



ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«КОЛЛЕДЖ СВЯЗИ №54»
ИМЕНИ П.М. ВОСТРУХИНА

115172, Москва, ул. Б.Каменщики, д. 7;

тел., факс: (495) 134 1234; e-mail:

spo-54@edu.mos.ru

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по междисциплинарному курсу

МДК 02.01 Технология разработки программного обеспечения

профессионального модуля

ПМ.02 Осуществление интеграции программных модулей

Москва, 2021


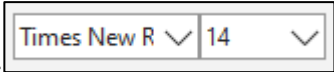
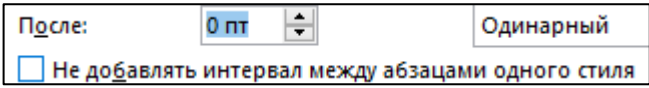
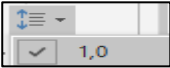
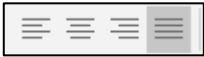

СОДЕРЖАНИЕ

1. Описание предметной области
2. Описание базы данных
3. Проектирование базы данных
4. Проектирование визуального интерфейса
5. Реализация функционала.

Каждый пункт лабораторного практикума должен быть отражен в отчете со скриншотом.

По окончании разработки информационной системы, студент сдает отчет, архив с программой и скриптом, и защищает ее.

Внешний вид отчета:

Word , Times New Roman 14 , интервал между абзацами одного стиля: , между строк , ориентация по ширине , красная строка 1,5 .

1. Описание предметной области



Мебельный магазин Room Interior занимается продажей мебели по всей России

Информационная система разрабатывается для мебельного магазина Room Interior.

В системе планируется организовать возможность работы трех пользователей:

1. Пользователь-покупатель, со следующими функциями:

- 1.1. Просмотр товаров;
- 1.2. Просмотр описания товара;
- 1.3. Редактирование собственного профиля;
- 1.4. Добавление товара в корзину;
- 1.5. Оформление заказа;
- 1.6. Просмотр описания компании.

2. Пользователь-менеджер:

- 2.1. Редактирование карточек товаров;
- 2.2. Просмотр заказов пользователей.

3. Пользователь-администратор:

- 3.1. Добавление/удаление товара;

В информационной системе должна использоваться определенная цветовая схема компании:

| Фон | Акцент | Текст |
|-----------------------------|------------------------|-----------------------|
| #EEEEEE RGB(238,238,238) | #00ADB5 RGB(0,173,181) | #1B2026 RGB(27,32,38) |

Шрифт – Candara.

На всех экранных формах должен присутствовать логотип и название компании.

2. Описание базы данных

Исходя из описанного функционала можно сделать вывод, что в системе необходимо предусмотреть возможность хранения информации о пользователях, при том, разделенных по ролям. Соответственно нужно две сущности – пользователи и роли. Так же, необходимо хранить информацию о продуктах, так как в мебельном магазине по умолчанию предусмотрено разделение товаров по категориям, необходимо еще две сущности – товары и категории товаров.

Так же в задании сказано, что пользователь может просматривать товары и добавлять их в корзину, значит нужна сущность, которая бы хранила данные из корзины.

Сущность пользователи должна содержать ФИО пользователя, контактный телефон, почту для любимого спама, фото, а также логин и пароль.

Сущность роль будет содержать в себе название ролей – администратор, менеджер, пользователь.

В товарах необходимо предусмотреть возможность хранения следующей информации – цена товара, название, материал, описание товара, а также категория товара.

В сущности, категория только номер и название категории.

Также, у пользователя существует возможность добавление товара в корзину, а у менеджера просмотр заказа.


Соответственно, сущность корзина будет содержать следующие атрибуты: номер корзины, номер продукта, номер пользователя, цена корзины.

Сущность заказ будет содержать номер заказа, номер корзины, дата заказа.


3. Проектирование базы данных

3.1. Создадим описанные таблицы.


3.1.1. Сущность пользователь:

| User | | | |
|---|-------------|----------------|-------------------------------------|
| | Имя столбца | Тип данных | Разрешить ... |
|  | IdUser | int | <input type="checkbox"/> |
| | IdRole | int | <input type="checkbox"/> |
| | FirstName | nvarchar(100) | <input type="checkbox"/> |
| | LastName | nvarchar(100) | <input type="checkbox"/> |
| | Patronymic | nvarchar(100) | <input type="checkbox"/> |
| | Phone | nvarchar(10) | <input type="checkbox"/> |
| | Email | nvarchar(50) | <input type="checkbox"/> |
| | PhotoUser | varbinary(MAX) | <input checked="" type="checkbox"/> |
| | Login | nvarchar(50) | <input type="checkbox"/> |
| | Password | nvarchar(50) | <input type="checkbox"/> |


3.1.2. Сущность роль:

| Role | | | |
|---|-------------|---------------|--------------------------|
| | Имя столбца | Тип данных | Разрешить ... |
|  | IdRole | int | <input type="checkbox"/> |
| | NameRole | nvarchar(100) | <input type="checkbox"/> |


3.1.3. Сущность продукт:

| Product | | | |
|---|--------------|----------------|-------------------------------------|
| | Имя столбца | Тип данных | Разрешить ... |
|  | IdProduct | int | <input type="checkbox"/> |
| | IdCategory | int | <input type="checkbox"/> |
| | Price | decimal(12, 2) | <input type="checkbox"/> |
| | PhotoProduct | varbinary(MAX) | <input checked="" type="checkbox"/> |
| | Material | nvarchar(100) | <input type="checkbox"/> |
| | Description | nvarchar(MAX) | <input type="checkbox"/> |
| | NameProduct | nvarchar(50) | <input checked="" type="checkbox"/> |


3.1.4. Сущность категория:

| Category | | | |
|---|--------------|---------------|--------------------------|
| | Имя столбца | Тип данных | Разрешить ... |
|  | IdCategory | int | <input type="checkbox"/> |
| | NameCategory | nvarchar(100) | <input type="checkbox"/> |

3.1.5. Сущность корзина:

| Basket | | | |
|---|-------------|----------------|--------------------------|
| | Имя столбца | Тип данных | Разрешить ... |
|  | IdBasket | int | <input type="checkbox"/> |
| | IdProduct | int | <input type="checkbox"/> |
| | IdUser | int | <input type="checkbox"/> |
| | Price | decimal(12, 2) | <input type="checkbox"/> |

3.1.6. Сущность заказ:

| Order | | | |
|---|-------------|------------|--------------------------|
| | Имя столбца | Тип данных | Разрешить ... |
|  | IdOrder | int | <input type="checkbox"/> |
| | IdBasket | int | <input type="checkbox"/> |
| | Date | date | <input type="checkbox"/> |

3.2. Организуем связи в базе данных:

3.2.1. Связь корзины и заказа:

Таблицы и столбцы

Имя связи:

FK_Order_Basket

Таблица первичного ключа:

Basket

Таблица внешнего ключа:

Order

IdBasket

IdBasket

3.2.2. Связь продукта и корзины:

Таблицы и столбцы

Имя связи:

FK_Basket_Product

Таблица первичного ключа:

Product

Таблица внешнего ключа:

Basket

IdProduct

IdProduct

3.2.3. Связь категории и продукта:

Таблицы и столбцы

Имя связи:

FK_Product_Category

Таблица первичного ключа:

Category

Таблица внешнего ключа:

Product

IdCategory

IdCategory

3.2.4. Связь роли и пользователя:

Таблицы и столбцы

Имя связи:

FK_User_Role

Таблица первичного ключа:

Role

Таблица внешнего ключа:

User

IdRole

IdRole

3.2.5. Связь пользователя и корзины:

Таблицы и столбцы ? X

Имя связи:

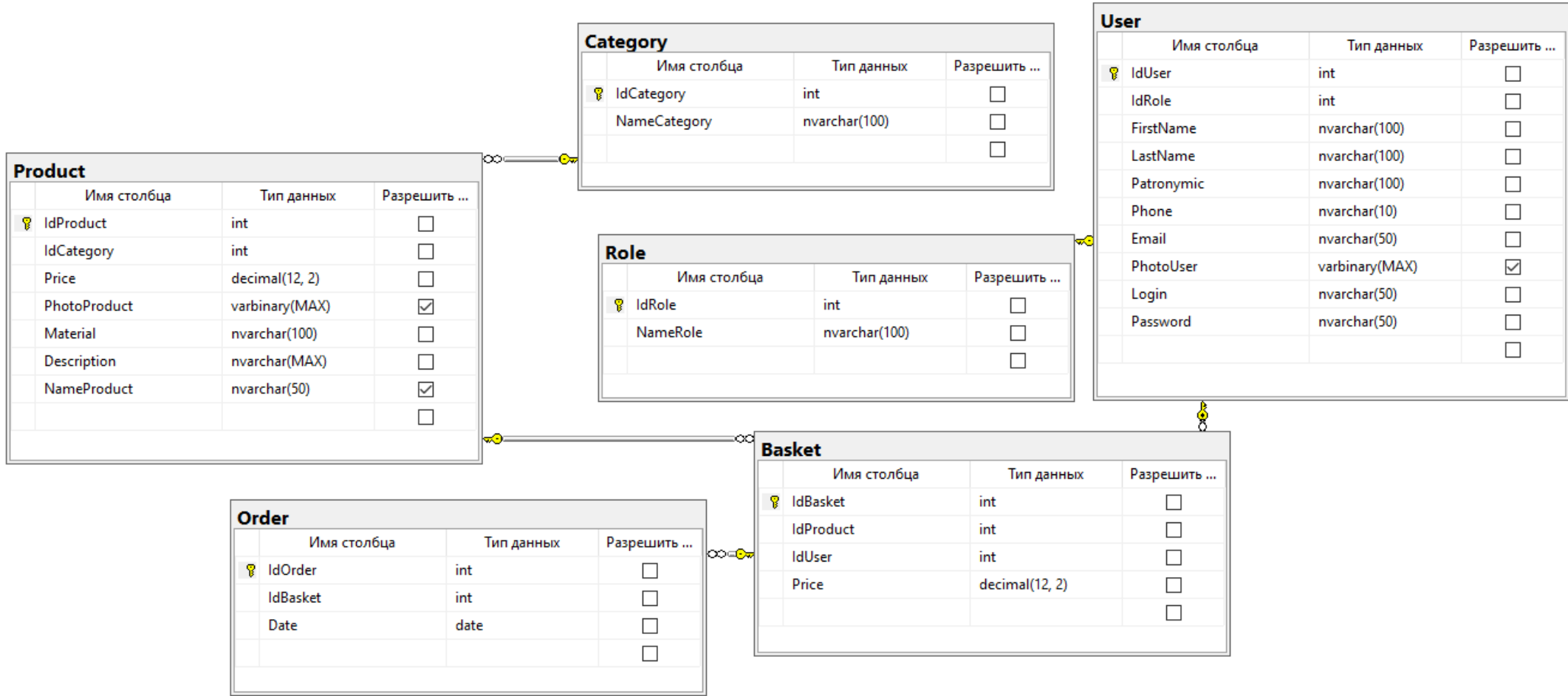
FK_Basket_User

Таблица первичного ключа: Таблица внешнего ключа:

User Basket

IdUser IdUser

4. Проверьте вашу базу по эталону диаграммы БД для разрабатываемой информационной системы:



3.3. Заполните сущности тестовыми данными:

3.3.1 Сущность Роль с тестовыми данными:

| | IdRole | NameRole |
|--|--------|---------------|
| | 1 | Администратор |
| | 2 | Менеджер |
| | 3 | Пользователь |

3.3.2. Сущность Пользователи с тестовыми данными:

| Id... | IdR... | FirstName | LastNa... | Patrony... | Phone | Email | PhotoUser | Login | Password |
|-------|--------|-----------|------------|-------------|----------|------------|-----------|---------|----------|
| 1 | 3 | Иван | Иванов | Иванович | 97777... | ivanii@... | NULL | ivanov | ivanov |
| 2 | 2 | Кира | Кирилова | Кириллов... | 96666... | kirakk... | NULL | manager | manager |
| 3 | 1 | Александр | Алексан... | Александ... | 95555... | alexand... | NULL | admin | admin |

Добавить фото можно несколькими способами, мы будем использовать функцию OpenRowSet в режиме BULK.

Выполните следующий запрос, имейте в виду что путь до файла должен быть соответствующий расположению ваших картинок.

```
UPDATE [User]
SET PhotoUser = (Select * FROM OPENROWSET(BULK N'C:\user-1.jpg', SINGLE_BLOB) AS PhotoUser)
WHERE IdUser = 1
```

Добавьте фото всем остальным вашим пользователям

3.3.3. Сущность Категория с тестовыми данными:

| | IdCategory | NameCategory |
|--|------------|-----------------------------|
| | 1 | Спальня |
| | 2 | Кухня |
| | 3 | Ванная комната |
| | 4 | Гостинная |
| | 5 | Офис |
| | 6 | Домашняя техника |
| | 7 | Офисная техника |
| | 8 | Материалы для строительства |
| | 9 | Материалы для отделки |

3.3.4. Сущность Продукты с тестовыми данными:

| IdProduct | IdCat... | Price | PhotoProduct | Material | Description | NameProduct |
|-----------|----------|-----------|--------------|-----------------|-------------------|----------------|
| 1 | 1 | 15000,00 | NULL | Дерево | Деревянная кр... | Кровать |
| 2 | 1 | 12000,00 | NULL | Железо | Железная кров... | Кровать |
| 3 | 1 | 13000,00 | NULL | Дерево | Деревянная кр... | Кровать |
| 4 | 2 | 8000,00 | NULL | Дерево | Обеденный ст... | Обеденный стол |
| 5 | 2 | 9999,00 | NULL | Железо | Обеденный ст... | Обеденный стол |
| 6 | 2 | 16000,00 | NULL | Гранит | Гранитный об... | Обеденный стол |
| 7 | 3 | 5000,00 | NULL | Гранит | Гранитный ум... | Умывальник |
| 8 | 3 | 6000,00 | NULL | Дерево | Деревянный у... | Умывальник |
| 9 | 3 | 7000,00 | NULL | Железо | Железный ум... | Умывальник |
| 10 | 4 | 16000,00 | NULL | Дерево, стекло | Деревянный га... | Гарнитур |
| 11 | 4 | 18000,00 | NULL | Гранит | Гарнитур для г... | Гарнитур |
| 12 | 4 | 6000,00 | NULL | Дерево | Деревянный д... | Диван |
| 13 | 5 | 12000,00 | NULL | Железо | Пылесос 900B... | Пылесос |
| 14 | 6 | 999000,00 | NULL | Железо, пластик | Игровой комп... | Персональны... |

Фотокарточки товаров также добавьте с помощью SQL – запроса.

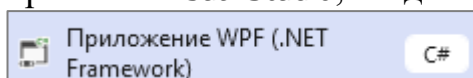
4. Проектирование визуального интерфейса.

4.1. WPF (Windows Presentation Foundation) – аналог WinForms, система для построения клиентских приложений с визуальными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML.

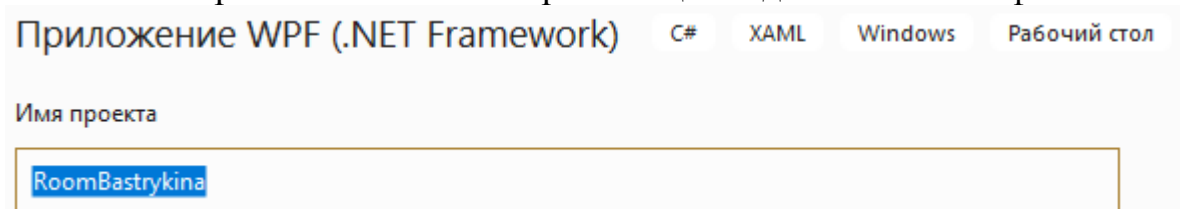
Настольное приложение (Desktop application) – программное обеспечение, предназначенное для работы на персональных компьютерах.

Элемент управления (Control) – элемент, предназначенный для взаимодействия с пользователем или для отображения ему какой-либо информации.

4.2. Запустить среду разработки Visual Studio, создать новый WPF проект:

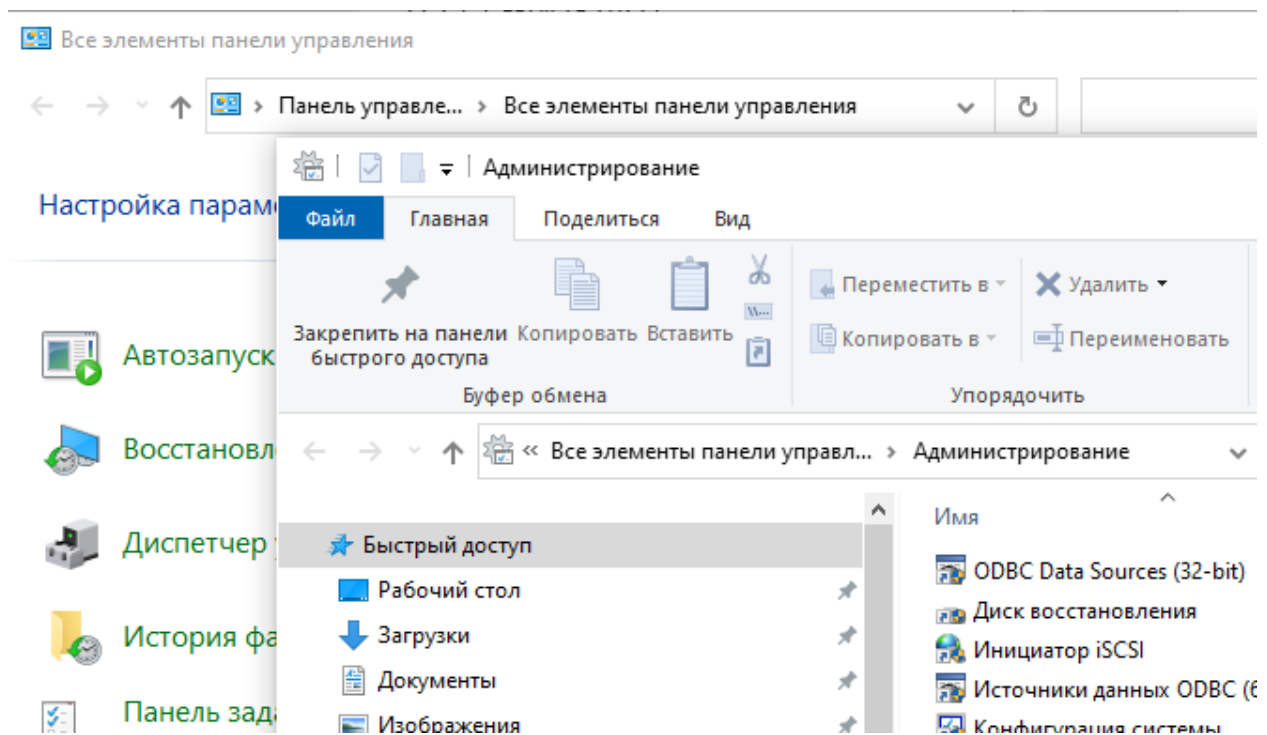


4.3. Назвать проект по названию организации и добавить свою фамилию:

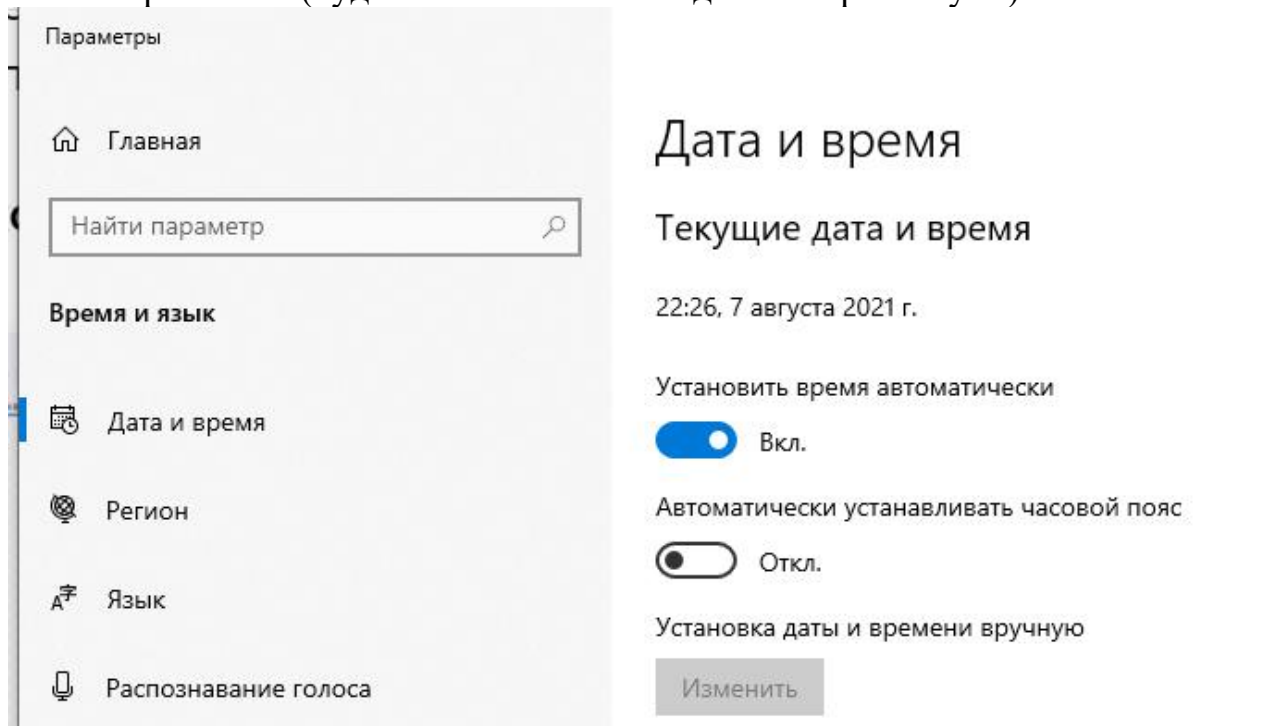


4.4. При разработке интерфейсов разработчик может использовать две модели:

4.4.1. Оконная:

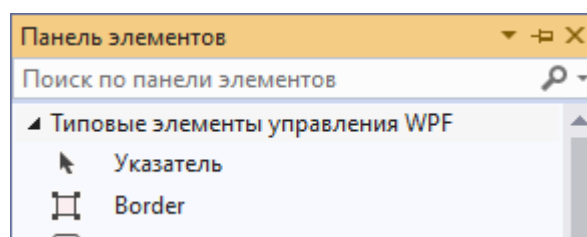


4.4.2. Страничная (будет использоваться в данном практикуме):



4.5. Интерфейс любого приложения состоит из элементов управления, с которыми взаимодействует пользователь, или которые отображают какую-либо информацию.

Доступные элементы управления можно просмотреть в панели элементов (ToolBox):



4.6. Для разметки окна или страницы в приложении можно использовать контейнеры компоновки:

4.6.1. Grid – наиболее мощный и часто используемый контейнер, похож на таблицу. Данный контейнер содержит столбцы и строки, количество которых можно задать вручную. Для определения строк используется свойство RowDefinition:

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="85*" />
    <RowDefinition Height="283*" />
    <RowDefinition Height="51*" />
  </Grid.RowDefinitions>
```

Для определения столбцов используется свойство ColumnDefinition:

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="373*" />
    <ColumnDefinition Width="420*" />
  </Grid.ColumnDefinitions>
```

Для привязки элемента управления к конкретной ячейке необходимо использовать свойства Grid.Column и Grid.Row, причем нумерация строк и столбцов начинается с нуля:

```
<Grid Grid.Row="0" Grid.ColumnSpan="2">
  <Image Source="logo.png" Stretch="Uniform" HorizontalAlignment="Left" />
</Grid>
```

4.6.2. StackPanel – позволяет размещать элементы управления поочередно друг за другом:

```
<StackPanel Grid.Row="1">
  ...
</StackPanel>
```

С помощью свойства Orientation можно выбрать размещение:

```
<StackPanel Grid.Row="1" Orientation="Horizontal">
  ...
</StackPanel>
```

4.7. Разметьте главное окно приложения как показано на скриншоте с помощью контейнера компоновки Grid:

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="400*" />
    <RowDefinition Height="50*" />
  </Grid.RowDefinitions>
```

Выделенный знак * говорит о том, что высота контейнера будет меняться в соответствии с высотой окна.

4.8. Установите минимальную высоту и ширину окна с помощью свойств MinWidth и MinHeight.

```
Title="Room Interior" MinHeight="450" MinWidth="800" Icon="Resources\logo.png">
```

4.9. Исходя из того, что в задании сказано, что каждый интерфейс системы должен иметь логотип и название компании назовите окно интерфейса Room Interior:

```
Title="Room Interior" MinHeight="450" MinWidth="800" Icon="Resources\logo.png">
```

4.10. Внизу справа вы можете заметить окно свойств, установите иконку из папки «Ресурсы»

Свойства

4.11. Нажмите кнопку отладки  в панели сверху, чтобы убедиться, что вы все сделали верно:



4.12. Также, в задании сказано использовать определенные цвета и определенный шрифт.

Перейдите в окно App.xaml  App.xaml

Мы будем использовать несколько постоянных элементов:

1. TextBlock. Для того, чтобы указать стиль определенному элементу после Style необходимо использовать TargetType="Название элемента", в нашем случае TextBlock.

Используя свойство FontFamily установим постоянный шрифт Candara, свойство FontSize позволит тексту всегда быть одного размера, и цвет шрифта установим с помощью свойства Foreground:

```
<Style TargetType="TextBlock">
  <Setter Property="FontFamily" Value="Candara"/>
  <Setter Property="FontSize" Value="14"/>
  <Setter Property="Foreground" Value="#1B2026"/>
</Style>
```

2. TextBox. Этому элементу помимо одинакового текста необходимо указать ширину и высоту, для акцента на элемент установим рамку элемента с помощью свойства BorderBrush:

```
<Style TargetType="TextBox">
  <Setter Property="FontFamily" Value="Candara"/>
  <Setter Property="FontSize" Value="14"/>
  <Setter Property="Foreground" Value="#1B2026"/>
  <Setter Property="Width" Value="150"/>
  <Setter Property="Height" Value="25"/>
  <Setter Property="Margin" Value="5"/>
  <Setter Property="BorderBrush" Value="#00ADB5"/>
</Style>
```

3. ComboBox:

```
<Style TargetType="ComboBox">
  <Setter Property="FontFamily" Value="Candara"/>
  <Setter Property="FontSize" Value="14"/>
  <Setter Property="Foreground" Value="#1B2026"/>
  <Setter Property="Width" Value="150"/>
  <Setter Property="Height" Value="25"/>
  <Setter Property="Margin" Value="5"/>
</Style>
```

4. Button. Нам понадобится две разных кнопки. 1 стиль будет стандартным для всех кнопок, второй для стрелочек перелистывания. Чтобы для каждого элемента указывать определенный стиль необходимо прописать x:Key:

Стиль для стандартных кнопок:

```
<Style TargetType="Button" x:Key="ButtonOne">
  <Setter Property="FontFamily" Value="Candara"/>
  <Setter Property="FontSize" Value="14"/>
  <Setter Property="Foreground" Value="#1B2026"/>
  <Setter Property="Width" Value="150"/>
  <Setter Property="Height" Value="25"/>
  <Setter Property="Margin" Value="5"/>
  <Setter Property="Background" Value="#00ADB5"/>
  <Setter Property="BorderBrush" Value="#00ADB5"/>
</Style>
```

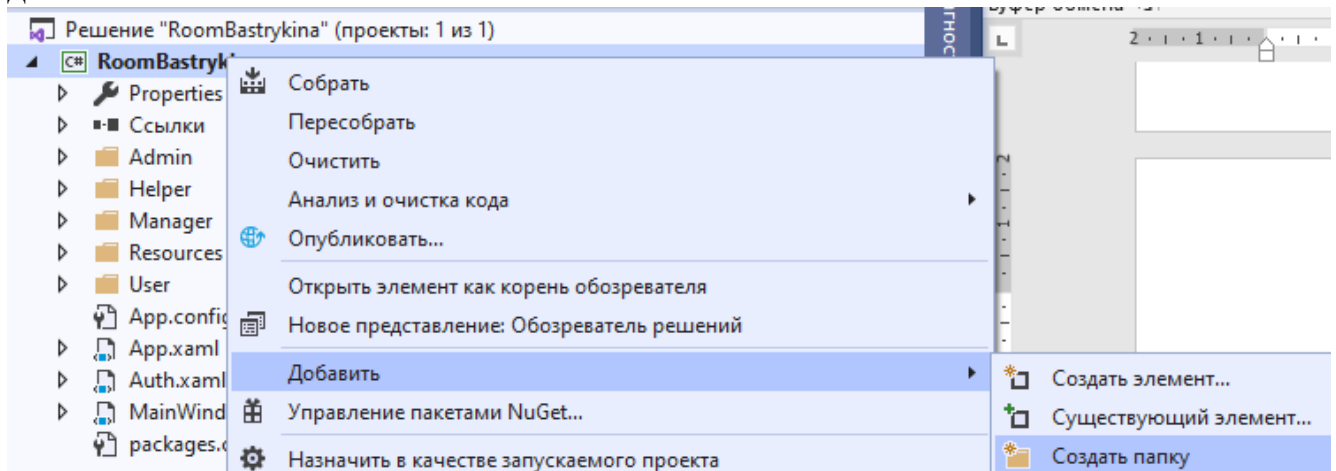
Стиль для стрелочек:

```
<Style TargetType="Button" x:Key="ButtonTwo">
    <Setter Property="FontFamily" Value="Candara"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Foreground" Value="#182026"/>
    <Setter Property="Width" Value="25"/>
    <Setter Property="Height" Value="25"/>
    <Setter Property="Margin" Value="5"/>
    <Setter Property="Background" Value="#00ADB5"/>
    <Setter Property="BorderBrush" Value="#00ADB5"/>
</Style>
```

5. PasswordBox опишите также как и TextBox, можно было бы использовать последний. Но правильным будет скрыть пароль от посторонних глаз.

```
<Style TargetType="PasswordBox">
    <Setter Property="FontFamily" Value="Candara"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Foreground" Value="#182026"/>
    <Setter Property="Width" Value="150"/>
    <Setter Property="Height" Value="25"/>
    <Setter Property="Margin" Value="5"/>
    <Setter Property="BorderBrush" Value="#00ADB5"/>
</Style>
```

4.13. Для навигации по проекту, и в случае, если в вашем проекте в дальнейшем будут заниматься другие программисты, необходимо разграничить сам проект, для этого нажмите ПКМ на ваш проект в обозревателе решений и добавьте папки:

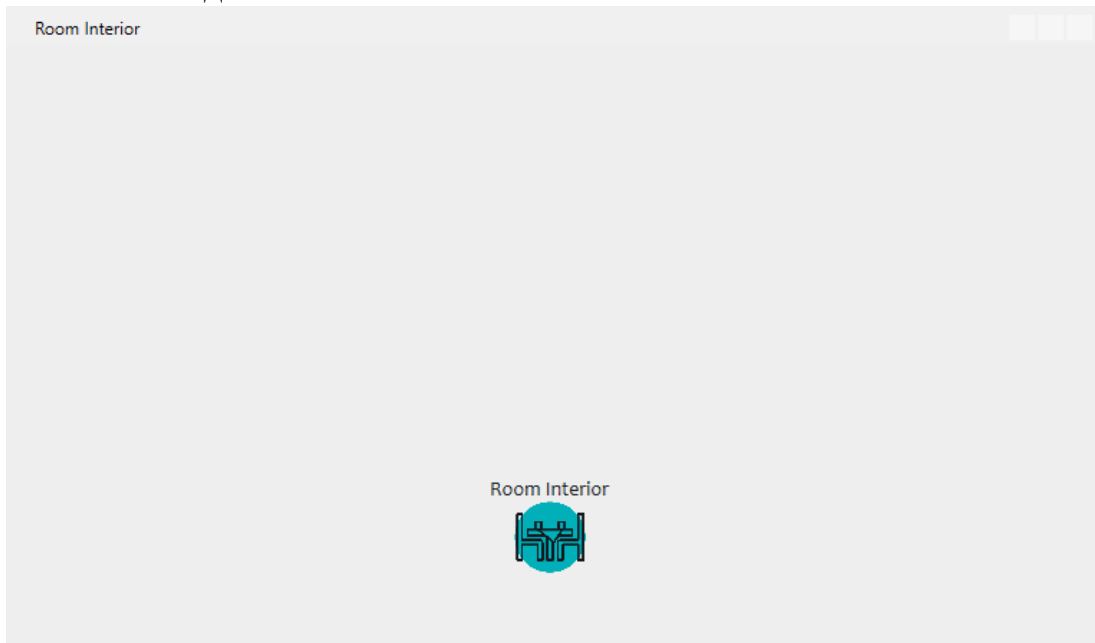


Так как в системе работают 3 роли пользователей нам понадобятся три папки разграничения функционала, папка с ресурсами, куда мы положим логотип и папка с моделью базы данных.

4.14. Опишем поочередно все интерфейсы системы:

4.1.14 MainWindow.xaml – это главное окно системы, которое будет запускаться при запуске приложения. В связи с ограничением по ролям, на это окно будет запускаться страница авторизации. Соответственно, для окна никаких особенный функционал не понадобится, кроме расположения логотипа, названия и элемента Frame, который позволит загрузить на данное окно страницу авторизации:

Внешний вид:



В предыдущих пунктах практикума вы уже узнали, как задать максимальную и минимальную величину окна системы, назвать его и установить иконку из ресурсов. Дополните содержимое раскраской фона с помощью свойства `Background` и цветом фона.

