# Создание формы вывода истории реализации продукции партнера

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

Первое что нам необходимо это исправить x:Class с:

x:Class="/Views.PartnerSalesView"

на:

x:Class="DemoApp.Views.PartnerSalesView"

Также необходимо открыть файл PartnerSalesView.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 PartnerSalesViewModel:

Изменим наш файл следующим образом:

public partial class PartnerSalesViewModel : ViewModelBase

{

private readonly DbDemoFdbContext \_context;

private readonly MainWindowViewModel \_mainViewModel;

private readonly int \_partnerId;

}

Здесь мы объявили объект для работы с базой данных, родительскую модель представления и идентификатор партнера.

Добавим в наш каталог новую модуль с имемен Sale

Содержимое класса:

public class 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 ₽";

public Sale(Entities.Sale sale)

{

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

}

}

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

[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";

Далее нам необходимо создать конструктор для данной модели представления

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

{

\_context = context;

\_mainViewModel = mainViewModel;

\_partnerId = partnerId;

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

LoadSales();

}

В качестве параметров конструктор принимает объект для работы с базой данных, полями партнера и родительскую модель представления с дальнейшем присвоением значений.

В конструкторе необходимо вызвать метод загрузки истории реализации продукции:

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<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 ₽";

}

}

Кроме этого, нам необходимо создать метод, чтобы вернуться на предыдущую форму:

[RelayCommand]

private void Back()

{

\_mainViewModel.ShowPartnersList();

}

Подключим модель представления к нашему представлению для этого отредактируем файл PartnerSalesView.axaml

Добавим в заголовок следующие строки:

xmlns:vm="using:DemoApp.ViewModels"

x:DataType="vm:PartnerSalesViewModel"

Следующим шагом будет разметка нашего контрола для этого вместо:

Welcome to Avalonia!

необходимо вставить:

<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}"

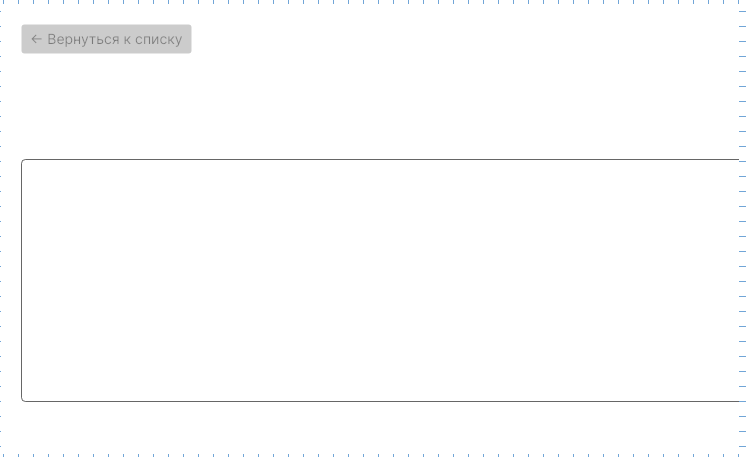
FontSize="18"

FontWeight="Bold"

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

</Grid>

Наш пользовательский элемент управления выглядит следующим образом:



Необходимо добавить переход с пользовательского элемента управления вывода истории реализации продукции. Для этого откроем файл PartnersViewModel.cs

Изменим метод

[RelayCommand]

private void ViewSalesHistory(Models.Partner? partner)

{

if (partner is null) return;

\_mainViewModel.ShowContent(new PartnerSalesView

{

DataContext = new PartnerSalesViewModel(\_context, \_mainViewModel, partner.\_partner.PartnerId, partner.\_partner.PartnerName)

});

}

Исправим разметку у кнопки

<Button Content="История продаж"

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

CommandParameter="{Binding}"/>

Запустите приложение.

