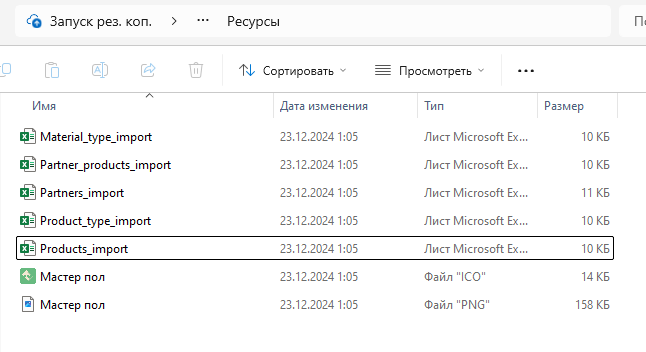
**Модуль № 1: Разработка, администрирование и защита баз данных**

**Описание предметной области представлено в приложении 1.**

После анализа предметной области необходимо создать базу данных с необходимыми таблицами:

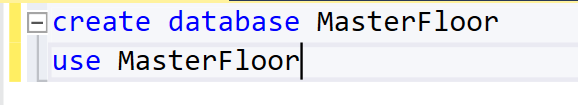
1. Продукция
2. Тип продукции
3. Партнер
4. Тип партнера
5. Материал
6. Сотрудник
7. Роль

Для создания таблиц подготовлен импорт, который хранится в папке Ресурсы



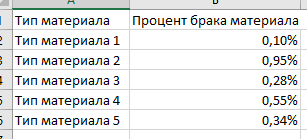
Создадим БД в SSMS и таблицы, определенные выше.

Компания называется «Мастер пол», поэтому название БД соответствующее

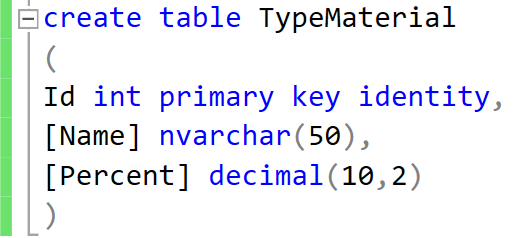


Открываем данные и создаем поля.

**Данные «Material\_type\_import»**

****

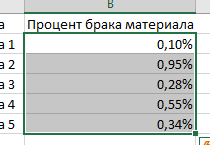
**Таблица «TypeMaterial»:**

****

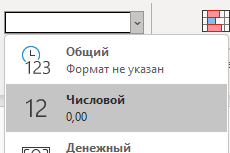
|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| Name | Наименование | nvarchar – текстовое поле |
| Percent | Процент | decimal (10,2) – вещественное число с 2 знаками после запятой |

Чтобы импортировать данные, необходимо удалить знак процента:

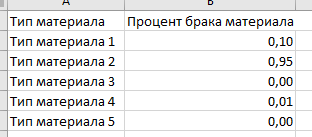
1. Выделяем данные



1. Выбираем числовой тип данных

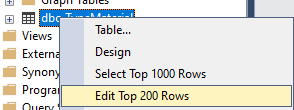


**Итог:**

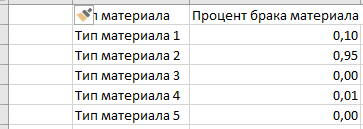
****

Данные подготовлены, копируем их и вставляем в таблицу в БД:

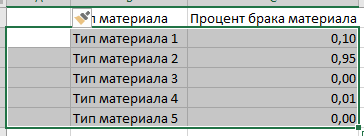
1. ПКМ по таблице и нажимаем поле «Изменить 200 строк»

****

1. В Excel добавляем пустой столбец слева от данных (это будет код в таблице, столбец пустой так как прописан **identity**).

****

1. Выделяем данные, которые хотим импортировать CTRL+C

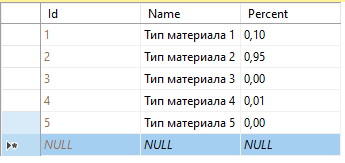
****

**ДАННЫЕ, КОТОРЫЕ ПЕРЕНОСЯТСЯ, ДОЛЖНЫ БЫТЬ СКОПИРОВАНЫ В ТОМ ЖЕ ПОРЯДКЕ, ЧТО И В БД**

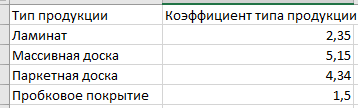
1. Нажимаем на поле выделения в БД

****

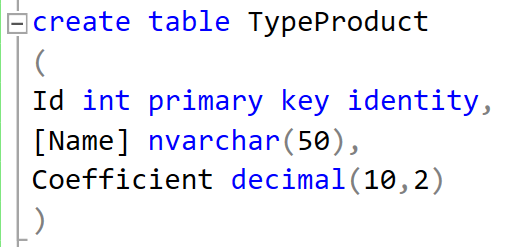
1. Вставляем данные CTRL+V

****

**Данные «Product\_type\_import»**

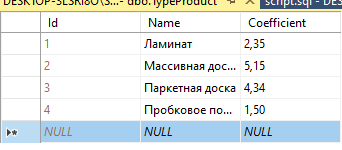
****

**Таблица «TypeProduct»:**

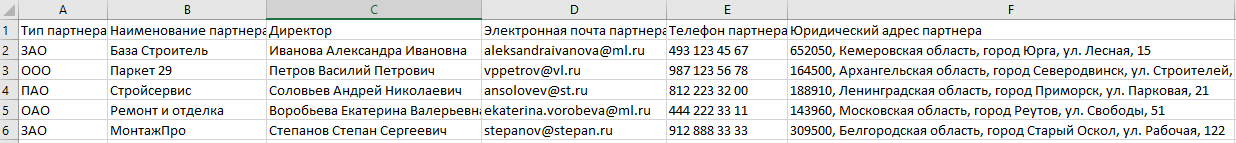
****

|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| Name | Наименование | nvarchar – текстовое поле |
| Coefficient | Коэффициент типа продукции | decimal (10,2) – вещественное число с 2 знаками после запятой |

Проделываем те же шаги для импорта в БД. Копируем данные и вставляем.

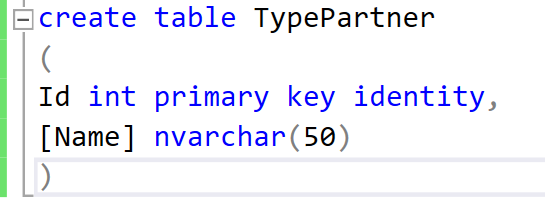


**Данные «Partners\_import»**

****

Прежде чем создать таблицу **Partner**, необходимо создать таблицу **TypePartner**

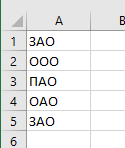
**Таблица «TypePartner»:**

****

|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| Name | Наименование | nvarchar – текстовое поле |

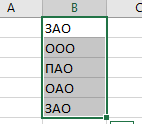
Чтобы импортировать данные, скопируем столбец с существующими типами и перенесем на новый лист в Excel





Удалим дубликаты для правильного импорта:

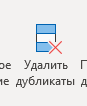
1. Выделяем данные



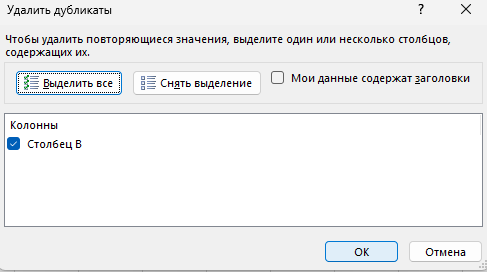
1. Переходим на вкладку «Данные»



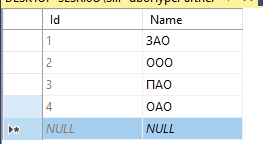
1. Выбираем удаление дубликатов



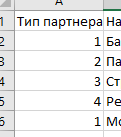
1. Нажимаем «Ок»



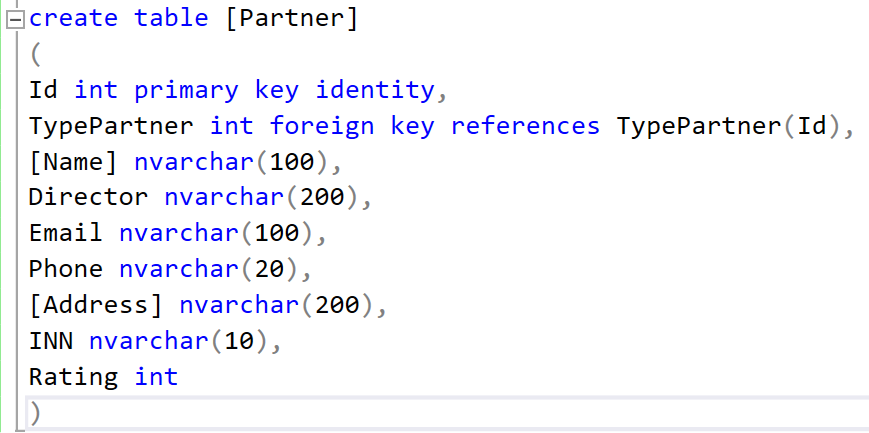
Для импорта проделываем те же шаги. Копируем и вставляем.



Подготовим данные для импорта **«Partners\_import».** Заменяем тип партнера на коды из БД.

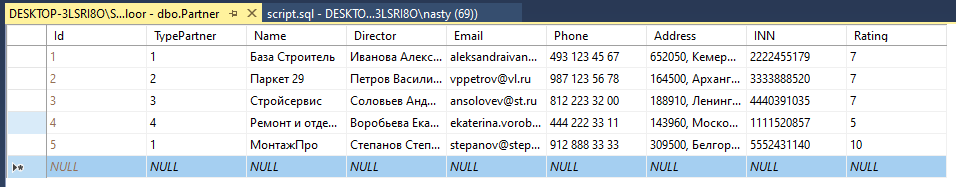


**Таблица «Partner»:**

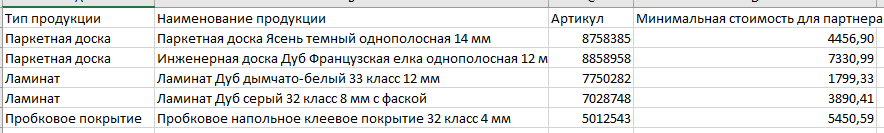
****

|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| TypePartner | Тип партнера | int foreign key references TypePartner(Id) – ссылка на таблицу TypePartner |
| Name | Наименование | nvarchar – текстовое поле |
| Director | Директор | nvarchar – текстовое поле |
| Email | Электронная почта партнера | nvarchar – текстовое поле |
| Phone | Телефон партнера | nvarchar – текстовое поле |
| Address | Юридический адрес партнера | nvarchar – текстовое поле |
| INN | ИНН | nvarchar – текстовое поле |
| Rating | Рейтинг | int – целое число |

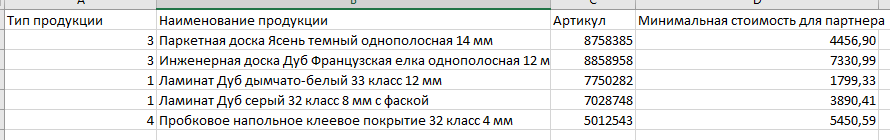
Проделываем те же шаги для импорта. Копируем и вставляем в БД.



