

**Федеральное государственное образовательное бюджетное  
учреждение высшего образования  
«Финансовый университет при Правительстве Российской Федерации»  
(Финансовый университет)**

Колледж информатики и программирования

**ОТЧЁТ**  
**По учебной практике**


Специальность 09.02.07 «Информационные системы и программирование»  
(код) (наименование)

Профессиональный модуль ПМ.02 Разработка модулей программного обеспечения для компьютерных систем  
(код) (наименование)

Междисциплинарный курс МДК.01.04 Системное программирование  
(код) (наименование)

Выполнил:

Студент 4 курса 4ИСИП-321 учебной группы  
(номер) (номер)

  
(подпись)

Ф.А.Татарников  
(инициалы, фамилия)

Проверил:

Руководитель практики от Колледжа  
информатики и программирования  
Преподаватель 1КК, к.п.н. Е.Л.Альшакова  
(квалификационная категория или звание, должность) (инициалы, фамилия)

Москва – 2024

### Перечень работ, выполненных в ходе учебной практики

№ п/п	Виды работ	Оценка
1	Разработка структуры, перечня артефактов и протоколов проекта	Отлично
2	Командная работа над проектом с учетом системы контроля версий	Отлично
3	Отладка программного проекта	Отлично

## **СОДЕРЖАНИЕ**

ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 1 .....	4
ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 2 .....	8

Федеральное государственное образовательное бюджетное  
учреждение высшего образования  
**«Финансовый университет при Правительстве Российской Федерации»**  
**(Финансовый университет)**  
Колледж информатики и программирования

## **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 1**

**Тема: Формализация и алгоритмизация поставленных задач**

Студенты: Татарников Ф.А.

Группа: 4ИСИП-321

Преподаватель: Альшакова Е.Л.

Дата: 15.12.2024

Цель работы: разработать подсистему для работы с партнерами компании, обеспечивающую эффективное взаимодействие и контроль за их деятельностью. Подсистема должна обеспечить удобное управление данными о партнерах, отслеживание их продаж и расчёт индивидуальных скидок.

1. Сущности и атрибуты:

Таблица Material\_types:

- Тип материала (nvarchar) — Название материала.
- Процент брака материала (float) — Процент брака для данного типа материала.

Таблица Partner\_products:

- Продукция (nvarchar) — Название продукции.
- Наименование партнера (nvarchar) — Наименование партнера, который реализует продукцию.
- Количество продукции (float) — Количество проданной продукции.
- Дата продажи (datetime) — Дата продажи продукции.

Таблица Partners:

- Тип партнера (nvarchar) — Тип партнера.
- Наименование партнера (nvarchar) — Название компании партнера.
- Директор (nvarchar) — ФИО директора компании.
- Электронная почта партнера (nvarchar) — Email партнера.
- Телефон партнера (nvarchar) — Телефон партнера.
- Юридический адрес партнера (nvarchar) — Юридический адрес партнера.
- ИНН (float) — ИНН партнера.
- Рейтинг (float) — Рейтинг партнера.

Таблица Product\_type:

- Тип продукции (nvarchar) — Название типа продукции.

- Коэффициент типа продукции (float) — Коэффициент типа продукции.

Таблица Products:

- Тип продукции (nvarchar) — Тип продукции.
- Наименование продукции (nvarchar) — Название продукции.
- Артикул (float) — Уникальный артикул продукции.
- Минимальная стоимость для партнера (float) — Минимальная стоимость для партнера.

## 2. Взаимосвязи между таблицами:

Таблица Partners и Partner\_products:

- Взаимосвязь между таблицами: один партнер может продавать несколько видов продукции.
- В таблице Partner\_products поле Наименование партнера будет связано с полем Наименование партнера в таблице Partners.

Таблица Products и Partner\_products:

- Взаимосвязь между таблицами: одна продукция может продаваться многими партнерами.
- В таблице Partner\_products поле Продукция связано с полем Наименование продукции в таблице Products.

Таблица Product\_type и Products:

- Взаимосвязь между таблицами: каждая продукция имеет определенный тип.
- Поле Тип продукции в таблице Products связано с полем Тип продукции в таблице Product\_type.

Таблица Material\_type и Product\_type:

- Взаимосвязь между таблицами: каждый тип продукции может требовать использования определенного типа материала с процентом брака.

## 3. Функциональные требования:

Просмотр списка партнеров: отображение всех партнеров с их основными данными (например, наименование, директор, рейтинг).

Добавление/редактирование партнера: возможность добавления нового партнера или редактирования информации о существующем.

Просмотр истории реализации продукции: для каждого партнера должна быть возможность посмотреть историю проданных продуктов (по данным из таблицы Partner\_products), включая наименование продукции, количество и дату продажи.

