

**Технически университет**

**София**

**Проект за освобождаване от изпит по**

**„Програмни среди”**

**Тема – КАТ система**

**Изпълнена от:**

**Студент: Красимир Етов**

**Факултет: ФКСТ**

**Специалност: КСИ**

**Факултетен № 121216142**

**Група: 51**

**Дата: 26.05.2019 Проверил:**

**Град: София**

**План за реализация**

1. Функционалност:

Ползвателят на системата ще може:

* Да регистрира ново превозно средство.
* Да регистрира нов потребител.
* Да експортира PDF файл с: всички нарушения извършени от даден потребител, всички нарушения направени с определено превозно средство.
* Да вижда списък с всички нарушения извършени от потребителите с възможност за филтриране на данните.
* Да изтрива запис от списъка с нарушения.
* Да добавя нов запис в списъка с нарушения

1. Използвани технологии и библиотеки:

* WPF – За графичен интерфейс.
* Microsoft SQL Server – База данни.
* Entity Framework ORM – За връзка между базата данни и C# кода.
* Autofac – За Dependency Injection контейнер в WPF частта. Грижи се за осигуряването на инстанции от обектите, когато са поискани.
* Automapper – За по-бързо и лесно преобразуване на Database моделите към ViewModel моделите използвани от WPF частта (графичния интефейс).
* Itext7 – За експортиране и запис на данните във файл с .PDF формат.
* Mahapps Metro UI – За модерен дизайн на WPF частта.

1. Архитектура

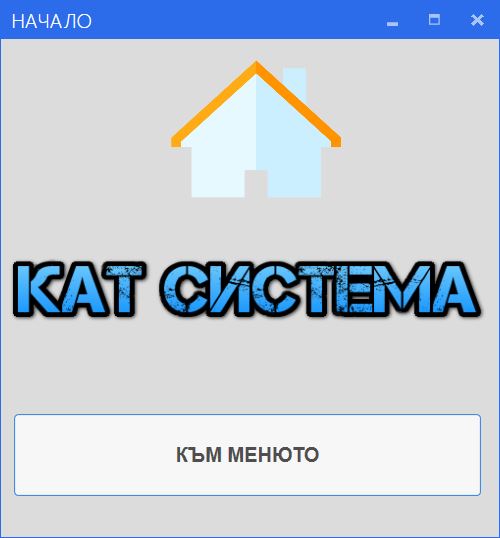
* CarSystem.App – Стартиращата част на приложението. Тук се намира WPF имплементацията,
* CarSystem.Data – Тук се намира връзката с базата данни. Използван е Entity Framework ORM с Code First подход. Автоматичните миграции са позволени, което означава, че при стартиране на приложението ще създаде базата, ако няма такава без да е необходимо ползвателя ръчно да изпълнява команди създаващи я.
* CarSystem.Data.Models – Тук се намират моделите използвани за базата. Използвани са конвенциите от Entity Framework за изграждане на връзките между таблиците и базата данни.
* CarSystem.Services – Частта съдържаща CRUD операциите с базата данни и логиката за пренос на данните към WPF частта (графичния интерфейс).

**Съдържание**

1. Увод.................................................................................................................4
2. Проектиране....................................................................... ...........................5
3. Реализация....................................................................................................14
4. Заключение...................................................................................................60
5. Използвана литература................................................................................61
6. Приложение.................................................................................................61

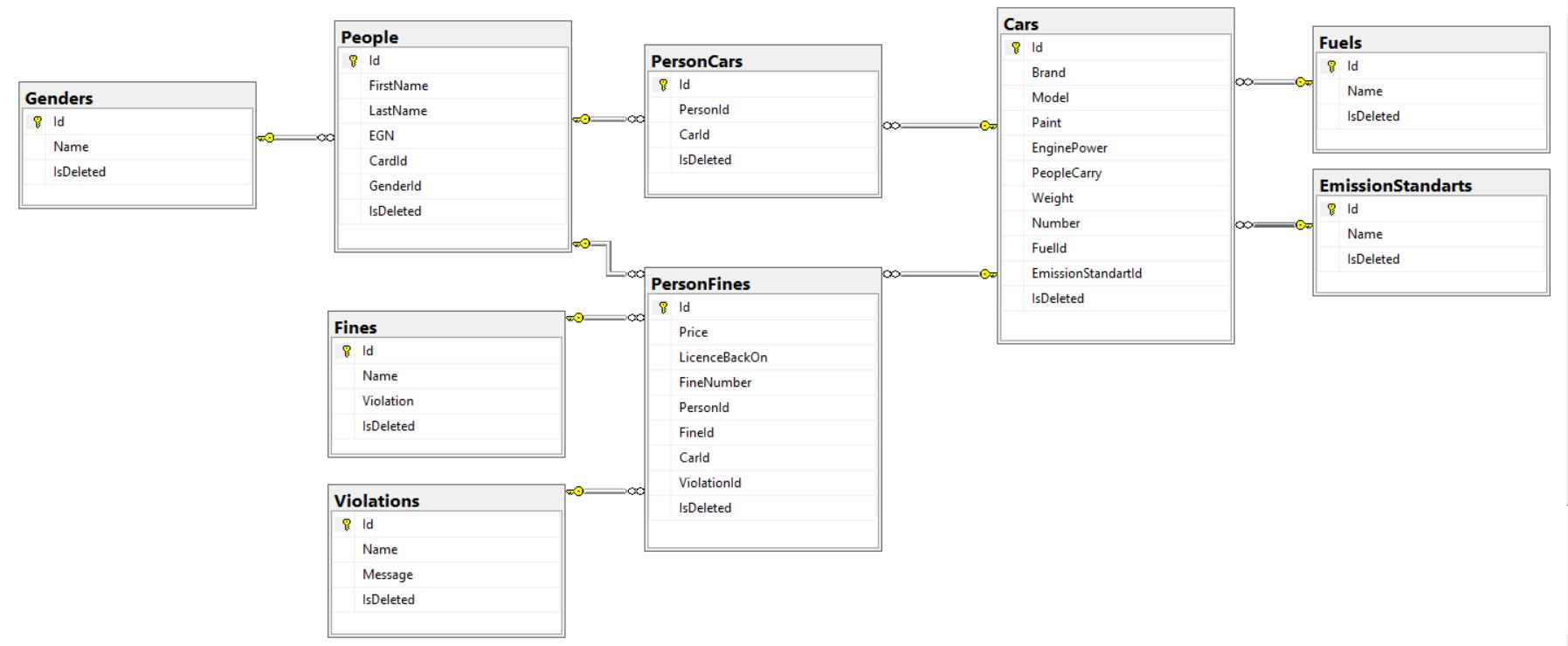
**Увод**

Следенето на информация в писмен вид в наши дни може да се укаже доста трудоемко, поради голямото количество данни. Тук на помощ идват компютрите. Те ни позволяват лесно и бързо да управляваме големи количества данни. Трудността да ги съхраняваме на хартиен носител идва от това, че са трудно копируеми и не толкова удобни за поддръжка. Едно приложение за управляване на потребителите и извършените от тях нарушения позволява лесно и интуитивно справяне с проблема. С помощта на това приложение обработката, намирането и извършването на различни действия бива многократно улеснено и удобно за ползване.



**Проектиране**

**ER диаграма:**



**Кой ще използва продукта?**

* Продукта ще се използва от хора работещи в КАТ с цел потребителите и превозните средства да бъдат по-лесно и бързо обслужвани.
* Интерфейсът на приложението е интуитивен, не сложен за употреба и доставящ удоволствие при ползване.

**Какви данни ще се използват?**

1. Прозорец нарушения:

* Показва се таблица, която първоначално зарежда всички налични нарушения извършени от всички потребители.
* Възможност за филтриране на данните по следните полета: № на лична карта, ЕГН, № на превозно срество, № на фиш, вид на нарушението като след избраните филтри е необходимо да се натисне бутона „Филтрирай“. За премахване на филтрите е необходимо да бъде натиснат бутона – „Изчисти филтрите“.
* Възможност за изтриване на запис като с мишката се посочи запис от таблицата и се натисне „Изтрий“. След, което се отваря прозорец – диалог, за да потвърди дали ползвателя е сигурен в избора си.
* Възможност за добавяне на ново нарушение чрез клик на бутона за slide-менюто. Отваря се форма, която след всяко попълване на информация филтрира следващите полета на база на предишно попълненото такова.
* Ползвателя има възможност да се откаже, да изчисти попълнените полета или да запише новото нарушение. При отказ или запис се показва прозорец – диалог потвърждаваш действиет на ползвателя.
* Данните за формите и таблицата са свързани с Binding и при промяна на информацията реагират динамично и промяната се визуализира в UI.

1. Прозорец „Справки“:

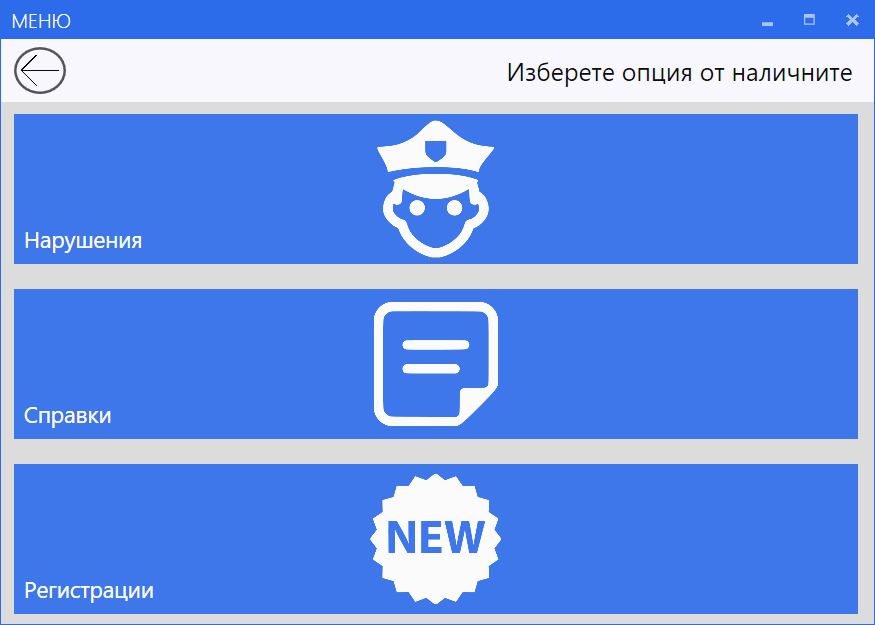
* Възможност за избор на табове служещи за навигиране между видовете справки - „Потребители“ или „Превозни средства“.
* Възможност за избор на потребител или превозно средство от списък с записи взети от базата данни.
* Възможност за сваляне на .PDF файл с информация за потребителя извършил нарушението или превозното средство, с което са извършени нарушения.
* Слез натискане на бутона за сваляне на .PDF файл се показва прозорец, в който въвежда име на файл и избира дестинация, където да бъде записан сайта.
* След правилно/неправилно добавяне се показва прозорец – диалог, който пита ползвателя дали е сигурен и дали иска да продължи.
* Данните за формите са свързани с Binding и при промяна на информацията реагират динамично и промяната се визуализира в UI.

1. Прозорец „Регистрации“:

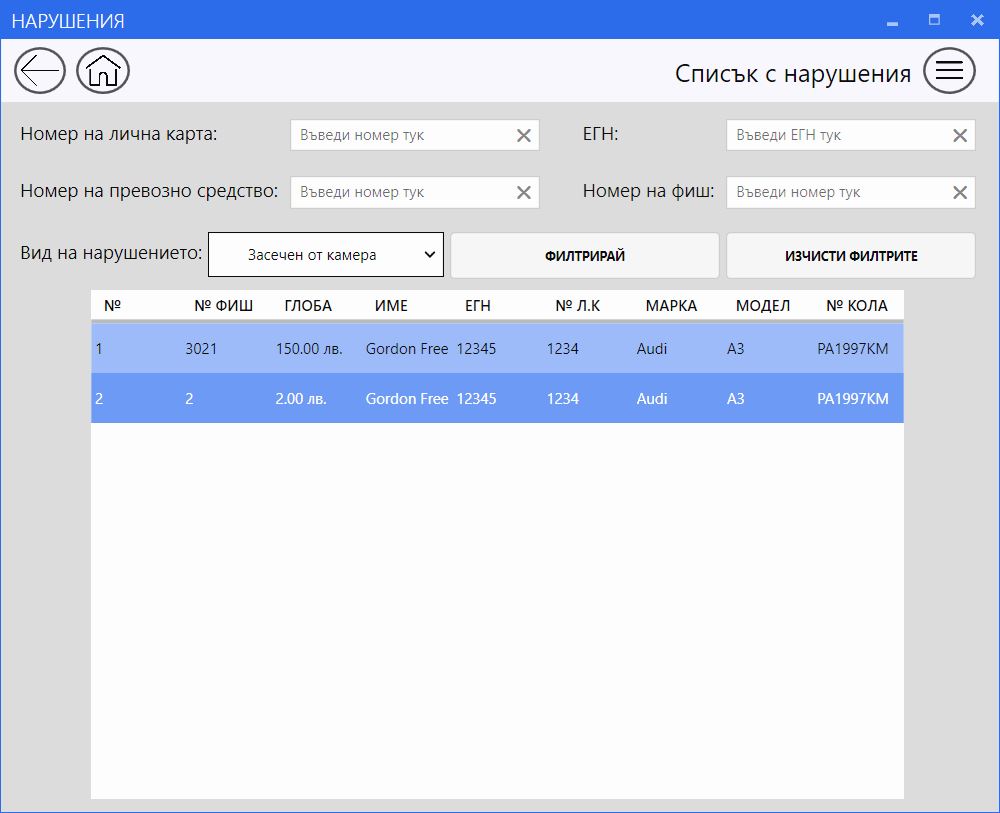
* Възможност за добавяне на потребител – ползвателя попълва форма с различни полета отговарящи на свойствата на потребител класа.
* Възможност за добавяне на превозно средство – ползвателя попълва форма с различни полета отговарящи на свойствата на класа за провеозно средство.
* След попълване на избраните форми се създава нов запис в базата данни според вида на добавения обект (потребител, превозно средство).
* Данните за формите са свързани с Binding и при промяна на информацията реагират динамично и промяната се визуализира в UI.