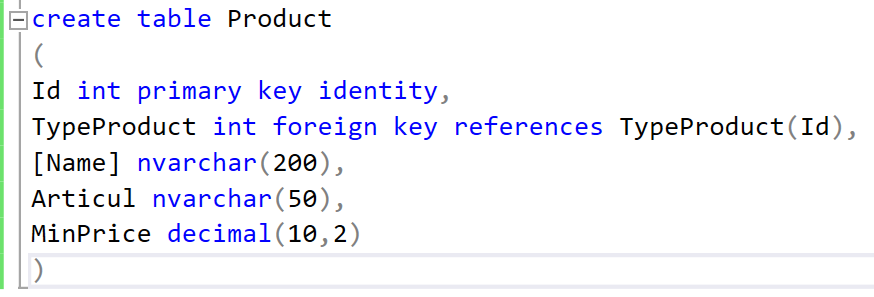
**Данные «Products\_import»**

****

Меняем тип продукции на коды из БД

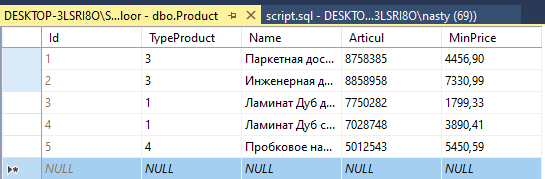


**Таблица «Product»:**

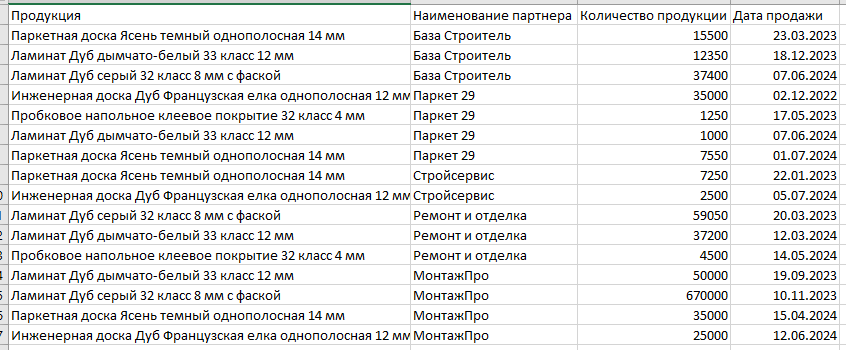
****

|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| TypeProduct | Тип продукта | int foreign key references TypeProduct (Id) – ссылка на таблицу TypeProduct |
| Name | Наименование | nvarchar – текстовое поле |
| Articul | Артикул | nvarchar – текстовое поле |
| MinPrice | Минимальная стоимость для партнера | decimal (10,2) – вещественное число с 2 знаками после запятой |

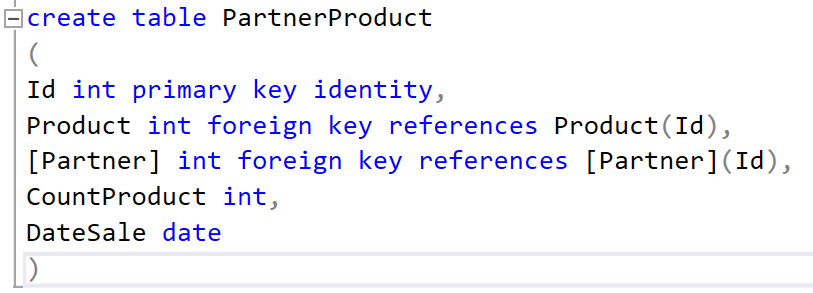
Переносим данные. Копируем и вставляем в БД.



**Данные «Partner\_products\_import»**

****

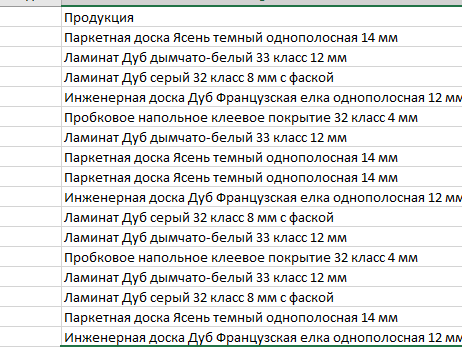
**Таблица «PartnerProduct»:**

****

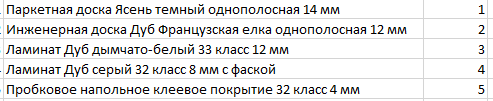
|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| Product | Продукт | int foreign key references Product (Id) – ссылка на таблицу Product |
| Partner | Партнер | int foreign key references Partner (Id) – ссылка на таблицу Partner |
| CountProduct | Количество продукции | int – целое число |
| DateSale | Дата продажи | date - дата |

Необходимо заменить данные на коды из БД. Если данных много, то можно воспользоваться функцией ВПР - она заменяет текст на то, что мы укажем, в нашем случае на коды.

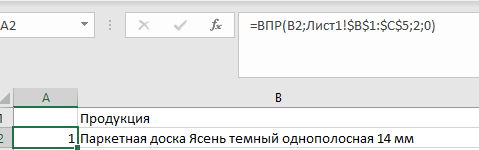
Для этого слева от заменяемых данных, вставим пустой столбец.



Далее возьмем данные о продукции и перенесем на другой лист, нам нужны только наименования. Справа от названий поставим коды из БД.

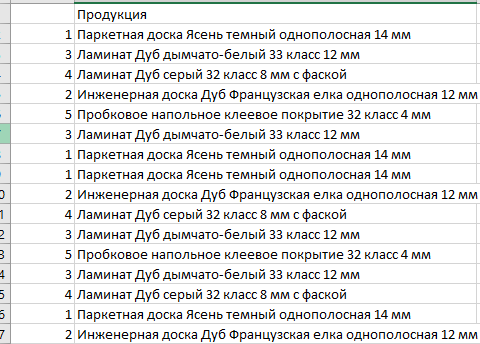


Для функции ВПР необходимо указать данные, которые мы хотим найти, это столбец справа, после указываем таблицу, в которой будем искать, эта таблица на другом листе, после указываем столбец и что нам нужно точное совпадение.

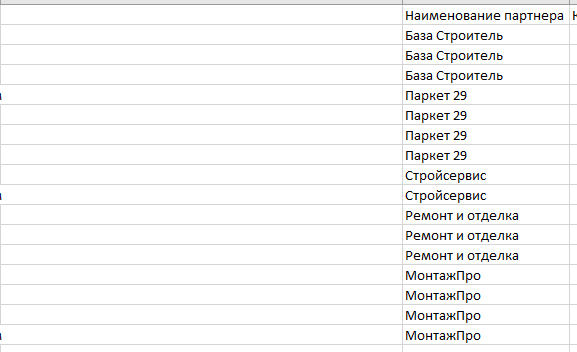


Перед и после букв необходимо поставить знак $, чтобы таблица не менялась.

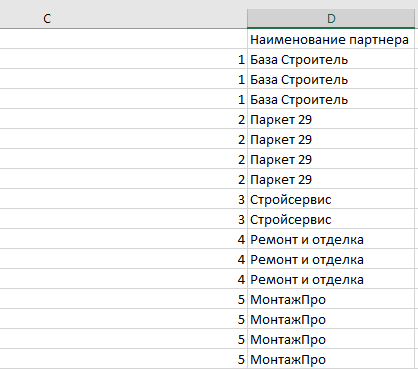
Проводим до конца и данные автоматически заполнятся.



То же самое делаем с партнерами

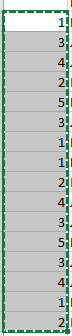






Теперь для импорта нам необходимо удалить столбцы с наименованиями, но, чтобы не слетели ссылки, нам необходимо скопировать коды и вставить только значения.

1. Копируем



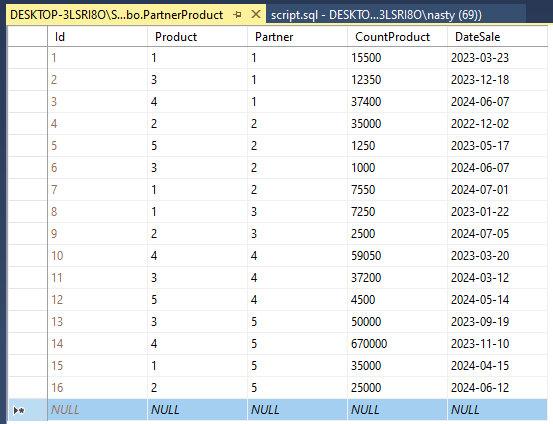
1. Вставляем туда же и выбираем вставить только значения



Удаляем столбец с наименованиями, тоже самое проделываем с партнерами.



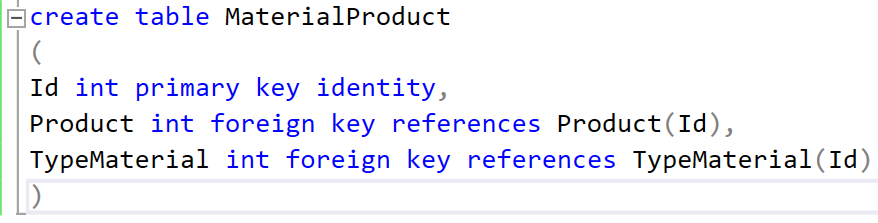
Для импорта проделываем те же шаги. Копируем и вставляем в БД.



Все данные для импорта находятся теперь в БД.

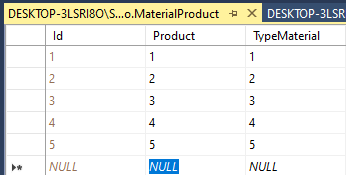
Таблица **«TypeMaterial»** находится без связи, поэтому добавим таблицу **«MaterialProduct»,** в которой будут находится материалы, необходимые для создания продукции.

**Таблица «MaterialProduct»:**

****

|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| Product | Продукт | int foreign key references Product (Id) – ссылка на таблицу Product |
| TypeMaterial | Материал | int foreign key references TypeMaterial (Id) – ссылка на таблицу TypeMaterial |

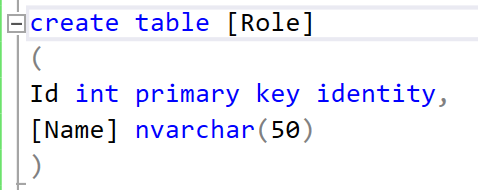
После создания заполним любыми данными.



Также не хватает таблиц **«Employee» ‑ сотрудники**, **«Role» - должность**

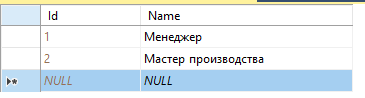
Создадим их в БД и заполним данными (если данных нет в приложении, то заполняем любыми).

