Модуль 1:
Запускаем сервис

```
systemctl start firebird
```

Создаем папку

```
sudo mkdir /db
sudo chown reddatabase /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)
);

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)
);

CREATE TABLE MATERIAL(
MATERIAL_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
MATERIAL_NAME VARCHAR(100),
UNIT_COST DECIMAL(15,2),
QUANTITY_IN_STOCK DECIMAL(15,2),
MINIMAL_QUANTITY DECIMAL(15,2),
QUANTITY_PER_PACKAGE DECIMAL(15,2),
UNIT VARCHAR(5),
MATERIAL_TYPE INT,
FOREIGN KEY (MATERIAL_TYPE) REFERENCES MATERIAL_TYPE(MATERIAL_TYPE_ID)
);

CREATE TABLE SUPPLIER_TYPE(
SUPPLIER_TYPE_ID  INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
SUPPLIER_TYPE_NAME VARCHAR(100)
);

CREATE TABLE SUPPLIER(
SUPPLIER_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
SUPPLIER_NAME VARCHAR(100),
SUPPLIER_INN VARCHAR(100),
SUPPLIER_RATING INT,
SUPPLIER_DATE DATE,
SUPPLIER_TYPE INT,
FOREIGN KEY (SUPPLIER_TYPE) REFERENCES SUPPLIER_TYPE(SUPPLIER_TYPE_ID)
);

CREATE TABLE MATERIAL_SUPPLIER(
MATERIAL_SUPPLIER_ID INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
MATERIAL_NAME INT,
SUPPLIER_NAME INT,
FOREIGN KEY (SUPPLIER_NAME) REFERENCES SUPPLIER(SUPPLIER_ID),
FOREIGN KEY (MATERIAL_NAME) REFERENCES MATERIAL(MATERIAL_ID)
);

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 DemoApp3.Entities;
using DemoApp3.Views;

namespace DemoApp3.ViewModels;

public partial class MainWindowViewModel : ViewModelBase
{
private readonly DbDemoFdbContext _context;
[ObservableProperty]
private Control? _content;
public void ShowContent(Control content)
{
Content = content;
}
public void ShowMaterialsList()
{
Content = new MaterialView
{
DataContext = new MaterialViewModel(_context, this)
};
}
public MainWindowViewModel()
{
_context = new DbDemoFdbContext();
ShowMaterialsList();
}
public void ShowEditMaterial(EditMaterialView editControl)
{
Content = editControl;
}
}
```

Создаем MaterialView: **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="DemoApp3.Views.MaterialView"
xmlns:vm="using:DemoApp3.ViewModels"
x:DataType="vm:MaterialViewModel">
<Grid Margin="20">
<StackPanel Spacing="10" Classes="main">
<TextBlock Text="Список партнеров" Classes="h1" />
```

```
<StackPanel Orientation="Horizontal" Spacing="10">
<Button Content="Добавить материал" Command="{Binding AddMaterialCommand}" />
<Button Content="Редактировать" Command="{Binding EditMaterialCommand}" />
</StackPanel>
<ListBox ItemsSource="{Binding Materials}" SelectedItem="{Binding SelectedMaterial}"
MaxHeight="800">
<Interaction.Behaviors>
<EventTriggerBehavior EventName="SelectionChanged">
<InvokeCommandAction Command="{Binding EditMaterialCommand}" CommandParameter="{Binding
SelectedMaterial}"/>
</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 MaterialTypeDescription}" TextWrapping="Wrap" />
<TextBlock Text=" | " TextWrapping="Wrap" />
<TextBlock Text="{Binding _material.MaterialName}" TextWrapping="Wrap" Width="250" />
</StackPanel>
<TextBlock Text="{Binding _material.MinimalQuantity}" TextWrapping="Wrap" Width="250"
HorizontalAlignment="Left" />
<TextBlock Text="{Binding _material.QuantityInStock}" TextWrapping="Wrap" Width="250"
HorizontalAlignment="Left" />
<TextBlock Text="{Binding _material.UnitCost}" TextWrapping="Wrap" Width="250"
HorizontalAlignment="Left" />
</StackPanel>
<StackPanel Orientation="Vertical" Spacing="10">
<TextBlock Text="{Binding BatchCostDisplay}" TextWrapping="Wrap" Width="150" />
<Button Content="История продаж" />
</StackPanel>
</StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>
</StackPanel>
</Grid>
</UserControl>
```

Создаем в папке ViewModels MaterialViewModel.cs