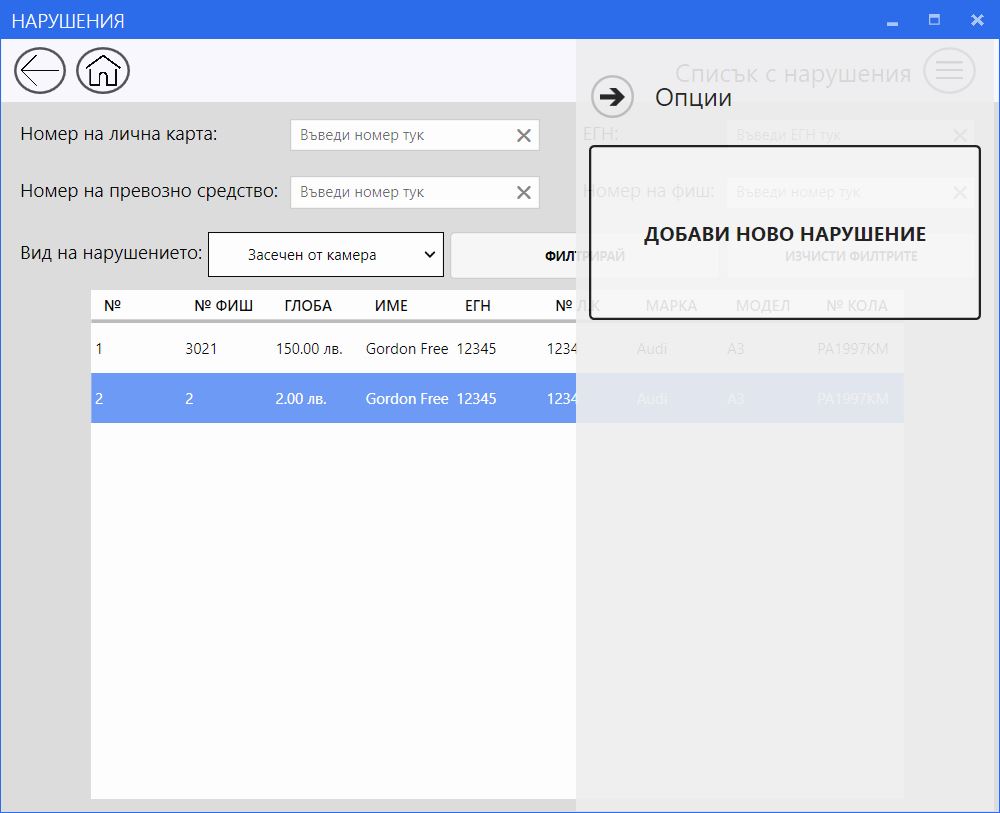
**Как ще бъдат достъпени функционалностите от ползвателя?**

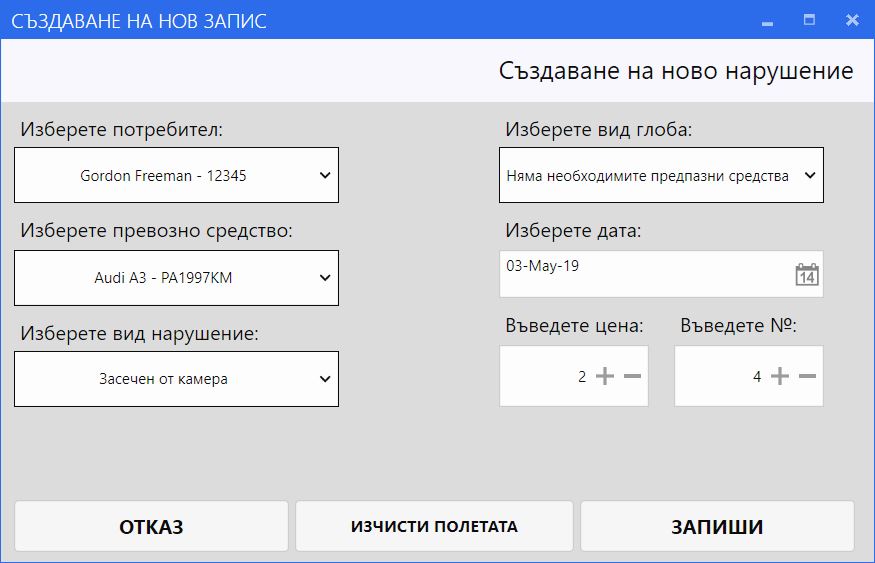
* Меню – след преминаване от началният екран към менюто ще се покажат 3 плочки (бутона) за избор на функционалност, заедно с бутони в горния ляв ъгъл за навигиране към предишния екран.

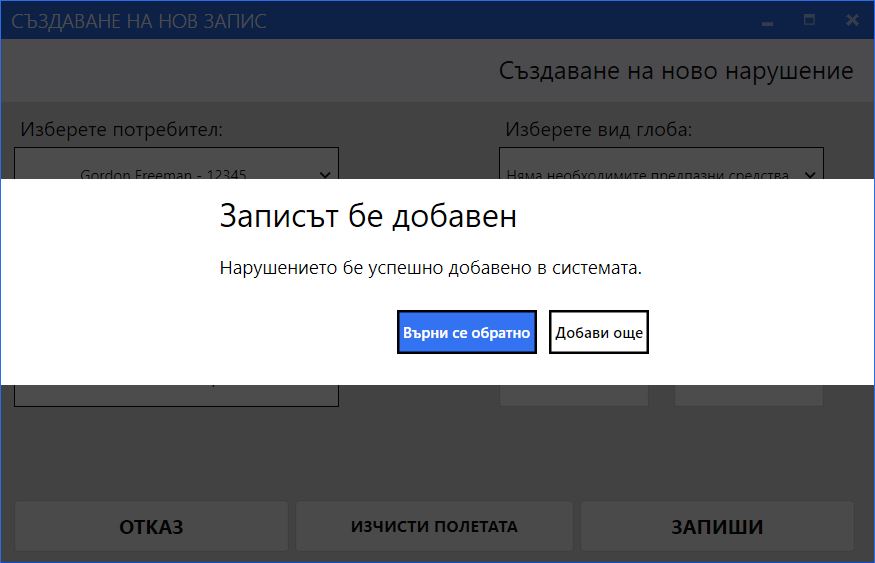
****

* Нарушения – Визуализира се таблица (DataGrid) показваща всички нарушения от базата данни заедно с TextBox полета и SplitButton контрола за избора на гориво. Налични са и бутоните за навигиране към предишния или началния екран, както и бутона за отварянето на прозореца с допълнителните опции включващи добавянето на ново нарушение.

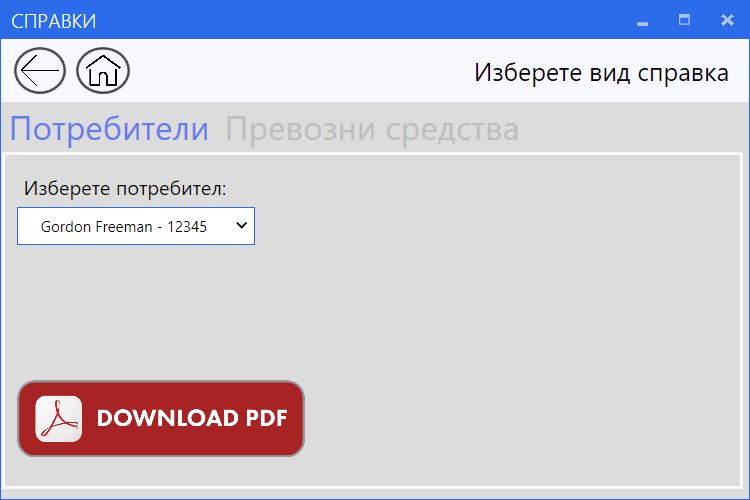




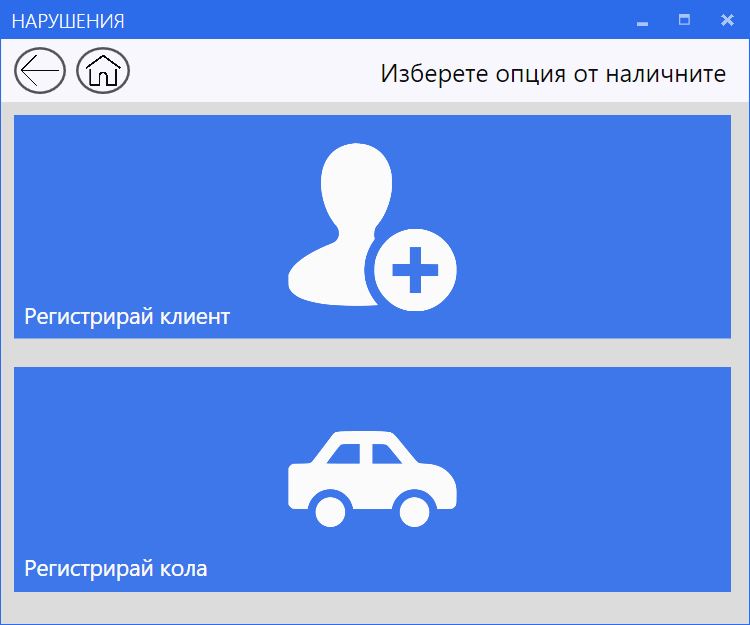


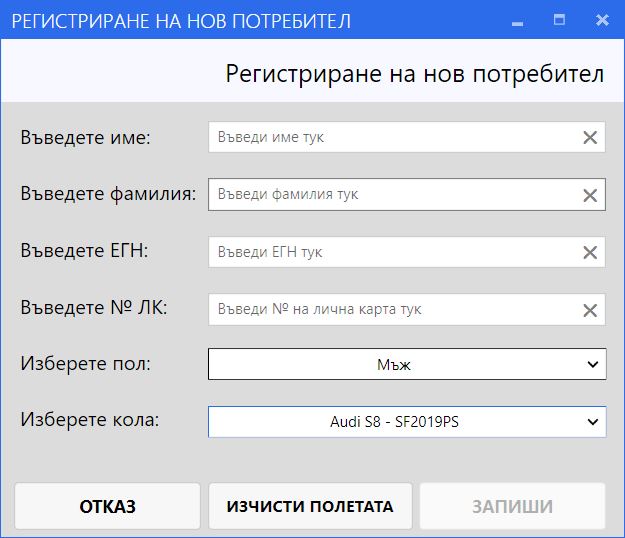


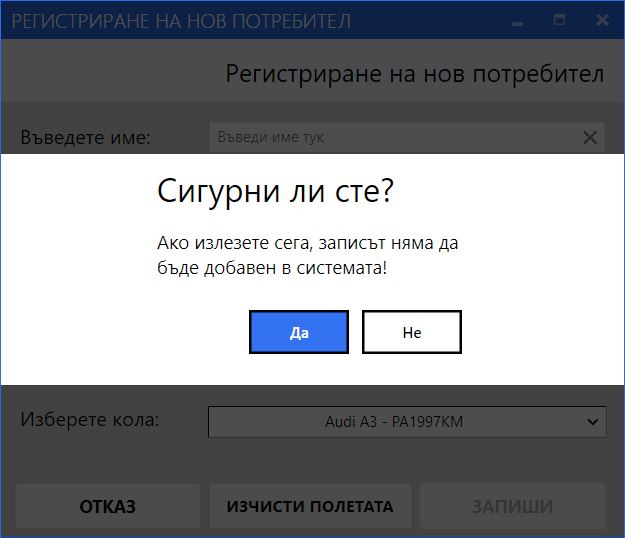
* Справки – Визуализира се (TabControl) показващ двата налични вида справки поддържани от системат. Ползвателя избира от SplitButton потребител или превозно средство, след което се визуализира бутона за сваляне на PDF файл. Налични са и бутоните за навигиране към предишния или началния екран, както и бутона за отварянето на прозореца с допълнителните опции включващи добавянето на ново нарушение.