**Таблица «Role»:**

****

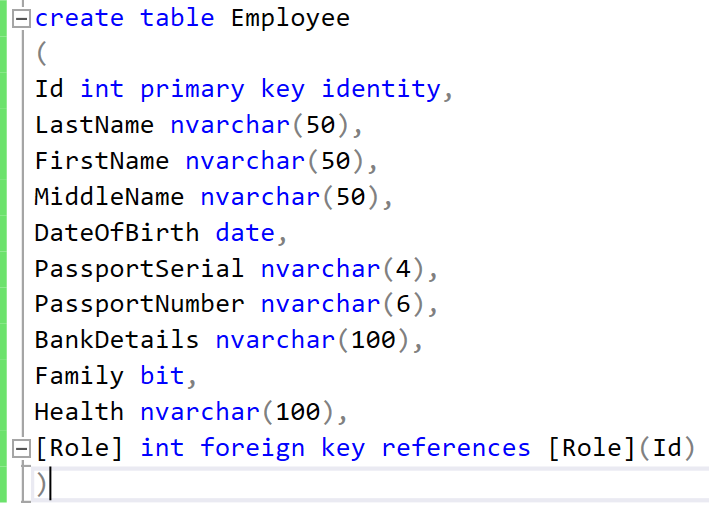
|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| Name | Наименование | nvarchar – текстовое поле |

Данные взяты из приложения 1:

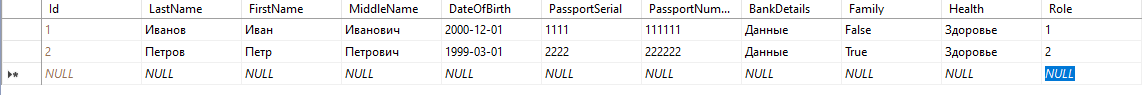


**Таблица «Employee»:**



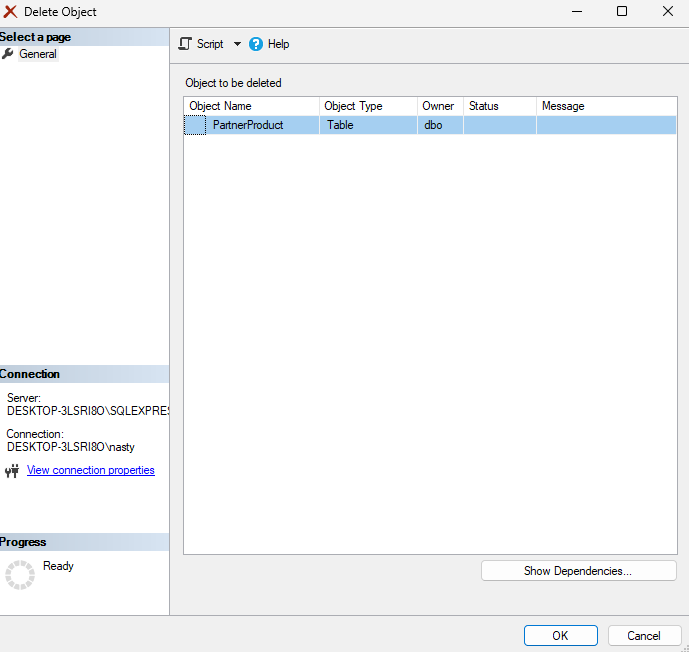
****

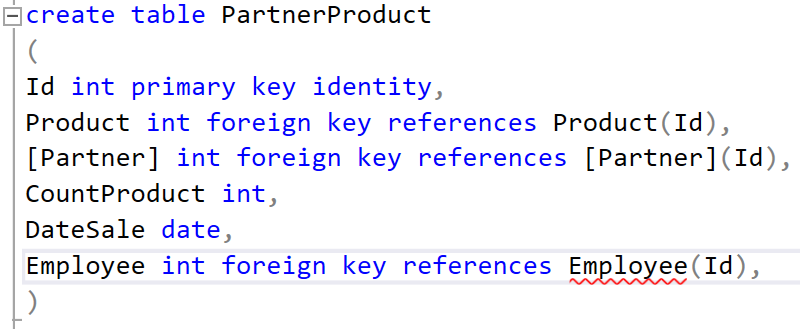
|  |  |  |
| --- | --- | --- |
| Идентификатор | Наименование | Примечание |
| Id | Код | Необходим для всех таблиц, всегда **identity**, всегда **primary key (первичный ключ)** |
| LastName | Фамилия | nvarchar – текстовое поле |
| FirstName | Имя | nvarchar – текстовое поле |
| MiddleName | Отчество | nvarchar – текстовое поле |
| DateOfBirth | Дата рождения | date - дата |
| PassportSerial | Серия паспорта | nvarchar – текстовое поле |
| PassportNumber | Номер паспорта | nvarchar – текстовое поле |
| BankDetails | Банковские реквизиты | nvarchar – текстовое поле |
| Family | Наличие семьи | bit – true/false |
| Health | Состояние здоровья | nvarchar – текстовое поле |
| Role | Должность | int foreign key references Role (Id) – ссылка на таблицу Role |



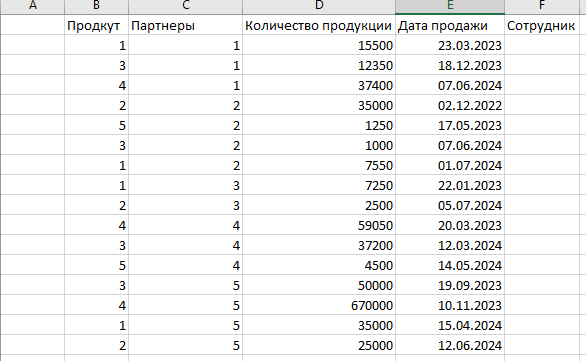
Сотрудник оформляет заказы партнеров на продукцию, значит в таблице **«PartnerProduct»** должен хранится внешний ключ на таблицу **«Employee».**

Придется удалить таблицу, добавить поле и заново загрузить данные.



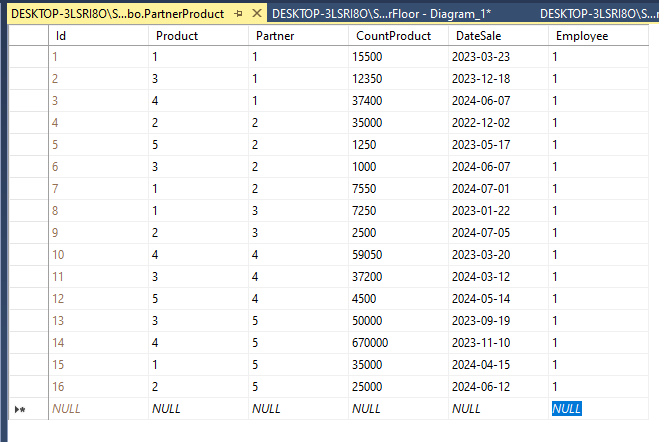


Так как появилось новое поле, в данных нужно добавить пустой столбец.



Теперь переносим данные. Копируем и вставляем.

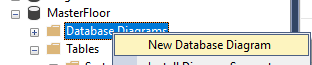




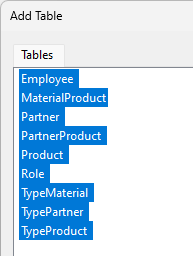
Добавляем любые данные в поле Employee.

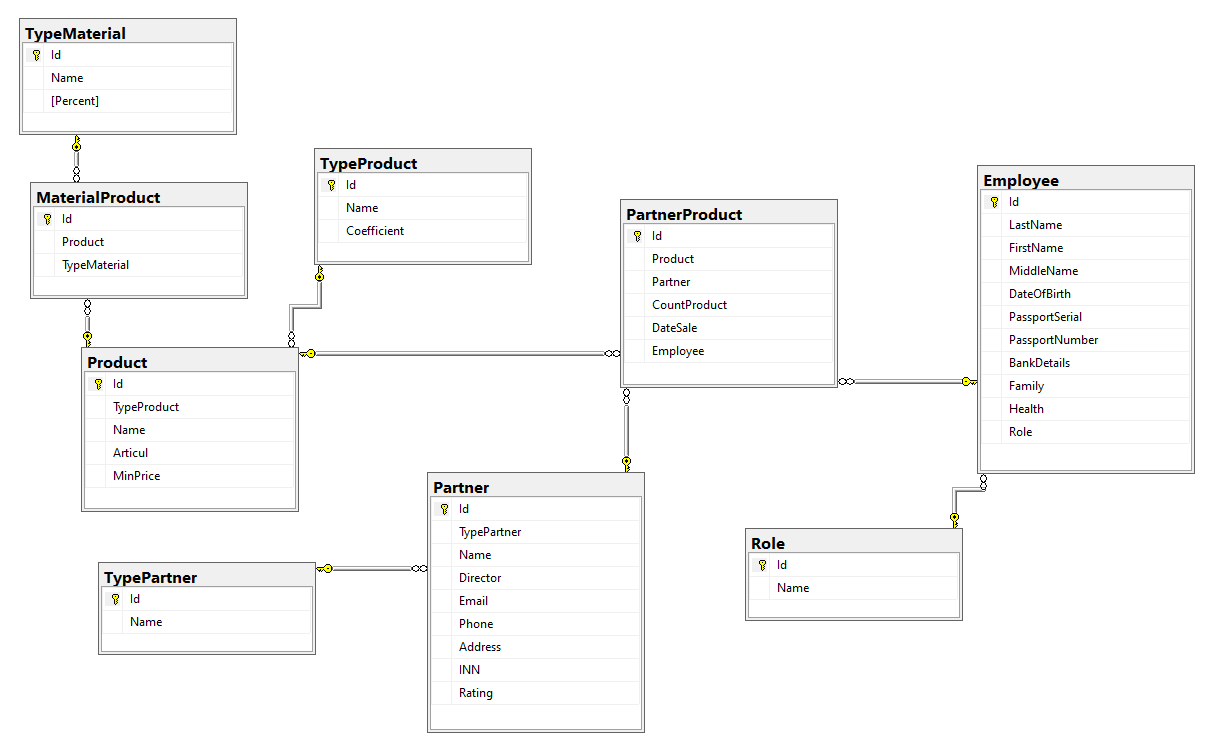
**По заданию требуется сделать ER – диаграмму**. Для этого построим ее в MS SQL и после сохраним в формате PDF.

1. Нажимаем ПКМ по полю «Диаграммы БД».



1. Выбираем все таблицы и нажимаем добавить



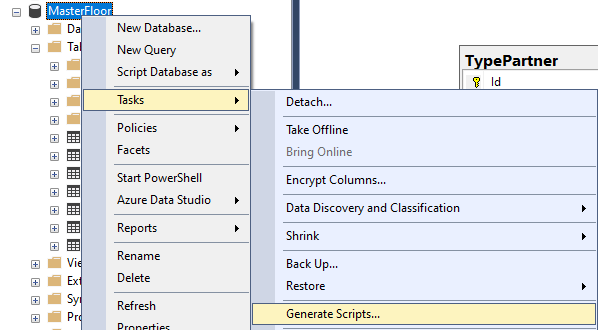


Вырежем диаграмму с помощью ножниц и вставим в WORD. Сохраним в формате PDF.