```
using System.Collections.ObjectModel;
using System.Linq;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using DemoApp3.Entities;
using DemoApp3.Views;
using Microsoft.EntityFrameworkCore;

namespace DemoApp3.ViewModels
```

```
{
public partial class MaterialViewModel : ObservableObject
{
[ObservableProperty]
private ObservableCollection<Models.Material> _materials = new();

[ObservableProperty]
private Models.Material? _selectedMaterial;

private readonly DbDemoFdbContext _context;
internal readonly MainWindowViewModel _mainViewModel;
public MaterialViewModel(DbDemoFdbContext context, MainWindowViewModel mainViewModel)
{
_context = context;
_mainViewModel = mainViewModel;
LoadMaterials();
}
public void LoadMaterials()
{
var entities = _context.Materials.Include(p => p.MaterialTypeNavigation).ToList();
Materials = new ObservableCollection<Models.Material>(
entities.Select(e => new Models.Material(e))
);
}

[RelayCommand]
private void EditMaterial()
{
if (SelectedMaterial == null) return;
var MaterialEntity = _context.Materials.Find(SelectedMaterial._material.MaterialId);
if (MaterialEntity == null) return;

var viewModel = new EditMaterialViewModel(_context, MaterialEntity, this);
var editControl = new EditMaterialView { DataContext = viewModel };
_mainViewModel.ShowEditMaterial(editControl);
}

[RelayCommand]
private void AddMaterial()
{
var viewModel = new EditMaterialViewModel(_context, parentViewModel: this);
var editControl = new EditMaterialView { DataContext = viewModel };
_mainViewModel.ShowEditMaterial(editControl);
}
[RelayCommand]
private void ViewSalesHistory(Material? Material)
{
//история продаж
}
}
}
```

Создаем EditMaterialView.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="DemoApp3.Views.EditMaterialView"
xmlns:vm="using:DemoApp3.ViewModels"
x:DataType="vm:EditMaterialViewModel">
<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 MaterialName}" />
<TextBlock Grid.Row="1" Grid.Column="0" Text="ИНН:" />
<TextBox Grid.Row="1" Grid.Column="1" Text="{Binding MinimalQuantity}" />
<TextBlock Grid.Row="2" Grid.Column="0" Text="Директор:" />
<TextBox Grid.Row="2" Grid.Column="1" Text="{Binding QuantityInStock}" />
<TextBlock Grid.Row="3" Grid.Column="0" Text="Телефон:" />
<MaskedTextBox Grid.Row="3" Grid.Column="1" Text="{Binding UnitCost}"
Mask="+7 (000) 000 0000" />
<TextBlock Grid.Row="4" Grid.Column="0" Text="Email:" />
<TextBox Grid.Row="4" Grid.Column="1" Text="{Binding Unit}"
Watermark="example@domain.com" />
<TextBlock Grid.Row="6" Grid.Column="0" Text="Тип материала:" />
<ComboBox Grid.Row="6" Grid.Column="1"
ItemsSource="{Binding MaterialTypes}"
SelectedValue="{Binding SelectedMaterialType}"
SelectedValueBinding="{Binding MaterialTypeId}"
DisplayMemberBinding="{Binding MaterialTypeName}" />
</Grid>
<StackPanel Orientation="Horizontal" Spacing="10">
<Button Content="Сохранить" Command="{Binding SaveCommand}" />
<Button Content="Отмена" Command="{Binding CancelCommand}" />
</StackPanel>
</StackPanel>
</Grid>
</UserControl>
```

Создаем EditMaterialViewModel.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 DemoApp3.Entities;
using DemoApp3.Helpers;
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;

namespace DemoApp3.ViewModels
{
public partial class EditMaterialViewModel : ViewModelBase
{
private readonly DbDemoFdbContext _context;
private readonly MaterialViewModel? _parentViewModel;
private readonly Material _MaterialEntity;
[ObservableProperty]
private string? _materialName;