#### 4. Алгоритм для расчета скидки:

Условия: скидка для партнера рассчитывается на основе общего количества реализованной продукции:

- до 10000 единиц — 0%
- от 10000 до 50000 единиц — 5%
- от 50000 до 300000 единиц — 10%
- более 300000 единиц — 15%

Федеральное государственное образовательное бюджетное  
учреждение высшего образования  
**«Финансовый университет при Правительстве Российской Федерации»**  
**(Финансовый университет)**  
Колледж информатики и программирования

## **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 2**

**Тема: Написание программного кода с использованием языков  
программирования, определения и манипулирования данными**

Студенты: Татарников Ф.А.

Группа: 4ИСИП-321

Преподаватель: Альшакова Е.Л.

Дата: 16.12.2024



Цель работы: разработка программы и базы данных для работы с партнерами компании с использованием SSMS и Visual Studio. Подсистема должна обеспечить эффективное управление данными о партнерах, отслеживание их продаж и расчёт индивидуальных скидок, а также интеграцию с базой данных для хранения и обработки информации.

Прежде чем разрабатывать приложение, необходимо разработать структуру рабочей базы данных, которая будет содержать все нужные данные.

Диаграмма базы данных, которая будет у нас представлена на рисунке 1.

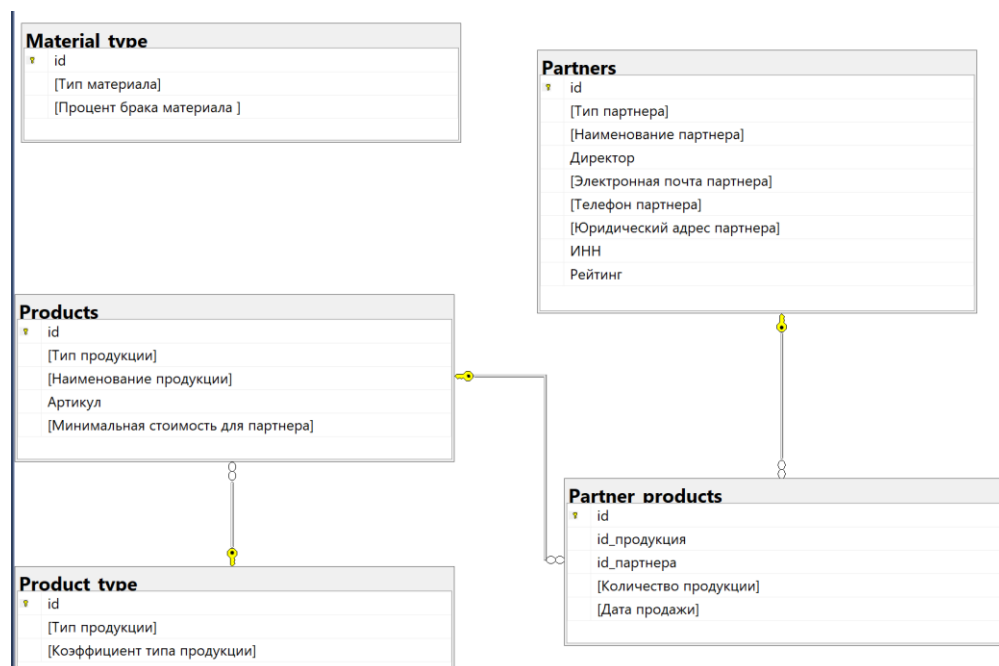


Рис. 1 Диаграмма базы данных.

