**ДИСКЛЕЙМЕР**

Я не отвечаю за работоспособность этого кода,

так как я его украла и адаптировала под свой проект

но у меня все работает

А еще тут есть объяснения каждого действия, комментарии к коду и рисунки

похвала и картинки мемных котят

и других смешных животных



В отдельном файле есть решение первого модуля.

Его тут нет, тк первый модуль супер изи, но на всякий случай он все равно существует

Методичка, кстати, стоит 50.000 рублей

Да, за не оригинальный код, и что

Зато все работает и расписано, ага

**МОДУЛЬ 2 Создание проекта**

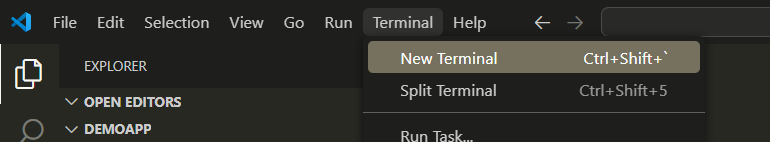
**Установить пакеты для работы**

* **Установить шаблоны Avalonia**

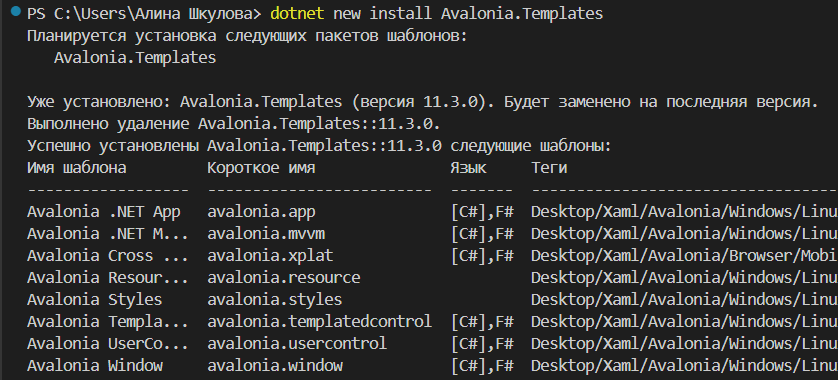
Ввести код в терминале программы

dotnet new install Avalonia.Templates

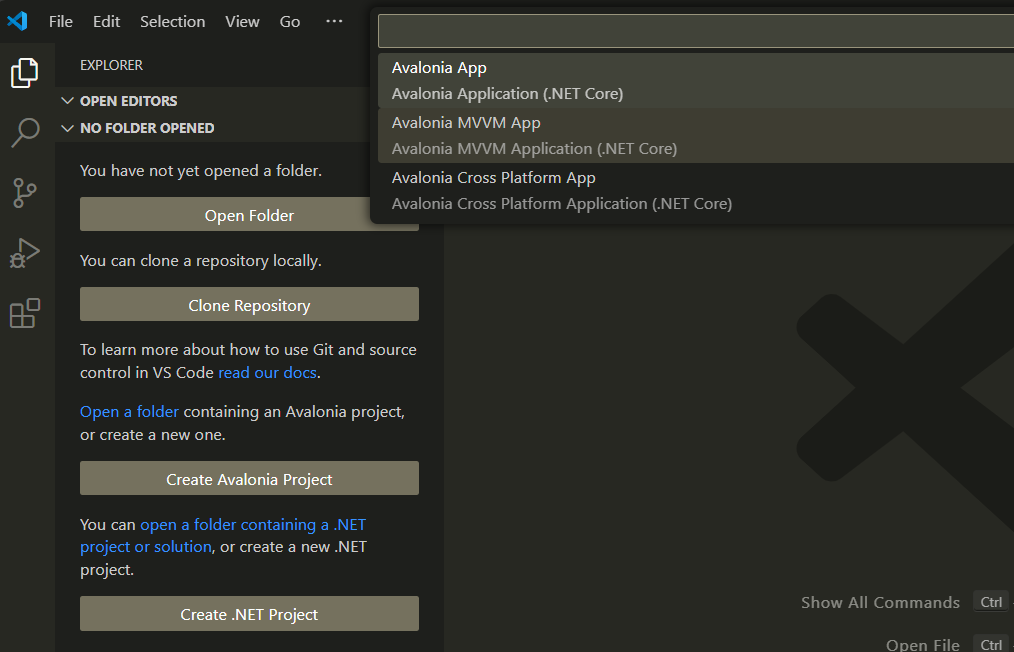
Терминал, кстати, здесь



Устанавливаются шаблоны Авалония для создания приложений



* **Создать проект Avalonia MVVM App**



* **Добавить в проект пакеты через терминал**

dotnet add package Microsoft.EntityFrameworkCore.Tools --version 9.0.3

dotnet add package Microsoft.EntityFrameworkCore.Design --version 9.0.3

dotnet add package FirebirdSql.EntityFrameworkCore.Firebird --version 12.0.0

dotnet add package Avalonia.ReactiveUI --version 11.2.6

dotnet new tool-manifest

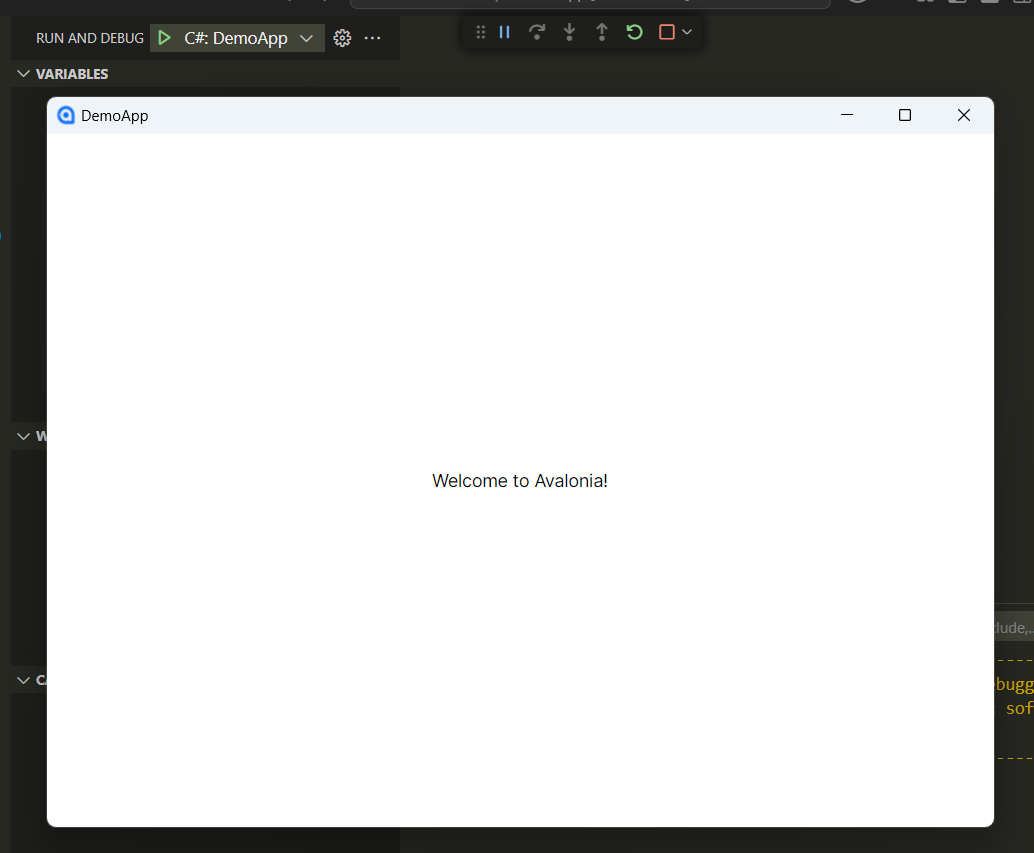
dotnet tool install dotnet-ef --version 9.0.3