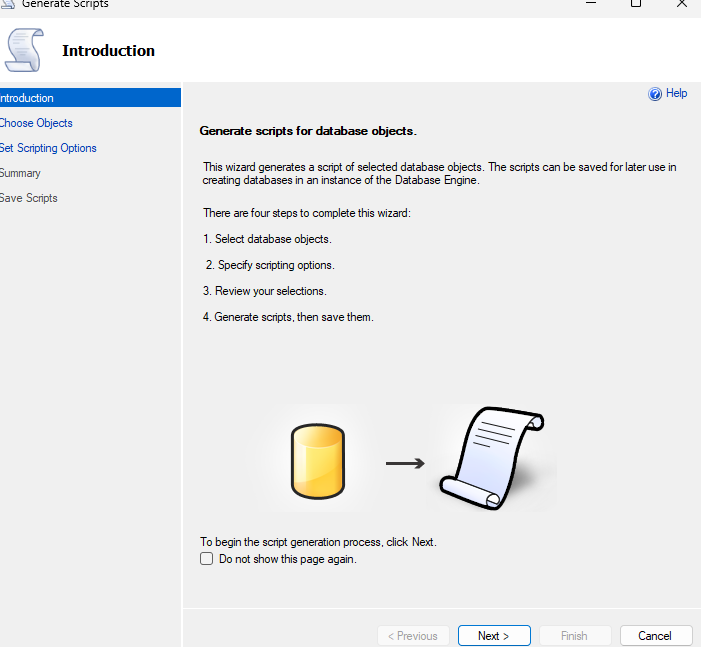


**По заданию требуется сохранить полученные результаты: создать скрипт БД.**

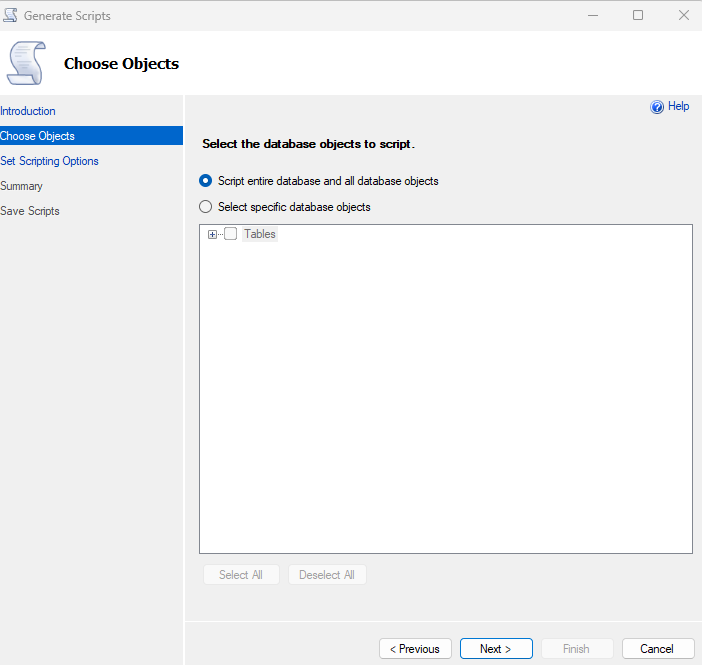
Для этого нажмем ПКМ по БД и выберем сгенерировать скрипт.



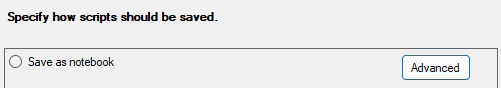
Нажимаем далее.



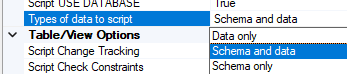
Нажимаем далее.

****

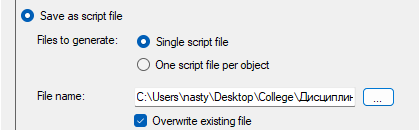
Выбираем подробнее.



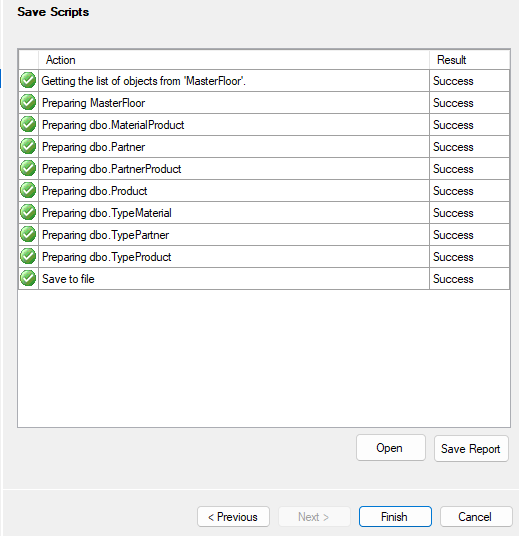
В поле тип данных выбираем схема и данные



Выбираем куда сохранить



И нажимаем далее. Итог:



Перенесем PDF и созданный скрипт в папку, созданную для задания.

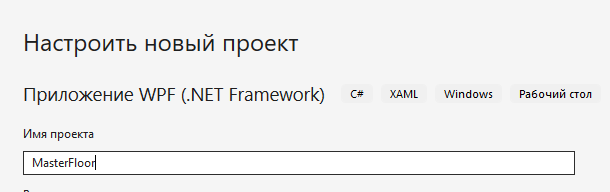


**Модуль № 2: Разработка модулей программного обеспечения для компьютерных систем**

После создания БД необходимо создать приложение. Для этого используем WPF, создадим проект.

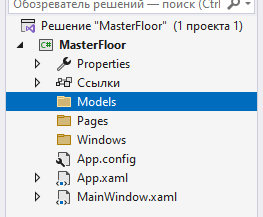


**Название должно соответствовать названию компании.**

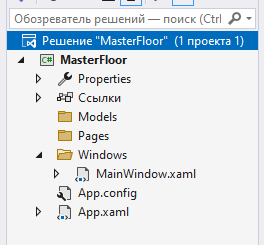


Для начала необходимо подготовить проект, чтобы он был структурирован создадим в нем несколько папок: для окон, для страниц и для будущего взаимодействия с БД.

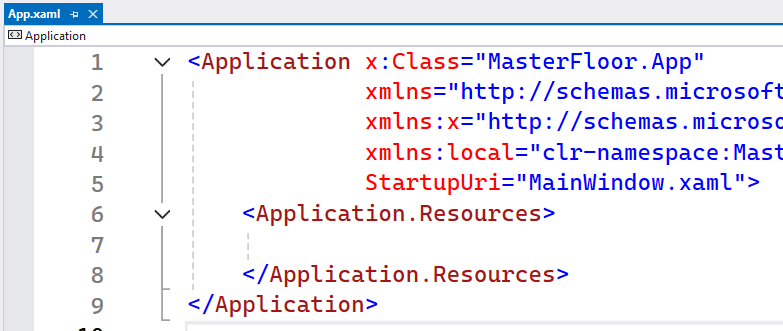
1. **Папка «Windows» - для окон**
2. **Папка «Pages» - для страниц**
3. **Папка «Models» - для будущего взаимодействия с БД**



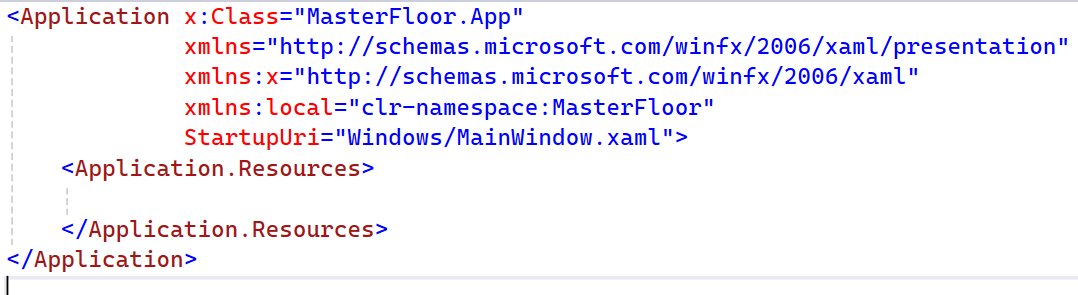
Теперь перенесем главное окно в папку «Windows».



Так как главное окно теперь имеет другой путь, необходимо его поменять в файле **«App.xaml».**



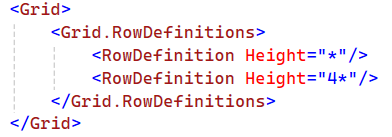
Изменим StartupUri



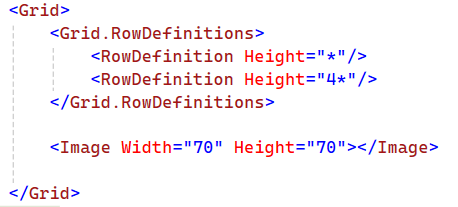
Мы будем использовать 1 окно и с помощью **Frame** перемещаться между **страницами**. Для этого сделаем разметку на окне **MainWindow.**

Так как необходимо создать интерфейс, который будет отображать логотип компании и заголовки на каждой странице, а также кнопку назад, то разделим наше окно на 2 строки.

В 1 строке будут располагаться 3 компонента: Image – для логотипа, TextBlock – для заголовков, Button – для последовательных переходов между страницами (Кнопка «Назад»).



Положим **«Image»,** зададим размеры с помощью **Width** - **ширина** и **Height – высота.**



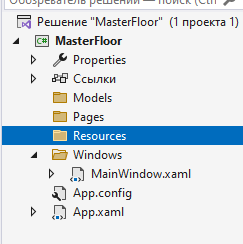
Чтобы логотип располагался слева, укажем свойство **HorizontalAligment –** выравнивание по горизонтали.



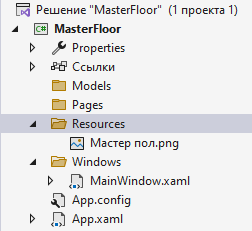
Чтобы логотип не приклеивался к границам, зададим внешние отступы с помощью **Margin.**



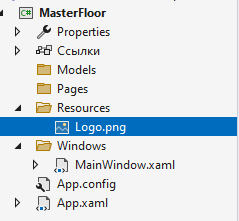
Чтобы положить в компонент логотип, добавим его в программу, создадим папку **«Resources»**

****

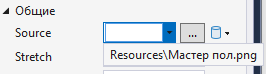
Перенесем картинку в эту папку с помощью перетаскивания.



Переименуем его как **Logo.**

****

Откроем окно свойств компонента **Image** и выберем нашу картинку.



Для отображения заголовков, разместим по центру **TextBlock**.



Расположим **Frame** во 2 строке.