На листинге 1 представлен скрипт базы данных

Листинг 1. Скрипт базы данных.

```
USE [master]
GO

/***** Object: Database [bc]    Script Date: 20.12.2024 1:59:28 *****/
CREATE DATABASE [bc]
    CONTAINMENT = NONE
    ON PRIMARY
    ( NAME = N'bc_Data', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\bc.mdf' , SIZE = 8192KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
```

```
LOG ON

( NAME = N'bc_Log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\bc.ldf' , SIZE = 8192KB , MAXSIZE =
2048GB , FILEGROWTH = 10%)

WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [bc].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [bc] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [bc] SET ANSI_NULLS OFF
GO

ALTER DATABASE [bc] SET ANSI_PADDING OFF
GO

ALTER DATABASE [bc] SET ANSI_WARNINGS OFF
GO

ALTER DATABASE [bc] SET ARITHABORT OFF
GO

ALTER DATABASE [bc] SET AUTO_CLOSE ON
GO

ALTER DATABASE [bc] SET AUTO_SHRINK OFF
GO

ALTER DATABASE [bc] SET AUTO_UPDATE_STATISTICS ON
GO

ALTER DATABASE [bc] SET CURSOR_CLOSE_ON_COMMIT OFF
GO

ALTER DATABASE [bc] SET CURSOR_DEFAULT GLOBAL
GO
```

```
ALTER DATABASE [bc] SET CONCAT_NULL_YIELDS_NULL OFF  
GO
```

```
ALTER DATABASE [bc] SET NUMERIC_ROUNDABORT OFF  
GO
```

```
ALTER DATABASE [bc] SET QUOTED_IDENTIFIER OFF  
GO
```

```
ALTER DATABASE [bc] SET RECURSIVE_TRIGGERS OFF  
GO
```

```
ALTER DATABASE [bc] SET  ENABLE_BROKER  
GO
```

```
ALTER DATABASE [bc] SET AUTO_UPDATE_STATISTICS_ASYNC OFF  
GO
```

```
ALTER DATABASE [bc] SET DATE_CORRELATION_OPTIMIZATION OFF  
GO
```

```
ALTER DATABASE [bc] SET TRUSTWORTHY OFF  
GO
```

```
ALTER DATABASE [bc] SET ALLOW_SNAPSHOT_ISOLATION OFF  
GO
```

```
ALTER DATABASE [bc] SET PARAMETERIZATION SIMPLE  
GO
```

```
ALTER DATABASE [bc] SET READ_COMMITTED_SNAPSHOT OFF  
GO
```

```
ALTER DATABASE [bc] SET HONOR_BROKER_PRIORITY OFF  
GO
```

```
ALTER DATABASE [bc] SET RECOVERY SIMPLE  
GO
```

```
ALTER DATABASE [bc] SET  MULTI_USER  
GO
```

```
ALTER DATABASE [bc] SET PAGE_VERIFY CHECKSUM
```

```
GO

ALTER DATABASE [bc] SET DB_CHAINING OFF
GO

ALTER DATABASE [bc] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO

ALTER DATABASE [bc] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO

ALTER DATABASE [bc] SET DELAYED_DURABILITY = DISABLED
GO

ALTER DATABASE [bc] SET ACCELERATED_DATABASE_RECOVERY = OFF
GO

ALTER DATABASE [bc] SET QUERY_STORE = ON
GO

ALTER DATABASE [bc] SET QUERY_STORE (OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30), DATA_FLUSH_INTERVAL_SECONDS
= 900, INTERVAL_LENGTH_MINUTES = 60, MAX_STORAGE_SIZE_MB = 1000,
QUERY_CAPTURE_MODE = AUTO, SIZE_BASED_CLEANUP_MODE = AUTO, MAX_PLANS_PER_QUERY
= 200, WAIT_STATS_CAPTURE_MODE = ON)
GO

ALTER DATABASE [bc] SET READ_WRITE
GO
```

Страница PartnersView предназначена для отображения и управления списком партнеров компании. Она позволяет просматривать информацию о партнерах, редактировать данные, а также просматривать историю продаж каждого партнера. Пользователи могут добавлять новых партнеров и изменять данные существующих, а также инициировать действия, такие как просмотр истории продаж и изменение информации о партнере. Показано на рисунке 2.

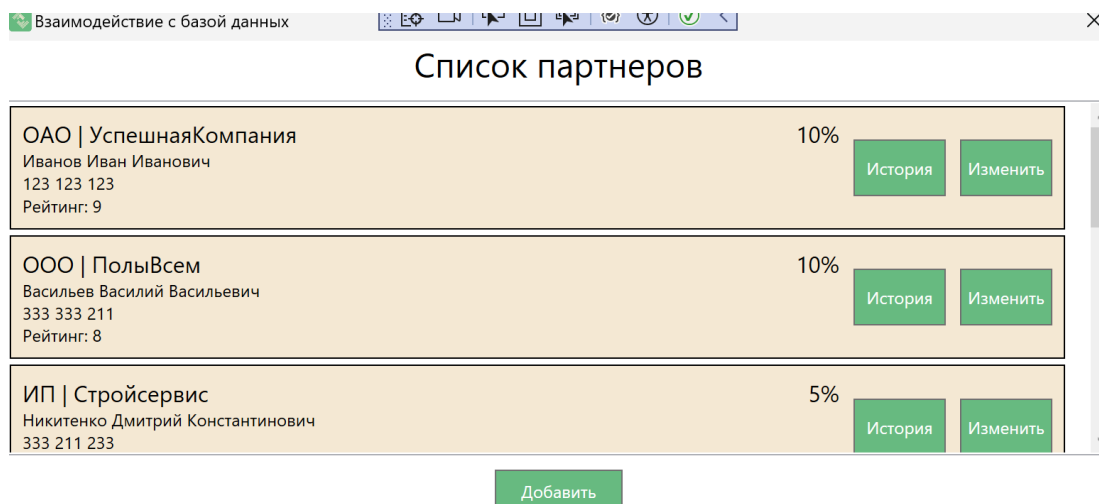


Рис. 2 Страница PartnersView.