dotnet add package CommunityToolkit.Mvvm

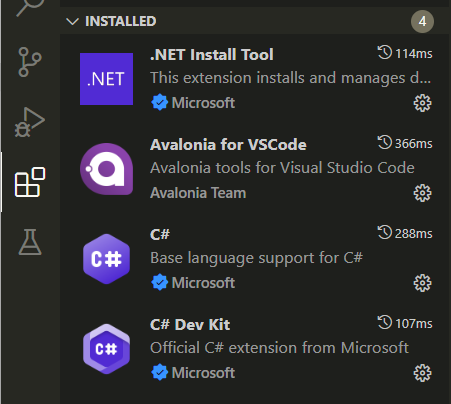
* **Запуск проекта**

Запустить проект на панели **RUN**, **используя C#**

Выбрать созданный проект



Перед началом кодинга не забудь про необходимые скаченные расширения для VS Code



**Подключение в созданной ранее БД**

В терминал проекта ввести

dotnet tool restore

dotnet tool run dotnet-ef dbcontext scaffold "DataSource=localhost;Port=3050;Database=

C:/db/basa.fdb;Username=sysdba;Password=123456 " FirebirdSql.EntityFrameworkCore.Firebird -o Entities -f

**Создание элемента управления. Настройка вида**

**Первый шаг:** создать файл **UserControl** с названием **PartnersView** используя шаблоны Avalonia.

Если в контекстном меню VS Code (или через Правой кнопкой → Добавить) нет пункта New Avalonia Template → UserControl, можно добавить файл через консоль:

Выбрать папку Views, открыть консоль и ввести

dotnet new avalonia.usercontrol -n PartnersView

имя файла

В этом файле содержится инфа о внешнем виде окна, вплоть до каждой кнопки.

Теперь в созданных файлах, нужно заменить код. Ok, let’s go…

***PartnersView.axaml***

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

d:DesignWidth="800"

d:DesignHeight="450"

x:Class="DemoApp.Views.PartnersView"

xmlns:vm="using:DemoApp.ViewModels"

x:DataType="vm:PartnersViewModel">

<Grid Margin="20">

<StackPanel Spacing="10" Classes="main">

<TextBlock Text="Список партнеров" FontSize="24" FontWeight="Bold" Classes="h1" />

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

<Button Content="Добавить партнера" Command="{Binding AddPartnerCommand}" />

<Button Content="Редактировать" Command="{Binding EditPartnerCommand}" />

</StackPanel>

<ListBox ItemsSource="{Binding Partners}"

SelectedItem="{Binding SelectedPartner}"

MaxHeight="800">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Margin="5" HorizontalAlignment="Left">

<StackPanel Orientation="Vertical" Margin="5" HorizontalAlignment="Left">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Left">

<TextBlock Text="{Binding PartnerTypeDescription}" TextWrapping="Wrap" />

<TextBlock Text=" | " TextWrapping="Wrap" />

<TextBlock Text="{Binding \_partner.PartnerName}" TextWrapping="Wrap" Width="250" />

</StackPanel>

<TextBlock Text="{Binding \_partner.Director}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

<TextBlock Text="{Binding \_partner.DirectorPhone}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

<TextBlock Text="{Binding RatingDisplay}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

</StackPanel>

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

<TextBlock Text="{Binding DiscountDisplay}" TextWrapping="Wrap" Width="150" />

<Button Content="История продаж"

Command="{Binding $parent[ListBox].((vm:PartnersViewModel)DataContext).ViewSalesHistoryCommand}"

CommandParameter="{Binding}" />

</StackPanel>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</StackPanel>

</Grid>

</UserControl>

***PartnersView.axaml.cs:***

using Avalonia.Controls;

using Avalonia.Markup.Xaml;

*namespace* DemoApp.Views

{

// Класс представления (View) для отображения списка партнеров

// Наследуется от UserControl — базового класса для пользовательских элементов управления в Avalonia

    public partial *class* PartnersView : UserControl

    {

        public PartnersView()

        {

// Инициализация компонентов, описанных в XAML (PartnersView.axaml)

            InitializeComponent();

        }

// Метод инициализации, загружает XAML-разметку для данного UserControl

        private void InitializeComponent()

        {

            AvaloniaXamlLoader.Load(this);

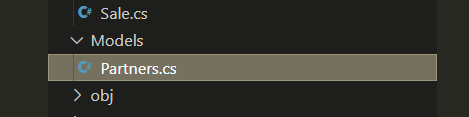
        }

    }

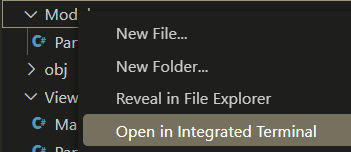
}

**Второй шаг**: Создание класса Партнеры в папке Модели.

Нам необходимо создать модель партнера в каталоге Models. В каталоге Models необходимо добавить новый Class, при помощи контекстного меню с именем Partner



Для этого выбрать папку **Models** и выбрать **создание нового класса #C,** но если такой графы нет, класс можно создать при помощи терминала:



dotnet new class -n Product

имя класса

В созданном файле вставить этот код:

***Models – Partner.cs***

using System.Linq;

using System.Collections.Generic;

*namespace* DemoApp.Models

{

// Partner наследуется от Entities.Partner и расширяет его функциональность

    public *class* Partner : Entities.Partner

    {

// Берет значения из Partner Entities

        public Entities.Partner \_partner { *get*; *set*; }

// Конструктор, принимающий объект Entities.Partner и сохраняющий его во внутреннем поле

        public Partner(Entities.Partner *partner*)

        {

            \_partner = *partner*;

        }

// Свойство, возвращающее описание типа партнера, используя навигационное свойство PartnerTypeNavigation

        public string? PartnerTypeDescription => \_partner.PartnerTypeNavigation?.PartnerTypeName;

// Свойство, формирующее строку с контактной информацией директора (телефон и почта)

        public string ContactInfo => $"{\_partner.DirectorPhone} | {\_partner.DirectorMail}";

        public string RatingDisplay => $"Рейтинг: {\_partner.PartnerRating ?? 0}";

// Свойство, отображающее рейтинг партнера, если рейтинг отсутствует, используется 0

        public int TotalProductsSold => \_partner.Sales?.Sum(*s* => *s*.ProductCount) ?? 0;

// Свойство, вычисляющее скидку партнера на основе общего количества проданных продуктов

        public decimal Discount => CalculatePartnerDiscount(TotalProductsSold);

        public string DiscountDisplay => $"Скидка: {Discount:P0}";

// Расчет скидки с зависимости от количества проданных продуктов

        public static decimal CalculatePartnerDiscount(int *totalProductsSold*)

        {

            return *totalProductsSold* switch

            {

                < 10000 => 0,

                < 50000 => 0.05m,

                < 300000 => 0.10m,

                \_ => 0.15m

            };