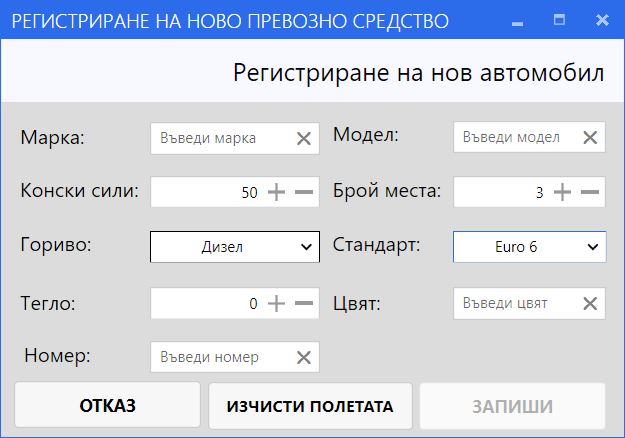


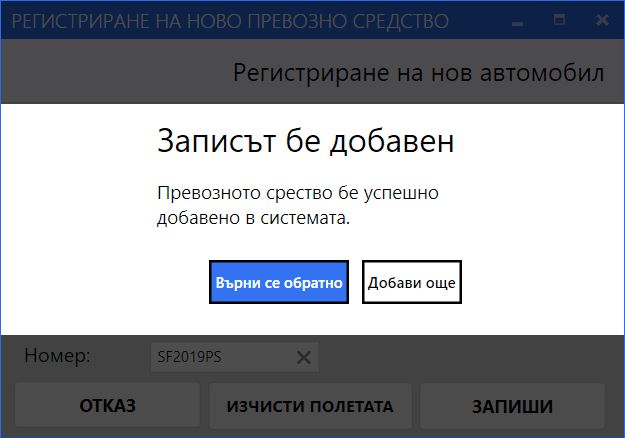
* Регистрации – Визуализират се две плочки (бутона) съответно за регистрация на потребител и превозно средство. Налични са и бутоните за навигиране към предишния или началния екран. След избиране на един от тях се показва динамична форма за създаването на запис заедно с потвърждаващи диалози.











**Реализация**

**Начален екран:**

MainWindow.xaml

<Controls:MetroWindow>

Title="Начало" Height="430" Width="400" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid>

        <Grid.RowDefinitions>

            <RowDefinition></RowDefinition>

            <RowDefinition></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Image x:Name="StartupImage" Grid.Row="0" HorizontalAlignment="Left" Height="124" Margin="136,10,0,0" VerticalAlignment="Top" Width="136" Source="Images/Startup.png" Stretch="Fill"/>

        <Image x:Name="CarSystemImage" Grid.Row="1" HorizontalAlignment="Left" Height="64" Margin="0,35.4,-0.4,0" VerticalAlignment="Top" Width="394" Source="Images/CarSystem.png" Stretch="Fill"/>

        <Button x:Name="StartupButton" Grid.Row="2" Content="Към менюто" HorizontalAlignment="Left" Margin="10,34.8,0,0" VerticalAlignment="Top" Width="374" Click="StartupButton\_Click" Cursor="Hand" Height="65" FontSize="16" FontWeight="Bold" FontFamily="Arial" BorderBrush="#FF3B8CE4" Foreground="#FF4F4D4D"/>

    </Grid>

</Controls:MetroWindow>

MainWindow.xaml.cs – Имаме бутон, който при кликване взима инстанция на MyMenu от Autofac контейнера, затваря текущия прозорец и отваря новия.

    public partial class MainWindow : MetroWindow

    {

        IContainer container = ContainerConfiguration.GetContainer();

        public MainWindow()

        {

            InitializeComponent();

        }

        private void StartupButton\_Click(object sender, RoutedEventArgs e)

        {

            var myMenu = container.Resolve<MyMenu>();

            this.Close();

            myMenu.ShowDialog();

        }

    }

}

**Меню:**

MyMenu.xaml – Тук имаме една дефинирана колона и 4 реда в Grid-a. заедно с Tile контролите, които играят ролята на бутон

<Controls:MetroWindow x:Name="MyMenuWindow">

Title="Меню" Height="500" Width="700" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid Opacity="0.95">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="140"/>

            <RowDefinition Height="140"/>

            <RowDefinition Height="140"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <Button x:Name="PreviousButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="10,6,0,0" VerticalAlignment="Top" Width="42" Cursor="Hand" ToolTip="Върни се към предишният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="PreviousButton\_Click">

            <Image Source="/CarSystem.App;component/Images/Previous.png" Height="33" Width="31" />

        </Button>

        <TextBlock HorizontalAlignment="Center" Margin="404,12,679.8,10.6" TextWrapping="Wrap" Text="Изберете опция от наличните" VerticalAlignment="Center" Height="27" Width="281" FontSize="20"/>

        <Controls:Tile x:Name="ViolationsTile" Grid.Column="0" Grid.Row="1" Margin="10,10,680,10" Title="Нарушения" Opacity="0.95" TitleFontSize="18" Cursor="Hand" Height="Auto" Width="675" Click="ViolationsTile\_Click">

            <Image Source="/CarSystem.App;component/Images/Violations.png" VerticalAlignment="Center" HorizontalAlignment="Center" Height="110"/>

        </Controls:Tile>

        <Controls:Tile x:Name="ReferencesTile" Grid.Column="0" Grid.Row="2" Margin="10,10,680,10" Title="Справки" Opacity="0.95" TitleFontSize="18" Cursor="Hand" Height="Auto" Width="675" Click="ReferencesTile\_Click">

            <Image Source="/CarSystem.App;component/Images/References.png" VerticalAlignment="Center" HorizontalAlignment="Center" Height="100"/>

        </Controls:Tile>

        <Controls:Tile x:Name="RegistrationsTile" Grid.Column="0" Grid.Row="3" Margin="10,10,680,10" Title="Регистрации" Opacity="0.95" TitleFontSize="18" Cursor="Hand" Height="Auto" Width="675" Click="RegistrationsTile\_Click">

            <Image Source="/CarSystem.App;component/Images/Registrations.png" VerticalAlignment="Center" HorizontalAlignment="Center" Height="105"/>

        </Controls:Tile>

    </Grid>

</Controls:MetroWindow>

MyMenu.xaml.cs – Тук имаме Click event за всяка плочка (бутон), в който използваме Autofac контейнера за инициализация на новия прозорец, затваряне на текущия отворен и показването на новия.

    public partial class MyMenu : MetroWindow

    {

        IContainer container = ContainerConfiguration.GetContainer();

        public MyMenu()

        {

            InitializeComponent();

        }

        private void ViolationsTile\_Click(object sender, System.Windows.RoutedEventArgs e)

        {

            var violationsWindow = container.Resolve<Violations>();

            this.Close();

            violationsWindow.ShowDialog();

        }

        private void PreviousButton\_Click(object sender, System.Windows.RoutedEventArgs e)

        {

            var startupWindow = container.Resolve<MainWindow>();

            this.Close();

            startupWindow.ShowDialog();

        }

        private void RegistrationsTile\_Click(object sender, System.Windows.RoutedEventArgs e)

        {

            var registrationsWindow = container.Resolve<Registrations>();

            this.Close();

            registrationsWindow.ShowDialog();

        }

        private void ReferencesTile\_Click(object sender, System.Windows.RoutedEventArgs e)

        {

            var referencesWindow = container.Resolve<References>();

            this.Close();

            referencesWindow.ShowDialog();

        }

    }

}

Violations.xaml – Визуализацията на таблицата с нарушенията. Ползва се DataGrid контрола за таблицата, FlyOut контрола за страничното меню за добавяне на ново нарушение, SplitButton контрола за dropdown менюто, в което се избира вид на нарушението

<Controls:MetroWindow x:Name="ViolationsWindow">

Title="Нарушения" Height="650" Width="800" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid Opacity="0.95">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="150"/>

            <RowDefinition Height="220"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <Button x:Name="PreviousButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="10,6,0,0" VerticalAlignment="Top" Width="42" Cursor="Hand" ToolTip="Върни се към предишният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="PreviousButton\_Click">

            <Image Source="/CarSystem.App;component/Images/Previous.png" Height="33" Width="31" />

        </Button>

        <Button x:Name="HomeScreenButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="60,6,0,0" VerticalAlignment="Top" Width="43" Cursor="Hand" ToolTip="Върни се към началният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="HomeScreenButton\_Click">

            <Image Source="/CarSystem.App;component/Images/HomeScreen.png" Height="32" Width="28" />

        </Button>

        <TextBlock HorizontalAlignment="Center" Margin="539,13,758.4,10.4" Grid.Row="0" TextWrapping="Wrap" Text="Списък с нарушения" VerticalAlignment="Center" Height="27" Width="193" FontSize="20"/>

        <Button x:Name="HamburgerMenuButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="737,6,0,0" VerticalAlignment="Top" Width="43" Cursor="Hand" ToolTip="Кликни за допълнителни опции" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="HamburgerMenuButton\_Click">

            <Image Source="/CarSystem.App;component/Images/HamburgerMenu.png" Height="22" Width="22" />

        </Button>

        <Label x:Name="CardIdLabel" Content="Номер на лична карта:" HorizontalAlignment="Left" Margin="10,9.6,0,0" VerticalAlignment="Top" Grid.Row="1" FontSize="15"/>

        <TextBox x:Name="CardIdTextBox" HorizontalAlignment="Left" Height="23" Margin="231,13.6,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи номер тук" VerticalAlignment="Top" Width="200" TextChanged="TextBoxChange"/>

        <Label x:Name="VehicleNumberLabel" Content="Номер на превозно средство:" HorizontalAlignment="Left" Margin="10,55,0,0" VerticalAlignment="Top" Grid.Row="1" FontSize="15"/>

        <TextBox x:Name="VehicleNumberTextBox" HorizontalAlignment="Left" Height="23" Margin="231,59.6,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи номер тук" VerticalAlignment="Top" Width="200" TextChanged="TextBoxChange"/>

        <Label x:Name="EGNLabel" Content="ЕГН:" HorizontalAlignment="Left" Margin="460,9.6,0,0" VerticalAlignment="Top" Grid.Row="1" FontSize="15"/>

        <TextBox x:Name="EGNTextBox" HorizontalAlignment="Left" Height="23" Margin="580,13.6,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи ЕГН тук" VerticalAlignment="Top" Width="200" TextChanged="TextBoxChange"/>

        <Label x:Name="FineNumberLabel" Content="Номер на фиш:" HorizontalAlignment="Left" Margin="460,55,0,0" VerticalAlignment="Top" Grid.Row="1" FontSize="15"/>

        <TextBox x:Name="FineNumberTextBox" HorizontalAlignment="Left" Height="23" Margin="580,59.6,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи номер тук" VerticalAlignment="Top" Width="200" TextChanged="TextBoxChange"/>

        <Button x:Name="FilterButton" Content="Филтрирай" HorizontalAlignment="Left" Margin="359,104.6,0,0" Grid.Row="1" VerticalAlignment="Top" Width="216" Height="37" Cursor="Hand" Click="FilterButton\_Click" IsEnabled="False"/>

        <Button x:Name="ClearFiltersButton" Content="Изчисти филтрите" HorizontalAlignment="Left" Margin="580,104.6,0,0" Grid.Row="1" VerticalAlignment="Top" Width="200" Height="37" Cursor="Hand" Click="ClearFiltersButton\_Click"/>

        <DataGrid x:Name="ViolationsDataGrid" ItemsSource="{Binding ViolationsViewModels}" AutoGenerateColumns="True" AutoGeneratingColumn="ViolationsDataGrid\_AutoGeneratingColumn" CanUserAddRows="False" CanUserResizeColumns="False" CanUserSortColumns="False" Grid.Row="2" Margin="0,0.4,696.4,-187.6" ColumnWidth="\*" RowHeight="40" Width="Auto" MaxWidth="650" Height="Auto" MaxHeight="800" Cursor="Hand" PreviewMouseRightButtonDown="DisableContextMenuOnDgHeaders\_PreviewMouseRightButtonDown">

            <DataGrid.ContextMenu>

                <ContextMenu>

                    <MenuItem Header="Изтрий" Click="DeleteItem\_Click" Cursor="Hand" />

                </ContextMenu>

            </DataGrid.ContextMenu>

        </DataGrid>

        <Controls:Flyout x:Name="HamburgerMenuFlyout" Margin="460,0,696.4,-197.6" Grid.RowSpan="3" Width="334" Opacity="0.9" Visibility="Hidden" Cursor="Hand" Position="Right" Header="Опции" FontSize="16" Background="#FFEEEEEE" Foreground="#FF050505" Theme="Light" AnimateOpacity="True">

            <Canvas>

                <Button x:Name="CreateButton" Content="Добави ново нарушение" HorizontalAlignment="Left" Width="314" Height="140" Canvas.Left="10" Canvas.Top="10" Cursor="Hand" FontSize="16" Click="CreateButton\_Click"/>

            </Canvas>

        </Controls:Flyout>

        <Label x:Name="ViolationPickerLabel" Content="Вид на нарушението:" HorizontalAlignment="Left" Margin="10,104.6,0,0" VerticalAlignment="Top" Grid.Row="1" FontSize="15"/>

        <Controls:SplitButton x:Name="ViolationPickerButton" ItemsSource="{Binding ViolationsViewModel}" DisplayMemberPath="Message" SelectionChanged="ViolationPickerButton\_SelectionChanged" Margin="165,104.6,1136.4,9.6" Grid.Row="1"/>

