuIconButton}"
Command="{Binding BoekVM.ContactenVM.DeleteCommand}">
                    <Button.Content>
                        <Image Height="30" Width="30"
Source="pack://siteoforigin:,,,/Resources/deleteIcon.png"/>
                    </Button.Content>
                </Button>
            </StackPanel>
```

60) We voegen nu een `JsonContactDataService` klasse toe onder de folder Services. Deze klasse implementeert de interface `IContactDataService`

```
public class JsonContactDataService : IContactDataService
{
    private readonly string _dataPath = "Resources/contacten.json";

    public IEnumerable<Contact> GeefContacten()
    {
        if (!File.Exists(_dataPath))
        {
            File.Create(_dataPath).Close();
        }

        var serializedContacts = File.ReadAllText(_dataPath);
        var contacts =
JsonConvert.DeserializeObject<IEnumerable<Contact>>(serializedContacts);

        if (contacts == null)
            return new List<Contact>();

        return contacts;
    }

    public void Bewaar(IEnumerable<Contact> contacten)
    {
        var serializedContacts = JsonConvert.SerializeObject(contacten);
        File.WriteAllText(_dataPath, serializedContacts);
    }
}
```

61) Open AppViewModel.cs en vervang de creatie van het object MockDataService door een object van de klasse JsonContactDataService :

```
    public AppViewModel()
    {
        IContactDataService dataService = new JsonContactDataService();//new MockDataService();
        IDialogService dialogService = new WindowDialogService();
        BoekVM = new BoekViewModel(dataService, dialogService);
        CurrentView = BoekVM;
    }
```

62) **Test de app, maak een paar contacten aan en bewaar deze. Herstart de app, je ziet dat deze contacten nu gepersisteerd zijn (in een testbestand .json formaat):**