        }

    }

}

**Третий шаг**: создать класс C# **PartnersViewModel** в папке **ViewModels**

если ОПЯТЬ нет контекстного меню ввести код в консоли:

dotnet new class -n PartnersViewModel

Имя класса

В созданном файле ввести код:

***PartnersViewModel.cs***

using System.Collections.ObjectModel;

using System.Linq;

using CommunityToolkit.Mvvm.ComponentModel;

using CommunityToolkit.Mvvm.Input;

using DemoApp.Entities;

using Microsoft.EntityFrameworkCore;

*namespace* DemoApp.ViewModels

{

    public partial *class* PartnersViewModel : ObservableObject

    {

        private readonly DbDemoFdbContext \_context;

// Ссылка на главную ViewModel, чтобы можно было взаимодействовать с главным окном (например, менять отображаемый контент)

        internal readonly MainWindowViewModel \_mainViewModel;

// Коллекция партнеров, которая будет отображаться в UI

        [ObservableProperty]

        private ObservableCollection<DemoApp.Models.Partner> \_partners = new();

        [ObservableProperty]

        private DemoApp.Models.Partner? \_selectedPartner;

// Конструктор, принимает контекст базы данных и ссылку на главную ViewModel

// Загружает список партнеров при создании

        public PartnersViewModel(DbDemoFdbContext *context*, MainWindowViewModel *mainViewModel*)

        {

            \_context = *context*;

            \_mainViewModel = *mainViewModel*;

            LoadPartners();

        }

// Загружаем данные

        public void LoadPartners()

        {

*var* entities = \_context.Partners

                .Include(*p* => *p*.Sales)

                .Include(*p* => *p*.PartnerTypeNavigation)

                .ToList();

            Partners = new ObservableCollection<DemoApp.Models.Partner>(

                entities.Select(*e* => new DemoApp.Models.Partner(*e*))

            );

        }

// Команда для редактирования выбранного партнера

        [RelayCommand]

        private void EditPartner()

        {

            // изменение партнера

        }

        [RelayCommand]

        private void AddPartner()

        {

            // добавление партнера

        }

        [RelayCommand]

        private void ViewSalesHistory(DemoApp.Models.Partner? *partner*)

        {

            // история продаж

        }

    }

}

**Четвёртый шаг**: изменить код в существующем файле **MainWindowViewModel**

***MainWindowViewModel.cs***

using Avalonia.Controls;

using CommunityToolkit.Mvvm.ComponentModel;

using DemoApp.Entities;

using DemoApp.Views;

using Microsoft.EntityFrameworkCore;

*namespace* DemoApp.ViewModels

{

// Основная ViewModel для главного окна приложения, наследуется от базового класса ViewModelBase

    public partial *class* MainWindowViewModel : ViewModelBase

    {

        private readonly DbDemoFdbContext \_context;

        [ObservableProperty]

        private Control? content;

// Метод для установки текущего отображаемого контента в главное окно

        public void ShowContent(Control *content*) => Content = *content*;

// Метод для отображения списка партнеров

// Создаёт новый экземпляр PartnersView и устанавливает его DataContext в новую PartnersViewModel, передавая в неё контекст базы данных и ссылку на текущую MainWindowViewModel

        public void ShowPartnersList()

        {

            Content = new PartnersView

            {

                DataContext = new PartnersViewModel(\_context, this)

            };

        }

//Инициация контента

        public MainWindowViewModel()

        {

            \_context = new DbDemoFdbContext();

            ShowPartnersList();

        }

    }

}

**Пятый шаг**: изменить код файлов **MainWindow** в папке **Views.**

***MainWindow.axaml***

<Window xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:vm="using:DemoApp.ViewModels"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="DemoApp.Views.MainWindow"

x:DataType="vm:MainWindowViewModel"

Icon="/Assets/avalonia-logo.ico"

<!—Тут указать путь к месту иконки -->

Title="Мастер Пол">

<!—Тут написать название программы-->

<Design.DataContext>

<vm:MainWindowViewModel/>

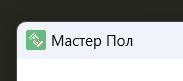
</Design.DataContext>

<ContentControl Content="{Binding Content}"/>

</Window>

Тут можно поменять иконку и название, кстати.

Просто по заданному пути добавить пичку и указать ее название в коде.



***MainWindow.axaml.cs***

using Avalonia.Controls;

namespace DemoApp.Views;

public partial class MainWindow : Window

{

public MainWindow()

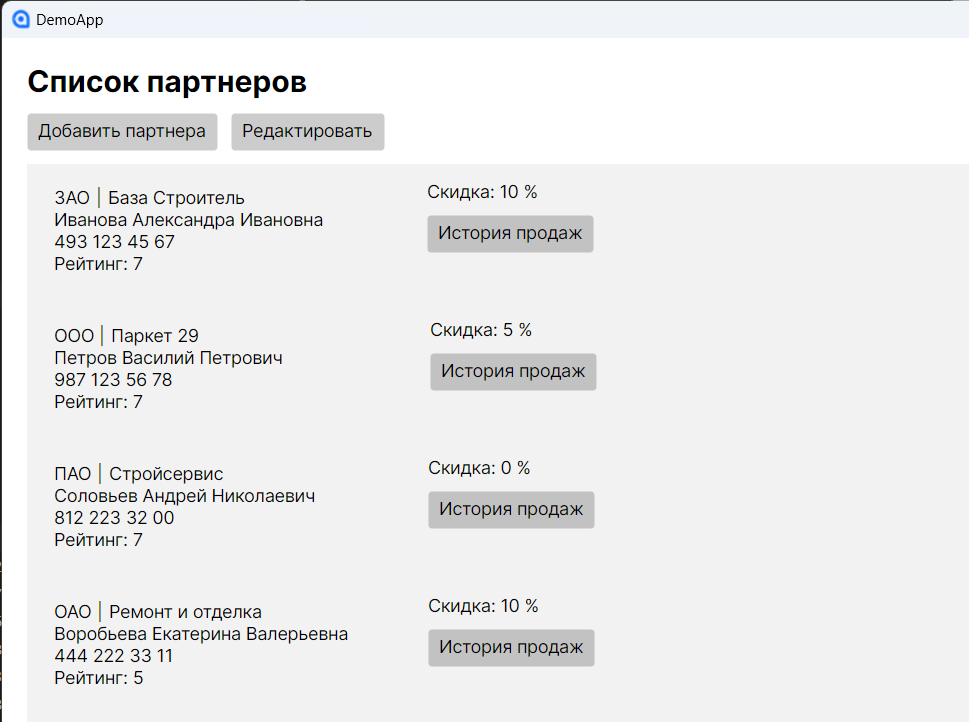
{

InitializeComponent();

}

}

На этом можно закончить. Запускаем программу!



Омагад, все получилось, ты молодец

**На этом можно закончить выполнение второго модуля…**

**И приступить к третьему!!**



**Примерное задание**

**Разработать интерфейс программного модуля для работы с партнерами.**

Реализовать последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами (страницами) в приложении (в том числе обратно, например, с помощью кнопки «*Назад*»). Обеспечить соответствующий заголовок на каждом окне (странице) приложения.

Реализовать обработку исключительных ситуаций в приложении. Необходимо уведомлять пользователя о совершаемых им ошибках или о запрещенных в рамках задания действиях, предупреждать о неотвратимых операциях. Окна сообщений соответствующих типов (например, ошибка, предупреждение, информация) должны отображаться с соответствующим заголовком и пиктограммой. Текст сообщения должен быть полезным и информативным, содержать полную информацию о совершенных ошибках пользователя и порядок действий для их исправления. Также можно использовать визуальные подсказки для пользователя при вводе данных.