    </Grid>

</Controls:MetroWindow>

Violations.xaml.cs – Тук имаме евенти за всеки тип бутон както и две публични свойства – Observable колекции, които служат за Binding-а на таблицата и dropdown менюто. Използва се Autofac контейнера за взимане на инстанцията към service-ите и след това се извиква помощен клас, използващ Automapper, за да преобразува database моделите върнати от service-ите в view model и да ги попълни в колекцията, за да се визуализират в UI частта.

    public partial class Violations : MetroWindow

    {

        Autofac.IContainer container = ContainerConfiguration.GetContainer();

        public ObservableCollection<ViolationsViewModel> ViolationsViewModels { get; set; }

        public ObservableCollection<ViolationViewModel> ViolationsViewModel { get; set; }

        public Violations()

        {

            ViolationsViewModels = new ObservableCollection<ViolationsViewModel>();

            ViolationsViewModel = new ObservableCollection<ViolationViewModel>();

            InitializeComponent();

            LoadViolationsViewModels();

            LoadViolationsViewModel();

            ViolationsDataGrid.ItemsSource = ViolationsViewModels;

            ViolationPickerButton.ItemsSource = ViolationsViewModel;

        }

        private void LoadViolationsViewModels()

        {

            var personFinesService = container.Resolve<IPersonFinesService>();

            var dbRecords = personFinesService.GetFilteredPersonFinesAsync().Result;

            var observableDtoModels = ModelHandler.PersonFinesToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(ViolationsViewModels, observableDtoModels);

        }

        private void LoadViolationsViewModel()

        {

            var violationService = container.Resolve<IViolationService>();

            var dbRecords = violationService.GetAllViolationsAsync().Result;

            var observableDtoModels = ModelHandler.ViolationsToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(ViolationsViewModel, observableDtoModels);

        }

        private void PreviousButton\_Click(object sender, RoutedEventArgs e)

        {

            var myMenu = container.Resolve<MyMenu>();

            this.Close();

            myMenu.ShowDialog();

        }

        private void HomeScreenButton\_Click(object sender, RoutedEventArgs e)

        {

            var startupWindow = container.Resolve<MainWindow>();

            this.Close();

            startupWindow.ShowDialog();

        }

        private void FilterButton\_Click(object sender, RoutedEventArgs e)

        {

            var personFinesService = container.Resolve<IPersonFinesService>();

            string violationName = string.Empty;

            var violationViewModel = ViolationPickerButton.SelectedItem as ViolationViewModel;

            if (violationViewModel != null)

            {

                violationName = violationViewModel.Name;

            }

            var dbRecords = personFinesService.GetFilteredPersonFinesAsync(violationName, CardIdTextBox.Text, EGNTextBox.Text, VehicleNumberTextBox.Text, FineNumberTextBox.Text).Result;

            var observableDtoModels = ModelHandler.PersonFinesToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(ViolationsViewModels, observableDtoModels);

        }

        private void ClearFiltersButton\_Click(object sender, RoutedEventArgs e)

        {

            CardIdTextBox.Text = "";

            FineNumberTextBox.Text = "";

            EGNTextBox.Text = "";

            VehicleNumberTextBox.Text = "";

            ViolationPickerButton.SelectedItem = null;

            FilterButton.IsEnabled = false;

            LoadViolationsViewModels();

        }

        private void TextBoxChange(object sender, TextChangedEventArgs e)

        {

            if (!string.IsNullOrEmpty(CardIdTextBox.Text) || !string.IsNullOrEmpty(FineNumberTextBox.Text) || !string.IsNullOrEmpty(EGNTextBox.Text) || !string.IsNullOrEmpty(VehicleNumberTextBox.Text))

            {

                FilterButton.IsEnabled = true;

            }

            else

            {

                FilterButton.IsEnabled = false;

            }

        }

        // Use DisplayName property to visualize column names and make text centered

        private void ViolationsDataGrid\_AutoGeneratingColumn(object sender, DataGridAutoGeneratingColumnEventArgs e)

        {

            if (e.PropertyDescriptor is PropertyDescriptor descriptor)

            {

                e.Column.Header = descriptor.DisplayName ?? descriptor.Name;

                //e.Column.HeaderStyle = new Style(typeof(DataGridColumnHeader));

                //e.Column.HeaderStyle.Setters.Add(new Setter(HorizontalContentAlignmentProperty, HorizontalAlignment.Center));

            }

        }

        private void HamburgerMenuButton\_Click(object sender, RoutedEventArgs e)

        {

            HamburgerMenuFlyout.IsOpen = true;

        }

        private void CreateButton\_Click(object sender, RoutedEventArgs e)

        {

            var myMenu = container.Resolve<CreateViolation>();

            myMenu.ShowDialog();

        }

        private async void DeleteItem\_Click(object sender, RoutedEventArgs e)

        {

            var rowData = ViolationsDataGrid.SelectedItem as ViolationsViewModel;

            this.Cursor = Cursors.Hand;

            var result = await this.ShowMessageAsync("Сигурни ли сте?", "Записът ще бъде изтрит от системата!", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Да",

                AnimateHide = true,

                AnimateShow = true,

                NegativeButtonText = "Не",

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                var service = container.Resolve<IPersonFinesService>();

                await service.DeletePersonFineAsync(rowData.PersonFineId);

                ViolationsViewModels.Remove(rowData);

            }

            this.Cursor = Cursors.Arrow;

        }

        // Disable contextMenu in data grid headers

        private void DisableContextMenuOnDgHeaders\_PreviewMouseRightButtonDown(object sender, MouseButtonEventArgs e)

        {

            DependencyObject DepObject = (DependencyObject)e.OriginalSource;

            while ((DepObject != null) && !(DepObject is DataGridColumnHeader) && !(DepObject is DataGridRow))

            {

                DepObject = VisualTreeHelper.GetParent(DepObject);

            }

            if (DepObject == null)

            {

                return;

            }

            if (DepObject is DataGridColumnHeader)

            {

                ViolationsDataGrid.ContextMenu.Visibility = Visibility.Collapsed;

            }

            else

            {

                ViolationsDataGrid.ContextMenu.Visibility = Visibility.Visible;

            }

        }

        private void ViolationPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            var violationViewModel = ViolationPickerButton.SelectedItem as ViolationViewModel;

            if (violationViewModel != null)

            {

                FilterButton.IsEnabled = true;

                ClearFiltersButton.IsEnabled = true;

            }

        }

    }

}

References.xaml – Тук имаме TabControl контрола, която се грижи за смяната на табовете за избор на вид справка.

<Controls:MetroWindow x:Name="ReferencesWindow">

Title="Справки" Height="400" Width="600" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="350"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <Button x:Name="PreviousButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="10,6,0,0" VerticalAlignment="Top" Width="42" Cursor="Hand" ToolTip="Върни се към предишният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="PreviousButton\_Click">

            <Image Source="/CarSystem.App;component/Images/Previous.png" Height="33" Width="31" />

        </Button>

        <Button x:Name="HomeScreenButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="60,6,0,0" VerticalAlignment="Top" Width="43" Cursor="Hand" ToolTip="Върни се към началният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="HomeScreenButton\_Click">

            <Image Source="/CarSystem.App;component/Images/HomeScreen.png" Height="32" Width="28" />

        </Button>

        <TextBlock HorizontalAlignment="Center" Margin="378,12,781.8,10.6" TextWrapping="Wrap" Text="Изберете вид справка" VerticalAlignment="Center" Height="27" Width="205" FontSize="20"/>

        <TabControl Name="ReferencesTabControl" HorizontalAlignment="Left" Height="310" Margin="0,0.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="594">

            <TabItem Name="PeopleTabItem" Header="Потребители" Background="Gainsboro" Selector.Selected="PeopleTabItem\_Selected">

                <Grid Background="Gainsboro">

                    <Label x:Name="PersonPickerLabel" Content="Изберете потребител:" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" FontSize="16"/>

                    <Controls:SplitButton x:Name="PersonPickerButton" ItemsSource="{Binding PersonViewModels}" DisplayMemberPath="DisplayName" Margin="10,42,388,192.8" SelectionChanged="PersonPickerButton\_SelectionChanged"/>

                    <Image Name="PersonPdfDownloadImage" Source="/CarSystem.App;component/Images/Pdf.png" Margin="10,177,348,18.8" Cursor="Hand" PreviewMouseLeftButtonDown="PersonPdfDownloadImage\_PreviewMouseLeftButtonDown" Visibility="Hidden" />

                    <DockPanel HorizontalAlignment="Left" Height="100" LastChildFill="False" Margin="26,126,0,0" VerticalAlignment="Top" Width="100">

                        <WrapPanel Height="99.2" VerticalAlignment="Top" Width="100"/>

                    </DockPanel>

                </Grid>

            </TabItem>

            <TabItem Name="CarsTabItem" Header="Превозни средства" Background="Gainsboro" Selector.Selected="CarsTabItem\_Selected">

                <Grid Background="Gainsboro">

                    <Label x:Name="CarPickerLabel" Content="Изберете превозно средство:" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" FontSize="16"/>

                    <Controls:SplitButton x:Name="CarPickerButton" ItemsSource="{Binding CarsViewModels}" DisplayMemberPath="DisplayName" Margin="10,42,349,192.8" SelectionChanged="CarPickerButton\_SelectionChanged"/>

                    <Image Name="CarPdfDownloadImage" Source="/CarSystem.App;component/Images/Pdf.png" Margin="10,177,348,18.8" Cursor="Hand" PreviewMouseLeftButtonDown="CarPdfDownloadImage\_PreviewMouseLeftButtonDown" Visibility="Hidden" />

                </Grid>

            </TabItem>

        </TabControl>

    </Grid>

</Controls:MetroWindow>

References.xaml.cs – Съдържа Autofac контейнер, чрез който взимаме необходимите ни инстанции, колекции в които популираме view model-ите използвани за Binding-а и диалог прозорците.

public partial class References : MetroWindow

    {

        IContainer container = ContainerConfiguration.GetContainer();

        public ObservableCollection<PersonViewModel> PersonViewModels { get; set; }

        public ObservableCollection<CarViewModel> CarViewModels { get; set; }

        public References()

        {

            PersonViewModels = new ObservableCollection<PersonViewModel>();

            CarViewModels = new ObservableCollection<CarViewModel>();

            InitializeComponent();

            LoadPersonViewModels();

            LoadCarViewModels();

            PersonPickerButton.ItemsSource = PersonViewModels;

            CarPickerButton.ItemsSource = CarViewModels;

        }

        private void LoadPersonViewModels()

        {

            var peopleService = container.Resolve<IPeopleService>();

            var dbRecords = peopleService.GetAllPersonsAsync().Result;

            var observableDtoModels = ModelHandler.PersonToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(PersonViewModels, observableDtoModels);

        }

        private void LoadCarViewModels()

        {

            var carService = container.Resolve<ICarService>();

            var dbRecords = carService.GetAllCarsAsync().Result;

            var observableDtoModels = ModelHandler.CarToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(CarViewModels, observableDtoModels);

        }

        private void PreviousButton\_Click(object sender, System.Windows.RoutedEventArgs e)

        {

            ReturnToPreviousScreen();

        }

        private void ReturnToPreviousScreen()

        {

            var myMenu = container.Resolve<MyMenu>();

            this.Close();

            myMenu.ShowDialog();

        }

        private void HomeScreenButton\_Click(object sender, RoutedEventArgs e)

        {

            var startupWindow = container.Resolve<MainWindow>();

            this.Close();

            startupWindow.ShowDialog();

        }

        private void CarsTabItem\_Selected(object sender, RoutedEventArgs e)

        {

            CarPickerButton.SelectedItem = null;

            CarPdfDownloadImage.Visibility = Visibility.Hidden;

        }

        private void PeopleTabItem\_Selected(object sender, RoutedEventArgs e)

        {

            PersonPickerButton.SelectedItem = null;

            PersonPdfDownloadImage.Visibility = Visibility.Hidden;

        }

        private async void PersonPdfDownloadImage\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

        {

            var personFinesService = container.Resolve<IPersonFinesService>();

            var exportService = container.Resolve<IExportService>();

        tryAgain:

            string fileName = SaveFileHelper();

            bool shouldRepeat = await RepeatHelperDialog(fileName);

            if (shouldRepeat)

            {

                goto tryAgain;

            }

            var person = PersonPickerButton.SelectedItem as PersonViewModel;

            var personFines = personFinesService.GetPersonFinesByPersonId(person.Id).Result;

            exportService.ExportPersonInformation(fileName, person.Name, personFines);

            SuccessfullHelperDialog(PersonPdfDownloadImage, PersonPickerButton);

        }

        private async void CarPdfDownloadImage\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

        {

            var personFinesService = container.Resolve<IPersonFinesService>();

            var exportService = container.Resolve<IExportService>();

        tryAgain:

            string fileName = SaveFileHelper();

            bool shouldRepeat = await RepeatHelperDialog(fileName);

            if (shouldRepeat)

            {

                goto tryAgain;

            }

            var car = CarPickerButton.SelectedItem as CarViewModel;

            var carFines = personFinesService.GetCarFinesByCarId(car.Id).Result;

            exportService.ExportCarInformation(fileName, car.DisplayName, carFines);

            SuccessfullHelperDialog(CarPdfDownloadImage, CarPickerButton);

        }

        private void PersonPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            PersonPdfDownloadImage.Visibility = Visibility.Visible;