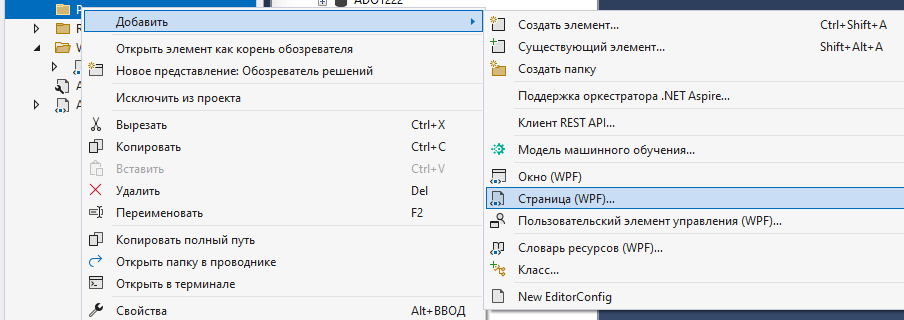
Зададим наименование и свойство **NavigationUIVivibility – Hidden**,



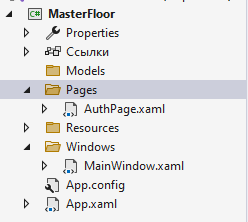
Чтобы у него текст у нашего заголовка менялся автоматически, в зависимости от названия страницы, зададим свойство **Text** и пропишем в нем **Binding – привязка данных**, так как страницы находятся во Frame, то привязка будет к этому элементу**. Content.Title –** необходим для того, чтобы обращаться к Title страницы.

****

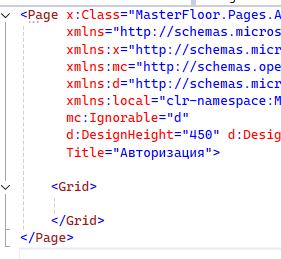
Перед тем, как расположить кнопку, проверим как отображаются заголовки и логотип, поэтому создадим первую страницу – страница авторизации.



Назовем ее – **AuthPage**.

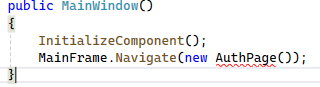


Изменим у страницы **Title.**

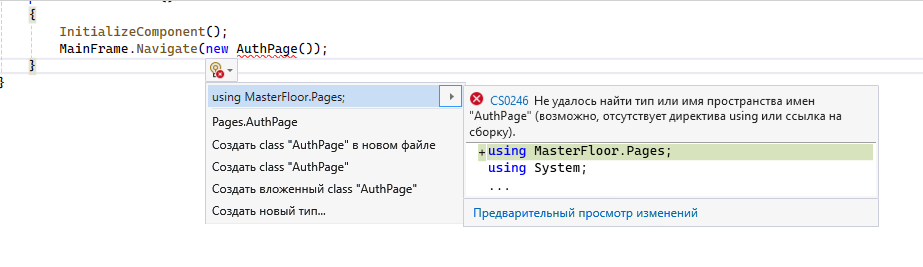


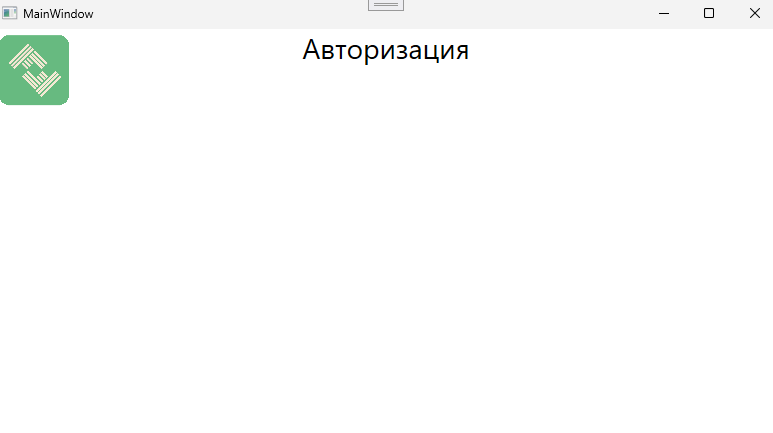
Теперь зададим начальную страницу у **Frame.** Перейдем в код **MainFrame.**

Чтобы передать во **Frame** страницу, обратимся к нему и с помощью метода **Navigate** и передадим страницу.



Чтобы страница была доступна, подключим пространство имен.



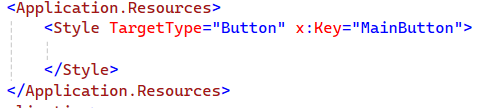


По заданию необходим единый стиль, для этого есть программные ресурсы, которые позволяют для конкретных компонентов создать стиль и после использовать по всей программе.

Зайдем в **приложение 2**.

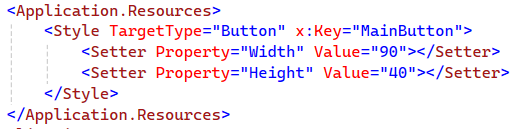
Основной фон будет белым, дополнительный фон использовать необязательно. Цвет для акцентирования внимания будем использовать во всех кнопках. По заданию необходим шрифт Segoe UI, поэтому его также пропишем в стилях.

Для начала создадим стиль для кнопки. Для этого необходимо указать **<Style>,** в котором указать компонент **(TargetType),** для которого написан стиль и ключ **(Key),** по которому будем обращаться.

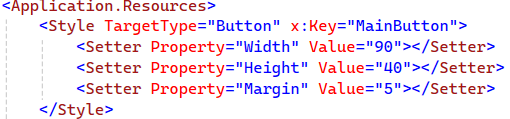


Теперь мы можем указывать любые свойства и из значения, для этого используется <**Setter**>, в котором указываются свойство (**Property**) и его значение (**Value**).

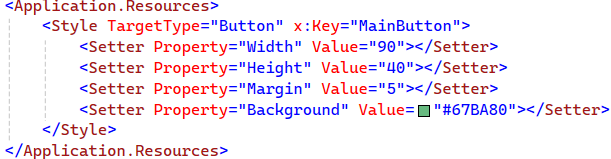
1. **Размеры** (ширина и высота).



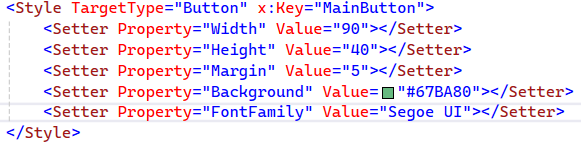
1. **Отступы** (Margin)



1. **Фон** (Background), берем из Приложения 2

****

1. **Шрифт** (FontFamily), берем из Приложения 2

****

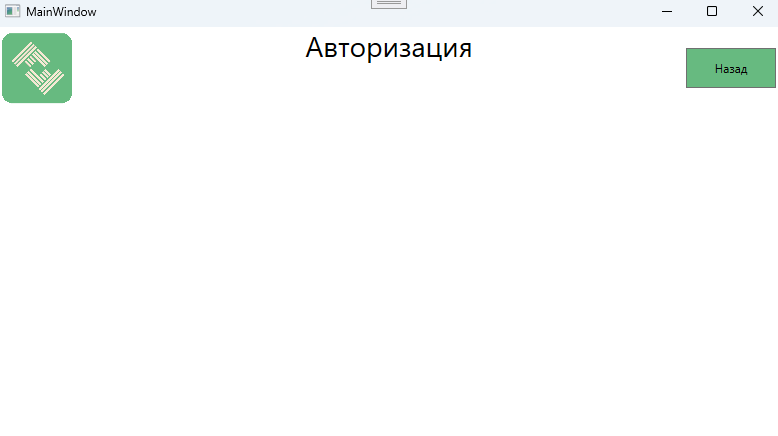
Теперь с помощью этого стиля будут создаваться все кнопки.

Создадим кнопку на главном окне справа в 1 строке.



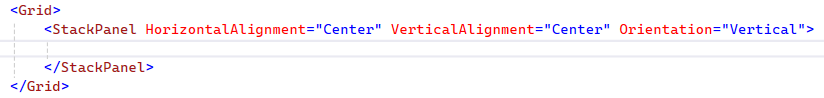
Укажем стиль, который только что создали.





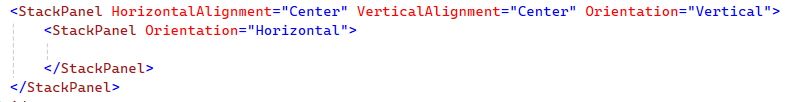
Перейдем к странице авторизации.

Создадим простой интерфейс.

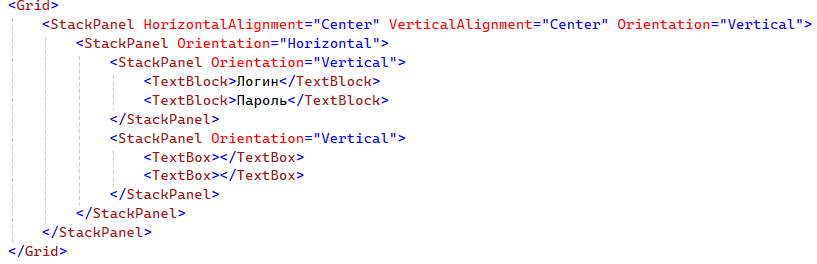


Нам необходим основной контейнер – **StackPanel**, у которого будут указаны несколько свойств: **HorizontalAlignment**="Center" (контейнер будет находится по центру) **VerticalAlignment**="Center" (контейнер будет находится по центру) **Orientation**="Vertical" (все элементы внутри контейнера будут располагаться по вертикали).

Внутри расположим контейнер, но уже с горизонтальной ориентацией.

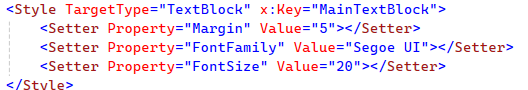


После этого располагаем 2 контейнера, в которых находятся 2 **TextBlock** и 2 **TextBox** для ввода логина и пароля.

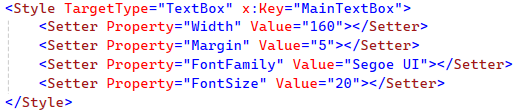


Чтобы все элементы были в едином стиле, для них тоже создадим их в файле **App.xaml**

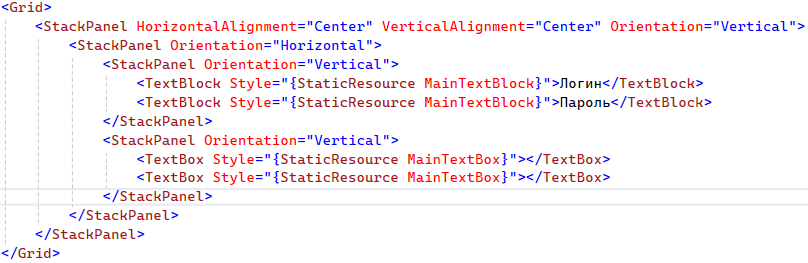
Создадим стиль для **TextBlock**

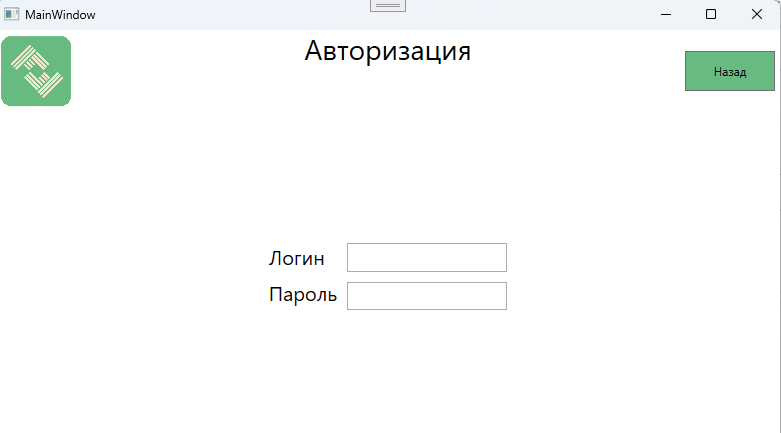


И для **TextBox**

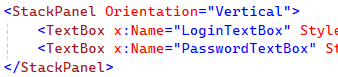


Применим эти стили к странице авторизации.