Необходимо использовать комментарии для пояснения неочевидных фрагментов кода. Комментарии должны присутствовать только в местах, которые требуют дополнительного пояснения.

Реализовать функции добавления/редактирования данных партнера в новом окне (странице) – форме для добавления/редактирования партнера. Переходы на эту форму должны быть реализованы из главной формы списка партнеров: для редактирования – при нажатии на конкретный элемент, для добавления – при нажатии кнопки.

На форме для добавления/редактирования партнера должны быть предусмотрены следующие поля: наименование, тип партнера (выпадающий список), рейтинг, адрес, ФИО директора, телефон и email компании.

Рейтинг партнера должен быть целыми неотрицательным числом.

При открытии формы для редактирования все поля выбранного объекта должны быть подгружены в соответствующие поля из базы данных, а таблица заполнена актуальными значениями.

После добавления/редактирования партнера данные в окне списка партнеров должны быть обновлены.

**МОДУЛЬ 3**

**Создание формы изменения партнера**

Добавить в проект пакеты:

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

dotnet add package MessageBox.Avalonia --version 3.2.0

Для создания внешнего вида окна изменения партнера нужно:

**Первое**: каталоге **Views** добавить новый **UserControl** с именем **EditPartnerView.** В этих файлах будет содержаться вся инфа про окно изменения партнера.

dotnet new avalonia.usercontrol -n EditPartnerView

В созданных файлах вставить код:

***EditPartnerView.axaml***

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="DemoApp.Views.EditPartnerView"

xmlns:vm="using:DemoApp.ViewModels"

x:DataType="vm:EditPartnerViewModel">

<Grid Margin="20">

<StackPanel Spacing="10">

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

<TextBlock Text="{Binding ErrorMessage}"

Foreground="Red"

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

TextWrapping="Wrap"/>

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

</Grid>

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

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

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

</StackPanel>

<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}" />

</StackPanel>

</Grid>

</UserControl>

***EditPartnerView.axaml.cs:***

using Avalonia;

using Avalonia.Controls;

using Avalonia.Markup.Xaml;

namespace DemoApp.Views;

public partial class EditPartnerView : UserControl

{

public EditPartnerView()

{

InitializeComponent();

}

}

**Второе**: добавим проверки для формы. Для этого создадим каталог **Helpers** и в него добавим класс **ValidationHelper.** В нем описываютсяформаты введённых данных для того, чтобы не написать лишнего (обработка данных и вывод ошибок).

dotnet new class -n ValidationHelper

ValidationHelper:



В созданном файле изменить код:

***ValidationHelper.cs:***

Хочет написать эл.почту в поле телефона

using System.Text.RegularExpressions;

public class ValidationHelper

{

// Проверка ввода ИНН

public static bool IsValidInn(string? inn)

{

// Если строка null или пуста, возвращаем false

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

// Проверяем соответствие строки шаблону ИНН (10 или 12 цифр) с использованием регулярного выражения

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

}

// Проверка ввода эл. почты

public static bool IsValidEmail(string? email)

{

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

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

}

// Проверка ввода номера телефона

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}$");

}

// Ввод рейтинга от 0 до 100

public static bool IsValidRating(int? rating)

{

return rating is >= 0 and <= 100;

}

}

Создать класс **EditPartnerViewModel** в папке **ViewModel**

dotnet new class -n EditPartnerViewModel

***EditPartnerViewModel.cs***

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using CommunityToolkit.Mvvm.ComponentModel;

using CommunityToolkit.Mvvm.Input;

using DemoApp.Entities;

using MsBox.Avalonia;

using MsBox.Avalonia.Enums;

namespace DemoApp.ViewModels

{

public partial class EditPartnerViewModel : ViewModelBase

{

private readonly DbDemoFdbContext \_context;

private readonly PartnersViewModel? \_parentViewModel;

private readonly Partner \_partnerEntity;

// Это ключевой атрибут из MVVM Toolkit. Когда он применяется к приватному полю (backing field, в данном случае \_partnerName), он автоматически генерирует: Публичное свойство с тем же именем, что и поле, но с первой буквой в верхнем регистре (PartnerName).

[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;

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

{

\_context = context;

\_parentViewModel = parentViewModel;

\_partnerEntity = partner ?? new Partner();

// Если partner == null, то создается новый экземпляр Partner.

LoadPartnerTypes();

LoadPartnerData();

}

private void LoadPartnerTypes()

{

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

PartnerTypes = new ObservableCollection<PartnerType>(types);

// Создание ObservableCollection для привязки к UI.

}

// Загружает данные партнера (из \_partnerEntity) в свойства ViewModel.

private void LoadPartnerData()

{

PartnerName = \_partnerEntity.PartnerName;

PartnerInn = \_partnerEntity.PartnerInn;

Director = \_partnerEntity.Director;

DirectorPhone = \_partnerEntity.DirectorPhone;

DirectorMail = \_partnerEntity.DirectorMail;

PartnerRating = \_partnerEntity.PartnerRating;

SelectedPartnerType = \_partnerEntity.PartnerType;

}

// Вывод сообщений об ошибке ввода данных

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;

}

// Атрибут CommunityToolkit.Mvvm для создания команды, привязанной к кнопке Save в UI.

[RelayCommand]

private async Task SaveAsync()

{

if (!ValidateData()) {

var box = MessageBoxManager

.GetMessageBoxStandard("Error", ErrorMessage,

ButtonEnum.Ok); // Создание стандартного MessageBox.

var result = await box.ShowAsync();

return;}

// Обновление свойств сущности партнера данными из ViewModel.

\_partnerEntity.PartnerName = PartnerName;

\_partnerEntity.PartnerInn = PartnerInn;

\_partnerEntity.Director = Director;

\_partnerEntity.DirectorPhone = DirectorPhone;

\_partnerEntity.DirectorMail = DirectorMail;

\_partnerEntity.PartnerRating = PartnerRating;

\_partnerEntity.PartnerType = SelectedPartnerType;

// Определяем, создаем новую запись или обновляем существующую.

if (\_partnerEntity.PartnerId == 0)

{

\_context.Partners.Add(\_partnerEntity); // новая запись

}

else

{

\_context.Partners.Update(\_partnerEntity); // обновление существующей

}

\_context.SaveChanges();

\_parentViewModel?.LoadPartners();

Cancel();

}

[RelayCommand]

private void Cancel()

{

\_parentViewModel?.\_mainViewModel.ShowPartnersList();

}

}

}

**Далее обновить код в файлах, ранее созданных**

***PartnersView.axaml***

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="DemoApp.Views.PartnersView"

xmlns:vm="using:DemoApp.ViewModels"

x:DataType="vm:PartnersViewModel">

<Grid Margin="20">

<StackPanel Spacing="10">

<TextBlock Text="Список партнеров" FontSize="24" FontWeight="Bold" />

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

<Button Content="Добавить партнера" Command="{Binding AddPartnerCommand}" />

<Button Content="Редактировать" Command="{Binding EditPartnerCommand}" />

</StackPanel>

<ListBox ItemsSource="{Binding Partners}" SelectedItem="{Binding SelectedPartner}"

MaxHeight="800">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Margin="5" HorizontalAlignment="Left">

<StackPanel Orientation="Vertical" Margin="5" HorizontalAlignment="Left">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Left">