        }

        private void CarPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            CarPdfDownloadImage.Visibility = Visibility.Visible;

        }

        private string SaveFileHelper()

        {

            SaveFileDialog saveFileDialog = new SaveFileDialog()

            {

                Filter = "Pdf files (\*.pdf)|\*.pdf",

                InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop)

            };

            saveFileDialog.ShowDialog();

            if (string.IsNullOrEmpty(saveFileDialog.FileName))

            {

                return string.Empty;

            }

            string fileName =

                    ($"{saveFileDialog.FileName.Substring(0, saveFileDialog.FileName.Length - 4)}-{DateTime.Now.Day}-{DateTime.Now.Month}-{DateTime.Now.Year}-" +

                    $"{DateTime.Now.Hour}-{DateTime.Now.Minute}-{DateTime.Now.Second}-{DateTime.Now.Millisecond}.pdf");

            return fileName;

        }

        private async Task<bool> RepeatHelperDialog(string fileName)

        {

            this.Cursor = Cursors.Hand;

            bool shouldRepeat = false;

            if (string.IsNullOrEmpty(fileName))

            {

                var result = await this.ShowMessageAsync("Не сте избрали име на файл", "Необходимо е да изберете име. Желаете ли да продължите?", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

                {

                    AffirmativeButtonText = "Да, ще добавя име",

                    NegativeButtonText = "Не, искам да изляза",

                    AnimateHide = true,

                    AnimateShow = true,

                    DefaultButtonFocus = MessageDialogResult.Affirmative,

                    DialogResultOnCancel = MessageDialogResult.Canceled,

                });

                if (result == MessageDialogResult.Affirmative)

                {

                    shouldRepeat = true;

                }

                else

                {

                    ReturnToPreviousScreen();

                }

                this.Cursor = Cursors.Arrow;

            }

            return shouldRepeat;

        }

        private async void SuccessfullHelperDialog(Image image, SplitButton splitButton)

        {

            this.Cursor = Cursors.Hand;

            var result = await this.ShowMessageAsync("Файлът бе записан успешно", "Желаете ли да експортирате още справки?", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Да",

                NegativeButtonText = "Не",

                AnimateHide = true,

                AnimateShow = true,

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                splitButton.SelectedItem = null;

                image.Visibility = Visibility.Hidden;

            }

            else

            {

                ReturnToPreviousScreen();

            }

            this.Cursor = Cursors.Arrow;

        }

    }

}

Registrations.xaml – съдържа бутоните за регистрация на потребител или превозно средство

<Controls:MetroWindow x:Name="RegistrationsWindow">

Title="Нарушения" Height="500" Width="600" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid Opacity="0.95">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="200"/>

            <RowDefinition Height="220"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <Button x:Name="PreviousButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="10,6,0,0" VerticalAlignment="Top" Width="42" Cursor="Hand" ToolTip="Върни се към предишният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="PreviousButton\_Click">

            <Image Source="/CarSystem.App;component/Images/Previous.png" Height="33" Width="31" />

        </Button>

        <Button x:Name="HomeScreenButton" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Margin="60,6,0,0" VerticalAlignment="Top" Width="43" Cursor="Hand" ToolTip="Върни се към началният екран" Style="{DynamicResource MahApps.Metro.Styles.MetroCircleButtonStyle}" BorderBrush="Black" Height="38" Click="HomeScreenButton\_Click">

            <Image Source="/CarSystem.App;component/Images/HomeScreen.png" Height="32" Width="28" />

        </Button>

        <TextBlock HorizontalAlignment="Center" Margin="303,13,906.4,9.6" Grid.Row="0" TextWrapping="Wrap" Text="Изберете опция от наличните" VerticalAlignment="Center" Height="27" Width="281" FontSize="20"/>

        <Controls:Tile x:Name="AddPersonTile" Grid.Row="1" Margin="10,10.4,906.4,10.4" Height="180" Width="Auto" Title="Регистрирай клиент" Opacity="0.95" TitleFontSize="18" Cursor="Hand" Click="AddPersonTile\_Click">

            <Image Source="/CarSystem.App;component/Images/AddPerson.png" Width="Auto" Height="135"/>

        </Controls:Tile>

        <Controls:Tile x:Name="AddCarTile" Grid.Row="2" Margin="10,9.6,906.4,25.4" Height="180" Width="Auto" Title="Регистрирай кола" Opacity="0.95" TitleFontSize="18" Cursor="Hand" Click="AddCarTile\_Click">

            <Image Source="/CarSystem.App;component/Images/Car.png" Width="Auto" Height="135"/>

        </Controls:Tile>

    </Grid>

</Controls:MetroWindow>

Registrations.xaml.cs - Съдържа Autofac контейнер, чрез който взимаме необходимите ни инстанции.

public partial class Registrations : MetroWindow

    {

        IContainer container = ContainerConfiguration.GetContainer();

        public Registrations()

        {

            InitializeComponent();

        }

        private void PreviousButton\_Click(object sender, System.Windows.RoutedEventArgs e)

        {

            var myMenu = container.Resolve<MyMenu>();

            this.Close();

            myMenu.ShowDialog();

        }

        private void HomeScreenButton\_Click(object sender, RoutedEventArgs e)

        {

            var startupWindow = container.Resolve<MainWindow>();

            this.Close();

            startupWindow.ShowDialog();

        }

        private void CameraRadarTile\_Click(object sender, RoutedEventArgs e)

        {

            var startupWindow = container.Resolve<Violations>();

            this.Close();

            startupWindow.ShowDialog();

        }

        private void AddPersonTile\_Click(object sender, RoutedEventArgs e)

        {

            var createPersonWindow = container.Resolve<CreatePerson>();

            createPersonWindow.ShowDialog();

        }

        private void AddCarTile\_Click(object sender, RoutedEventArgs e)

        {

            var createCarWindow = container.Resolve<CreateCar>();

            createCarWindow.ShowDialog();

        }

    }

}

CreateViolation.xaml – Съдържа SplitButton контроли за dropdown менютата използващи Binding и TextBlock полета използвани за попълване на информацията

<Controls:MetroWindow x:Name="CreateViolationWindow">

Title="Създаване на нов запис" Height="450" Width="700" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid Opacity="0.95">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="400"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <TextBlock HorizontalAlignment="Center" Margin="398,10,807.4,12.6" Grid.Row="0" TextWrapping="Wrap" Text="Създаване на ново нарушение" VerticalAlignment="Center" Height="27" Width="285" FontSize="20"/>

        <Label x:Name="PersonPickerLabel" Content="Изберете потребител:" HorizontalAlignment="Left" Margin="10,5.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:SplitButton x:Name="PersonPickerButton" ItemsSource="{Binding PersonViewModels}" DisplayMemberPath="DisplayName" SelectionChanged="PersonPickerButton\_SelectionChanged" Margin="10,36.4,1220.4,318.6" Grid.Row="1"/>

        <Label x:Name="CarPickerLabel" Content="Изберете превозно средство:" HorizontalAlignment="Left" Margin="10,86.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16" Visibility="Hidden"/>

        <Controls:SplitButton x:Name="CarPickerButton" ItemsSource="{Binding CarViewModels}" DisplayMemberPath="DisplayName" SelectionChanged="CarPickerButton\_SelectionChanged" Margin="10,118.4,1220.4,236.6" Grid.Row="1" Visibility="Hidden"/>

        <Label x:Name="ViolationPickerLabel" Content="Изберете вид нарушение:" HorizontalAlignment="Left" Margin="10,168.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16" Visibility="Hidden"/>

        <Controls:SplitButton x:Name="ViolationPickerButton" ItemsSource="{Binding ViolationViewModels}" DisplayMemberPath="Message" SelectionChanged="ViolationPickerButton\_SelectionChanged" Margin="10,199.4,1220.4,155.6" Grid.Row="1" Visibility="Hidden"/>

        <Label x:Name="FinePickerLabel" Content="Изберете вид глоба:" HorizontalAlignment="Left" Margin="398,5.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16" Visibility="Hidden"/>

        <Controls:SplitButton x:Name="FinePickerButton" ItemsSource="{Binding FineViewModels}" DisplayMemberPath="Violation" SelectionChanged="FinePickerButton\_SelectionChanged" Margin="398,36.4,832.4,318.6" Grid.Row="1" Visibility="Hidden"/>

        <Label x:Name="LicenceBackOnDatePickerLabel" Content="Изберете дата:" HorizontalAlignment="Left" Margin="398,86.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16" Visibility="Hidden"/>

        <DatePicker x:Name="LicenceBackOnDatePicker" SelectedDateChanged="LicenceBackOnDatePicker\_SelectedDateChanged" HorizontalAlignment="Left" Margin="398,118.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="260" Height="39" Visibility="Hidden"/>

        <Label x:Name="FinePriceLabel" Content="Въведете цена:" HorizontalAlignment="Left" Margin="398,162.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16" Visibility="Hidden"/>

        <Controls:NumericUpDown x:Name="FinePriceNumericUpDown" Minimum="0" Maximum="10000" Margin="398,194.4,972.4,155.6" Grid.Row="1" Visibility="Hidden"/>

        <Label x:Name="FineNumberLabel" Content="Въведете №:" HorizontalAlignment="Left" Margin="538,162.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16" Visibility="Hidden"/>

        <Controls:NumericUpDown x:Name="FineNumberNumericUpDown" Minimum="0" Margin="538,194.4,832.4,155.6" Grid.Row="1" Visibility="Hidden"/>

        <Button x:Name="SaveButton" Content="Запиши" HorizontalAlignment="Left" Margin="463,318.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="220" Height="42" FontSize="16" Click="SaveButton\_Click" Cursor="Hand" IsEnabled="False"/>

        <Button x:Name="CancelButton" Content="Отказ" HorizontalAlignment="Left" Margin="10,318.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="220" Height="42" FontSize="16" Click="CancelButton\_Click" Cursor="Hand"/>

        <Button x:Name="ClearButton" Content="Изчисти полетата" HorizontalAlignment="Left" Margin="235,318.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="223" Height="42" FontSize="13" Click="ClearButton\_Click" Cursor="Hand" IsEnabled="False"/>

    </Grid>

</Controls:MetroWindow>

CreateViolation.xaml.cs - Съдържа Autofac контейнер, чрез който взимаме необходимите ни инстанции, колекции в които популираме view model-ите използвани за Binding-а и диалог прозорците.

public partial class CreateViolation : MetroWindow

    {

        Autofac.IContainer container = ContainerConfiguration.GetContainer();

        public ObservableCollection<PersonViewModel> PersonViewModels { get; set; }

        public ObservableCollection<CarViewModel> CarViewModels { get; set; }

        public ObservableCollection<FineViewModel> FineViewModels { get; set; }

        public ObservableCollection<ViolationViewModel> ViolationViewModels { get; set; }

        public CreateViolation()

        {

            PersonViewModels = new ObservableCollection<PersonViewModel>();

            CarViewModels = new ObservableCollection<CarViewModel>();

            FineViewModels = new ObservableCollection<FineViewModel>();

            ViolationViewModels = new ObservableCollection<ViolationViewModel>();

            InitializeComponent();

            LoadPersonViewModels();

            PersonPickerButton.ItemsSource = PersonViewModels;

            CarPickerButton.ItemsSource = CarViewModels;

            ViolationPickerButton.ItemsSource = ViolationViewModels;

            FinePickerButton.ItemsSource = FineViewModels;

        }

        private void LoadPersonViewModels()

        {

            var peopleService = container.Resolve<IPeopleService>();

            var dbRecords = peopleService.GetAllPersonsAsync().Result;

            var observableDtoModels = ModelHandler.PersonToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(PersonViewModels, observableDtoModels);

        }

        private void LoadCarViewModels(int personId)

        {

            var carService = container.Resolve<ICarService>();

            var dbRecords = carService.GetPersonCarsAsync(personId).Result;

            var observableDtoModels = ModelHandler.CarToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(CarViewModels, observableDtoModels);

        }

        private void LoadViolationsViewModels()

        {

            var violationService = container.Resolve<IViolationService>();

            var dbRecords = violationService.GetAllViolationsAsync().Result;

            var observableDtoModels = ModelHandler.ViolationsToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(ViolationViewModels, observableDtoModels);

        }

        private void LoadFineViewModels()

        {

            var fineService = container.Resolve<IFineService>();

            var dbRecords = fineService.GetAllFinesAsync().Result;

            var observableDtoModels = ModelHandler.FinesToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(FineViewModels, observableDtoModels);

        }

        private async void SaveButton\_Click(object sender, RoutedEventArgs e)

        {

            this.Cursor = Cursors.Hand;

            var person = PersonPickerButton.SelectedItem as PersonViewModel;

            var car = CarPickerButton.SelectedItem as CarViewModel;

            var violation = ViolationPickerButton.SelectedItem as ViolationViewModel;

            var fine = FinePickerButton.SelectedItem as FineViewModel;

            var finePrice = decimal.Parse(FinePriceNumericUpDown.Value.Value.ToString());

            var fineNumber = FineNumberNumericUpDown.Value.Value.ToString();

            var licenceBackOn = DateTime.Parse(LicenceBackOnDatePicker.Text);

            var personFinesService = container.Resolve<IPersonFinesService>();

            await personFinesService.CreatePersonFineAsync(person.Id, car.Id, violation.Id, fine.Id, finePrice, fineNumber, licenceBackOn);

            var result = await this.ShowMessageAsync("Записът бе добавен", "Нарушението бе успешно добавено в системата.", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Върни се обратно",