Листинг этой страницы представлен в листинге 2 и 3.

## Листинг 2. Xaml код страницы PartnersView

```
<UserControl x:Class="WpfAppl.view.PartnersView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:WpfAppl.view"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <StackPanel Orientation="Vertical"
            Grid.Row="0">
            <!-- Заголовок -->
            <TextBlock Text="Список партнеров"
                FontSize="24"
                HorizontalAlignment="Center"
                Margin="0,0,0,10">
```

```

Grid.Row="0"/>

<!-- Основной блок -->
<ScrollViewer VerticalScrollBarVisibility="Hidden"
HorizontalScrollBarVisibility="Disabled"
PanningMode="VerticalOnly"
Grid.Row="1">
<ListView ItemsSource="{Binding PartnersData}"
Height="250">
<ListView.ItemsPanel>
<ItemsPanelTemplate>
<StackPanel Orientation="Vertical"/>
</ItemsPanelTemplate>
</ListView.ItemsPanel>
<ListView.ItemTemplate>
<DataTemplate>
<Border BorderBrush="Black" BorderThickness="1"
Padding="8" Background="#F4E8D3">
<StackPanel Orientation="Horizontal">
<Grid>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="475"/>
<ColumnDefinition Width="100"/>
<ColumnDefinition Width="150"/>
</Grid.ColumnDefinitions>
<!-- Первый StackPanel для данных -->
<StackPanel Orientation="Vertical"
Grid.Column="0">
<StackPanel
Orientation="Horizontal">
<TextBlock Text="{Binding
type}" FontSize="16"/>
<TextBlock Text=" | "
FontSize="16"/>
<TextBlock Text="{Binding
name}" FontSize="16"/>
</StackPanel>
<TextBlock Text="{Binding
director}"/>
<TextBlock Text="{Binding
phone}"/>

```

```

                                <StackPanel
Orientation="Horizontal">
                                <TextBlock      Text="{Binding
rating, StringFormat='Рейтинг: {0}'}"/>
                                </StackPanel>
                                </StackPanel>

                                <!--      TextBlock      для      Discount,
выровненный вправо -->
                                <TextBlock      Text="{Binding      discount,
StringFormat='{{0}}%'      "      FontSize="16"      HorizontalAlignment="Right"
Grid.Column="1"/>

                                <!-- кнопки элементов списка -->
                                <StackPanel Orientation="Horizontal"
Grid.Column="2"

HorizontalAlignment="Right">

                                <Button Content="История"
Grid.Column="2"
Style="{StaticResource
button_partner_edit}"

                                Command="{Binding
DataContext.checkHistoryCommand,
AncestorType=ListView}"

                                CommandParameter="{Binding
id}"

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

                                <Button Content="Изменить"
Grid.Column="2"
Style="{StaticResource
button_partner_edit}"

                                Command="{Binding
DataContext.editMemberCommand,
AncestorType=ListView}"

                                CommandParameter="{Binding
id}" />

                                </StackPanel>
                                </Grid>

```

```

        </StackPanel>

        </Border>

    </DataTemplate>

</ListView.ItemTemplate>

</ListView>

</ScrollViewer>

</StackPanel>

<!-- кнопка -->
<Button Content="Добавить"
        Style="{StaticResource button_standard}"
        Command="{Binding addMemberCommand}"
        HorizontalAlignment="Center"
        Margin="0,10,0,0"
        Grid.Row="1"/>

</Grid>
</UserControl>

```

Логика страницы представлена на листинге 3.

### Листинг 3. Код логики страницы PartnersView

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WpfAppl.entities;
using WpfAppl.utility;

namespace WpfAppl.models
{
    public class PartnersModel
    {
        // список партнеров
        public List<PartnersData> partners;

        public PartnersModel()
        {
            // инициализируем наш список
            partners = new List<PartnersData>(GetPartnersData());
        }
    }
}

```