<TextBlock Text="{Binding PartnerTypeDescription}" TextWrapping="Wrap" />

<TextBlock Text=" | " TextWrapping="Wrap" />

<TextBlock Text="{Binding \_partner.PartnerName}" TextWrapping="Wrap" Width="250" />

</StackPanel>

<TextBlock Text="{Binding \_partner.Director}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

<TextBlock Text="{Binding \_partner.DirectorPhone}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

<TextBlock Text="{Binding RatingDisplay}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

</StackPanel>

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

<TextBlock Text="{Binding DiscountDisplay}" TextWrapping="Wrap" Width="150" />

<Button Content="История продаж" />

</StackPanel>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

<Interaction.Behaviors>

<EventTriggerBehavior EventName="SelectionChanged">

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

</EventTriggerBehavior>

</Interaction.Behaviors>

</ListBox>

</StackPanel>

</Grid>

</UserControl>

***PartnersView.axaml.cs***

using Avalonia;

using Avalonia.Controls;

using Avalonia.Markup.Xaml;

namespace DemoApp.Views;

public partial class PartnersView : UserControl

{

public PartnersView()

{

InitializeComponent();

}

}

***PartnersViewModel.cs:***

using System.Collections.ObjectModel;

using System.Linq;

using CommunityToolkit.Mvvm.ComponentModel;

using CommunityToolkit.Mvvm.Input;

using DemoApp.Entities;

using DemoApp.Views;

using Microsoft.EntityFrameworkCore;

namespace DemoApp.ViewModels

{

public partial class PartnersViewModel : ObservableObject

{

private readonly DbDemoFdbContext \_context;

internal readonly MainWindowViewModel \_mainViewModel;

[ObservableProperty]

private ObservableCollection<Models.Partner> \_partners = new();

[ObservableProperty]

private Models.Partner? \_selectedPartner;

public PartnersViewModel(DbDemoFdbContext context, MainWindowViewModel mainViewModel)

{

\_context = context;

\_mainViewModel = mainViewModel;

LoadPartners();

}

public void LoadPartners()

{

var entities = \_context.Partners.Include(p => p.Sales).Include(p => p.PartnerTypeNavigation).ToList();

Partners = new ObservableCollection<Models.Partner>(

entities.Select(e => new Models.Partner(e))

);

}

[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);

}

[RelayCommand]

private void AddPartner()

{

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

var editControl = new EditPartnerView { DataContext = viewModel };

\_mainViewModel.ShowEditPartner(editControl);

}

[RelayCommand]

private void ViewSalesHistory(Partner? partner)

{

//история продаж

}

}

}

***MainWindowViewModel.cs:***

using Avalonia.Controls;

using CommunityToolkit.Mvvm.ComponentModel;

using DemoApp.Entities;

using DemoApp.Views;

using Microsoft.EntityFrameworkCore;

namespace DemoApp.ViewModels;

public partial class MainWindowViewModel : ViewModelBase

{

[ObservableProperty]

private Control? \_content;

private readonly DbDemoFdbContext \_context;

public void ShowContent(Control content)

{

Content = content;

}

public void ShowPartnersList()

{

Content = new PartnersView

{

DataContext = new PartnersViewModel(\_context, this)

};

}

public MainWindowViewModel()

{

\_context = new DbDemoFdbContext();

ShowPartnersList();

}

public void ShowEditPartner(EditPartnerView editControl)

{

Content = editControl;

}

}

***Partner.cs*** *в каталоге Entities*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DemoApp.Entities;

public partial class Partner

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int? PartnerId { get; set; }

public string? PartnerName { get; set; }

public string? Director { get; set; }

public string? DirectorMail { get; set; }

public string? DirectorPhone { get; set; }

public string? PartnerLegalAddress { get; set; }

public string? PartnerInn { get; set; }

public int? PartnerRating { get; set; }

public int? PartnerType { get; set; }

public virtual PartnerType? PartnerTypeNavigation { get; set; }

public virtual ICollection<Sale> Sales { get; set; } = new List<Sale>();

}

***Partner.cs***

using System.Linq;

namespace DemoApp.Models

{

public class Partner : Entities.Partner

{

public Entities.Partner \_partner { get; set; }

public Partner(Entities.Partner partner)

{

\_partner = partner;

}

public string? PartnerTypeDescription => \_partner.PartnerTypeNavigation?.PartnerTypeName;

public string ContactInfo => $"{\_partner.DirectorPhone} | {\_partner.DirectorMail}";

public string RatingDisplay => $"Рейтинг: {\_partner.PartnerRating ?? 0}";

public int TotalProductsSold => \_partner.Sales?.Sum(s => s.ProductCount) ?? 0;

public decimal Discount => CalculatePartnerDiscount(TotalProductsSold);

public string DiscountDisplay => $"Скидка: {Discount:P0}";

public static decimal CalculatePartnerDiscount(int totalProductsSold)

{

return totalProductsSold switch

{

< 10000 => 0,

< 50000 => 0.05m,

< 300000 => 0.10m,

\_ => 0.15m

};

}

}

}

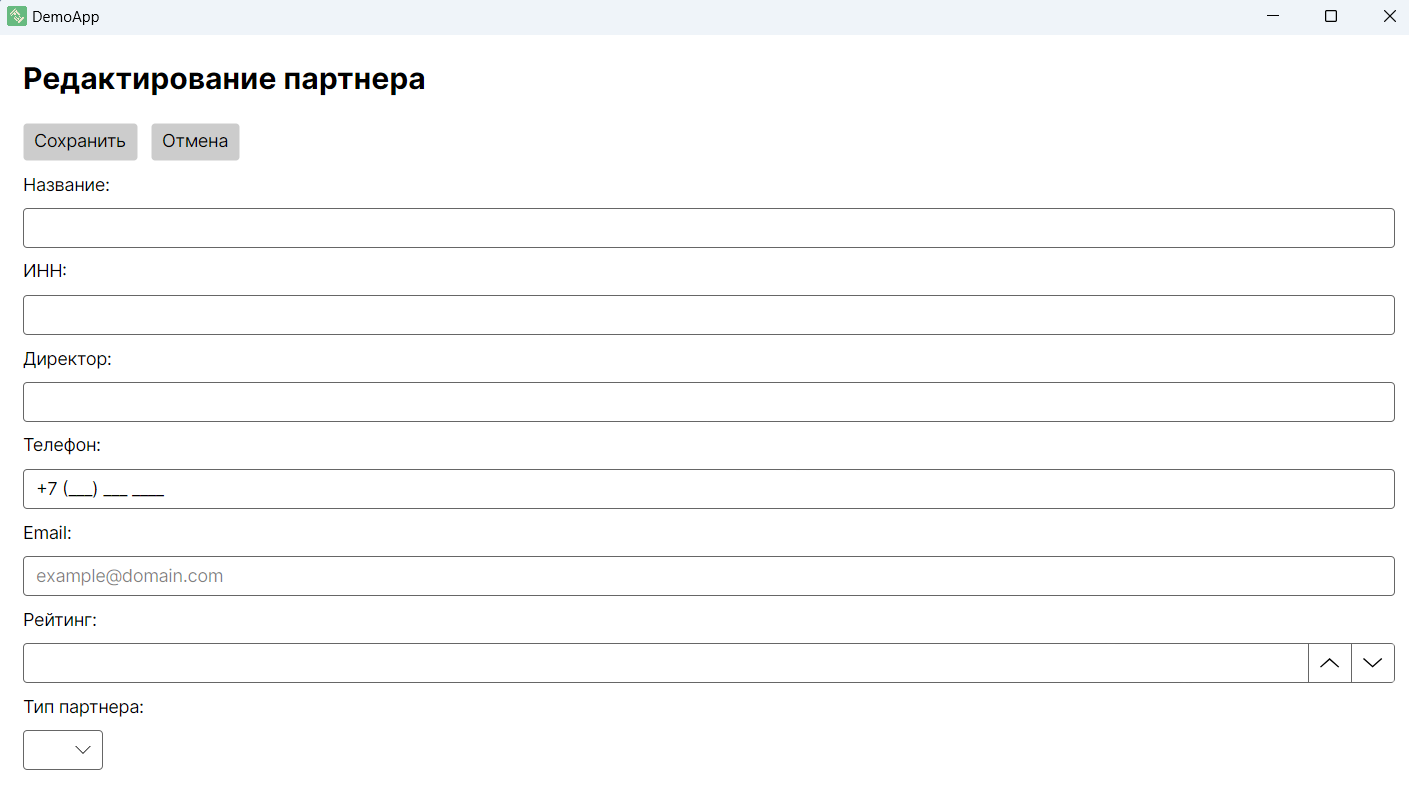
На этом все. Запуск!