                NegativeButtonText = "Добави още",

                AnimateHide = true,

                AnimateShow = true,

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                this.Close();

            }

            else

            {

                ClearFields();

            }

            this.Cursor = Cursors.Arrow;

        }

        private async void CancelButton\_Click(object sender, RoutedEventArgs e)

        {

            this.Cursor = Cursors.Hand;

            var result = await this.ShowMessageAsync("Сигурни ли сте?", "Ако излезете сега, записът няма да бъде добавен в системата!", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Да",

                AnimateHide = true,

                AnimateShow = true,

                NegativeButtonText = "Не",

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                this.Close();

            }

            this.Cursor = Cursors.Arrow;

        }

        private void ClearButton\_Click(object sender, RoutedEventArgs e)

        {

            ClearFields();

        }

        private void ClearFields()

        {

            PersonPickerButton.SelectedItem = null;

            ClearButton.IsEnabled = false;

            SaveButton.IsEnabled = false;

            CarPickerLabel.Visibility = Visibility.Hidden;

            CarPickerButton.Visibility = Visibility.Hidden;

            CarPickerButton.SelectedItem = null;

            ViolationPickerLabel.Visibility = Visibility.Hidden;

            ViolationPickerButton.Visibility = Visibility.Hidden;

            ViolationPickerButton.SelectedItem = null;

            FinePickerLabel.Visibility = Visibility.Hidden;

            FinePickerButton.Visibility = Visibility.Hidden;

            FinePickerButton.SelectedItem = null;

            LicenceBackOnDatePickerLabel.Visibility = Visibility.Hidden;

            LicenceBackOnDatePicker.Visibility = Visibility.Hidden;

            LicenceBackOnDatePicker.Text = "";

            FinePriceLabel.Visibility = Visibility.Hidden;

            FinePriceNumericUpDown.Visibility = Visibility.Hidden;

            FinePriceNumericUpDown.Value = 0;

            FineNumberLabel.Visibility = Visibility.Hidden;

            FineNumberNumericUpDown.Visibility = Visibility.Hidden;

            FineNumberNumericUpDown.Value = 0;

        }

        private void PersonPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            CarPickerLabel.Visibility = Visibility.Visible;

            CarPickerButton.Visibility = Visibility.Visible;

            ClearButton.IsEnabled = true;

            var personViewModel = PersonPickerButton.SelectedItem as PersonViewModel;

            if (personViewModel != null)

            {

                LoadCarViewModels(personViewModel.Id);

            }

        }

        private void CarPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            ViolationPickerLabel.Visibility = Visibility.Visible;

            ViolationPickerButton.Visibility = Visibility.Visible;

            var carViewModel = CarPickerButton.SelectedItem as CarViewModel;

            if (carViewModel != null)

            {

                LoadViolationsViewModels();

            }

        }

        private void FinePickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            var fineViewModel = FinePickerButton.SelectedItem as FineViewModel;

            if (fineViewModel != null)

            {

                LicenceBackOnDatePickerLabel.Visibility = Visibility.Visible;

                LicenceBackOnDatePicker.Visibility = Visibility.Visible;

            }

        }

        private void ViolationPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            FinePickerLabel.Visibility = Visibility.Visible;

            FinePickerButton.Visibility = Visibility.Visible;

            var violationViewModel = ViolationPickerButton.SelectedItem as ViolationViewModel;

            if (violationViewModel != null)

            {

                LoadFineViewModels();

            }

        }

        private void LicenceBackOnDatePicker\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

        {

            if (!String.IsNullOrEmpty(LicenceBackOnDatePicker.Text))

            {

                FinePriceLabel.Visibility = Visibility.Visible;

                FinePriceNumericUpDown.Value = 0;

                FinePriceNumericUpDown.Visibility = Visibility.Visible;

                FineNumberLabel.Visibility = Visibility.Visible;

                FineNumberNumericUpDown.Visibility = Visibility.Visible;

                FineNumberNumericUpDown.Value = 0;

                SaveButton.IsEnabled = true;

            }

        }

    }

}

CreateCar.xaml – Съдържа SplitButton контроли за dropdown менютата използващи Binding и TextBlock полета използвани за попълване на информацията

<Controls:MetroWindow x:Name="CreateCarWindow">

Title="Регистриране на ново превозно средство" Height="350" Width="500" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="300"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <TextBlock HorizontalAlignment="Center" Margin="185,12,1006.4,10.6" Grid.Row="0" TextWrapping="Wrap" Text="Регистриране на нов автомобил" VerticalAlignment="Center" Height="27" Width="299" FontSize="20"/>

        <Label x:Name="BrandLabel" Content="Mарка:" HorizontalAlignment="Left" Margin="10,11.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="BrandTextBox" HorizontalAlignment="Left" Height="23" Margin="119,16.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи марка" VerticalAlignment="Top" Width="136" TextChanged="TextBoxChange"/>

        <Label x:Name="ModelLabel" Content="Mодел:" HorizontalAlignment="Left" Margin="260,9.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="ModelTextBox" HorizontalAlignment="Left" Height="23" Margin="361,15.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи модел" VerticalAlignment="Top" Width="123" TextChanged="TextBoxChange"/>

        <Label x:Name="EnginePowerLabel" Content="Конски сили:" HorizontalAlignment="Left" Margin="10,54.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:NumericUpDown x:Name="EnginePowerNumericUpDown" Value="50" Minimum="50" Margin="119,59.4,1235.4,215.4" Grid.Row="1"/>

        <Label x:Name="PeopleCarryLabel" Content="Брой места:" HorizontalAlignment="Left" Margin="260,54.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:NumericUpDown x:Name="PeopleCarryNumericUpDown" Value="3" Minimum="3" Margin="361,59.4,1006.4,215.4" Grid.Row="1"/>

        <Label x:Name="FuelPickerLabel" Content="Гориво:" HorizontalAlignment="Left" Margin="10,97.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:SplitButton x:Name="FuelPickerButton" ItemsSource="{Binding FuelViewModels}" DisplayMemberPath="Name" SelectionChanged="FuelPickerButton\_SelectionChanged" Margin="119,103.4,1235.4,171.4" Grid.Row="1"/>

        <Label x:Name="EmissionStandartLabel" Content="Стандарт:" HorizontalAlignment="Left" Margin="260,97.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:SplitButton x:Name="EmissionStandartPickerButton" ItemsSource="{Binding EmissionStandartViewModels}" DisplayMemberPath="Name" SelectionChanged="EmissionStandartPickerButton\_SelectionChanged" Margin="361,103.4,1006.4,171.4" Grid.Row="1"/>

        <Label x:Name="WeightLabel" Content="Тегло:" HorizontalAlignment="Left" Margin="10,144.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:NumericUpDown x:Name="WeightNumericUpDown" Value="0" Minimum="0" Margin="119,148.4,1235.4,126.4" Grid.Row="1"/>

        <Label x:Name="ColorLabel" Content="Цвят:" HorizontalAlignment="Left" Margin="260,144.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="ColorTextBox" HorizontalAlignment="Left" Height="23" Margin="361,148.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи цвят" VerticalAlignment="Top" Width="123" TextChanged="TextBoxChange"/>

        <Label x:Name="NumberLabel" Content="Номер:" HorizontalAlignment="Left" Margin="13,186.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="NumberTextBox" HorizontalAlignment="Left" Height="23" Margin="119,191.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи номер" VerticalAlignment="Top" Width="136" TextChanged="TextBoxChange"/>

        <Button x:Name="SaveButton" Content="Запиши" HorizontalAlignment="Left" Margin="334,224.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="150" Height="38" FontSize="14" Click="SaveButton\_Click" Cursor="Hand" IsEnabled="False"/>

        <Button x:Name="CancelButton" Content="Отказ" HorizontalAlignment="Left" Margin="10,223.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="150" Height="38" FontSize="14" Click="CancelButton\_Click" Cursor="Hand"/>

        <Button x:Name="ClearButton" Content="Изчисти полетата" HorizontalAlignment="Left" Margin="165,224.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="164" Height="38" FontSize="13" Click="ClearButton\_Click" Cursor="Hand" IsEnabled="False"/>

    </Grid>

</Controls:MetroWindow>

CreateCar.xaml.cs - Съдържа Autofac контейнер, чрез който взимаме необходимите ни инстанции, колекции в които популираме view model-ите използвани за Binding-а и диалог прозорците.

public partial class CreateCar : MetroWindow

    {

        Autofac.IContainer container = ContainerConfiguration.GetContainer();

        public ObservableCollection<FuelViewModel> FuelViewModels { get; set; }

        public ObservableCollection<EmissionStandartViewModel> EmissionStandartViewModels { get; set; }

        public CreateCar()

        {

            FuelViewModels = new ObservableCollection<FuelViewModel>();

            EmissionStandartViewModels = new ObservableCollection<EmissionStandartViewModel>();

            InitializeComponent();

            LoadFuelViewModels();

            LoadEmissionStandartViewModels();

            FuelPickerButton.ItemsSource = FuelViewModels;

            EmissionStandartPickerButton.ItemsSource = EmissionStandartViewModels;

        }

        private void LoadFuelViewModels()

        {

            var fuelService = container.Resolve<IFuelService>();

            var dbRecords = fuelService.GetAllFuelsAsync().Result;

            var observableDtoModels = ModelHandler.FuelsToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(FuelViewModels, observableDtoModels);

        }

        private void LoadEmissionStandartViewModels()

        {

            var emissionStandartService = container.Resolve<IEmissionStandartService>();

            var dbRecords = emissionStandartService.GetAllEmissionStandartsAsync().Result;

            var observableDtoModels = ModelHandler.EmissionStandartsToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(EmissionStandartViewModels, observableDtoModels);

        }

        private void TextBoxChange(object sender, TextChangedEventArgs e)

        {

            ResolveClearButtonStatus();

            ResolveSaveButtonStatus();

        }

        private void FuelPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            ResolveClearButtonStatus();

            ResolveSaveButtonStatus();

        }

        private void EmissionStandartPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            ResolveClearButtonStatus();

            ResolveSaveButtonStatus();

        }

        private async void SaveButton\_Click(object sender, RoutedEventArgs e)

        {

            this.Cursor = Cursors.Hand;

            var carBrand = BrandTextBox.Text;

            var carModel = ModelTextBox.Text;

            var enginePower = int.Parse(EnginePowerNumericUpDown.Value.ToString());

            var peopleCarry = int.Parse(PeopleCarryNumericUpDown.Value.ToString());

            var weight = int.Parse(WeightNumericUpDown.Value.ToString());

            var color = ColorTextBox.Text;

            var number = NumberTextBox.Text;

            var fuel = FuelPickerButton.SelectedItem as FuelViewModel;

            var emissionStandart = EmissionStandartPickerButton.SelectedItem as EmissionStandartViewModel;

            var carService = container.Resolve<ICarService>();

            await carService.CreateCarAsync(carBrand, carModel, enginePower, peopleCarry, weight, color, fuel.Id, emissionStandart.Id, number);

            var result = await this.ShowMessageAsync("Записът бе добавен", "Превозното срество бе успешно добавено в системата.", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Върни се обратно",

                NegativeButtonText = "Добави още",

                AnimateHide = true,

                AnimateShow = true,

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                this.Close();

            }

            else

            {

                ClearFields();

            }

            this.Cursor = Cursors.Arrow;

        }

        private async void CancelButton\_Click(object sender, RoutedEventArgs e)

        {

            this.Cursor = Cursors.Hand;

            var result = await this.ShowMessageAsync("Сигурни ли сте?", "Ако излезете сега, записът няма да бъде добавен в системата!", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Да",

                AnimateHide = true,

                AnimateShow = true,

                NegativeButtonText = "Не",

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                this.Close();

            }

            this.Cursor = Cursors.Arrow;

        }

        private void ClearButton\_Click(object sender, RoutedEventArgs e)

        {

            ClearFields();

        }

        private void ClearFields()

        {

            BrandTextBox.Text = "";

            ModelTextBox.Text = "";

            EnginePowerNumericUpDown.Value = 50;

            PeopleCarryNumericUpDown.Value = 3;

            WeightNumericUpDown.Value = 0;

            ColorTextBox.Text = "";

            NumberTextBox.Text = "";

            ClearButton.IsEnabled = false;

            SaveButton.IsEnabled = false;

            FuelPickerButton.SelectedItem = null;

            EmissionStandartPickerButton.SelectedItem = null;

        }

        private void ResolveClearButtonStatus()

        {

            bool buttonEnabled = false;

            var fuel = FuelPickerButton.SelectedItem as FuelViewModel;

            var emissionStandart = EmissionStandartPickerButton.SelectedItem as EmissionStandartViewModel;

            if (!string.IsNullOrEmpty(BrandTextBox.Text) || !string.IsNullOrEmpty(ModelTextBox.Text) || !string.IsNullOrEmpty(ColorTextBox.Text) || !string.IsNullOrEmpty(NumberTextBox.Text))

            {

                buttonEnabled = true;

            }

            if (fuel != null || emissionStandart != null)

            {

                buttonEnabled = true;