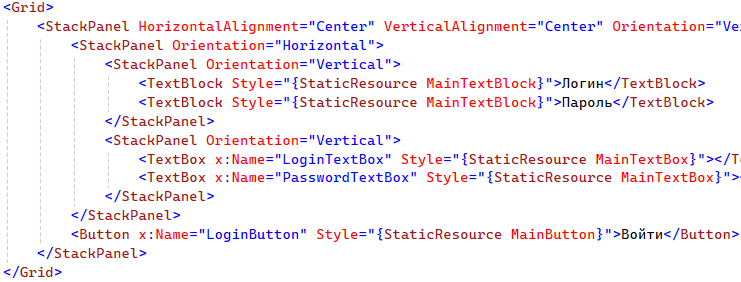




У **TextBox** необходимо указать имена, чтобы обращаться в коде.

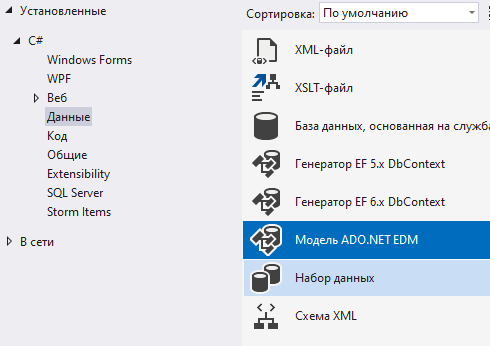


Для входа необходима кнопка, расположим ее на странице, зададим наименование и стиль.



Страница авторизации готова, перейдем к подключению БД.

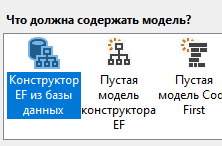
Для подключения к БД в папке **Models** создадим элемент **Модель ADO.NET EDM**



Назовем ее Model



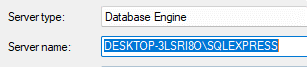
Необходимо выбрать **Конструктор EF**



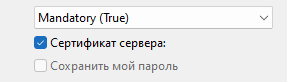
После создать соединение



Вводим имя сервера, его берем из БД



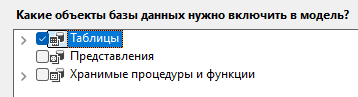
Ставим галочку **Сертификат сервера**



Пишем нашу БД



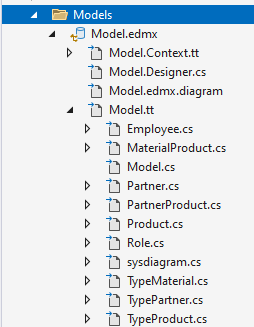
Нажимаем далее и при включении данных выбираем **Таблицы**



Ставим галочку и нажимаем **Готово**



Должна сформироваться диаграмма, а также создаться классы в папке.



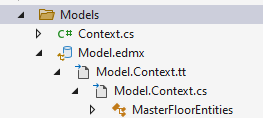
Теперь напишем авторизацию пользователей.

Создадим событие **Click** у кнопки.

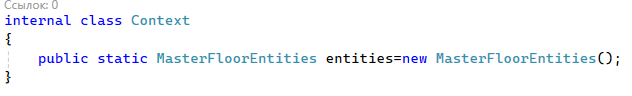


Для начала создадим подключение, для этого создадим класс в папке **Models,** назовем его **Context.**

Сформируем строчку подключения. Чтобы найти название, раскроем файл **Model.edmx**

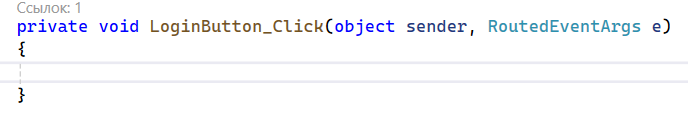


Создадим объект этого подключения.

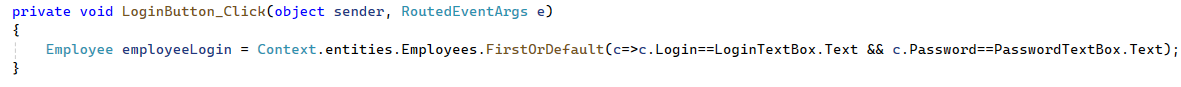


Теперь во всей программе мы будем обращаться к этому подключению.

Вернемся к коду в кнопке.

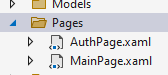


Найдем пользователя по данным, введенным в наши текстовые поля, для этого обратимся к методу **FirstOrDefault.**

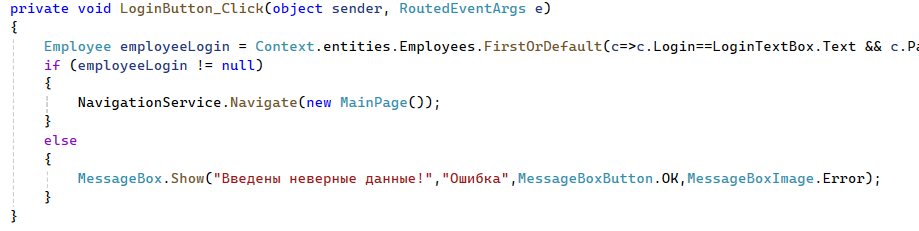


1. **Employee employeeLogin**:
   * Здесь создается переменная employeeLogin типа Employee, которая будет хранить информацию о сотруднике, если он будет найден.
2. **Context.entities.Employees**:
   * Context — это объект контекста базы данных, который позволяет взаимодействовать с данными. entities — это свойство контекста, которое содержит доступ ко всем сущностям (таблицам) в базе данных. Employees — это коллекция сотрудников.
3. **FirstOrDefault(...)**:
   * Этот метод используется для поиска первого элемента в коллекции, который соответствует заданному условию. Если такой элемент не найден, он вернет null.
4. **c => c.Login == LoginTextBox.Text && c.Password == PasswordTextBox.Text**:
   * Это лямбда-выражение, которое задает условие для поиска. Здесь c представляет собой объект сотрудника из коллекции Employees. Условие проверяет, совпадает ли логин (c.Login) с текстом в LoginTextBox и пароль (c.Password) с текстом в PasswordTextBox.

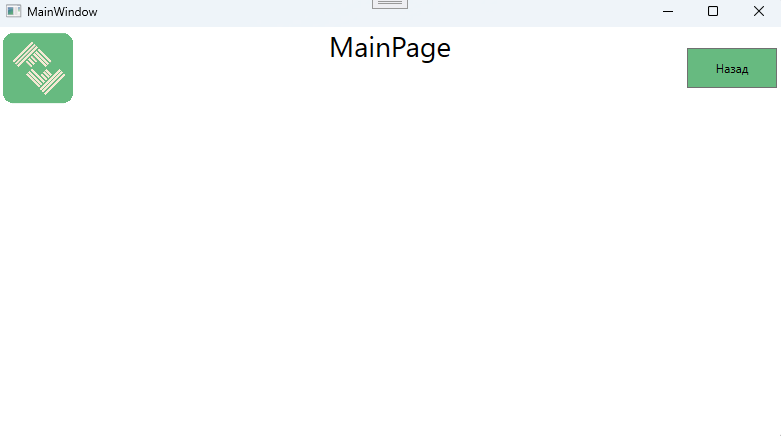
Проверим, если в переменной **null**, то будем выводить ошибку, если мы нашли совпадение в БД, то перейдем на главную страницу, создадим ее.



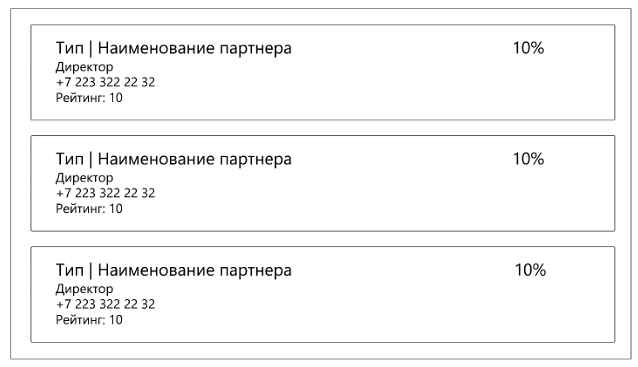
Пропишем условие. **NavigationService** позволяет осуществлять переход между страницами.



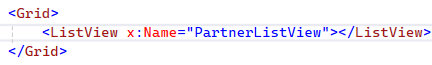
Таким образом, при правильно введенных данных, мы перейдем на главную страницу.



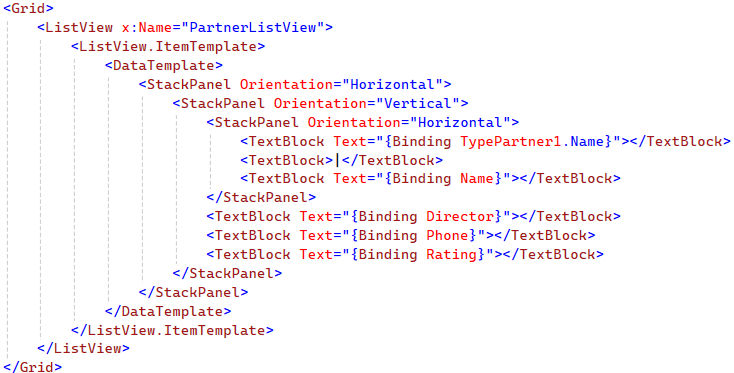
По заданию, нам необходимо отобразить список партнеров по шаблону.



Для этого на главной странице расположим **ListView.** У него укажем имя



Теперь нам необходимо сделать ячейку листа по шаблону.

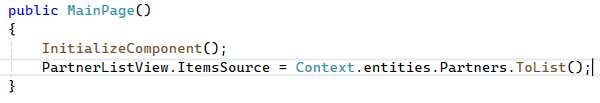


1. **<ListView x:Name="PartnerListView">**:
   * Создается элемент управления ListView, который будет отображать список данных. x:Name задает имя для этого элемента, чтобы к нему можно было обращаться из кода.
2. **<ListView.ItemTemplate>**:
   * Здесь задается шаблон для отображения каждого элемента в ListView. Это означает, что каждый элемент списка будет представлен по заданному шаблону.
3. **<DataTemplate>**:
   * Определяет шаблон, который будет использоваться для представления данных. Внутри него можно размещать различные элементы управления.
4. **<StackPanel Orientation="Horizontal">**:
   * Используется для размещения дочерних элементов в горизонтальном направлении. В данном случае используется для группировки элементов.
5. **Вложенные StackPanel**:
   * Внутренние StackPanel используются для организации элементов в вертикальном и горизонтальном направлениях, что позволяет лучше структурировать отображение данных.
6. **<TextBlock>**:
   * Этот элемент используется для отображения текстовой информации. В данном коде он используется для отображения различных свойств объекта, привязанных к данным.

