Подсистема для работы с партнерами

Описание предметной области Производственная компания «Мастер пол» выпускает напольные покрытия. Компания занимается производством и реализует свою продукцию через партнеров, которые доставляют продукцию компании до конечных потребителей.

С целью оптимизации деятельности компании разрабатывается система, в которой заказчик выделил несколько подсистем:

* продукция;
* склад и материалы;
* производство;
* сотрудники;
* партнеры.

В данной области определены следующие основные составляющие:

1. Партнеры: тип, наименование компании, юридический адрес, ИНН, ФИО директора, контактные данные (телефон, email), логотип, рейтинг, места продаж, история реализации продукции для последующего расчета скидок. Партнеры покупают продукцию со скидками, которые зависят от общего объема их продаж. Скидки мотивируют партнеров к большему объему продаж. Партнеры реализуют товары в розничных и оптовых магазинах, интернет-магазинах, другим компаниям по продаже товаров и оказанию услуг. Взаиморасчеты производственной осуществляются в рублях. компании с партнерами

2. Менеджеры: поиск и регистрация партнеров, решение об изменении рейтинга партнера, ведение истории изменений рейтинга, прием заявок от партнеров, формирование для них предложения на основании статистики истории продаж, полученной от аналитиков компании.

3. Заявка: если партнер ранее производил закупку продукции и предоставил данные о ее реализации, то на основании этих данных и текущих объемах готовой продукции менеджер формирует предложение. Если работа ведется с новым партнером, то менеджер предоставляет каталог готовой продукции и принимает заявку. Затем менеджер должен указать стоимость и дату производства каждой указанной в заявке единицы продукции. После согласования списка продукции с количеством, стоимостью и сроком изготовления заявка считается созданной. Партнер может отменить заявку до момента внесения предоплаты. Если предоплата не поступает в течение трех дней, то заявка отменяется автоматически, о чем обязательно уведомляется партнер. После поступления предоплаты продукция, указанная в заявке и отсутствующая на складе в нужном количестве, запускается в производство. По готовности всей продукции менеджер должен предложить организовать доставку. В момент получения продукции (путем доставки или самовывоза) партнер производит полную оплату. После проверки количества и качества продукции партнером заявка считается выполненной.

4. Сотрудники: ФИО, дата рождения, паспортные данные, банковские реквизиты, наличие семьи, состояние здоровья.

5. Кадры: допуск к работе с конкретным оборудованием. Мастер производства занимается технологией производства, контролем качества готовой продукции, заказом материалов у поставщиков.

6. Доступ: двери открываются при использовании карты сотрудника, система турникетов передает данные о перемещении сотрудников в приложение для формирования аналитиками статистики по сотрудникам.

7. Материалы (сырье): тип, наименование, поставщик, количество в упаковке, единица измерения, описание, изображение, стоимость, количество на складе, минимальное допустимое количество, история изменений количества материалов на складе.

8. Склад: регистрация поступлений материалов, резервирование материалов для производства, отпуск и списание материалов со склада в производство, поступление и выбытие готовой продукции, получение информации о текущих остатках и движении товарно-материальных ценностей по складу.

9. Поставщики: тип, наименование, ИНН, история поставок материалов. Менеджеры контролируют список поставщиков, аналитики формируют информацию о качестве поставляемых материалов для принятия решения о прекращении работы с определенными компаниями.

10. Продукция: артикул, тип, наименование, описание, изображение, минимальная стоимость для партнера, размер упаковки (длина, ширина, высота), вес без упаковки, вес с упаковкой, сертификат качества (скан документа), номер стандарта, история изменений минимальной стоимости для партнера, время изготовления, себестоимость, номер цеха, количество человек на производстве, необходимые материалы для производства.

Сейчас нам не нужны все сущности, только то, что необходимо для функционала

Функционал:

* просмотр списка партнеров;
* добавление/редактирование данных о партнере;
* просмотр истории реализации продукции партнером.