            }

            ClearButton.IsEnabled = buttonEnabled ? true : false;

        }

        private void ResolveSaveButtonStatus()

        {

            bool buttonEnabled = true;

            var fuel = FuelPickerButton.SelectedItem as FuelViewModel;

            var emissionStandart = EmissionStandartPickerButton.SelectedItem as EmissionStandartViewModel;

            if (string.IsNullOrEmpty(BrandTextBox.Text) || string.IsNullOrEmpty(ModelTextBox.Text) || string.IsNullOrEmpty(ColorTextBox.Text) || string.IsNullOrEmpty(NumberTextBox.Text))

            {

                buttonEnabled = false;

            }

            if (fuel == null || emissionStandart == null)

            {

                buttonEnabled = false;

            }

            SaveButton.IsEnabled = buttonEnabled ? true : false;

        }

    }

}

CreatePerson.xaml - Съдържа SplitButton контроли за dropdown менютата използващи Binding и TextBlock полета използвани за попълване на информацията

<Controls:MetroWindow x:Name="CreatePersonWindow">

Title="Регистриране на нов потребител" Height="430" Width="500" Background="Gainsboro" WindowStartupLocation="CenterScreen">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="Auto"/>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="50"/>

            <RowDefinition Height="380"/>

        </Grid.RowDefinitions>

        <Rectangle Grid.Row="0" Fill="GhostWhite" />

        <TextBlock HorizontalAlignment="Center" Margin="179,13,1006.4,9.6" Grid.Row="0" TextWrapping="Wrap" Text="Регистриране на нов потребител" VerticalAlignment="Center" Height="27" Width="305" FontSize="20"/>

        <Label x:Name="FirstNameLabel" Content="Въведете име:" HorizontalAlignment="Left" Margin="10,11.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="FirstNameTextBox" HorizontalAlignment="Left" Height="23" Margin="165,15.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи име тук" VerticalAlignment="Top" Width="319" TextChanged="TextBoxChange"/>

        <Label x:Name="LastNameLabel" Content="Въведете фамилия:" HorizontalAlignment="Left" Margin="10,56.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="LastNameTextBox" HorizontalAlignment="Left" Height="23" Margin="165,61.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи фамилия тук" VerticalAlignment="Top" Width="319" TextChanged="TextBoxChange"/>

        <Label x:Name="EGNLabel" Content="Въведете ЕГН:" HorizontalAlignment="Left" Margin="10,102.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="EGNTextBox" HorizontalAlignment="Left" Height="23" Margin="165,107.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи ЕГН тук" VerticalAlignment="Top" Width="319" TextChanged="TextBoxChange"/>

        <Label x:Name="CardIdLabel" Content="Въведете № ЛК:" HorizontalAlignment="Left" Margin="10,147.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <TextBox x:Name="CardIdTextBox" HorizontalAlignment="Left" Height="23" Margin="165,153.4,0,0" Grid.Row="1" TextWrapping="Wrap" Controls:TextBoxHelper.ClearTextButton="True" Controls:TextBoxHelper.Watermark="Въведи № на лична карта тук" VerticalAlignment="Top" Width="319" TextChanged="TextBoxChange"/>

        <Label x:Name="GenderPickerLabel" Content="Изберете пол:" HorizontalAlignment="Left" Margin="10,192.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:SplitButton x:Name="GenderPickerButton" ItemsSource="{Binding GenderViewModels}" DisplayMemberPath="Name" SelectionChanged="GenderPickerButton\_SelectionChanged" Margin="165,197.4,1006.4,157.4" Grid.Row="1"/>

        <Label x:Name="CarPickerLabel" Content="Изберете кола:" HorizontalAlignment="Left" Margin="10,237.4,0,0" Grid.Row="1" VerticalAlignment="Top" FontSize="16"/>

        <Controls:SplitButton x:Name="CarPickerButton" ItemsSource="{Binding CarViewModels}" DisplayMemberPath="DisplayName" SelectionChanged="CarPickerButton\_SelectionChanged" Margin="165,243.4,1006.4,111.4" Grid.Row="1"/>

        <Button x:Name="SaveButton" Content="Запиши" HorizontalAlignment="Left" Margin="334,304.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="150" Height="38" FontSize="14" Click="SaveButton\_Click" Cursor="Hand" IsEnabled="False"/>

        <Button x:Name="CancelButton" Content="Отказ" HorizontalAlignment="Left" Margin="10,304.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="150" Height="38" FontSize="14" Click="CancelButton\_Click" Cursor="Hand"/>

        <Button x:Name="ClearButton" Content="Изчисти полетата" HorizontalAlignment="Left" Margin="165,304.4,0,0" Grid.Row="1" VerticalAlignment="Top" Width="164" Height="38" FontSize="13" Click="ClearButton\_Click" Cursor="Hand" IsEnabled="False"/>

    </Grid>

</Controls:MetroWindow>

CreatePerson.xaml.cs - Съдържа Autofac контейнер, чрез който взимаме необходимите ни инстанции, колекции в които популираме view model-ите използвани за Binding-а и диалог прозорците.

ublic partial class CreatePerson : MetroWindow

    {

        Autofac.IContainer container = ContainerConfiguration.GetContainer();

        public ObservableCollection<GenderViewModel> GenderViewModels { get; set; }

        public ObservableCollection<CarViewModel> CarViewModels { get; set; }

        public CreatePerson()

        {

            GenderViewModels = new ObservableCollection<GenderViewModel>();

            CarViewModels = new ObservableCollection<CarViewModel>();

            InitializeComponent();

            LoadGenderViewModels();

            LoadCarViewModels();

            GenderPickerButton.ItemsSource = GenderViewModels;

            CarPickerButton.ItemsSource = CarViewModels;

        }

        private void LoadGenderViewModels()

        {

            var gendersService = container.Resolve<IGendersService>();

            var dbRecords = gendersService.GetAllGendersAsync().Result;

            var observableDtoModels = ModelHandler.GendersToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(GenderViewModels, observableDtoModels);

        }

        private void LoadCarViewModels()

        {

            var carService = container.Resolve<ICarService>();

            var dbRecords = carService.GetAllCarsAsync().Result;

            var observableDtoModels = ModelHandler.CarToObservableDto(dbRecords);

            ModelHandler.ProcessObservableDtoModels(CarViewModels, observableDtoModels);

        }

        private void GenderPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            ResolveClearButtonStatus();

            ResolveSaveButtonStatus();

        }

        private void CarPickerButton\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            ResolveClearButtonStatus();

            ResolveSaveButtonStatus();

        }

        private async void SaveButton\_Click(object sender, RoutedEventArgs e)

        {

            this.Cursor = Cursors.Hand;

            var gender = GenderPickerButton.SelectedItem as GenderViewModel;

            var person = new Person()

            {

                FirstName = FirstNameTextBox.Text,

                LastName = LastNameTextBox.Text,

                EGN = EGNTextBox.Text,

                CardId = CardIdTextBox.Text,

                GenderId = gender.Id

            };

            var car = CarPickerButton.SelectedItem as CarViewModel;

            var personCarsService = container.Resolve<IPersonCarsService>();

            await personCarsService.CreatePersonCarAsync(person, car.Id);

            var result = await this.ShowMessageAsync("Записът бе добавен", "Потребителят бе успешно добавен в системата.", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Върни се обратно",

                NegativeButtonText = "Добави още",

                AnimateHide = true,

                AnimateShow = true,

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                this.Close();

            }

            else

            {

                ClearFields();

            }

            this.Cursor = Cursors.Arrow;

        }

        private async void CancelButton\_Click(object sender, RoutedEventArgs e)

        {

            this.Cursor = Cursors.Hand;

            var result = await this.ShowMessageAsync("Сигурни ли сте?", "Ако излезете сега, записът няма да бъде добавен в системата!", MessageDialogStyle.AffirmativeAndNegative, MetroDialogOptions = new MetroDialogSettings()

            {

                AffirmativeButtonText = "Да",

                AnimateHide = true,

                AnimateShow = true,

                NegativeButtonText = "Не",

                DefaultButtonFocus = MessageDialogResult.Affirmative,

                DialogResultOnCancel = MessageDialogResult.Canceled,

            });

            if (result == MessageDialogResult.Affirmative)

            {

                this.Close();

            }

            this.Cursor = Cursors.Arrow;

        }

        private void ClearButton\_Click(object sender, RoutedEventArgs e)

        {

            ClearFields();

        }

        private void ClearFields()

        {

            FirstNameTextBox.Text = "";

            LastNameTextBox.Text = "";

            EGNTextBox.Text = "";

            CardIdTextBox.Text = "";

            ClearButton.IsEnabled = false;

            SaveButton.IsEnabled = false;

            GenderPickerButton.SelectedItem = null;

            CarPickerButton.SelectedItem = null;

        }

        private void TextBoxChange(object sender, TextChangedEventArgs e)

        {

            ResolveClearButtonStatus();

            ResolveSaveButtonStatus();

        }

        private void ResolveClearButtonStatus()

        {

            bool buttonEnabled = false;

            var gender = GenderPickerButton.SelectedItem as GenderViewModel;

            var car = CarPickerButton.SelectedItem as CarViewModel;

            if (!string.IsNullOrEmpty(FirstNameTextBox.Text) || !string.IsNullOrEmpty(LastNameTextBox.Text) || !string.IsNullOrEmpty(EGNTextBox.Text) || !string.IsNullOrEmpty(CardIdTextBox.Text))

            {

                buttonEnabled = true;

            }

            if (gender != null || car != null)

            {

                buttonEnabled = true;

            }

            ClearButton.IsEnabled = buttonEnabled ? true : false;

        }

        private void ResolveSaveButtonStatus()

        {

            bool buttonEnabled = true;

            var gender = GenderPickerButton.SelectedItem as GenderViewModel;

            var car = CarPickerButton.SelectedItem as CarViewModel;

            if (string.IsNullOrEmpty(FirstNameTextBox.Text) || string.IsNullOrEmpty(LastNameTextBox.Text) || string.IsNullOrEmpty(EGNTextBox.Text) || string.IsNullOrEmpty(CardIdTextBox.Text))

            {

                buttonEnabled = false;

            }

            if (gender == null || car == null)

            {

                buttonEnabled = false;

            }

            SaveButton.IsEnabled = buttonEnabled ? true : false;

        }

    }

}

CarSystemDbContext.cs – Клас указващ връзката с базата данни. Тук са DbSet свойствата, които се преобразуват в таблици.

    public class CarSystemDbContext : DbContext

    {

        public CarSystemDbContext() : base("CarSystemDb")

        {

            Database.SetInitializer(new MigrateDatabaseToLatestVersion<CarSystemDbContext, Migrations.Configuration>());

        }

        public DbSet<Person> People { get; set; }

        public DbSet<Gender> Genders { get; set; }

        public DbSet<Car> Cars { get; set; }

        public DbSet<EmissionStandart> EmissionStandarts { get; set; }

        public DbSet<Fine> Fines { get; set; }

        public DbSet<Fuel> Fuels { get; set; }

        public DbSet<PersonCars> PersonCars { get; set; }

        public DbSet<PersonFines> PersonFines { get; set; }

        public DbSet<Violation> Violations { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)

        {

            base.OnModelCreating(modelBuilder);

        }

        public override Task<int> SaveChangesAsync()

        {

            this.ApplyDeletionRules();

            return base.SaveChangesAsync();

        }

        private void ApplyDeletionRules()

        {

            var entitiesForDeletion = this.ChangeTracker.Entries()

                .Where(e => e.State == EntityState.Deleted && e.Entity is IDeletable);

            foreach (var entry in entitiesForDeletion)

            {

                var entity = (IDeletable)entry.Entity;

                entity.IsDeleted = true;

                entry.State = EntityState.Modified;

            }

        }

    }

}

Configuration.cs - Предварително попълване на базата с данни:

    internal sealed class Configuration : DbMigrationsConfiguration<CarSystem.Data.CarSystemDbContext>

    {

        public Configuration()

        {

            AutomaticMigrationsEnabled = true;

            AutomaticMigrationDataLossAllowed = true;

        }

        protected override void Seed(CarSystem.Data.CarSystemDbContext context)

        {

            // Seed genders

            context.Genders

                .AddOrUpdate(x => x.Id,

                new Gender() { Id = 1, Name = "Мъж" },

                new Gender() { Id = 2, Name = "Жена" }

                );

            // Seed emission standarts

            context.EmissionStandarts

                .AddOrUpdate(x => x.Id,

                new EmissionStandart() { Id = 1, Name = "Euro 1" },

                new EmissionStandart() { Id = 2, Name = "Euro 2" },

                new EmissionStandart() { Id = 3, Name = "Euro 3" },

                new EmissionStandart() { Id = 4, Name = "Euro 4" },

                new EmissionStandart() { Id = 5, Name = "Euro 5" },

                new EmissionStandart() { Id = 6, Name = "Euro 6" },

                new EmissionStandart() { Id = 7, Name = "Euro 6 RDE" }

                );

            // Seed fuels

            context.Fuels

                .AddOrUpdate(x => x.Id,

                new Fuel() { Id = 1, Name = "Дизел" },

                new Fuel() { Id = 2, Name = "Бензин" },

                new Fuel() { Id = 3, Name = "Газ" }

                );