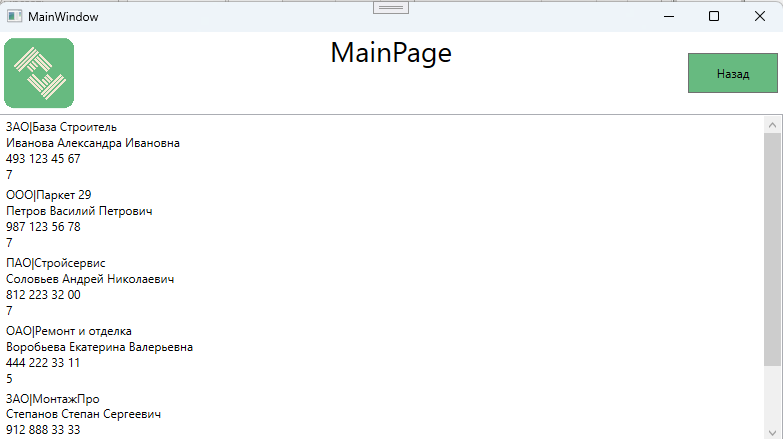
**Привязка данных**

* **Text="{Binding TypePartner1.Name}"**:
  + Привязывает текст к свойству Name объекта TypePartner1, который является частью данных, связанных с элементом списка.
* **Text="{Binding Name}"**:
  + Привязывает текст к свойству Name текущего объекта партнера.
* **Text="{Binding Director}", Text="{Binding Phone}", Text="{Binding Rating}"**:
  + Эти привязки отображают информацию о директоре, телефоне и рейтинге партнера соответственно.

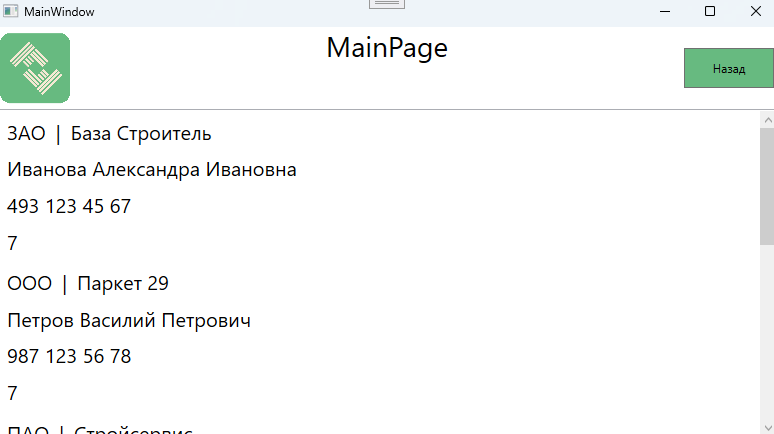
Теперь в этот лист отправим данные.



1. **PartnerListView.ItemsSource**:
   * Это свойство элемента управления ListView, которое определяет источник данных, который будет отображаться в списке. Устанавливая это свойство, мы говорим ListView, откуда брать данные для отображения.
2. **Context.entities.Partners**:
   * Context — это объект контекста базы данных, который позволяет взаимодействовать с данными в базе. entities — это коллекция всех сущностей (таблиц) в базе данных. Partners — это коллекция объектов партнеров, которая представляет таблицу партнеров.
3. **ToList()**:
   * Этот метод преобразует коллекцию Partners в список (List). Это необходимо, так как ItemsSource ожидает объект, реализующий интерфейс IEnumerable, и ToList() возвращает именно такой объект.

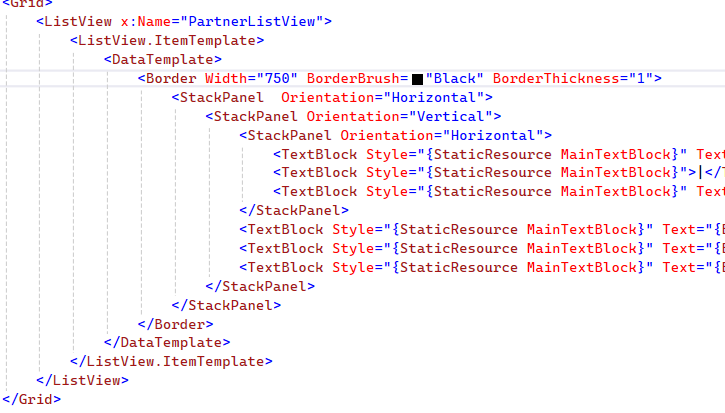


Добавим наши стили в разметку.



Чтобы сделать у ячеек границы, добавим поверх основного **StackPanel – Border.**

Зададим ширину, цвет границ и ширину границ.



1. **<Border>**:
   * Это элемент, который создает рамку вокруг других элементов управления или содержимого. Он может использоваться для визуального выделения содержимого.
2. **Width="750"**:
   * Это свойство задает фиксированную ширину рамки в 750 пикселей. Элемент Border будет иметь эту ширину, независимо от содержимого внутри него.
3. **BorderBrush="Black"**:
   * Это свойство устанавливает цвет границы. В данном случае граница будет черной. Можно использовать различные цвета.
4. **BorderThickness="1"**:
   * Это свойство задает толщину границы. В данном случае граница будет иметь толщину 1 пиксель. Можно указать разные значения для каждой стороны, например, BorderThickness="1,2,3,4" (слева, сверху, справа, снизу).

Дадим странице **Title.**



Установим иконку приложения. Зайдем в **Properties.**



Выберем значок из папки.

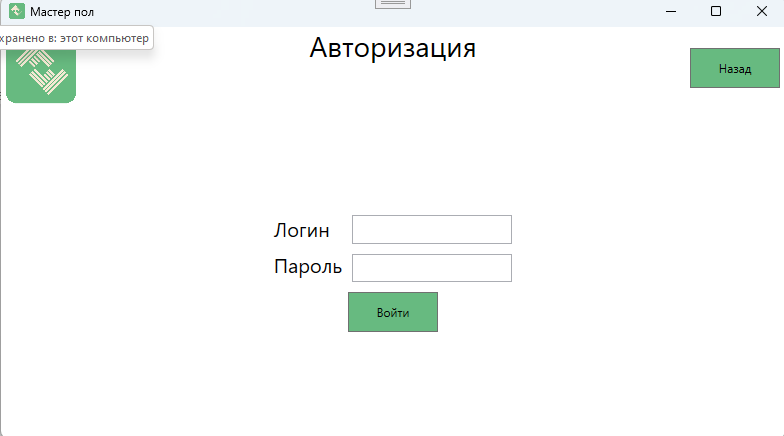
****

Теперь при запуске иконка будет отображаться.



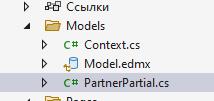
Также назовем нашу программу, перейдем на окно и зададим **Title**



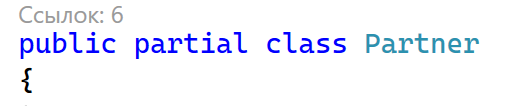


Для отображения процентов сделаем расширение класса.

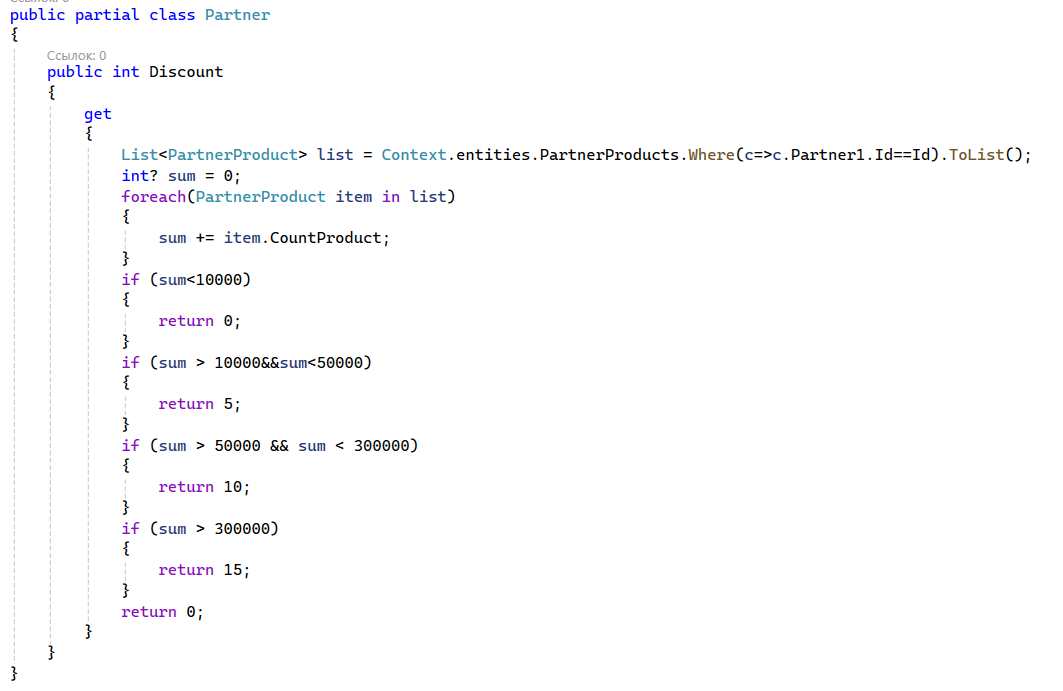
Назовем его PartnerPartial.



После перепишем его код.



Создадим поле, которое будет отвечать за вычисление скидки.



Класс Partner содержит одно свойство Discount, которое вычисляет скидку на основе количества продуктов, связанных с партнером.

1. **Получение списка продуктов**:

List<PartnerProduct> list = Context.entities.PartnerProducts.Where(c => c.Partner1.Id == Id).ToList();

Здесь мы получаем список продуктов (PartnerProduct), связанных с текущим партнером (Partner). Мы используем метод Where для фильтрации продуктов по идентификатору партнера (Id).

1. **Суммирование количества продуктов**:

int? sum = 0;

foreach (PartnerProduct item in list)

{

sum += item.CountProduct;

}

Мы инициализируем переменную sum как 0 и затем проходим по каждому продукту в списке, добавляя количество продуктов (CountProduct) к sum.

1. **Условия для определения скидки**: После того как мы получили общее количество продуктов, мы применяем условия для определения скидки:

if (sum < 10000)

{

return 0; // Нет скидки

}

if (sum > 10000 && sum < 50000)

{

return 5; // Скидка 5%

}

if (sum > 50000 && sum < 300000)

{

return 10; // Скидка 10%

}

if (sum > 300000)

{

return 15; // Скидка 15%

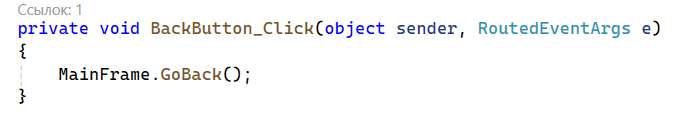
}

* + Если сумма меньше 10,000, скидка составляет 0%.
  + Если сумма больше 10,000, но меньше 50,000, скидка составляет 5%.
  + Если сумма больше 50,000, но меньше 300,000, скидка составляет 10%.
  + Если сумма больше 300,000, скидка составляет 15%.