**Модуль 1 (Представление алгоритма)**

1. Создать блок-схему алгоритма работы программы

Изображение выглядит как диаграмма, зарисовка, Технический чертеж, План

Автоматически созданное описание

Рисунок 1 – Блок-схема основного алгоритма

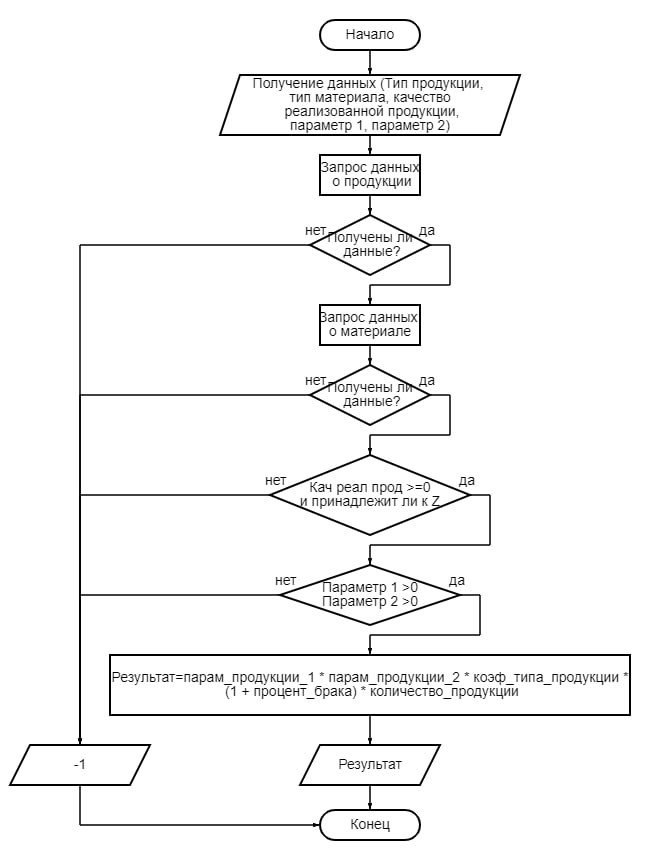


Рисунок 2– Блок-схема детализированного алгоритма

**Модуль 2 (Создание БД)**

1. Создать БД (3НФ). (Таблицы брать только те, что нужны для выполнения необходимой функциональности, то есть те данные, что нужно импортировать).

Для работы всех функций программного модуля необходимо наличие хорошо спроектированной базы данных. Таблицы в базе данных должны содержать связи между собой, а сама база данных должна находиться в 3НФ.

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых. Согласно определению 3НФ, неключевой атрибут таблицы не должен зависеть от других неключевых атрибутов этой же таблицы.

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа. Функционально полная зависимость означает, что если потенциальный ключ является составным, то атрибут зависит от всего ключа и не зависит от его частей.

Переменная отношения находится в первой нормальной форме [тогда и только тогда](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B3%D0%B4%D0%B0_%D0%B8_%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE_%D1%82%D0%BE%D0%B3%D0%B4%D0%B0), когда в любом допустимом значении отношения каждый его [кортеж](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