            // Seed fines

            context.Fines

                .AddOrUpdate(x => x.Id,

                new Fine() { Id = 1, Name = "Превишена скорост", Violation = "Движи се с превишена скорост" },

                new Fine() { Id = 2, Name = "Оборудване", Violation = "Няма необходимите предпазни средства" },

                new Fine() { Id = 3, Name = "Неспазване на закон", Violation = "Не спазва указанията на знаците" },

                new Fine() { Id = 4, Name = "Гуми", Violation = "Не е с необходимия за сезона вид гуми" }

                );

            // Seed violations

            context.Violations

                .AddOrUpdate(x => x.Id,

                new Violation() { Id = 1, Name = "Camera", Message = "Засечен от камера" },

                new Violation() { Id = 2, Name = "Slip", Message = "Получил фиш" },

                new Violation() { Id = 3, Name = "Act", Message = "Получил акт" }

                );

            // Seed Person

            context.People

                .AddOrUpdate(x => x.Id,

                new Person() { Id = 1, CardId = "1234", EGN = "12345", FirstName = "Gordon", LastName = "Freeman", GenderId = 1 }

                );

            // Seed Car

            context.Cars

                .AddOrUpdate(x => x.Id,

                new Car() { Id = 1, Brand = "Audi", EmissionStandartId = 1, EnginePower = 125, FuelId = 3, Model = "A3", Number = "PA1997KM", Paint = "Black", PeopleCarry = 5, Weight = 3500 }

                );

            // Seed Person Cars

            context.PersonCars

                .AddOrUpdate(x => x.Id,

                new PersonCars() { Id = 1, PersonId = 1, CarId = 1 }

                );

            // Seed Person Fines

            context.PersonFines

                .AddOrUpdate(x => x.Id,

                new PersonFines() { Id = 1, CarId = 1, FineId = 1, FineNumber = "3021", PersonId = 1, LicenceBackOn = DateTime.Now, Price = 150, ViolationId = 1 }

                );

        }

    }

}

Модели:

    public class BaseEntity : IDeletable

    {

        public int Id { get; set; }

        public bool IsDeleted { get; set; }

    }

}

    public class Car : BaseEntity

    {

        public string Brand { get; set; }

        public string Model { get; set; }

        public string Paint { get; set; }

        public int EnginePower { get; set; }

        public int PeopleCarry { get; set; }

        public decimal Weight { get; set; }

        public string Number { get; set; }

        public int FuelId { get; set; }

        public Fuel Fuel { get; set; }

        public int EmissionStandartId { get; set; }

        public EmissionStandart EmissionStandart { get; set; }

        public virtual ICollection<PersonCars> PersonCars { get; set; }

        public virtual ICollection<PersonFines> PersonFines { get; set; }

    }

}

    public class EmissionStandart : BaseEntity

    {

        public string Name { get; set; }

        public ICollection<Car> Cars { get; set; }

    }

}

    public class Fine : BaseEntity

    {

        public string Name { get; set; }

        public string Violation { get; set; }

        public virtual ICollection<PersonFines> PersonFines { get; set; }

    }

}

    public class Fuel : BaseEntity

    {

        public string Name { get; set; }

        public ICollection<Car> Cars { get; set; }

    }

    public class Gender : BaseEntity

    {

        public string Name { get; set; }

        public ICollection<Person> Persons { get; set; }

    }

}

    public class Person : BaseEntity

    {

        public string FirstName { get; set; }

        public string LastName { get; set; }

        public string EGN { get; set; }

        public string CardId { get; set; }

        public int GenderId { get; set; }

        public Gender Gender { get; set; }

        public virtual ICollection<PersonCars> PersonCars { get; set; }

        public virtual ICollection<PersonFines> PersonFines { get; set; }

    }

}

    public class Violation : BaseEntity

    {

        public string Name { get; set; }

        public string Message { get; set; }

        public virtual ICollection<PersonFines> PersonFines { get; set; }

    }

}

    public class PersonCars : BaseEntity

    {

        public int PersonId { get; set; }

        public int CarId { get; set; }

        public virtual Person Person { get; set; }

        public virtual Car Car { get; set; }

    }

}

    public class PersonFines : BaseEntity

    {

        public decimal Price { get; set; }

        public DateTime LicenceBackOn { get; set; }

        public string FineNumber { get; set; }

        public int PersonId { get; set; }

        public int FineId { get; set; }

        public int CarId { get; set; }

        public int ViolationId { get; set; }

        public virtual Person Person { get; set; }

        public virtual Fine Fine { get; set; }

        public virtual Car Car { get; set; }

        public virtual Violation Violation { get; set; }

    }

}

Сървиси – ползват се за извличане на информация от базата данни и връщането на информацията към WPF частта.

public class CarService : ICarService

    {

        private readonly CarSystemDbContext context;

        public CarService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<Car>> GetPersonCarsAsync(int personId)

        {

            return this.context.PersonCars

                .Where(x => x.PersonId == personId)

                .Select(x => x.Car)

                .ToListAsync();

        }

        public Task<List<Car>> GetAllCarsAsync()

        {

            return this.context.Cars

                .ToListAsync();

        }

        public async Task CreateCarAsync(string carBrand, string carModel, int enginePower, int peopleCarry, int weight, string color, int fuelId, int emissionStandartId, string number)

        {

            var car = new Car()

            {

                Brand = carBrand,

                EmissionStandartId = emissionStandartId,

                EnginePower = enginePower,

                FuelId = fuelId,

                Model = carModel,

                PeopleCarry = peopleCarry,

                Paint = color,

                Weight = weight,

                Number = number

            };

            this.context.Cars.Add(car);

            await this.context.SaveChangesAsync();

        }

    }

}

    public class EmissionStandartService : IEmissionStandartService

    {

        private readonly CarSystemDbContext context;

        public EmissionStandartService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<EmissionStandart>> GetAllEmissionStandartsAsync()

        {

            return this.context.EmissionStandarts

                .ToListAsync();

        }

    }

}

public class ExportService : IExportService

    {

        public void ExportPersonInformation(string fileName, string personName, List<PersonFines> personFines)

        {

            var writer = new PdfWriter(fileName);

            var pdf = new PdfDocument(writer);

            var document = new Document(pdf, PageSize.A4);

            document.Add(new Paragraph($"Violations made by {personName}"));

            document.Add(new Paragraph(Environment.NewLine));

            if (!personFines.Any())

            {

                document.Add(new Paragraph("This person has no violations."));

            }

            else

            {

                document.Add(new Paragraph($"Person information:\nName: {personFines[0].Person.FirstName} {personFines[0].Person.LastName}\r\nEGN: {personFines[0].Person.EGN}\r\nCard Id: {personFines[0].Person.CardId}\r\nOwned vehicles: {personFines[0].Person.PersonCars.Count}\r\nNumber of fines: {personFines[0].Person.PersonFines.Count}"));

                document.Add(new Paragraph(Environment.NewLine));

                document.Add(new Paragraph("Violations:\n"));

                foreach (var item in personFines)

                {

                    document.Add(new Paragraph($"Type: {item.Fine.Name}\r\nViolation: {item.Fine.Violation}\r\nFine number: {item.FineNumber}\r\nVehicle: {item.Car.Brand} {item.Car.Model} {item.Car.Number}\r\n"));

                }

            }

            document.Close();

            writer.Close();

        }

        public void ExportCarInformation(string fileName, string carInfo, List<PersonFines> carFines)

        {

            var writer = new PdfWriter(fileName);

            var pdf = new PdfDocument(writer);

            var document = new Document(pdf, PageSize.A4);

            document.Add(new Paragraph($"Violations made by car {carInfo}"));

            document.Add(new Paragraph(Environment.NewLine));

            if (!carFines.Any())

            {

                document.Add(new Paragraph("This person has no violations."));

            }

            else

            {

                document.Add(new Paragraph($"Person information:\nName: {carFines[0].Person.FirstName} {carFines[0].Person.LastName}\r\nEGN: {carFines[0].Person.EGN}\r\nCard Id: {carFines[0].Person.CardId}\r\nOwned vehicles: {carFines[0].Person.PersonCars.Count}\r\nNumber of fines: {carFines[0].Person.PersonFines.Count}"));

                document.Add(new Paragraph(Environment.NewLine));

                document.Add(new Paragraph("Violations:\n"));

                foreach (var item in carFines)

                {

                    document.Add(new Paragraph($"Type: {item.Fine.Name}\r\nViolation: {item.Fine.Violation}\r\nFine number: {item.FineNumber}\r\nVehicle: {item.Car.Brand} {item.Car.Model} {item.Car.Number}\r\n"));

                }

            }

            document.Close();

            writer.Close();

        }

    }

}

    public class FineService : IFineService

    {

        private readonly CarSystemDbContext context;

        public FineService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<Fine>> GetAllFinesAsync()

        {

            return this.context.Fines

                .ToListAsync();

        }

    }

}

    public class FuelService : IFuelService

    {

        private readonly CarSystemDbContext context;

        public FuelService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<Fuel>> GetAllFuelsAsync()

        {

            return this.context.Fuels

                .ToListAsync();

        }

    }

}

    public class GendersService : IGendersService

    {

        private readonly CarSystemDbContext context;

        public GendersService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<Gender>> GetAllGendersAsync()

        {

            return this.context.Genders

                .ToListAsync();

        }

    }

}

    public class PeopleService : IPeopleService

    {

        private readonly CarSystemDbContext context;

        public PeopleService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<Person>> GetAllPersonsAsync()

        {

            return this.context.People

                .ToListAsync();

        }

    }

}

    public class PersonCarsService : IPersonCarsService

    {

        private readonly CarSystemDbContext context;

        public PersonCarsService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public async Task CreatePersonCarAsync(Person person, int carId)

        {

            var personCarRecord = new PersonCars()

            {

                Person = person,

                CarId = carId

            };

            this.context.PersonCars.Add(personCarRecord);

            await this.context.SaveChangesAsync();

        }

    }

}

    public class PersonFinesService : IPersonFinesService

    {

        private readonly CarSystemDbContext context;

        public PersonFinesService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<PersonFines>> GetFilteredPersonFinesAsync(string violationName = "", string cardId = "", string egn = "", string carNumber = "", string fineNumber = "")

        {

            var personFines = this.context.PersonFines

                .Where(x => x.Violation.Name.Contains(violationName) && !x.IsDeleted)

                .AsQueryable();

            if (!string.IsNullOrEmpty(cardId))

            {

                personFines = personFines

                    .Where(x => x.Person.CardId.Contains(cardId))

                    .AsQueryable();

            }

            if (!string.IsNullOrEmpty(egn))

            {

                personFines = personFines

                    .Where(x => x.Person.EGN.Contains(egn))

                    .AsQueryable();

            }

            if (!string.IsNullOrEmpty(carNumber))

            {

                personFines = personFines

                    .Where(x => x.Car.Number.Contains(carNumber))

                    .AsQueryable();

            }

            if (!string.IsNullOrEmpty(fineNumber))

            {

                personFines = personFines.

                    Where(x => x.FineNumber.Contains(fineNumber))

                    .AsQueryable();

            }

            return personFines.ToListAsync();

        }

        public async Task DeletePersonFineAsync(int personFineId)

        {

            var dbRecord = await this.context.PersonFines

                .FirstOrDefaultAsync(x => x.Id == personFineId);

            if (dbRecord != null)

            {

                this.context.PersonFines.Remove(dbRecord);

            }

            await this.context.SaveChangesAsync();

        }

        public async Task CreatePersonFineAsync(int personId, int carId, int violationId, int fineId, decimal finePrice, string fineNumber, DateTime licenceBackOn)

        {

            var personFineRecord = new PersonFines()

            {

                PersonId = personId,

                CarId = carId,

                ViolationId = violationId,

                FineId = fineId,

                FineNumber = fineNumber,

                Price = finePrice,

                LicenceBackOn = licenceBackOn

            };

            this.context.PersonFines.Add(personFineRecord);

            await this.context.SaveChangesAsync();

        }

        public Task<List<PersonFines>> GetPersonFinesByPersonId(int personId)

        {

            return this.context.PersonFines

                .Where(x => x.PersonId == personId)

                .ToListAsync();

        }

        public Task<List<PersonFines>> GetCarFinesByCarId(int carId)

        {

            return this.context.PersonFines

                .Where(x => x.CarId == carId)

                .ToListAsync();

        }

    }

}

    public class ViolationService : IViolationService

    {

        private readonly CarSystemDbContext context;

        public ViolationService(CarSystemDbContext context)

        {

            this.context = context;

        }

        public Task<List<Violation>> GetAllViolationsAsync()

        {

            return this.context.Violations

                .ToListAsync();

        }

    }

}

**Заключение**

Продуктът е практически използваем и върши нещата, които има по предназначение. Има възможност лесно добавяне на нова функционалност и разширяване на текущата.

Къде може да се подобри – повече валидации, изчистване на наличен повтарящ се код, добавяне на тестове.

**Използвана литература**

1. <https://stackoverflow.com/>
2. <https://docs.microsoft.com>
3. <https://mahapps.com/> - modern WPF UI
4. <https://autofac.org/> - Dependency Injection container
5. <https://automapper.org/> - Map between models

**Приложение**

**Линк към GitHub -** [**https://github.com/KrasimirEtov/CarSystem**](https://github.com/KrasimirEtov/CarSystem)