

Модуль 1:

Запускаем сервис

```
systemctl start firebird
```

Создаем папку

```
sudo mkdir /db
```

```
sudo chown reddbatabase /db
```

В главном окне Red Expert нажмите на кнопку "Создать базу данных" (или выберите соответствующий пункт в меню).

В открывшемся окне укажите параметры базы данных:

Имя подключения: DemoConnection

Файл базы данных: /db/demo.fdb

Имя пользователя: sysdba

Пароль: пароль администратора

Нажмите "Create", чтобы создать базу данных.

С учетом связей в таблицах необходимо создать таблицы:

```
CREATE TABLE PRODUCT_TYPE(  
PRODUCT_TYPE_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
PRODUCT_TYPE_NAME VARCHAR(100),  
PRODUCT_TYPE_COEFFICIENT DECIMAL(10,2)  
);
```

Импортируем тип продукта

```
CREATE TABLE MATERIAL_TYPE(  
MATERIAL_TYPE_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
MATERIAL_TYPE_NAME VARCHAR(100),  
PERCENTAGE_DEFECTIVE_MATERIAL DECIMAL(5,4)  
);
```

Импортируем тип материала

```
CREATE TABLE PRODUCT(  
PRODUCT_ARTICLE VARCHAR(7) PRIMARY KEY, --указываем varchar с учетом возможных 30%  
изменений в задании ДЭ, можно указать INT, нужно будет исходить из задания, varchar усложняет  
импорт  
PRODUCT_NAME VARCHAR(100),  
MINIMUM_COST_FOR_PARTNER DECIMAL(15,2),  
PRODUCT_TYPE INT,  
FOREIGN KEY (PRODUCT_TYPE) REFERENCES PRODUCT_TYPE(PRODUCT_TYPE_ID)  
);
```

Меняем тип продукции на ид из БД. Импортируем продукты

```
CREATE TABLE PARTNER_TYPE(  
PARTNER_TYPE_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
PARTNER_TYPE_NAME VARCHAR(100)  
);
```

Импортируем тип партнера

```
CREATE TABLE PARTNER(  

```

```
PARTNER_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
PARTNER_NAME VARCHAR(100),  
DIRECTOR VARCHAR(100),  
DIRECTOR_MAIL VARCHAR(100),  
DIRECTOR_PHONE VARCHAR(25),  
PARTNER_LEGAL_ADDRESS VARCHAR(100),  
PARTNER_INN VARCHAR(100),  
PARTNER_RATING INT,  
PARTNER_TYPE INT,  
FOREIGN KEY (PARTNER_TYPE) REFERENCES PARTNER_TYPE(PARTNER_TYPE_ID)  
);
```

Меняем тип партнера на ид из БД. Импортируем партнеров

```
CREATE TABLE SALE(  
SALE_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
PRODUCT_ARTICLE VARCHAR(7), --указать тот тип данных, который был указан в PRODUCT  
PARTNER INT,  
PRODUCT_COUNT INT,  
SALE_DATE DATE,  
FOREIGN KEY (PARTNER) REFERENCES PARTNER(PARTNER_ID),  
FOREIGN KEY (PRODUCT_ARTICLE) REFERENCES PRODUCT(PRODUCT_ARTICLE)  
);
```

Заменить продукцию на артикул и партнера на ид. Импортируем данные

```
CREATE TABLE PRODUCT_MATERIAL_TYPE(  
PRODUCT_MATERIAL_TYPE_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
PRODUCT VARCHAR(7),  
MATERIAL_TYPE INT,  
FOREIGN KEY (PRODUCT) REFERENCES PRODUCT(PRODUCT_ARTICLE),  
FOREIGN KEY (MATERIAL_TYPE) REFERENCES MATERIAL_TYPE(MATERIAL_TYPE_ID)  
);
```

Данные создаем с помощью вкладки Data Generator

Создать ER диаграмму Tools>Er-diagram>Реверс инженеринг

Модуль 2 и 3

Создаем проект:

Необходимо в боковом меню выбрать пункт **Explorer** (1) и далее выбрать пункт **Create Avalonia Project** (2)

Далее выбираем пункт **Avalonia MVVM App**

Вводим название нашего проекта DemoApp

После выбираем директорию, где необходимо сохранить наш проект

После создания проекта у нас должно появиться уведомление с предложением открыть нашу папку в редакторе. Нажимаем на **Open**