Наполнение таблиц

|  |  |
| --- | --- |
| Название | Поля |
| Тип материала (MaterialType)  Хранит информацию о типах материалов, используемых в производстве. | * ID (INT IDENTITY) * Тип материала (Title) (VARCHAR) * Процент брака (ScrapRate) (DECIMAL) |
| Тип продукции (ProductType)  Хранит информацию о различных типах продукции, которые производит компания. | * ID (INT IDENTITY) * Тип продукции (Title) (VARCHAR) * Коэффициент типа продукции (Ratio) (DECIMAL) |
| Партнер продукции (PartnerProducts)  Хранит информацию о продажах продукции партнерам. | * ID (INT IDENTITY) * Продукция (ProductID) (INT) * Наименование партнера (PartnerID) (VARCHAR) * Количество продукции (CountProduct) (INT) * Дата продажи (DateSale) (DATE) |
| Партнеры (Partners)  Хранит информацию о партнерах компании (поставщиках, клиентах, дистрибьюторах). | * ID (INT IDENTITY) * Тип партнера (Type) (VARCHAR) * Наименование партнера (Title) (VARCHAR) * Директор (Director) (VARCHAR) * Электронная почта партнера (Email) (VARCHAR) * Телефон партнера (Phone) (VARCHAR) * Юридический адрес партнера (LegalAddress) (VARCHAR) * ИНН (INN) (VARCHAR) * Рейтинг (Rating) (INT) |
| Продукция (Products)  Хранит информацию о конкретных продуктах, производимых компанией. | * ID (INT IDENTITY) * Тип продукции (ProductTypeID) (INT) * Наименование продукции (Title) (VARCHAR) * Артикул (INT) * Минимальная стоимость для партнера (MinPartnerPrice) (DECIMAL) |

Создание скрипта БД

Для начала необходимо подключиться к серверу и написать скрипт для создания пустой БД. После уведомления об успешном создании её можно найти в списке существующих БД.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3 - Скрипт создания БД компании «Master\_Paul»

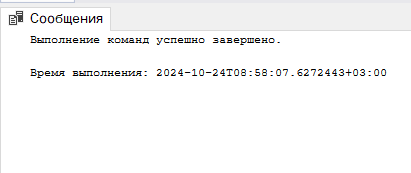


Рисунок 4 – Сообщение об успешно созданной базе

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 5 – Список существующих БД

Для успешного написания скрипта создания таблиц БД, важно учитывать наличие внешних и первичных ключей. При просмотре данных в Exel внешние ключи определяются по повторяющимся полям в разных таблицах.

Скрипт:

Use Master\_Paul; - обращение к БД

- Создание таблицы "Тип материала"

CREATE TABLE MaterialType (

ID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(255),

ScrapRate DECIMAL(7,2),

);

- Создание таблицы "Тип продукции"

CREATE TABLE ProductType(

ID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(255),

Ratio DECIMAL(7,2),

);

- Создание таблицы "Продукция"

CREATE TABLE Products (

ID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(255),

Article VARCHAR(255),

MinPartnerPrice DECIMAL(7,2),

ProductTypeID INT,

FOREIGN KEY (ProductTypeID) REFERENCES ProductType(ID) – внешний ключ

ON UPDATE CASCADE – допустимо обновление

ON DELETE CASCADE, – допустимо удаление

);

- Создание таблицы "Партнеры"

CREATE TABLE Partners (

ID INT IDENTITY(1,1) PRIMARY KEY,

TypePartners VARCHAR(255),

Title VARCHAR(255),

Director VARCHAR(255),

Email VARCHAR(255),

Phone VARCHAR(255),

LegalAddress VARCHAR(255),

INN VARCHAR(255),

Rating INT,

);

- Создание таблицы "Партнеры продукции"

CREATE TABLE PartnerProducts (

ID INT IDENTITY(1,1) PRIMARY KEY,

CountProduct INT,

DataSale DATE,

ProductID INT,

PartnerID INT,

FOREIGN KEY (ProductID) REFERENCES Products(ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY (PartnerID) REFERENCES Partners(ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

);

1. Получить ER-диаграмму (средствами СУБД). Представить в PDF формате. (скрин в ворд и сохранить в формате PDF)

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 6 - ER-диаграммa

1. Подготовить данные из EXEL для импорта и загрузить в базу (подготовка данных заключается в преобразовании некоторых полей во внешние ключи, и в преобразовании формата ячеек в некоторых полях).

