# Создание формы изменения партнера

В каталоге Views необходимо добавить новый UserControl, при помощи контекстного меню с именем EditPartnerView:

Первое что нам необходимо это исправить x:Class с:

x:Class="/Views.EditPartnerView"

на:

x:Class="DemoApp.Views.EditPartnerView"

Также необходимо открыть файл EditPartnerView.axaml.cs и изменить его с:

namespace /Views;

на:

namespace DemoApp.Views;

Теперь нам необходимо создать ViewModel для нашего UserControl. В каталоге ViewModels необходимо добавить новый Class при помощи контекстного меню с именем EditPartnerViewModel:

Изменим наш файл следующим образом:

namespace DemoApp.ViewModels

{

public partial class EditPartnerViewModel : ViewModelBase

{

private readonly DbDemoFdbContext \_context;

private readonly PartnersViewModel? \_parentViewModel;

private readonly Partner \_partnerEntity;

}

}

Здесь мы объявили объект для работы с базой данных, сущность партнер и родительскую модель представления.

Далее нам необходимо добавить свойства, которые у нас будут расположены на пользовательском элементе управления:

[ObservableProperty]

private string? \_partnerName;

[ObservableProperty]

private string? \_partnerInn;

[ObservableProperty]

private string? \_director;

[ObservableProperty]

private string? \_directorPhone;

[ObservableProperty]

private string? \_directorMail;

[ObservableProperty]

private int? \_partnerRating;

[ObservableProperty]

private int? \_selectedPartnerType;

[ObservableProperty]

private ObservableCollection<PartnerType> \_partnerTypes = new();

[ObservableProperty]

private string? \_errorMessage;

Далее нам необходимо создать конструктор для данной модели представления

public EditPartnerViewModel(DbDemoFdbContext context, Partner? partner = null, PartnersViewModel? parentViewModel = null)

{

\_context = context;

\_parentViewModel = parentViewModel;

\_partnerEntity = partner ?? new Partner();

LoadPartnerTypes();

LoadPartnerData();

}

В качестве параметров конструктор принимает объект для работы с базой данных, сущность партнер и родительскую модель представления с дальнейшем присвоением значений. В конструкторе необходимо вызвать два метода: первый для загрузки типов партнеров, второй для загрузки данных о партнере:

Добавим реализацию этих методов

private void LoadPartnerTypes()

{

var types = \_context.PartnerTypes.ToList();

PartnerTypes = new ObservableCollection<PartnerType>(types);

}

private void LoadPartnerData()

{

PartnerName = \_partnerEntity.PartnerName;

PartnerInn = \_partnerEntity.PartnerInn;

Director = \_partnerEntity.Director;

DirectorPhone = \_partnerEntity.DirectorPhone;

DirectorMail = \_partnerEntity.DirectorMail;

PartnerRating = \_partnerEntity.PartnerRating;

SelectedPartnerType = \_partnerEntity.PartnerType;

}

Кроме того, нам необходимо добавить два метода для кнопок, которые будут расположены на форме

[RelayCommand]

private void Save()

{

//сохранение

}

[RelayCommand]

private void Cancel()

{

//отмена

}

Подключим модель представления к нашему представлению. Для этого отредактируем файл EditPartnerView.axaml.

Добавим в заголовок следующие строки:

xmlns:vm="using:DemoApp.ViewModels"

x:DataType="vm:EditPartnerViewModel"

Следующим шагом будет разметка нашего контрола.

Для этого вместо:

Welcome to Avalonia!

необходимо вставить:

<Grid Margin="20">

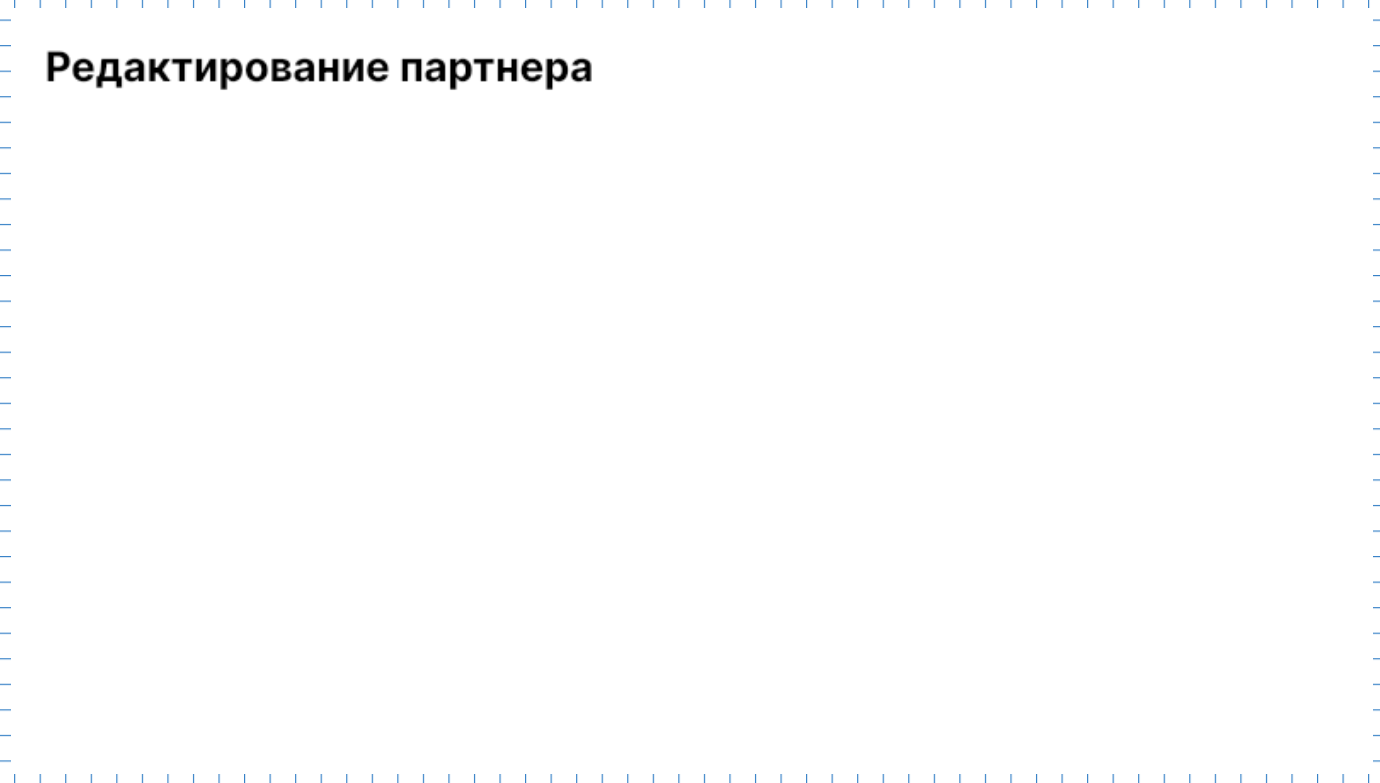
<StackPanel Spacing="10">

<TextBlock Text="Редактирование партнера" FontSize="24" FontWeight="Bold" />

</StackPanel>

</Grid>

Наш пользовательский элемент управления выглядит следующим образом:



После элемента TextBlock добавим элемент для отображения ошибок:

<TextBlock Text="{Binding ErrorMessage}"

Foreground="Red"

IsVisible="{Binding ErrorMessage, Converter={x:Static ObjectConverters.IsNotNull}}"

TextWrapping="Wrap"/>

Визуально наша форма не изменится

Ниже мы расположим таблицу для отображения данных о партнере:

<Grid ColumnDefinitions="150,\*" RowDefinitions="Auto,Auto,Auto,Auto,Auto,Auto,Auto,Auto,Auto">

</Grid>

За таблицей мы расположим кнопки для сохранения и отмены действия:

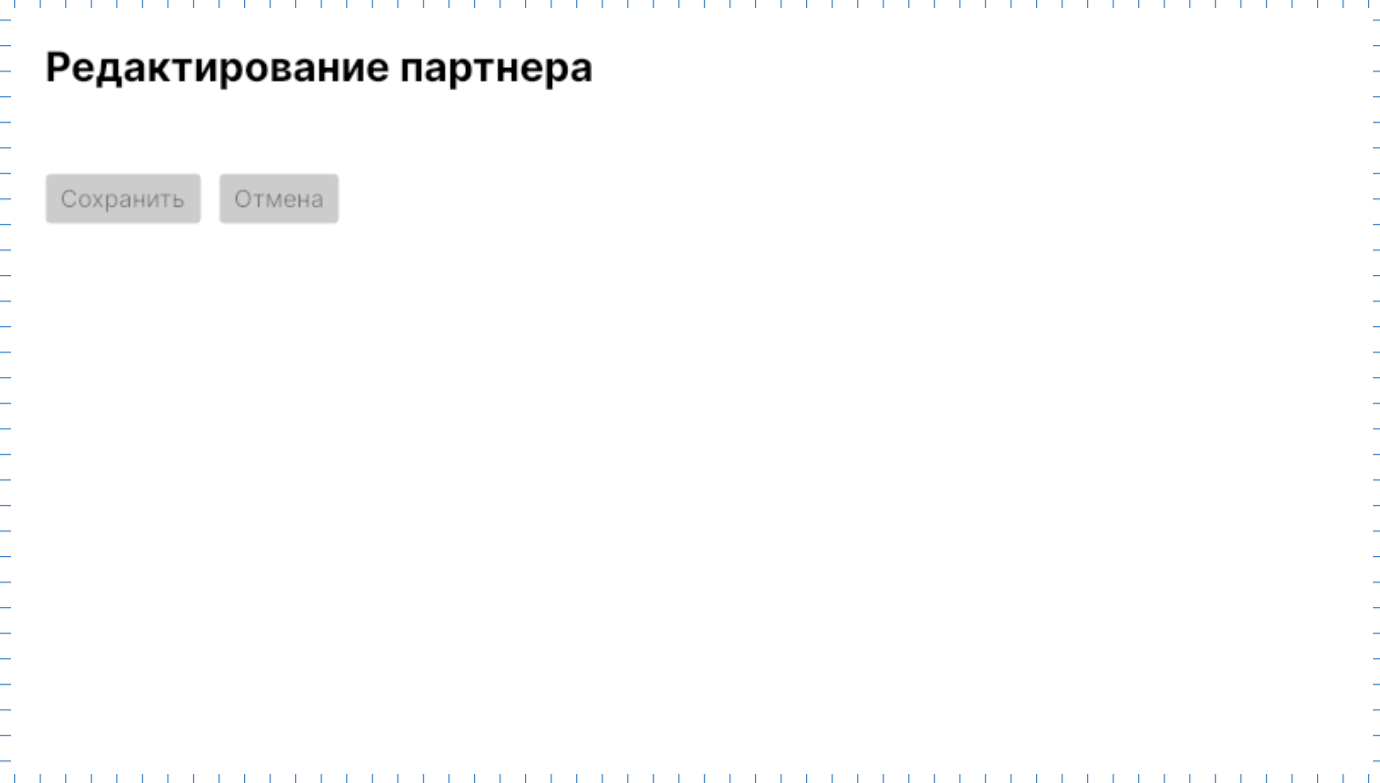
<StackPanel Orientation="Horizontal" Spacing="10">

<Button Content="Сохранить" Command="{Binding SaveCommand}" />

<Button Content="Отмена" Command="{Binding CancelCommand}" />

</StackPanel>

Наш пользовательский элемент управления выглядит следующим образом:



Добавим поля в таблицу для отображения информации о партнере:

<TextBlock Grid.Row="0" Grid.Column="0" Text="Название:" />

<TextBox Grid.Row="0" Grid.Column="1" Text="{Binding PartnerName}" />

<TextBlock Grid.Row="1" Grid.Column="0" Text="ИНН:" />

<TextBox Grid.Row="1" Grid.Column="1" Text="{Binding PartnerInn}" />

<TextBlock Grid.Row="2" Grid.Column="0" Text="Директор:" />

<TextBox Grid.Row="2" Grid.Column="1" Text="{Binding Director}" />

<TextBlock Grid.Row="3" Grid.Column="0" Text="Телефон:" />

<MaskedTextBox Grid.Row="3" Grid.Column="1" Text="{Binding DirectorPhone}"

Mask="+7 (000) 000 0000" />

<TextBlock Grid.Row="4" Grid.Column="0" Text="Email:" />

<TextBox Grid.Row="4" Grid.Column="1" Text="{Binding DirectorMail}"

Watermark="example@domain.com" />

<TextBlock Grid.Row="5" Grid.Column="0" Text="Рейтинг:" />

<NumericUpDown Grid.Row="5" Grid.Column="1" Value="{Binding PartnerRating}"

Minimum="0" Maximum="100" />

<TextBlock Grid.Row="6" Grid.Column="0" Text="Тип партнера:" />

<ComboBox Grid.Row="6" Grid.Column="1"