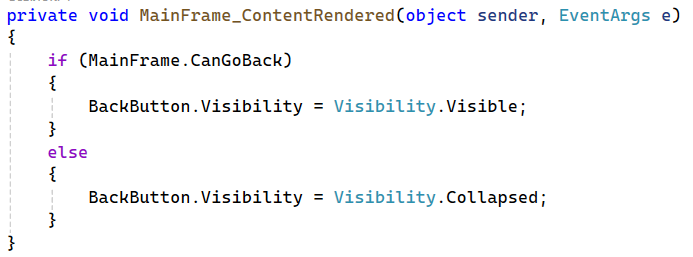
Задание по модулю 2 реализовано. Осталась кнопка назад.

Вернемся к главному окну.

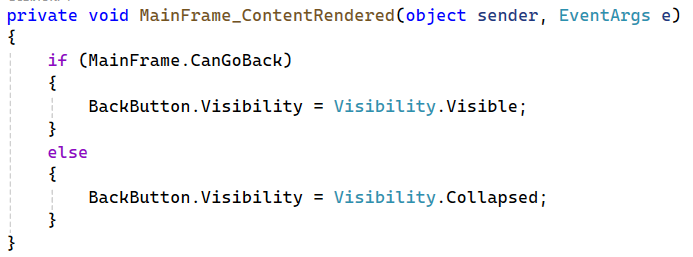
1. **Метод BackButton\_Click**:



* + **Назначение**: Этот метод обрабатывает событие нажатия кнопки "Назад" (BackButton).
  + **Функция**: Когда пользователь нажимает на кнопку, вызывается метод GoBack() у объекта MainFrame. Это позволяет вернуться на предыдущую страницу или окно в навигационной истории, если таковая имеется.



1. **Метод MainFrame\_ContentRendered**:



**Назначение:** Этот метод обрабатывает событие ContentRendered, которое срабатывает каждый раз, когда содержимое MainFrame загружено и отображено.

**Функция:** Внутри метода проверяется, может ли MainFrame вернуться на предыдущую страницу с помощью свойства CanGoBack.

Если CanGoBack равно true, это означает, что есть предыдущая страница, и кнопка "Назад" (BackButton) становится видимой (Visibility.Visible).

Если CanGoBack равно false, кнопка "Назад" скрывается (Visibility.Collapsed).

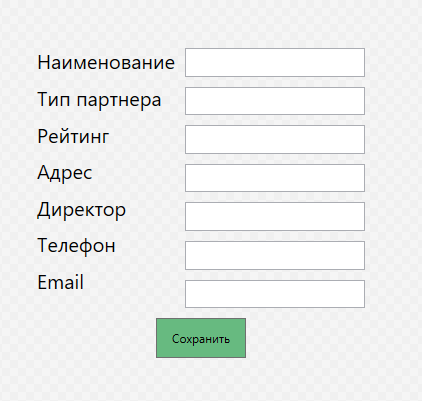
### Основные моменты

* **Навигация**: Код реализует простую навигацию между страницами в приложении. Пользователь может вернуться на предыдущую страницу, если она существует.
* **Состояние кнопки**: Кнопка "Назад" отображается или скрывается в зависимости от наличия предыдущих страниц в истории навигации, что улучшает пользовательский интерфейс и делает его более интуитивным.

**Модуль № 3: Сопровождение и обслуживание программного обеспечения компьютерных**

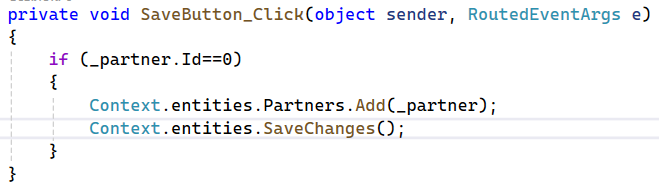
**систем**

Создадим страницу для добавления/редактирования партнеров.



Необходимо для каждого TextBox прописать **Binding**, как в предыдущем модуле 2.

При нажатии на кнопку будет отправляться новый партнер в БД. Условие нужно для проверки на то, хотим мы добавить или изменить.



Переменная \_partner объявлена глобально.

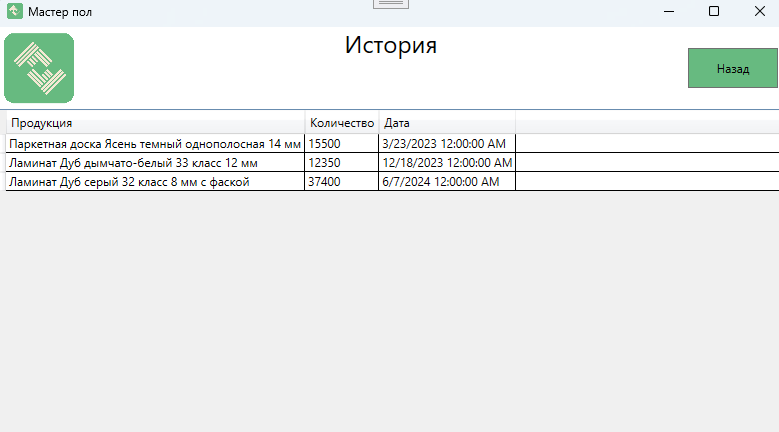


Реализуйте изменение самостоятельно, для этого необходимо сначала передать с другой страницы партнера и передать в \_partner, после сохранить изменения.

**Модуль № 4: Осуществление интеграции программных модулей**

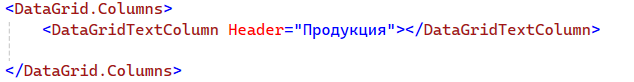
Необходимо осуществить вывод истории реализации продукции партнером с указанием наименования продукции, количества и даты продажи.

Создадим страницу для истории.



Для вывода списка будем использовать DataGrid.

Для указания столбцов необходимо указать тег Columns и после перечислить все необходимые.



Для отображения данные у колонок необходимо прописать заголовок с помощью Header и Binding следующим образом

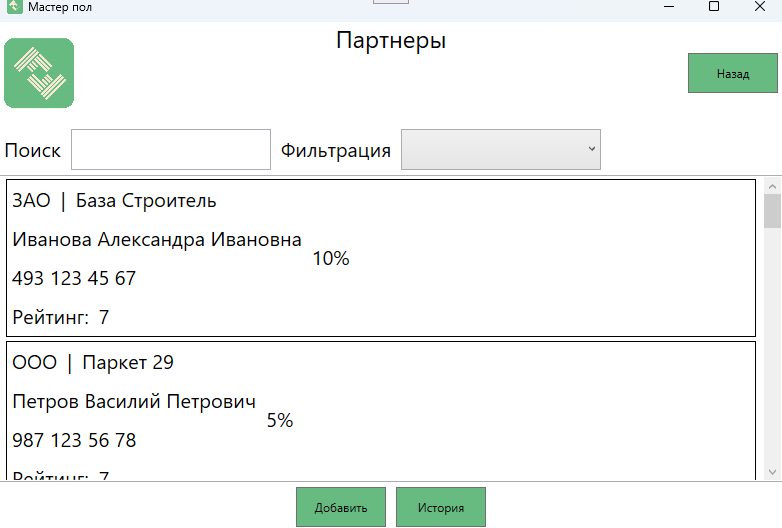
Binding="{Binding обращение к полю}"

Для указания данных в коде обратиться к свойству ItemsSource и передать желаемый лист.

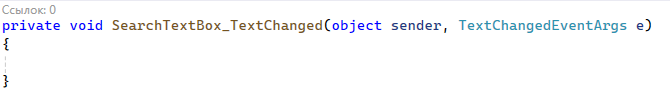
Метод для расчета количества материала необходимо реализовать самостоятельно.

**Дополнительный модуль: Поиск, фильтрация, GIT**

Для реализации поиска и фильтрации необходимо создать компоненты на странице, где будет происходить данный функционал.



Создадим событие TextChanged.



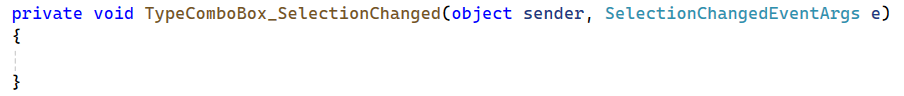
В примере будем осуществлять поиск по имени партнера.

Для этого обратимся к БД и будем передавать в список только тех партнеров, у которых в имени есть введенный текст из TextBox.



Для фильтрации сначала заполним ComboBox. Заполним данные по типу партнера.

Для фильтрации создадим метод SelectionChanged.



При выборе любого элемента будет срабатывать это событие. Получим выбранный и сделаем сравнение при выборке из БД.



В выборке указываем условие и сравниваем с тем объектом, который получили из ComboBox.

Чтобы отобразить всех, сделаем кнопку сброса. При нажатии на нее будем передавать все данные.



Для работы с гитом необходимо быть зарегистрированным на сайте <https://github.com/>.

Перейдем в репозитории и создадим один.

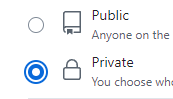




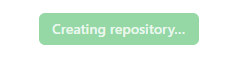
Впишем название.



Сделаем непубличным.

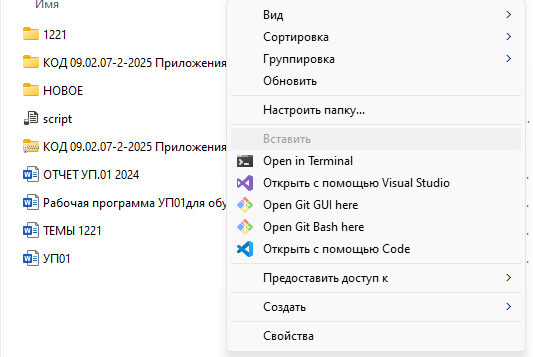


Нажмем создать.



Чтобы загрузить проект и все данные, создайте общую папку на рабочем столе и положите в нее все необходимые данные.

Нажимаем ПКМ. Выбираем **Open jn Terminal**.



И пропишем ряд команд.

git init

git add .

git commit -m "first commit"

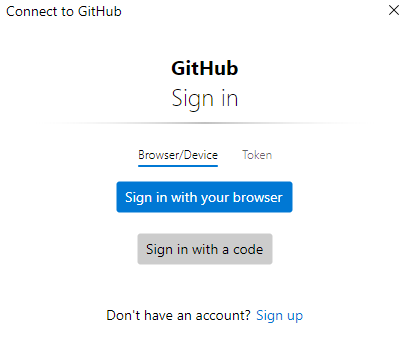
git remote add origin ССЫЛКА

ССЫЛКА



git push -u origin main

если выдаст ошибку, то попробуйте, main заменить на master

После этих команд он вас перекинет на страницу авторизации  


Авторизуйтесь и данные отправятся на git.