[ObservableProperty]
private decimal? _minimalQuantity;

[ObservableProperty]
private decimal? _quantityInStock;

[ObservableProperty]
private decimal? _unitCost;

[ObservableProperty]
private string? _unit;

[ObservableProperty]
private int? _MaterialRating;

[ObservableProperty]
private int? _selectedMaterialType;

[ObservableProperty]
private ObservableCollection<MaterialType> _MaterialTypes = new();

[ObservableProperty]
private string? _errorMessage;

public EditMaterialViewModel(DbDemoFdbContext context, Material? Material = null,
MaterialViewModel? parentViewModel = null)
{
_context = context;
_parentViewModel = parentViewModel;
_MaterialEntity = Material ?? new Material();
```

```
LoadMaterialTypes();
LoadMaterialData();
}
private void LoadMaterialTypes()
{
var types = _context.MaterialTypes.ToList();
MaterialTypes = new ObservableCollection<MaterialType>(types);
}

private void LoadMaterialData()
{
MaterialName = _MaterialEntity.MaterialName;
MinimalQuantity = _MaterialEntity.MinimalQuantity;
QuantityInStock = _MaterialEntity.QuantityInStock;
UnitCost = _MaterialEntity.UnitCost;
Unit = _MaterialEntity.Unit;
SelectedMaterialType = _MaterialEntity.MaterialType;
}

[RelayCommand]
private async Task SaveAsync()
{
if (!ValidateData()) {
var box = MessageBoxManager
.GetMessageBoxStandard("Error", ErrorMessage,
ButtonEnum.Ok);

var result = await box.ShowAsync();
return;}

_MaterialEntity.MaterialName = MaterialName;
_MaterialEntity.MinimalQuantity = MinimalQuantity;
_MaterialEntity.QuantityInStock = QuantityInStock;
_MaterialEntity.UnitCost = UnitCost;
_MaterialEntity.Unit = Unit;
_MaterialEntity.MaterialType = SelectedMaterialType;

if (_MaterialEntity.MaterialId == 0)
{
_context.Materials.Add(_MaterialEntity);
}

_context.SaveChanges();
_parentViewModel?.LoadMaterials();
Cancel();
}


[RelayCommand]
private void Cancel()
{
```

```
_parentViewModel?._mainViewModel.ShowMaterialsList();
}

private bool ValidateData()
{
/*if (string.IsNullOrWhiteSpace(MaterialName))
{
ErrorMessage = "Название партнера обязательно для заполнения";
return false;
}

if (!ValidationHelper.IsValidInn(MinimalQuantity))
{
ErrorMessage = "ИНН должен содержать 10 или 12 цифр";
return false;
}

if (string.IsNullOrWhiteSpace(QuantityInStock))
{
ErrorMessage = "Имя директора обязательно для заполнения";
return false;
}

if (!ValidationHelper.IsValidPhoneNumber(UnitCost))
{
ErrorMessage = "Неверный формат номера телефона. Пример: +7(999)999-99-99";
return false;
}

if (!ValidationHelper.IsValidEmail(SelectedMaterialType))
{
ErrorMessage = "Неверный формат электронной почты";
return false;
}

if (SelectedMaterialType == null)
{
ErrorMessage = "Выберите тип партнера";
return false;
}

ErrorMessage = null;*/
return true;
}
}
}
```

Создаем в Models Material.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace DemoApp3.Models
{
public class Material : Entities.Material
{
public Entities.Material _material { get; set; }
public Material(Entities.Material material)
{
_material = material;
}
public string? MaterialTypeDescription => _material.MaterialTypeNavigation?.MaterialTypeName;
public string Minimal => $"{_material.MinimalQuantity}";
public string Quantity => $"{_material.QuantityInStock}";
public decimal Cost => _material.UnitCost ?? 0;
public string Un => $"{_material.Unit}";
public decimal BatchCost => CalculatePartnerBatchCost();
public string BatchCostDisplay => $"Стоимость: {BatchCost:P0}";
public static decimal CalculatePartnerBatchCost()
{
return 0;
}
}
}
```

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.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;
}
}
}
```

Меняем стили App.examl

```
<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.View