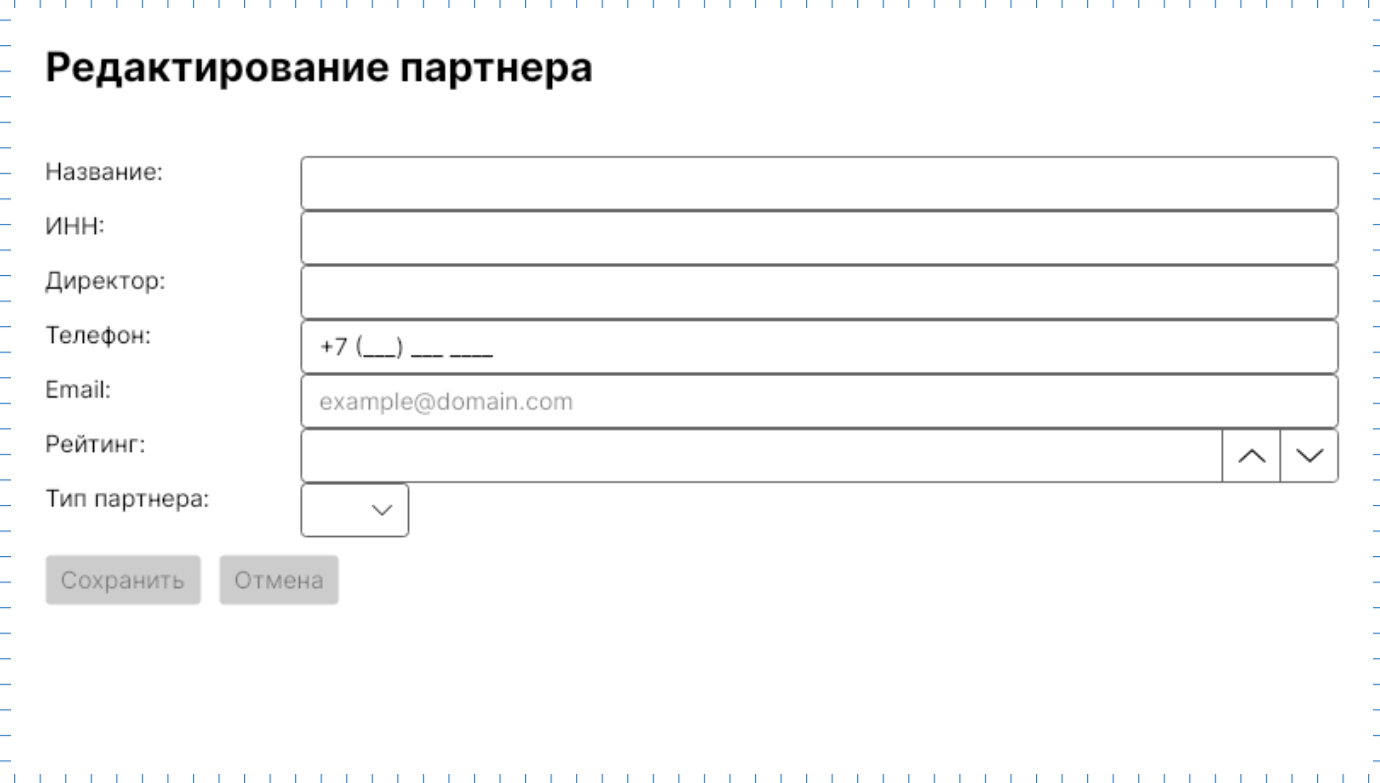
ItemsSource="{Binding PartnerTypes}"

SelectedValue="{Binding SelectedPartnerType}"

SelectedValueBinding="{Binding PartnerTypeId}"

DisplayMemberBinding="{Binding PartnerTypeName}" />

Наш пользовательский элемент управления выглядит следующим образом:



Необходимо добавить переход с пользовательского элемента управления для вывода списка партнеров. Для этого откроем файл PartnersViewModel.cs.

Изменим метод для добавления партнера

[RelayCommand]

private void AddPartner()

{

var viewModel = new EditPartnerViewModel(\_context, parentViewModel: this);

var editControl = new EditPartnerView { DataContext = viewModel };

\_mainViewModel.ShowEditPartner(editControl);

}

Необходимо добавить метод в класс MainWindowViewModel.cs

public void ShowEditPartner(EditPartnerView editControl)

{

Content = editControl;

}

Изменим метод для изменения партнера

[RelayCommand]

private void EditPartner()

{

if (SelectedPartner == null) return;

var partnerEntity = \_context.Partners.Find(SelectedPartner.\_partner.PartnerId);

if (partnerEntity == null) return;

var viewModel = new EditPartnerViewModel(\_context, partnerEntity, this);

var editControl = new EditPartnerView { DataContext = viewModel };

\_mainViewModel.ShowEditPartner(editControl);

}

Реализуем работу кнопки отмена, для этого отредактируем метод Cancel класса EditPartnerViewModel.cs:

[RelayCommand]

private void Cancel()

{

\_parentViewModel?.\_mainViewModel.ShowPartnersList();

}

Добавим проверки для формы. Для этого создадим каталог Helpers и в него добавим класс ValidationHelper.cs со следующим содержимым:

public class ValidationHelper

{

public static bool IsValidInn(string? inn)

{

if (string.IsNullOrWhiteSpace(inn)) return false;

return Regex.IsMatch(inn, @"^\d{10}$|^\d{12}$");

}

public static bool IsValidEmail(string? email)

{

if (string.IsNullOrWhiteSpace(email)) return false;

return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$");

}

public static bool IsValidPhoneNumber(string? phone)

{

if (string.IsNullOrWhiteSpace(phone)) return false;

return Regex.IsMatch(phone, @"^\+?[78]\s?\(?\d{3}\)?\s?\d{3}[-\s]?\d{2}[-\s]?\d{2}$");

}

public static bool IsValidRating(int? rating)

{

return rating is >= 0 and <= 100;

}

}

Добавим вызов наших проверок в классе

private bool ValidateData()

{

if (string.IsNullOrWhiteSpace(PartnerName))

{

ErrorMessage = "Название партнера обязательно для заполнения";

return false;

}

if (!ValidationHelper.IsValidInn(PartnerInn))

{

ErrorMessage = "ИНН должен содержать 10 или 12 цифр";

return false;

}

if (string.IsNullOrWhiteSpace(Director))

{

ErrorMessage = "Имя директора обязательно для заполнения";

return false;

}

if (!ValidationHelper.IsValidPhoneNumber(DirectorPhone))

{

ErrorMessage = "Неверный формат номера телефона. Пример: +7(999)999-99-99";

return false;

}

if (!ValidationHelper.IsValidEmail(DirectorMail))

{

ErrorMessage = "Неверный формат электронной почты";

return false;

}

if (!ValidationHelper.IsValidRating(PartnerRating))

{

ErrorMessage = "Рейтинг должен быть от 0 до 100";

return false;

}

if (SelectedPartnerType == null)

{

ErrorMessage = "Выберите тип партнера";

return false;

}

ErrorMessage = null;

return true;

}

Отредактируем метод Save класса EditPartnerViewModel.cs

[RelayCommand]

private void Save()

{

if (!ValidateData()) return;

\_partnerEntity.PartnerName = PartnerName;

\_partnerEntity.PartnerInn = PartnerInn;

\_partnerEntity.Director = Director;

\_partnerEntity.DirectorPhone = DirectorPhone;

\_partnerEntity.DirectorMail = DirectorMail;

\_partnerEntity.PartnerRating = PartnerRating;

\_partnerEntity.PartnerType = SelectedPartnerType;

if (\_partnerEntity.PartnerId == 0)

{

\_context.Partners.Add(\_partnerEntity);

}

\_context.SaveChanges();

\_parentViewModel?.LoadPartners();

Cancel();

}

Установим библиотеку

dotnet add package Avalonia.Xaml.Behaviors --version 11.2.0.14

Далее нам необходимо исправить изменение. Для этого в файле PartnersView.axaml добавим следующие строки в ListBox

<Interaction.Behaviors>

<EventTriggerBehavior EventName="SelectionChanged">

<InvokeCommandAction Command="{Binding EditPartnerCommand}" CommandParameter="{Binding SelectedPartner}"/>

</EventTriggerBehavior>

</Interaction.Behaviors>

Установим библиотеку

dotnet add package MessageBox.Avalonia --version 3.2.0

Изменим метод Save класса EditPartnerViewModel.cs

[RelayCommand]

private async Task SaveAsync()

{

if (!ValidateData()) {

var box = MessageBoxManager

.GetMessageBoxStandard("Error", ErrorMessage,

ButtonEnum.Ok);

var result = await box.ShowAsync();

return;}

\_partnerEntity.PartnerName = PartnerName;

\_partnerEntity.PartnerInn = PartnerInn;

\_partnerEntity.Director = Director;

\_partnerEntity.DirectorPhone = DirectorPhone;

\_partnerEntity.DirectorMail = DirectorMail;

\_partnerEntity.PartnerRating = PartnerRating;

\_partnerEntity.PartnerType = SelectedPartnerType;

if (\_partnerEntity.PartnerId == 0)

{

\_context.Partners.Add(\_partnerEntity);

}

\_context.SaveChanges();

\_parentViewModel?.LoadPartners();

Cancel();

}