Добавление стилей

<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>

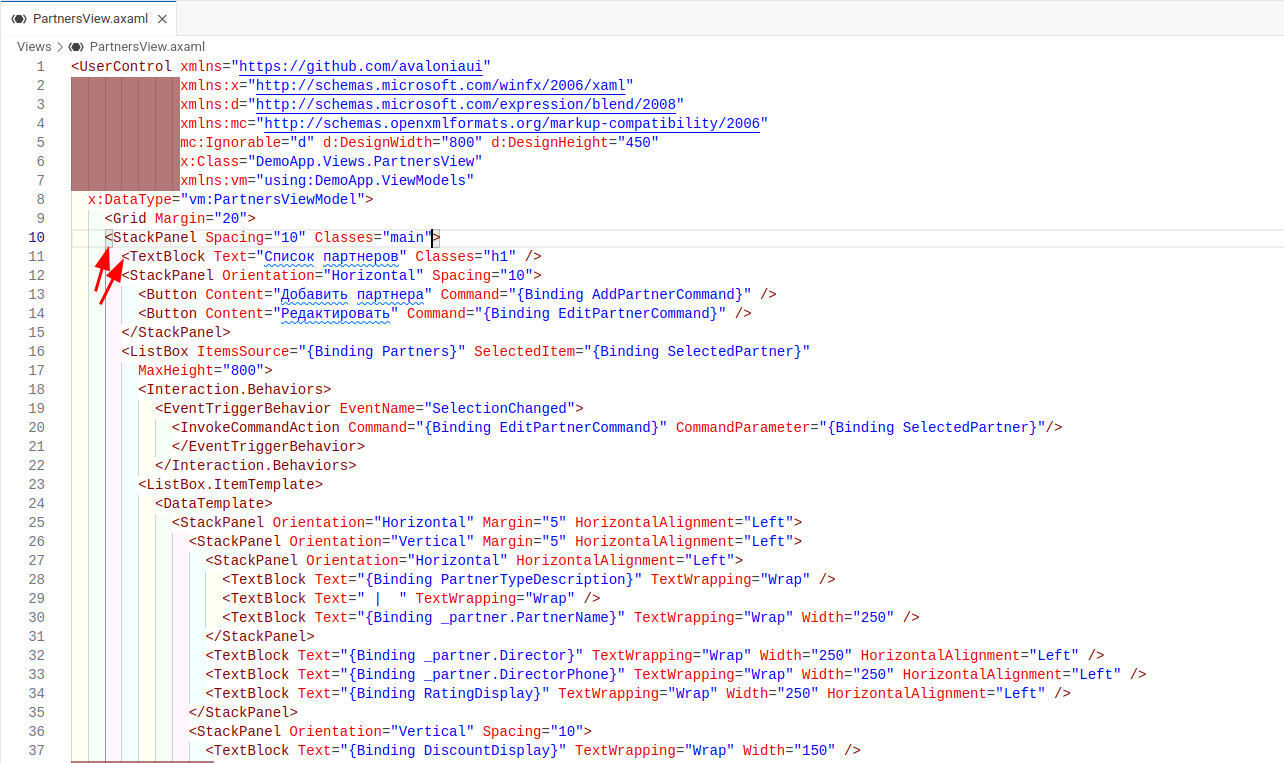
</Application>

Обращение к стилям

Необходимо отрыть файл PartnersView.axaml и изменить свойства элементов

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

<TextBlock Text="Список партнеров" Classes="h1" />



Добавление иконки. Поместите файл \*.ico в папку Assets измените название на logo.ico

Измените свойство Icon у главного окна

Icon="/Assets/logo.ico"

Необходимо дополнить класс ValidationHelper.cs методом CalculateRequiredMaterial

public static int CalculateRequiredMaterial(

int productTypeId,

int materialTypeId,

int productQuantity,

decimal parameter1,

decimal parameter2)

{

DbDemoFdbContext context = new DbDemoFdbContext();

try

{

if (productQuantity <= 0 || parameter1 <= 0 || parameter2 <= 0)

return -1;

var productType = context.ProductTypes.FirstOrDefault(pt => pt.ProductTypeId == productTypeId);

var materialType = context.MaterialTypes.FirstOrDefault(mt => mt.MaterialTypeId == materialTypeId);

if (productType == null || materialType == null)

return -1;

// Расчет базового количества материала на единицу продукции

decimal? baseAmount = parameter1 \* parameter2 \* productType.ProductTypeCoefficient;

// Расчет общего количества с учетом количества продукции

decimal? totalAmount = baseAmount \* productQuantity;

// Учет процента брака

decimal? wasteMultiplier = 1 + materialType.PercentageDefectiveMaterial;

decimal? finalAmount = totalAmount \* wasteMultiplier;

// Округление до целого числа в большую сторону

return (int)Math.Ceiling(finalAmount ?? 0);

}

catch

{

return -1;

}

}