**Кстати, если с первого раза не добавляется партнер, нужно попробовать давить его 4 раза и на 5-й раз он добавится. Магия!**



Проблема возникает из-за того, что в базе данных уже есть партнеры с ID №1, №2, №3, №4 и программа не может добавить нового партнера по уже существующему ID. Его можно только редактировать. После перебора создастся ID №5 и добавится новый партнер.

Не знаю как это фиксить, но все равно все работает, просто не так как задумано.



Ура, все работает

****

**Приступаем к последнему, самому жестокому модулю четвёртому модулю…**



**МОДУЛЬ 4**

**Функционал кнопки «История продаж» у каждого партнера.**

Найти файл **App.axaml** и заменить код на:

***App.axaml***

<Application xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

x:Class="DemoApp.App"

xmlns:local="using:DemoApp"

RequestedThemeVariant="Default">

<!-- "Default" ThemeVariant follows system theme variant. "Dark" or "Light" are other available options. -->

<Application.DataTemplates>

<local:ViewLocator />

</Application.DataTemplates>

<Application.Styles>

<FluentTheme />

<Style Selector="TextBlock.h1">

<Setter Property="FontSize" Value="24" />

<Setter Property="FontWeight" Value="Bold" />

<Setter Property="Foreground" Value="Green" />

</Style>

<Style Selector="StackPanel.main">

<Setter Property="Background" Value="#F4E8D3" />

</Style>

<Style Selector="Button">

<Setter Property="Background" Value="#67BA80" />

</Style>

</Application.Styles>

</Application>

Здесь мы задали тему программы.

Создать файлы авалония для настройки внешнего вида окна Стоимости в папке **Views**:

dotnet new avalonia.usercontrol -n PartnerSalesView

***PartnerSalesView.axaml***

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:vm="using:DemoApp.ViewModels"

x:Class="DemoApp.Views.PartnerSalesView"

x:DataType="vm:PartnerSalesViewModel">

<Grid Margin="20">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Spacing="10">

<Button Content="← Вернуться к списку"

Command="{Binding BackCommand}"

HorizontalAlignment="Left"/>

<TextBlock Text="{Binding Title}"

FontSize="24"

FontWeight="Bold"/>

</StackPanel>

<StackPanel Grid.Row="1" Spacing="5" Margin="0,10">

<TextBlock Text="{Binding SummaryInfo}"

FontSize="16"/>

<TextBlock Text="{Binding DiscountInfo}"

FontSize="16"

Foreground="Green"/>

</StackPanel>

<Border Grid.Row="2"

BorderBrush="{StaticResource SystemBaseMediumColor}"

BorderThickness="1"

CornerRadius="4">

<Grid>

<ScrollViewer IsVisible="{Binding HasSales}">

<ItemsControl ItemsSource="{Binding Sales}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Margin="5"

Padding="10"

BorderBrush="{StaticResource SystemBaseLowColor}"

BorderThickness="1"

CornerRadius="4">

<Grid ColumnDefinitions="2\*,3\*,2\*,2\*">

<StackPanel Grid.Column="0" Margin="5">

<TextBlock Text="Дата продажи:"

Foreground="{StaticResource SystemBaseMediumColor}"/>

<TextBlock Text="{Binding SaleDateFormatted}"

FontWeight="SemiBold"/>

</StackPanel>

<StackPanel Grid.Column="1" Margin="5">

<TextBlock Text="Наименование:"

Foreground="{StaticResource SystemBaseMediumColor}"/>

<TextBlock Text="{Binding ProductName}"

FontWeight="SemiBold"/>

</StackPanel>

<StackPanel Grid.Column="2" Margin="5">

<TextBlock Text="Количество:"

Foreground="{StaticResource SystemBaseMediumColor}"/>

<TextBlock Text="{Binding \_sale.ProductCount}"

FontWeight="SemiBold"/>

</StackPanel>

<StackPanel Grid.Column="3" Margin="5">

<TextBlock Text="Стоимость:"

Foreground="{StaticResource SystemBaseMediumColor}"/>

<TextBlock Text="{Binding TotalCostFormatted}"

FontWeight="SemiBold"

Foreground="Green"/>

</StackPanel>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

<TextBlock Text="{Binding NoSalesMessage}"

IsVisible="{Binding !HasSales}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontSize="16"

Foreground="{StaticResource SystemBaseMediumColor}"/>

</Grid>

</Border>

<TextBlock Grid.Row="3"

Text="{Binding TotalInfo}"

FontSize="18"

FontWeight="Bold"

Margin="0,10,0,0"/>

</Grid>

</UserControl>

***PartnerSalesView.axaml.cs***

using Avalonia.Controls;

namespace DemoApp.Views

{

public partial class PartnerSalesView : UserControl

{

public PartnerSalesView()

{

InitializeComponent();

}

}

}

Создать класс **Sale** в папке **Models** для отображения данных и их схватывания.

dotnet new class -n Sale

***Sale.cs***

using System;

using DemoApp.Entities;

namespace DemoApp.Models

{

public class Sale

{

public Sale(Entities.Sale sale)

{

\_sale = sale ?? throw new ArgumentNullException(nameof(sale));

}

public Entities.Sale \_sale { get; set; }

public string ProductName => \_sale.ProductArticleNavigation?.ProductName ?? "Нет данных";

public decimal? TotalCost => \_sale.ProductCount \* (\_sale.ProductArticleNavigation?.MinimumCostForPartner ?? 0);

public string SaleDateFormatted => \_sale.SaleDate?.ToString("dd.MM.yyyy") ?? "Дата не указана";

public string TotalCostFormatted => TotalCost.HasValue ? TotalCost.Value.ToString("C2") : "0.00 ₽";

}

}

Создать класс С# **PartnerSalesViewModel** в папке **ViewModels** для оформления окна истории продаж.

dotnet new class -n PartnerSalesViewModel

***PartnerSalesViewModel.cs:***

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using CommunityToolkit.Mvvm.ComponentModel;

using CommunityToolkit.Mvvm.Input;

using DemoApp.Entities;

using DemoApp.Models;

using DemoApp.Views;

using Microsoft.EntityFrameworkCore;

namespace DemoApp.ViewModels

{

public partial class PartnerSalesViewModel : ViewModelBase

{

private readonly DbDemoFdbContext \_context;

private readonly MainWindowViewModel \_mainViewModel;

private readonly int \_partnerId;