```

        // метод заполнения списка путем получения нужных данных из бд таблицы
Partners
private List<PartnersData> GetPartnersData()
{
    List<PartnersData> result = new List<PartnersData>();
    using (var context = new MyDbContext())
    {
        foreach (var i in context.Partners)
        {
            PartnersData partnersData = new PartnersData();
            partnersData.id = i.id;
            partnersData.type = i.PartnerType;
            partnersData.name = i.PartnerName;
            partnersData.director = i.Director;
            partnersData.phone = i.DirectorPhone;
            partnersData.rating = i.Rating;
            partnersData.discount = getDiscount(i.id);

            result.Add(partnersData);
        }
    }
    return result;
}

// метод высчитывания скидки партнера через его количество продаж
private int getDiscount(int partnerID)
{
    using (var context = new MyDbContext())
    {
        if (context.partnerProducts.Where(ch => ch.PartnerID ==
partnerID).Count() == 0) return 0;

        int totalAmount = (int)context.partnerProducts.Where(ch =>
ch.PartnerID == partnerID).Sum(ch => ch.SaleCount);
        if (totalAmount < 10000)
        {
            return 0;
        }
        else if (totalAmount > 10000 && totalAmount < 50000)
        {

```

```

        return 5;
    }
    else
    {
        return 10;
    }
}
}
}
}
}

```

Страница `AddAndEditUserView` используется для создания и редактирования данных партнера. Он состоит из двух основных частей: заголовка с текущим названием страницы, а также формы для ввода данных партнера, где пользователю предлагается заполнить различные поля, такие как наименование, тип партнера, ФИО директора, почта, телефон, адрес, ИНН и рейтинг. Работа страницы представлена на рисунке 3.

Взаимодействие с базой данных

### Добавление партнера

Наименование

Тип партнера

ФИО директора

Почта

Телефон

Адрес

ИНН

Рейтинг

Рис. 3 Страница `AddAndEditUserView`.

Листинг этой страницы представлен в листинге 4 и 5.

Листинг 4. Xaml код страницы `AddAndEditUserView`.

```
<UserControl x:Class="WpfApp1.view.AddAndEditUserView"
```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:WpfAppl.view"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>

    <!-- заголовок -->
    <TextBlock Text="{Binding currentPage}"
        FontSize="24"
        HorizontalAlignment="Center"
        Margin="0,0,0,10"
        Grid.Row="0"/>

    <!-- центральная часть страницы -->
    <Grid Grid.Row="1"
        Margin="0,0,0,25">
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <!-- блок надписей слева-->

```

```

<TextBlock Text="Наименование"
            Grid.Column="0"
            Grid.Row="0"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="Тип партнера"
            Grid.Column="0"
            Grid.Row="1"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="ФИО директора"
            Grid.Column="0"
            Grid.Row="2"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="Почта"
            Grid.Column="0"
            Grid.Row="3"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="Телефон"
            Grid.Column="0"
            Grid.Row="4"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="Адрес"
            Grid.Column="0"
            Grid.Row="5"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="ИНН"
            Grid.Column="0"
            Grid.Row="6"
            Style="{StaticResource AddAndEdit}"/>
<TextBlock Text="Рейтинг"
            Grid.Column="0"
            Grid.Row="7"
            Style="{StaticResource AddAndEdit}"/>

<!-- блок ввода справа-->
<TextBox Grid.Column="1"
          Grid.Row="0"
          Style="{StaticResource AddAndEditTextBox}"
          Text="{Binding NewName, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"/>
<ComboBox Grid.Column="1"
           Grid.Row="1"
           ItemsSource="{Binding Types}"
           Style="{StaticResource AddAndEditComboBox}"

```

```

        SelectedItem="{Binding NewType, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox Grid.Column="1"
                Grid.Row="2"
                Style="{StaticResource AddAndEditTextBox}"
                Text="{Binding NewFio, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox Grid.Column="1"
                Grid.Row="3"
                Style="{StaticResource AddAndEditTextBox}"
                Text="{Binding NewMail, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox Grid.Column="1"
                Grid.Row="4"
                Style="{StaticResource AddAndEditTextBox}"
                Text="{Binding NewPhone, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                MaxLength="15"/>
        <TextBox Grid.Column="1"
                Grid.Row="5"
                Style="{StaticResource AddAndEditTextBox}"
                Text="{Binding NewAddress, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox Grid.Column="1"
                Grid.Row="6"
                Style="{StaticResource AddAndEditTextBox}"
                Text="{Binding NewINN, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                MaxLength="10"/>
        <TextBox Grid.Column="1"
                Grid.Row="7"
                Style="{StaticResource AddAndEditTextBox}"
                Text="{Binding NewRating, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0}}}"
                MaxLength="2"/>

</Grid>

<Grid Grid.Row="2"
        Margin="0,0,0,50">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />

```

```

        </Grid.ColumnDefinitions>

        <!-- кнопки под основным блоком -->
        <Button Content="Назад"
                Style="{StaticResource button_standard}"
                Command="{Binding goBackCommand}"
                Grid.Column="0"
                HorizontalAlignment="Right"
                Margin="0,0,50,0"/>
        <Button Content="Подтвердить"
                Style="{StaticResource button_standard}"
                Command="{Binding saveDataCommand}"
                Grid.Column="1"
                HorizontalAlignment="Left"
                Margin="50,0,0,0" />

    </Grid>
</Grid>
</UserControl>