ДОБАВЛЕНИЕ ПАКЕТОВ В ПРОЕКТ:

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

```
dotnet add package MessageBox.Avalonia --version 3.2.0
```

ТЕСТОВЫЙ ЗАПУСК ПРИЛОЖЕНИЯ:

Для запуска приложения в боковом меню нам необходимо выбрать пункт **Run and Debug** (1) далее на открывшейся панели необходимо нажать на кнопку **Run and Debug** (2) и выбрать **C#**

Далее необходимо выбрать наш проект

ФОРМИРОВАНИЯ КЛАССОВ СУЩНОСТЕЙ:

```
dotnet ef dbcontext scaffold
```

```
"DataSource=localhost;Port=3050;Database=db/demo.fdb;Username=sysdba;Password=student"
FirebirdSql.EntityFrameworkCore.Firebird -o Entities -f
```

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"
Title="DemoApp">
```

```
<Design.DataContext>
<!-- This only sets the DataContext for the previewer in an IDE,
to set the actual DataContext for runtime, set the DataContext property in code (look at App.axaml.cs) -->
<vm:MainWindowViewModel/>
</Design.DataContext>
```

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

```
</Window>
```

MainWindowViewModel.cs:

```
using Avalonia.Controls;
using CommunityToolkit.Mvvm.ComponentModel;
using DemoApp.Entities;
using DemoApp.Views;
```

```
namespace DemoApp.ViewModels;
```

```
public partial class MainWindowViewModel : ViewModelBase
{
    private readonly DbDemoFdbContext _context;
    [ObservableProperty]
    private Control? _content;
    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;
    }
}
```

Создаем **PartnersView: Views>New Avalonia Template>UserControl** (Не создаем ViewModel)

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" 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">
<Interaction.Behaviors>
<EventTriggerBehavior EventName="SelectionChanged">
<InvokeCommandAction Command="{Binding EditPartnerCommand}" CommandParameter="{Binding
SelectedPartner}"/>
</EventTriggerBehavior>
</Interaction.Behaviors>
<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>
</ListBox>
</StackPanel>
</Grid>
</UserControl>

```

Создаем в папке ViewModels 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
    {
        [ObservableProperty]
        private ObservableCollection<Models.Partner> _partners = new();

        [ObservableProperty]
        private Models.Partner? _selectedPartner;

        private readonly DbDemoFdbContext _context;
        internal readonly MainWindowViewModel _mainViewModel;
        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
}
}
}
```

Создаем 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">
<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}" />
</Grid>
<StackPanel Orientation="Horizontal" Spacing="10">
<Button Content="Сохранить" Command="{Binding SaveCommand}" />
<Button Content="Отмена" Command="{Binding CancelCommand}" />
</StackPanel>
</StackPanel>
</Grid>
</UserControl>
```

Создаем EditPartnerViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using DemoApp.Entities;
using DemoApp.Helpers;
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;
        [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();
```

```
    LoadPartnerTypes();  
    LoadPartnerData();  
}  
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;  
}
```

```
[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();  
}
```

```
[RelayCommand]  
private void Cancel()  
{  
    _parentViewModel?._mainViewModel.ShowPartnersList();  
}
```

```
private bool ValidateData()  
{  
    if (string.IsNullOrEmpty(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.IsNullOrEmpty(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;
}
}
}

```

Создаем в Models 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
};
}
}
}

```

Создаем пакет Helpers в нем создаем ValidationHelper.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace DemoApp.Helpers
{
    public class ValidationHelper
    {
        public static bool IsValidInn(string? inn)
        {
            if (string.IsNullOrEmpty(inn)) return false;
            return Regex.IsMatch(inn, @"^\d{10}$|^\d{12}$");
        }
    }
}

```

```

public static bool IsValidEmail(string? email)
{
    if (string.IsNullOrEmpty(email)) return false;
    return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$");
}

```

```

public static bool IsValidPhoneNumber(string? phone)
{
    if (string.IsNullOrEmpty(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;
}
}
}

```

Меняем стили App.xaml

```

<Application xmlns="https://github.com/avaloniaui"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="DemoApp2.App"

```

```
xmlns:local="using:DemoApp2"
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>
```

Во всех .axaml.cs файлах заменить /View на DemoApp.Views