```
<Window x:Class="RoomBastrykina.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        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"
        xmlns:local="clr-namespace:RoomBastrykina"
        mc:Ignorable="d"
        Title="Room Interior" MinWidth="800" MaxWidth="800"
        MinHeight="450" MaxHeight="400"
        Icon="Resources\logo.png" Background="#EEEEEE">
```

Далее необходимо разметить окно системы с помощью элемента `Grid`:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="300"/>
        <RowDefinition Height="150"/>
    </Grid.RowDefinitions>
```

В нулевую по индексу строку (воспользуйтесь `Grid.Row`, для указания строки) вставьте элемент `Frame`, растягивать по ширине с помощью `Stretch` не обязательно. Дополнительно присвойте имя, по которому мы будем работать в коде с фреймом с помощью `x:Name`:

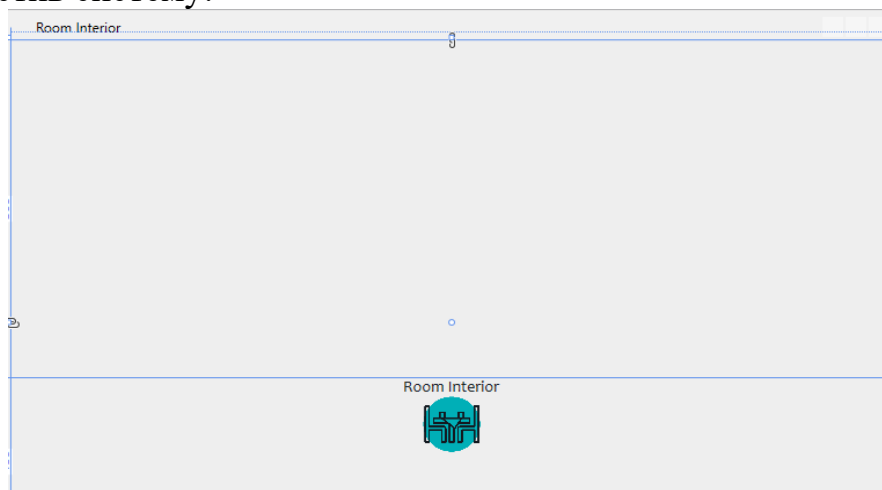
```
<Grid Grid.Row="0" HorizontalAlignment="Stretch">
    <Frame x:Name="frmMain"/>
</Grid>
```

В первой строке расположите элемент текст блока с названием фирмы и логотип:

```
<StackPanel Grid.Row="1" Orientation="Vertical">
    <TextBlock Text="Room Interior" HorizontalAlignment="Center" VerticalAlignment="Bottom"/>
    <Image Source="Resources\logo.png" Width="50"/>
</StackPanel>
```

`StackPanel` позволит расположить элементы вертикально с помощью свойства `Orientation`.

Проверяйте список ошибок, закрытие тэгов и сверьте сделанную работу с эталоном, запустив систему:



4.15. Создайте страницу авторизации `Auth.xaml` используя 4 TextBlock'a, 3 TextBox'a и 2 Button'a:

Разметим страницу авторизации с помощью грида:

```
<Grid.RowDefinitions>
  <RowDefinition Height="50"/>
  <RowDefinition Height="50"/>
  <RowDefinition Height="50"/>
  <RowDefinition Height="50"/>
  <RowDefinition Height="50"/>
</Grid.RowDefinitions>
```

В нулевой строке укажите название страницы:

```
<Grid Grid.Row="0" HorizontalAlignment="Center" VerticalAlignment="Center">
  <TextBlock Text="Авторизация"/>
</Grid>
```

Используйте 4 StackPanel чтобы красиво разметить страницу:

```
<StackPanel Grid.Row="1" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">
  <TextBlock Text="Введите логин:" />
  <TextBox x:Name="txbLogin"/>
</StackPanel>

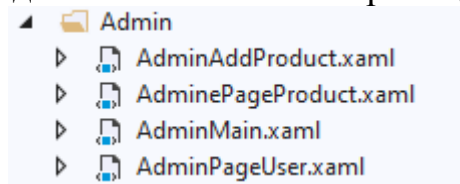
<StackPanel Grid.Row="2" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">
  <TextBlock Text="Введите пароль:" />
  <PasswordBox x:Name="psbPassword"/>
</StackPanel>

<StackPanel Grid.Row="3" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">
  <TextBlock Text="Повторите пароль:" />
  <PasswordBox x:Name="psbPasswordRepeat"/>
</StackPanel>
```

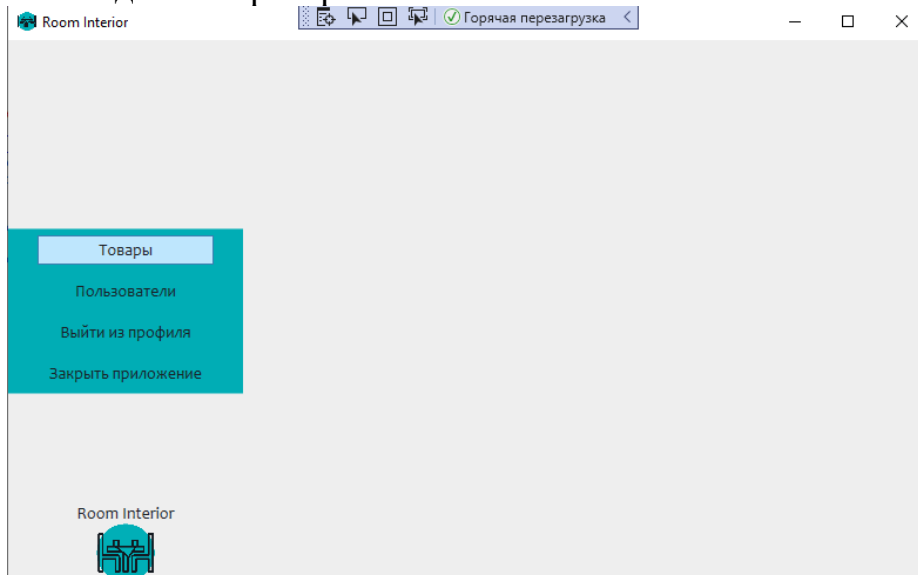
Для того чтобы кнопка определять стиль мы использовали `x:Key`, следовательно, с помощью свойства `Style` мы можем указать определенный стиль:


```
<StackPanel Grid.Row="4" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">
    <Button x:Name="btnExit" Content="Выйти" Click="btnExit_Click" Style="{StaticResource ButtonOne}"/>
    <Button x:Name="btnEnter" Content="Войти" Click="btnEnter_Click" Style="{StaticResource ButtonOne}"/>
</StackPanel>
```

В папке Admin нам понадобится 2 окна и 2 страницы:



4.16. Окно администратора:



Разметим окно с помощью колонок элемента Grid:

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="200"/>
        <ColumnDefinition Width="600"/>
    </Grid.ColumnDefinitions>
```

Воспользуемся StackPanel, чтобы элемент располагался в левом столбце грида воспользуйтесь свойством Grid.Column что бы разместить четыре кнопки:

```
<StackPanel Grid.Column="0" Orientation="Vertical" Background="#00ADB5" VerticalAlignment="Center">
    <Button x:Name="btnListProduct" Content="Товары" Click="btnListProduct_Click" Style="{StaticResource ButtonOne}"/>
    <Button x:Name="btnListUser" Content="Пользователи" Click="btnListUser_Click" Style="{StaticResource ButtonOne}"/>
    <Button x:Name="btnListExitProfile" Content="Выйти из профиля" Click="btnListExitProfile_Click" Style="{StaticResource ButtonOne}"/>
    <Button x:Name="btnListCloseApp" Content="Закреть приложение" Click="btnListCloseApp_Click" Style="{StaticResource ButtonOne}"/>
</StackPanel>
```

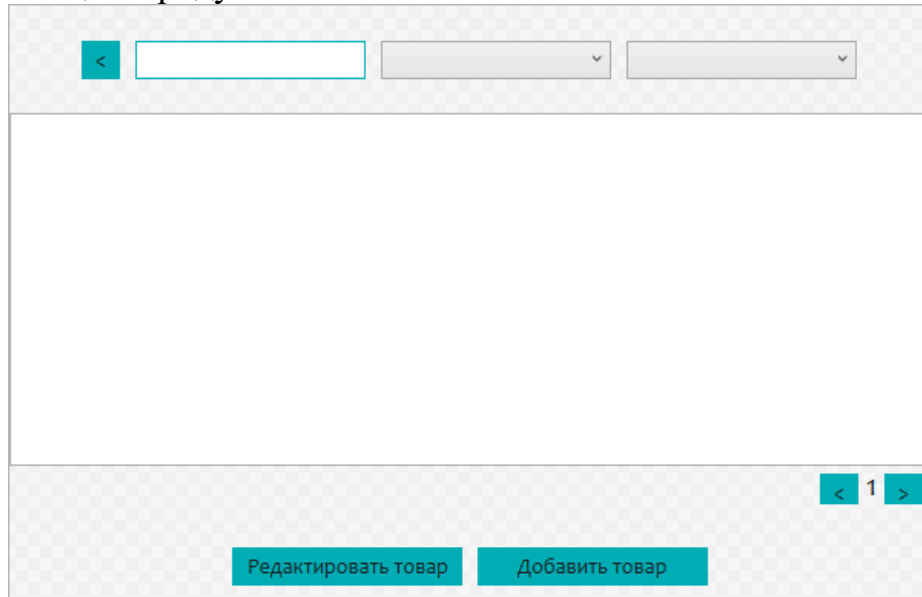
Внизу расположите название фирмы и логотип:

```
<StackPanel Grid.Column="0" Orientation="Vertical" VerticalAlignment="Bottom">
    <TextBlock Text="Room Interior" HorizontalAlignment="Center"/>
    <Image Source="/Resources/logo.png" Width="50" VerticalAlignment="Bottom"/>
</StackPanel>
```

В правом столбике расположите элемент Frame:


```
<Grid Grid.Column="1">
    <Frame x:Name="frmMainAdmin" Margin="0,0,30,0"/>
</Grid>
```

4.17. Страница «Продукты»:



Нам нужны 4 строки:

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="70"/>
    <RowDefinition Height="230"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="70"/>
  </Grid.RowDefinitions>
```

В нулевой строке будет кнопка возвращения на домашнее окно администратора, строка поиска, и 2 ComboBox'а для сортировки и фильтрации. Для кнопки используйте второй стиль, как мы и писали, он для стрелочек. Обратите внимание как написана стрелочка <; не описывайте SelectionChanged самостоятельно, воспользуйтесь энтером  <Новый обработчик событий>:

```
<WrapPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">
  <Button x:Name="btnBackFrm" Content="&lt;" Style="{StaticResource ButtonTwo}" Click="btnBackFrm_Click"/>
  <TextBox x:Name="txbSearch" TextChanged="txbSearch_TextChanged"/>
  <ComboBox x:Name="cmbFiltrationCategory" SelectedIndex="0" SelectionChanged="cmbFiltrationCategory_SelectionChanged"/>
  <ComboBox x:Name="cmbSortingPrice" SelectedIndex="0" SelectionChanged="cmbSortingPrice_SelectionChanged"/>
</WrapPanel>
```

Во второй строке будут располагаться кнопки перехода по страницам, событие Click также назначайте автоматически:

```
<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Right" VerticalAlignment="Center">
  <Button x:Name="btnLastPage" Content="&lt;" Style="{StaticResource ButtonTwo}" Click="btnLastPage_Click"/>
  <TextBlock x:Name="tbckPage" Text="1" FontSize="20"/>
  <Button x:Name="btnNextPage" Content="&gt;" Style="{StaticResource ButtonTwo}" Click="btnNextPage_Click"/>
</StackPanel>
```

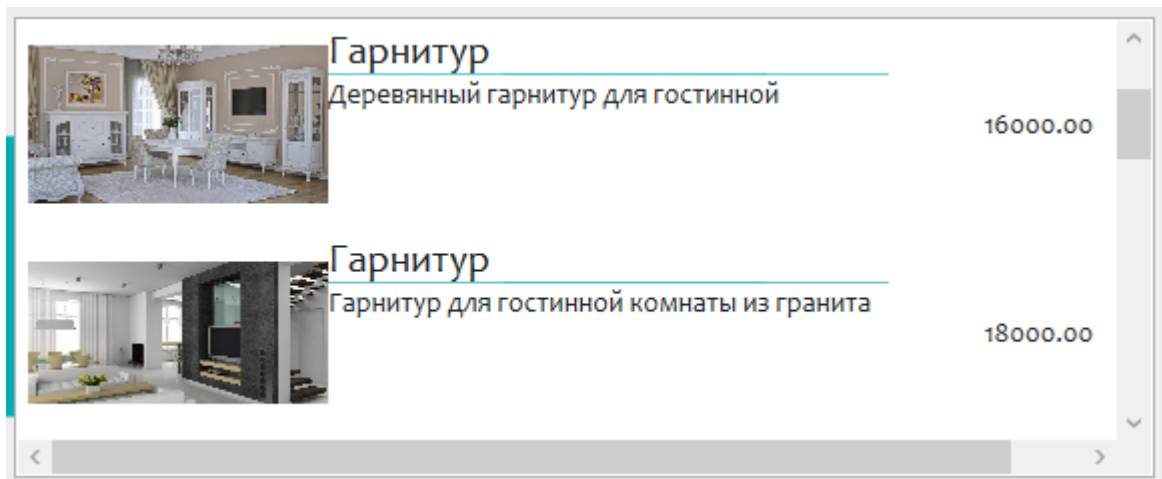
В третьей строке расположите две кнопки редактирования и добавления товара:

```
<StackPanel Grid.Row="3" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">
  <Button x:Name="btnEditProduct" Content="Редактировать товар" Style="{StaticResource ButtonOne}" Click="btnEditProduct_Click"/>
  <Button x:Name="btnAddProduct" Content="Добавить товар" Style="{StaticResource ButtonOne}" Click="btnAddProduct_Click"/>
</StackPanel>
```

Теперь воспользуемся элементом ListView:

```
<ListView Grid.Row="1" x:Name="LvProduct" SelectionChanged="LvProduct_SelectionChanged">
  <ListView.ItemTemplate>
    <DataTemplate>
      <!-- Content of the DataTemplate -->
    </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

С помощью известных вам элементов управления расположите данные из базы в соответствии с макетом:



ХAML-код:

```
<Grid Height="100" Width="580">
  <Grid.RowDefinitions>
    <RowDefinition Height="100"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="150"/>
    <ColumnDefinition Width="280"/>
    <ColumnDefinition Width="150"/>
  </Grid.ColumnDefinitions>

  <Image Grid.Column="0" Source="{Binding PhotoProduct}" HorizontalAlignment="Stretch"/>

  <StackPanel Grid.Column="1" Orientation="Vertical">
    <TextBlock x:Name="tbckNameProduct" Text="{Binding NameProduct}" FontSize="20"/>
    <Line X1="350" Y1="1" Stroke="#00ADB5"/>
    <TextBlock x:Name="tbckDescription" Text="{Binding Description}" FontSize="14"/>
  </StackPanel>

  <StackPanel Grid.Column="2" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">
    <TextBlock x:Name="tbckPrice" Text="{Binding Price}" FontSize="14"/>
  </StackPanel>
</Grid>
```

4.18. Страница «Пользователь»:

The screenshot shows a mobile application interface for the 'User' page. At the top, there is a back arrow. Below it is a table with five columns: 'Фамилия' (Family Name), 'Имя' (First Name), 'Отчество' (Patronymic), 'Номер телефона' (Phone Number), and 'Электронная почта' (Email). The table is currently empty. Below the table is a large empty area, and at the bottom, there is a light blue bar.

Вы уже работали с данным видом ListView. Ниже код, что бы вы могли вспомнить как размечается страница:

ХАМЛ-код:

```
<Grid.RowDefinitions>
    <RowDefinition Height="70"/>
    <RowDefinition Height="330"/>
</Grid.RowDefinitions>

<WrapPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Left" VerticalAlignment="Center">
    <Button x:Name="btnBackFrm" Content="&lt;" Style="{StaticResource ButtonTwo}" Click="btnBackFrm_Click"/>
</WrapPanel>

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Right" VerticalAlignment="Center"/>

<ListView Grid.Row="1" x:Name="LvUser" SelectionChanged="LvUser_SelectionChanged">
    <ListView.View>
        <GridView>
            <GridViewColumn Header="Фамилия" Width="100" DisplayMemberBinding="{Binding LastName}"/>
            <GridViewColumn Header="Имя" Width="100" DisplayMemberBinding="{Binding FirstName}"/>
            <GridViewColumn Header="Отчество" Width="100" DisplayMemberBinding="{Binding Patronymic}"/>
            <GridViewColumn Header="Номер телефона" Width="100" DisplayMemberBinding="{Binding Phone}"/>
            <GridViewColumn Header="Электронная почта" Width="150" DisplayMemberBinding="{Binding Email}"/>
        </GridView>
    </ListView.View>
</ListView>
```

Для того чтобы шапка таблицы имела цвет, указанный в задании, добавьте следующий код в App.xaml:

```
<Style TargetType="GridViewColumnHeader">
    <Setter Property="FontFamily" Value="Candara"/>
    <Setter Property="Foreground" Value="#1B2026"/>
    <Setter Property="BorderBrush" Value="#00ADB5"/>
    <Setter Property="Background" Value="#00ADB5"/>
</Style>
```

4.19. Окно добавления продукта:

Room Interior

Введите название товара:

Выберите категорию товара:

▼

Введите цену

Выберите изображение:


Обзор

Введите описание:

Введите материал товара:

Закреть окно Сохранить

Room Interior



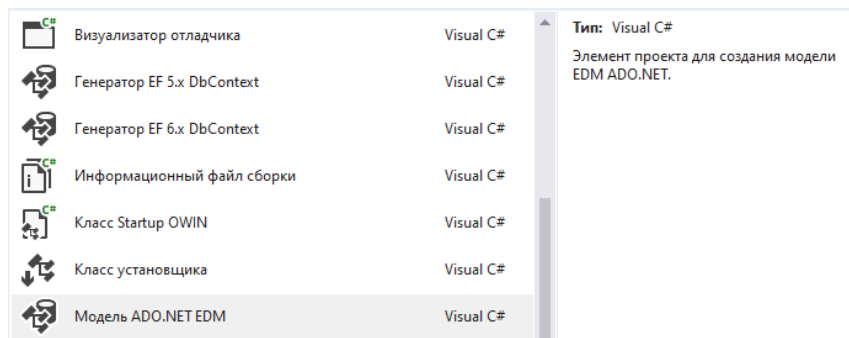
Не забудьте изменить фон окна.

5.1. Реализация функционала.

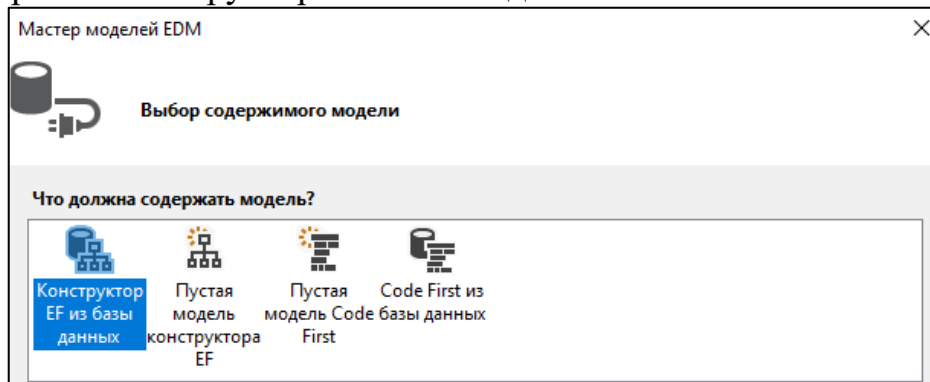
Для реализации функционала нам необходимо подключить базу:



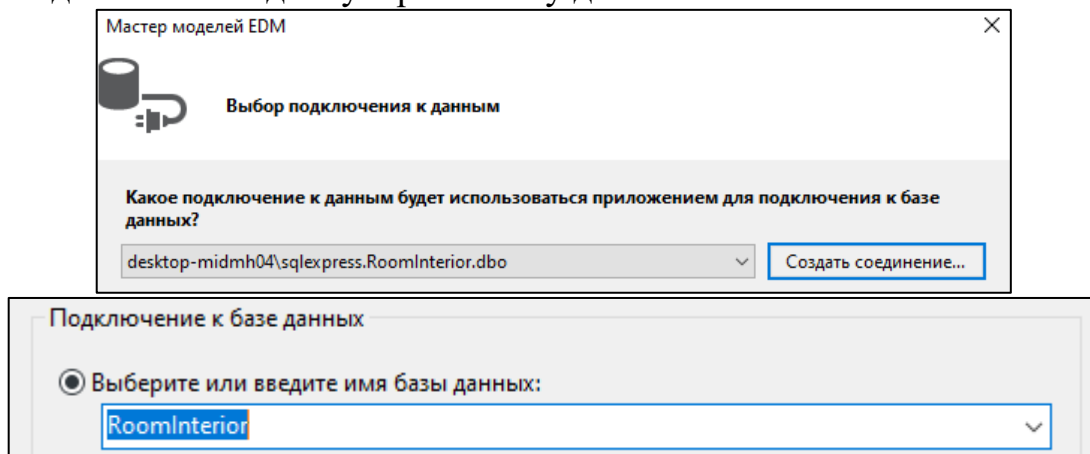
Выберите Model ADO.NET EDM



Выберите «Конструктор EF из базы данных»



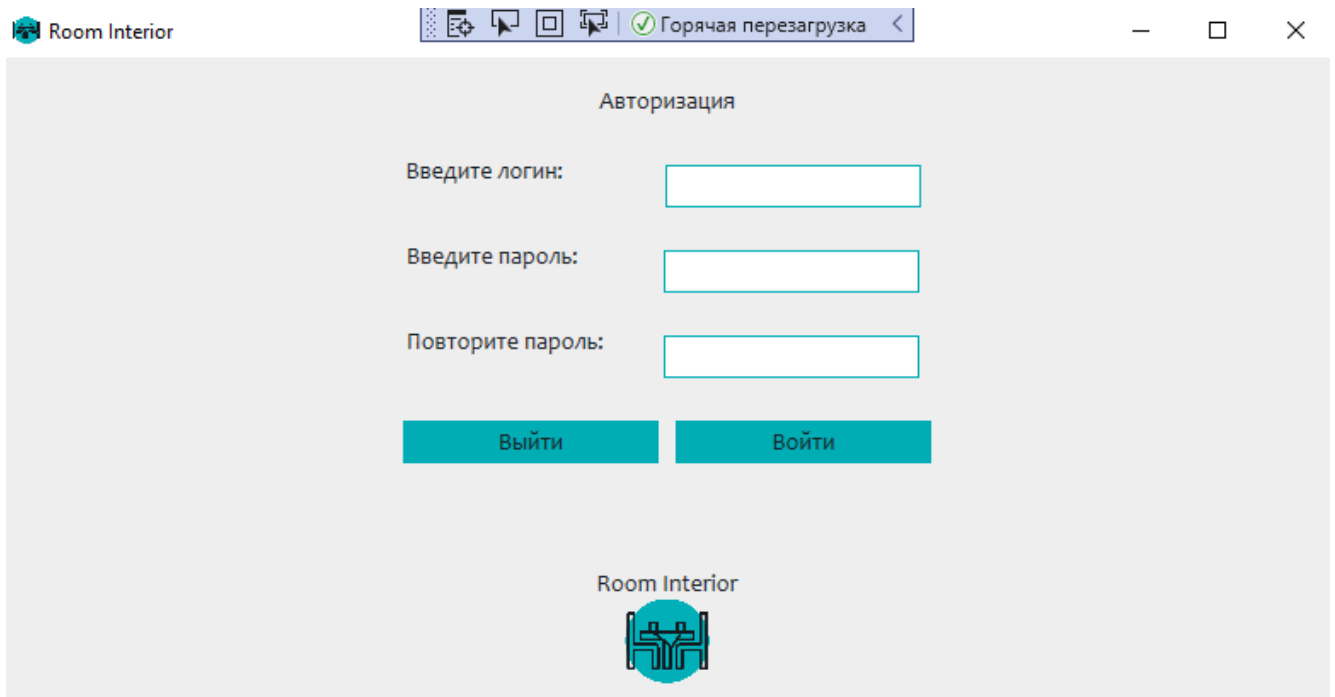
Подключите созданную ранее базу данных:



5.2. Для того, чтобы страница авторизации появилась на окне, воспользуйтесь следующим кодом:

```
public MainWindow()
{
    InitializeComponent();
    frmMain.Navigate(new Auth());
}
```

Проверьте работоспособность запустив проект:



5.3. Опишите функционал кнопки выйти с помощью Application, он позволит полностью закрыть приложение из фрейма:

```
private void btnExit_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}
```

5.4. Учитывая тот факт, что авторизация должна работать с базой, воспользуемся стандартным кодом сверки логина и пароля с базой:

```
var userModel = DB.entities.User.FirstOrDefault
(i => i.Login == txtLogin.Text && i.Password == psbPassword.Password);
```

5.5. Теперь пропишем условия, при котором, в зависимости от результата обработки нам открываются окна системы.

Если пароль и повтор пароля совпадет, система пустит нас на окно в зависимости от роли:

```
if (psbPassword.Password == psbPasswordRepeat.Password)
{
    if (userModel == null)
    {
        MessageBox.Show("Пользователь не найден, повторите попытку", "Пользователь не найден", MessageBoxButton.OK);
    }

    else if (userModel.IdRole == 1)
    {
        AdminMain aM = new AdminMain();
        aM.Show();
        Application.Current.MainWindow.Close();
    }
}
```

MessageBox позволит оповестить пользователя если пользователь не найден.

Если пароли не совпадают покажите пользователю сообщение с просьбой повторить ввод пароля:

```
else
{
    MessageBox.Show("Пароль не совпадает, повторите попытку",
        "Ошибка ввода пароля", MessageBoxButton.OK);
}
```

5.6. Попробуйте авторизоваться в системе, логин администратора – admin, пароль администратора – admin.

Если вы все сделали верно, система закроет текущее окно и откроет панель администратора.

5.7. Опишем функционал окна администратора.

5.7.1. Кнопка «Товар»

```
private void btnListProduct_Click(object sender, RoutedEventArgs e)
{
    frmMainAdmin.Navigate(new AdminPageProduct());
}
```

5.7.2. Кнопка «Пользователь»:

```
private void btnListUser_Click(object sender, RoutedEventArgs e)
{
    frmMainAdmin.Navigate(new AdminPageUser());
}
```

5.7.3. Кнопка «Выйти из профиля» должна отправить вас обратно на окно MainWindow с фреймом авторизации:

```
private void btnListExitProfile_Click(object sender, RoutedEventArgs e)
{
    MainWindow mw = new MainWindow();
    mw.Show();
    this.Close();
}
```

5.7.4. Как описать кнопку закрытия приложения вы уже знаете:

```
private void btnListCloseApp_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}
```

5.8. Функционал страницы товаров:

5.8.1. Чтобы в ListView выводило список товаров в соответствии с нашим описанием необходимо также обратиться к базе:

```
listProduct = DB.entities.Product.ToList();
```

5.8.2. TextBox с функцией поиска опишите следующим образом:

```
listProduct = listProduct.Where(i => i.NameProduct.ToLower().Contains(txbSearch.Text.ToLower())).ToList();
```

Чтобы данный код сработал, необходимо переопределить источник данных для ListView:

```
// переопределение источника данных для ListView
LvProduct.ItemsSource = listProduct;
```

5.8.3. Реализация функции фильтрации:

```
var selectFiltr = cmbFiltrationCategory.SelectedIndex;

if (selectFiltr != 0)
{
    listProduct = listProduct.Where(i => i.IdCategory == selectFiltr).ToList();
}
```

Для вывода категорий фильтрации в public AdminPageProduct() добавьте код:


```
var category = DB.entities.Category.ToList();
foreach (var i in category)
{
    listFiltr.Add(i.NameCategory);
}
```

```
listFiltr.Insert(0, "Все категории");
cmbFiltrationCategory.ItemsSource = listFiltr;
cmbFiltrationCategory.SelectedIndex = 0;
```

```
cmbSortingPrice.ItemsSource = listSort;
cmbSortingPrice.SelectedIndex = 0;
```

Он же добавит пункты в комбобокс сортировки.

5.8.4. Реализация функции сортировки:

Добавьте пункты сортировки в проект:

```
List<Product> listProduct = new List<Product>();
private int numPage = 0;
```

```
List<string> listSort = new List<string>()
{
    "Наименование (по возрастанию)",
    "Наименование (по убыванию)",
    "Стоимость(по возрастанию)",
    "Стоимость(по убыванию)"
};
```

```
List<string> listFiltr = new List<string>();
```

А также объявите переменные listSort и listFiltr.

```
//Сортировка
var selectSort = cmbSortingPrice.SelectedIndex;
switch (selectSort)
{
    case 0:
        listProduct = listProduct.OrderBy(i => i.NameProduct).ToList(); // по возрастанию
        break;

    case 1:
        listProduct = listProduct.OrderByDescending(i => i.NameProduct).ToList(); // по убыванию
        break;

    case 2:
        listProduct = listProduct.OrderBy(i => i.Price).ToList();
        break;

    case 3:
        listProduct = listProduct.OrderByDescending(i => i.Price).ToList();
        break;

    default:
        listProduct = listProduct.OrderBy(i => i.IdProduct).ToList();
        break;
}
```

5.8.5. Реализуйте функционал перехода по страницам:

```
private void btnNextPage_Click(object sender, RoutedEventArgs e)
{
    if (listProduct.Count > 0)
    {
        numPage++;
        tbckPage.Text = (numPage + 1).ToString();
    }
    Filtr();
}
```

ссылка: 1

```
private void btnLastPage_Click(object sender, RoutedEventArgs e)
{
    if (numPage > 0)
    {
        numPage--;
        tbckPage.Text = (numPage + 1).ToString();
    }
    Filtr();
}
```

5.8.6. Полный код описания функционала для страницы с продуктами. Не копируйте его:

```
using RoomBastrykina.Helper;
```

```
namespace RoomBastrykina.Admin
{
```

```
public partial class AdminPageProduct : Page
{
```

```
List<Product> listProduct = new List<Product>();
private int numPage = 0;
```

```
List<string> listSort = new List<string>()
{
    "Наименование (по возрастанию)",
    "Наименование (по убыванию)",
    "Стоимость(по возрастанию)",
    "Стоимость(по убыванию)"
};
```

```
List<string> listFiltr = new List<string>();
```

```
void Filtr()
{
    listProduct = DB.entities.Product.ToList();
```

```
//Сортировка
```

```
var selectSort = cmbSortingPrice.SelectedIndex;
```

```
switch (selectSort)
```

```
{
    case 0:
        listProduct = listProduct.OrderBy(i => i.NameProduct).ToList(); // по возрастанию
        break;
```

```
    case 1:
        listProduct = listProduct.OrderByDescending(i => i.NameProduct).ToList(); // по убыванию
        break;
```

```
    case 2:
        listProduct = listProduct.OrderBy(i => i.Price).ToList();
        break;
```

```

        case 3:
            listProduct = listProduct.OrderByDescending(i => i.Price).ToList();
            break;

        default:
            listProduct = listProduct.OrderBy(i => i.IdProduct).ToList();
            break;
    }

    //Фильтрация
    var selectFiltr = cmbFiltrationCategory.SelectedIndex;

    if (selectFiltr != 0)
    {
        listProduct = listProduct.Where(i => i.IdCategory == selectFiltr).ToList();
    }

    // Вывод по страницам
    listProduct = listProduct.Skip(10 * numPage).Take(10).ToList();

    // переопределение источника данных для ListView
    LvProduct.ItemsSource = listProduct;
}

public AdminPageProduct()
{
    InitializeComponent();
    LvProduct.ItemsSource = DB.entities.Product.ToList();

    var category = DB.entities.Category.ToList();
    foreach (var i in category)
    {
        listFiltr.Add(i.NameCategory);
    }

    listFiltr.Insert(0, "Все категории");
    cmbFiltrationCategory.ItemsSource = listFiltr;
    cmbFiltrationCategory.SelectedIndex = 0;

    cmbSortingPrice.ItemsSource = listSort;
    cmbSortingPrice.SelectedIndex = 0;
}

private void btnBackFrm_Click(object sender, RoutedEventArgs e)
{
    Content = null;
}

private void txbSearch_TextChanged(object sender, TextChangedEventArgs e)
{
    listProduct = listProduct.Where(i => i.NameProduct.ToLower().Contains(txbSearch.Text.ToLower())).ToList();
}

private void cmbFiltrationCategory_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    Filtr();
}

private void cmbSortingPrice_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    Filtr();
}

```

```

private void LvProduct_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    Filtr();
}

private void btnNextPage_Click(object sender, RoutedEventArgs e)
{
    if (listProduct.Count > 0)
    {
        numPage++;
        tbckPage.Text = (numPage + 1).ToString();
    }
    Filtr();
}

private void btnLastPage_Click(object sender, RoutedEventArgs e)
{
    if (numPage > 0)
    {
        numPage--;
        tbckPage.Text = (numPage + 1).ToString();
    }
    Filtr();
}

private void btnEditProduct_Click(object sender, RoutedEventArgs e)
{
}

private void btnAddProduct_Click(object sender, RoutedEventArgs e)
{
    AdminAddProduct addProd = new AdminAddProduct();
    addProd.Show();
}

```

5.9. Реализуйте функционал со списком пользователей:

```

InitializeComponent();
LvUser.ItemsSource = DB.entities.User.ToList();

```

5.9.1. Чтобы закрыть фрейм пользователя пропишите:

```

private void btnBackFrm_Click(object sender, RoutedEventArgs e)
{
    Content = null;
}

```

5.10. Страница добавления товара:

5.10.1. Основной функционал страницы будет прописан на кнопке «Сохранить». Перед сохранением важно уточнить у пользователя действительно ли он хочет сохранить введенные данные:

```

var result = MessageBox.Show("Вы хотите добавить материал?", "Подтверждение", MessageBoxButton.YesNo);

Product prodAdd = new Product();
prodAdd.NameProduct = txtNameProduct.Text;
prodAdd.IdCategory = cmbCategory.SelectedIndex + 1;
prodAdd.Price = Convert.ToDecimal(txtPrice.Text);

```

Допишите код самостоятельно.

5.10.2. Чтобы загрузить фото нам понадобится OpenFileDialog:

```
private void btnImageAdd_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog openFD = new OpenFileDialog();
    if (openFD.ShowDialog() == true)
    {
        string fileName = openFD.FileName;
        imgImageProduct.Source = new BitmapImage(new Uri(openFD.FileName));
        PathProduct = openFD.FileName;
    }
}
```

5.10.3. Сохраните изменения в базе данных:

```
DB.entities.Product.Add(prodAdd);
DB.entities.SaveChanges();
MessageBox.Show("Товар успешно добавлен", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);
```

Самостоятельная работа:

Используя всю проделанную работу допишите информационную систему самостоятельно в соответствии с заданием.