```

## Листинг 5. Код логики страницы AddAndEditUserView.

```

public class AddAndEditUserVM : MainViewModel
{
    // инициализация модели
    private AddAndEditUserModel _model = new AddAndEditUserModel();
    // ссылка на класс навигации
    private Navigation _navigation;

    // лист типов партнеров
    private List<string> _types;
    public List<string> Types
    {
        get { return _types; }
        set { _types = value; OnPropertyChanged(); }
    }

    // текущая страница (создание или редактирование)
    private string _currentPage;
    public string currentPage
    {
        get { return _currentPage; }
        set { _currentPage = value; OnPropertyChanged(); }
    }
}

```

```

}

// если человек перешел чтобы изменить партнера
private int editPartnerID;

// поля которые редактируются
private string _newName;
public string NewName
{
    get => _newName;
    set
    {
        if (_newName != value)
        {
            _newName = value;
            OnPropertyChanged();
        }
    }
}

private string _newType;
public string NewType
{
    get => _newType;
    set
    {
        if (_newType != value)
        {
            _newType = value;
            OnPropertyChanged();
        }
    }
}

private string _newFio;
public string NewFio
{
    get => _newFio;
    set
    {
        if (_newFio != value)
        {
            _newFio = value;

```

```

        OnPropertyChanged();
    }
}

private string _newMail;
public string NewMail
{
    get => _newMail;
    set
    {
        if (_newMail != value)
        {
            _newMail = value;
            OnPropertyChanged();
        }
    }
}

private string _newPhone;
public string NewPhone
{
    get => _newPhone;
    set
    {
        if (_newPhone != value)
        {
            _newPhone = value;
            OnPropertyChanged();
        }
    }
}

private string _newAddress;
public string NewAddress
{
    get => _newAddress;
    set
    {
        if (_newAddress != value)
        {
            _newAddress = value;

```



```

        OnPropertyChanged();
    }
}

private string _newINN;
public string NewINN
{
    get => _newINN;
    set
    {
        if (_newINN != value)
        {
            _newINN = value;
            OnPropertyChanged();
        }
    }
}

private float _newRating;
public float NewRating
{
    get => _newRating;
    set
    {
        if (_newRating != value)
        {
            _newRating = value;
            OnPropertyChanged();
        }
    }
}

// команды для кнопок
public ICommand goBackCommand { get; set; }
public ICommand saveDataCommand { get; set; }

public AddAndEditUserVM(Navigation navigation, int partnerID = -1)
{
    // устанавливаем изначальный заголовок как создание

```

```

        _currentPage = "Добавление партнера";
        // получаем айди партнера
        editPartnerID = partnerID;
        // по умолчанию айди равно -1 дабы отличать происходит сейчас создание
или редактирование
        // при редактировании мы получим айди строго больше или равно 0
        // если редактируем, то устанавливаем данные партнера
        if (editPartnerID != -1)
        {
            _currentPage = "Изменение партнера";
            Partners partner = _model.getEditPartner(editPartnerID);
            NewName = partner.PartnerName;
            NewType = partner.PartnerType;
            NewFio = partner.Director;
            NewMail = partner.DirectorMail;
            NewPhone = partner.DirectorPhone;
            NewAddress = partner.DirectorAddress;
            NewINN = partner.INN.ToString("0");
            NewRating = partner.Rating;
        }
        // привязываем данные
        _navigation = navigation;
        Types = _model.types;

        goBackCommand = new RelayCommand(goBack);
        saveDataCommand = new RelayCommand(saveData);
    }

    // метод нажатия назад
    private void goBack(object obj)
    {
        _navigation.CurrentView = new PartnersVM(_navigation);
    }

    // метод нажатия подтвердить
    private void saveData(object obj)
    {
        // проверка данных
        if (_model.checkName(NewName) == false)
        {
            MessageBox.Show("Наименование:\nНе должно быть пустым.\nНе должно
превышать длину в 255 символов.", "Изменение наименования",
MessageBoxButton.OK, MessageBoxImage.Error);