[ObservableProperty]

private ObservableCollection<Models.Sale> \_sales = new();

[ObservableProperty]

private string \_title;

[ObservableProperty]

private string \_summaryInfo;

[ObservableProperty]

private string \_discountInfo;

[ObservableProperty]

private string \_totalInfo;

[ObservableProperty]

private bool \_hasSales;

[ObservableProperty]

private string \_noSalesMessage = "У данного партнера пока нет истории продаж";

public PartnerSalesViewModel(DbDemoFdbContext context, MainWindowViewModel mainViewModel, int partnerId, string partnerName)

{

\_context = context;

\_mainViewModel = mainViewModel;

\_partnerId = partnerId;

Title = $"История продаж: {partnerName}";

LoadSales();

}

private void LoadSales()

{

var partner = \_context.Partners

.Include(p => p.Sales)

.ThenInclude(s => s.ProductArticleNavigation)

.FirstOrDefault(p => p.PartnerId == \_partnerId);

if (partner == null) return;

var sales = partner.Sales?

.Where(s => s.ProductCount.HasValue && s.ProductArticleNavigation != null)

.OrderByDescending(s => s.SaleDate)

.ToList() ?? new List<Entities.Sale>();

Sales = new ObservableCollection<Models.Sale>(

sales.Select(s => new Models.Sale(s))

);

HasSales = Sales.Any();

if (HasSales)

{

var totalProducts = sales.Sum(s => s.ProductCount ?? 0);

var totalCost = sales.Sum(s => (s.ProductCount ?? 0) \* (s.ProductArticleNavigation?.MinimumCostForPartner ?? 0));

var discount = Models.Partner.CalculatePartnerDiscount(totalProducts);

SummaryInfo = $"Всего продаж: {sales.Count}";

DiscountInfo = $"Текущая скидка партнера: {discount:P0}";

TotalInfo = $"Общая стоимость продаж: {totalCost:C2}";

}

else

{

SummaryInfo = "Нет данных о продажах";

DiscountInfo = "Текущая скидка партнера: 0%";

TotalInfo = "Общая стоимость продаж: 0.00 ₽";

}

}

[RelayCommand]

private void Back()

{

\_mainViewModel.ShowPartnersList();

}

}

}

Далее, обновить код в файлах:

***PartnersView.axaml***

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

d:DesignWidth="800"

d:DesignHeight="450"

x:Class="DemoApp.Views.PartnersView"

xmlns:vm="using:DemoApp.ViewModels"

x:DataType="vm:PartnersViewModel">

<Grid Margin="20">

<StackPanel Spacing="10" Classes="main">

<TextBlock Text="Список партнеров" FontSize="24" FontWeight="Bold" Classes="h1" />

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

<Button Content="Добавить партнера" Command="{Binding AddPartnerCommand}" />

<Button Content="Редактировать" Command="{Binding EditPartnerCommand}" />

</StackPanel>

<ListBox ItemsSource="{Binding Partners}"

SelectedItem="{Binding SelectedPartner}"

MaxHeight="800">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Margin="5" HorizontalAlignment="Left">

<StackPanel Orientation="Vertical" Margin="5" HorizontalAlignment="Left">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Left">

<TextBlock Text="{Binding PartnerTypeDescription}" TextWrapping="Wrap" />

<TextBlock Text=" | " TextWrapping="Wrap" />

<TextBlock Text="{Binding \_partner.PartnerName}" TextWrapping="Wrap" Width="250" />

</StackPanel>

<TextBlock Text="{Binding \_partner.Director}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

<TextBlock Text="{Binding \_partner.DirectorPhone}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

<TextBlock Text="{Binding RatingDisplay}" TextWrapping="Wrap" Width="250" HorizontalAlignment="Left" />

</StackPanel>

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

<TextBlock Text="{Binding DiscountDisplay}" TextWrapping="Wrap" Width="150" />

<Button Content="История продаж"

Command="{Binding $parent[ListBox].((vm:PartnersViewModel)DataContext).ViewSalesHistoryCommand}"

CommandParameter="{Binding}" />

</StackPanel>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</StackPanel>

</Grid>

</UserControl>

***PartnersView.axaml.cs***

using Avalonia;

using Avalonia.Controls;

using Avalonia.Markup.Xaml;

namespace DemoApp.Views;

public partial class PartnersView : UserControl

{

public PartnersView()

{

InitializeComponent();

}

}

***PartnersViewModel.cs***

using System.Collections.ObjectModel;

using System.Linq;

using CommunityToolkit.Mvvm.ComponentModel;

using CommunityToolkit.Mvvm.Input;

using DemoApp.Entities;

using DemoApp.Views;

using Microsoft.EntityFrameworkCore;

namespace DemoApp.ViewModels

{

public partial class PartnersViewModel : ObservableObject

{

private readonly DbDemoFdbContext \_context;

internal readonly MainWindowViewModel \_mainViewModel;

[ObservableProperty]

private ObservableCollection<Models.Partner> \_partners = new();

[ObservableProperty]

private Models.Partner? \_selectedPartner;

public PartnersViewModel(DbDemoFdbContext context, MainWindowViewModel mainViewModel)

{

\_context = context;

\_mainViewModel = mainViewModel;

LoadPartners();

}

public void LoadPartners()

{

var entities = \_context.Partners

.Include(p => p.Sales)

.Include(p => p.PartnerTypeNavigation)

.ToList();

Partners = new ObservableCollection<Models.Partner>(

entities.Select(e => new Models.Partner(e))

);

}

[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);

}

[RelayCommand]

private void AddPartner()

{

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

var editControl = new EditPartnerView { DataContext = viewModel };

\_mainViewModel.ShowEditPartner(editControl);

}

[RelayCommand]

private void ViewSalesHistory(Models.Partner? partner)

{

if (partner is null) return;

\_mainViewModel.ShowContent(new PartnerSalesView

{

DataContext = new PartnerSalesViewModel(

\_context,

\_mainViewModel,

partner.\_partner.PartnerId ?? 0,

partner.\_partner.PartnerName

)