```

```
        NewName = "";
        return;
    }

    if (_model.checkType(NewType) == false)
    {
        MessageBox.Show("Тип должен быть выбран.", "Изменение типа",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (_model.checkFio(NewFio) == false)
    {
        MessageBox.Show("ФИО:\nНе должно быть пустым.\nНе должно превышать
        длину в 255 символов.", "Изменение ФИО", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        NewFio = "";
        return;
    }

    if (_model.checkMail(NewMail) == false)
    {
        MessageBox.Show("Почта:\nНе должна быть пустой.\nНе должна
        превышать длину в 255 символов.", "Изменение почты", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        NewMail = "";
        return;
    }

    if (_model.checkPhone(NewPhone) == false)
    {
        MessageBox.Show("Телефон:\nНе должен быть пустым.\nНе должно
        превышать длину в 15 символов(8(999)888-77-66).", "Изменение телефона",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        NewPhone = "";
        return;
    }

    if (_model.checkAddress(NewAddress) == false)
    {
        MessageBox.Show("Адрес:\nНе должен быть пустым.\nНе должен
        превышать длину в 255 символов.", "Изменение адреса", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        NewAddress = "";
        return;
    }

    if (_model.checkINN(NewINN) == false)
```

```

        {
            MessageBox.Show("ИНН:\nНе должен быть пустым.\nНе должен превышать
длину в 10 символов.", "Изменение ИНН", MessageBoxButton.OK,
MessageBoxImage.Error);
            NewINN = "";
            return;
        }

        if (_model.checkRating(NewRating) == false)
        {
            MessageBox.Show("Рейтинг должен быть неотрицательным число от 0 до
10!", "Изменение рейтинга", MessageBoxButton.OK, MessageBoxImage.Error);
            NewRating = 0;
            return;
        }

        // проверка айди для создания или редактирования партнера
        if (editPartnerID == -1)
        {
            _model.AddNewUser(NewName, NewType, NewFio, NewMail, NewPhone,
NewAddress, NewINN, NewRating);
        }
        else
        {
            _model.EditUser(editPartnerID, NewName, NewType, NewFio, NewMail,
NewPhone, NewAddress, NewINN, NewRating);
        }

        // переход назад
        _navigation.CurrentView = new PartnersVM(_navigation);
    }
}

```

Страница `PartnerHistoryView` представляет собой интерфейс для отображения истории продаж партнера. Он позволяет пользователю увидеть информацию о продуктах, которые были проданы партнером, включая наименование продукции, количество и дату продажи. Также присутствует кнопка для возврата на предыдущий экран. Представлено на рисунке 4.

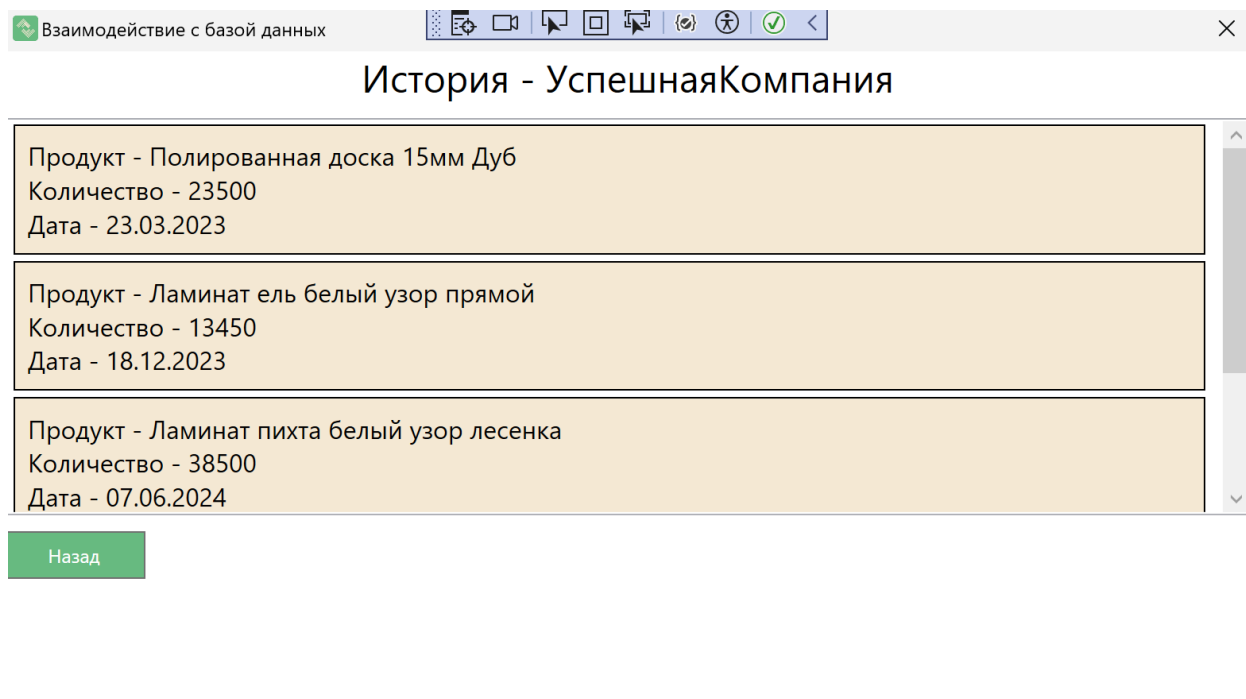


Рис. 4 Страница PartnerHistoryView

Листинг этой страницы представлен на листинге 6 и 7.

Листинг 6. Xaml код страницы PartnerHistoryView.

```
<UserControl x:Class="WpfAppl.view.PartnerHistoryView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:WpfAppl.view"
        mc:Ignorable="d"
        d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <!-- заголовок -->
        <StackPanel Orientation="Vertical"
            Grid.Row="0">
            <TextBlock
                Text="{Binding CurrentPartnerName,
StringFormat='История - {0}'}"
                FontSize="24"
```

```

        HorizontalAlignment="Center"
        Margin="0,0,0,10"
        Grid.Row="0"/>

<!-- Основной блок информации -->
<ScrollViewer VerticalScrollBarVisibility="Hidden"
    HorizontalScrollBarVisibility="Disabled"
    PanningMode="VerticalOnly"
    Grid.Row="1">
    <ListView ItemsSource="{Binding Products}"
        Height="250">
        <ListView.ItemsPanel>
            <ItemsPanelTemplate>
                <StackPanel Orientation="Vertical">

                    </StackPanel>
                </ItemsPanelTemplate>
            </ListView.ItemsPanel>
            <ListView.ItemTemplate>
                <DataTemplate>
                    <Border BorderBrush="Black" BorderThickness="1"
Padding="8" Width="750" Background="#F4E8D3">
                        <StackPanel Orientation="Vertical">
                            <TextBlock Text="{Binding name,
StringFormat='Продукт - {0}'}" FontSize="16"/>
                            <TextBlock Text="{Binding count,
StringFormat='Количество - {0}'}" FontSize="16"/>
                            <TextBlock Text="{Binding date,
StringFormat='Дата - {0:dd.MM.yyyy}'}" FontSize="16"/>
                        </StackPanel>
                    </Border>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </ScrollViewer>
</StackPanel>

<!-- кнопка -->
<Button Content="Назад"
    Style="{StaticResource button_standard}"
    Command="{Binding goBackCommand}"
    HorizontalAlignment="Center"

```

```
        Margin="0,10,0,0"
        Grid.Row="1"/>
    </Grid>
</UserControl>
```

## Листинг 7. Код логики страницы PartnerHistoryView.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WpfApp1.utility;

namespace WpfApp1.models
{
    public class PartnerHisotryModel
    {
        // создаем список продуктов которые продавал выбранный партнер
        public List<ProductHistory> products;

        // создаем айди партнера для которого нужно получить информацию
        private int partnerID;

        public PartnerHisotryModel(int partnerid)
        {
            // получаем айди
            partnerID = partnerid;
            // инициализируем данные
            products = fillProducts();
        }

        // метод получения имени партнера по айди
        public string getPartnerNameByID()
        {
            return DbManager.getPartnerByID(partnerID).PartnerName;
        }

        // метод заполнения продуктов выбранного партнера
        private List<ProductHistory> fillProducts()
        {
            return DbManager.getProductsHistoryByPartnerID(partnerID);
        }
    }
}
```

```
}  
}
```

Также было выполнено дополнительное задание. Код представлен в листинге 6.

Листинг 6. Код выполнения 4 пункта задания.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace WpfAppl.utility  
{  
    // класс для 4 пункта  
    // Метод должен рассчитывать целое количество материала, необходимого  
    // для производства указанного количества продукции, учитывая возможный  
    // брак материала.  
    internal class Zadanie4cs  
    {  
        public static int CalculateRequiredMaterial(double Length, double  
Width, double Height, double productCoefficient, double defectRate, int  
productCount)  
        {  
            // Проверка на корректность данных  
            if (Length <= 0 || Width <= 0 || Height <= 0 ||  
                productCoefficient <= 0 || defectRate < 0 || productCount <=  
0)  
            {  
                return -1;  
            }  
  
            try  
            {  
                // Рассчитываем объем материала на одну единицу продукции  
                double materialPerUnit = Length * Width * Height *  
productCoefficient;  
  
                // Учитываем процент брака  
                double defectMultiplier = 1 + defectRate / 100;
```



```
        double totalMaterial = materialPerUnit * defectMultiplier *  
productCount;  
  
        // Округляем до целого количества  
        return (int)Math.Ceiling(totalMaterial);  
    }  
    catch  
    {  
        // Обработка ошибок  
        return -1;  
    }  
}  
  
}
```