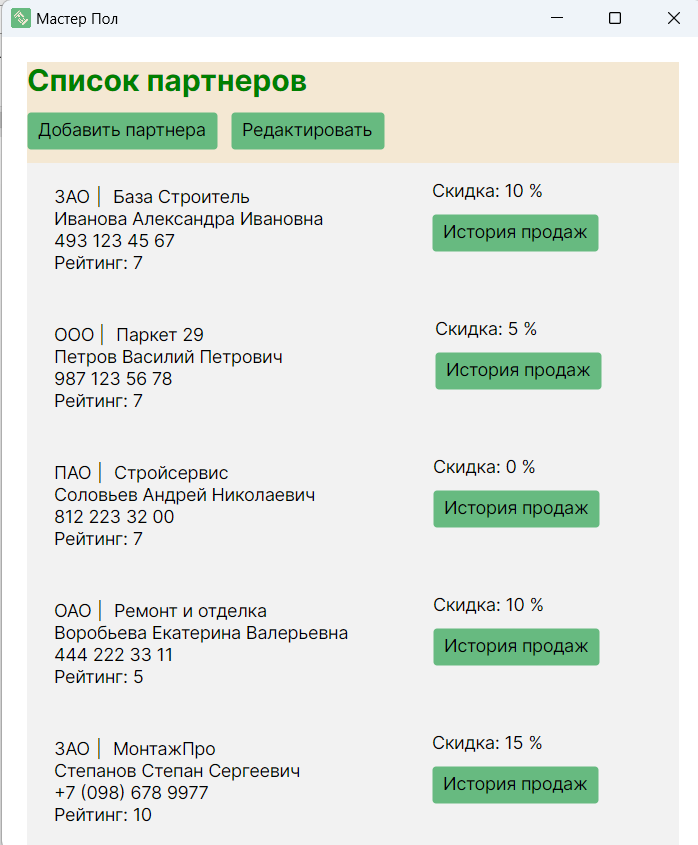
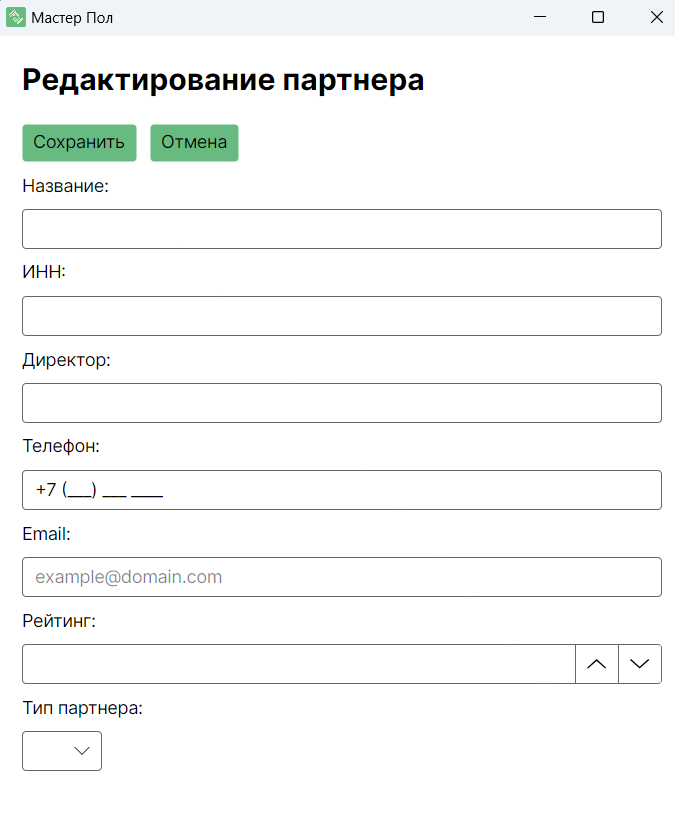
});

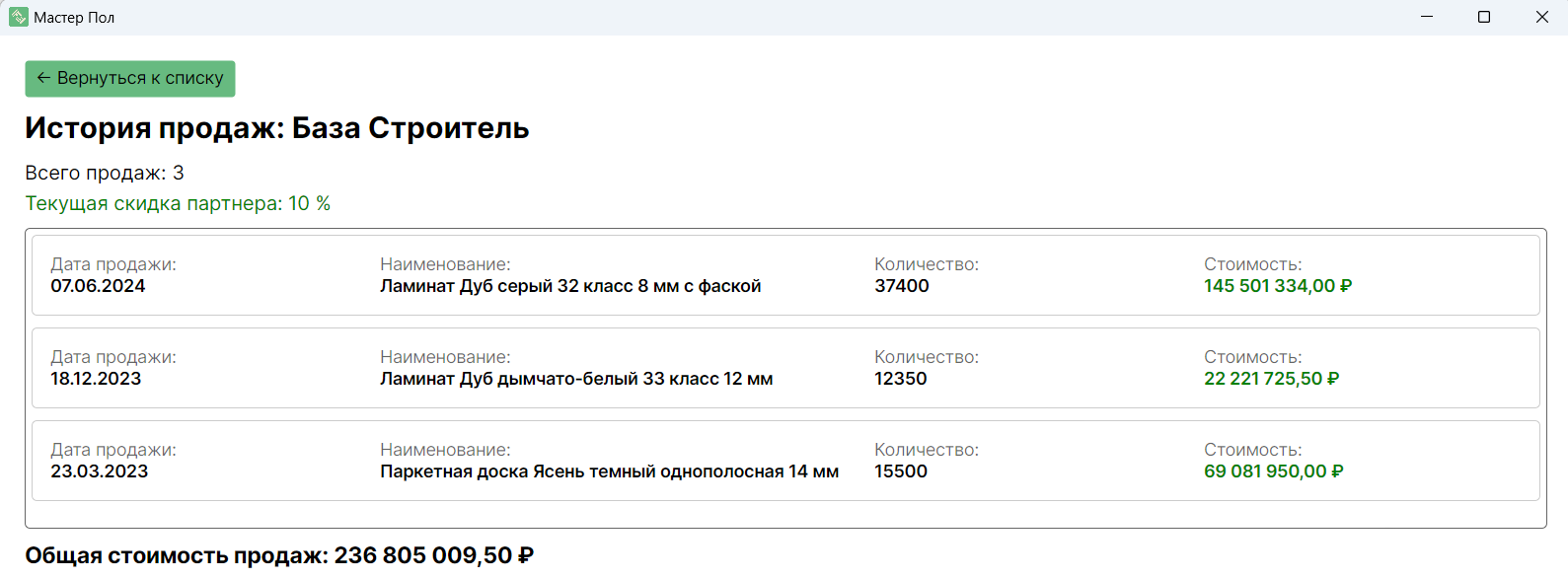
}

}

}

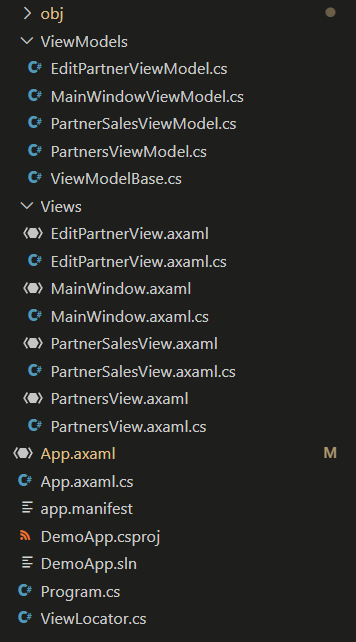
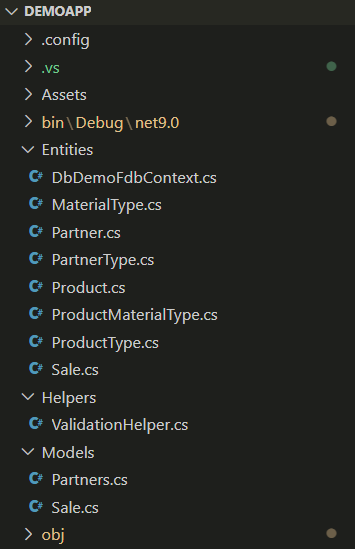
На этом все. Получаем при запуске:





**На этом можно закончить выполнение 4 модуля и решения демонстрационного экзамена.**

**Структура проекта:**



Ссылки гитхаб

Проект VS Code

<https://github.com/AlinaPipu/DemoApp>

Готовая база данных в Red Expert с отводными таблицами и триггерами.

<https://github.com/AlinaPipu/DBRedExpert>