Для того, чтобы грамотно перенести данные из Exel, необходимо:

1. Выбрать необходимую для работы таблицу.
2. Открыть таблицу в Excel.
3. Освободить первый столбец в таблице для ID, выделив данные и сместив их на 1 столбец вправо.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 7 – Перенос таблицы, для освобождения столбца ID

1. Сверить столбцы в Excel и в MsSQL (порядок столбцов). Если столбцы не сходятся, то изменить их в Excel.
2. Изменить данные, относящиеся к внешним ключам, на ID таблиц, внешний ключ которых приведен. (заменить текстовые названия на соответствующие идентификаторы из таблиц).

Например:

Необходимо обновить записи в таблицах, «Продукты», заменив текстовые названия продуктов на соответствующие идентификаторы из таблицы «Тип продуктов».

Представим, что в таблице «Тип продуктов» есть записи:

Изображение выглядит как текст, снимок экрана, линия, чек

Автоматически созданное описание

Рисунок 8 – Записи таблицы «Тип продуктов»

А в таблице «Продукты» есть записи:

Изображение выглядит как текст, снимок экрана, чек, линия

Автоматически созданное описание

Рисунок 9 – Записи таблицы «Продукты»

Задача - заменить названия продуктов в таблице «Продукты» на соответствующие идентификаторы из таблицы «Тип продуктов».

Изображение выглядит как линия, снимок экрана, чек, текст

Автоматически созданное описание

Рисунок 10 – Результат

1. Отредактировать столбцы таблицы по формату. (цифры – числовой тип)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 8 – Форматирование ячеек

1. Скопировать данные из Excel (без шапки) и вставить в первую строку в MsSQL. Далее нажать на треугольник \*, тем самым выделиться вся строка и вставить - CTRL + V.

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рисунок 9 – Копирование таблицы

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 10 – Вставка данных

1. Повторить все проделанные шаги для оставшихся таблиц.

Порядок заполнения таблиц (MaterialType, Partners, ProductType, Products, PartnerProducts). Если порядок будет изменен, то, данные могут быть вставлены не корректно, так как в некоторых таблицах есть ограничения по внешним ключам (в других вариантах ДЭ, порядок заполнения таблиц: сначала таблицы в которых нет ВК, затем таблицы в которых есть ВК).

Просмотр импортированных данных в таблицах:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 11 – Данные для таблицы «MaterialType»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 12 – Данные для таблицы «Partners»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13 – Данные для таблицы «ProductType»

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 14 – Данные для таблицы «Products»

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 15 – Данные для таблицы «PartnerProducts»

1. Создание резервной копии БД

Открываем MS SQL Management Studio. Кликаем правой кнопкой мыши по базе, для которой хотим сделать резервную копию - Задачи - Создать резервную копию:

В открывшемся окне оставляем полный тип копий и путь к резервному файлу (при необходимости, можно его поменять, удалив и создав снова. Можно указать как локальный диск, так и сетевой), также необходимо добавить имя файла:

После завершения процесса мы увидим сообщение «Резервное копирование базы ... успешно завершено».

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 16 – Создание резервной копии

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 17 – Добавление пути файла

**Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание**

Рисунок 18 – Уведомление об успешном создании Backup

1. Подключение БД к проекту
2. Создание нового приложения

Для создания нового приложения откроем Visual Studio и в меню справа нужно выбрать пункт «Создание проекта».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 19 – Создание проекта

Далее необходимо выбрать тип создаваемого приложения «Приложение WPF (.NET Framework)».

Изображение выглядит как текст, снимок экрана, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 20 – Выбор типа приложения

Далее указывается название приложения и его расположение, в поле «Платформа» ничего не меняем. Называть приложение нужно осознанно. Нажимаем на кнопку «Создать».

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 21 – Созданный проект

Перед созданием подключения к базе данных создадим папку, в которой будут храниться все данные, связанные с базой данных - модели данных, диаграмма и т. д.

Для создания папки нужно открыть контекстное меню для проекта найти пункт «Добавить» > «Создать папку».

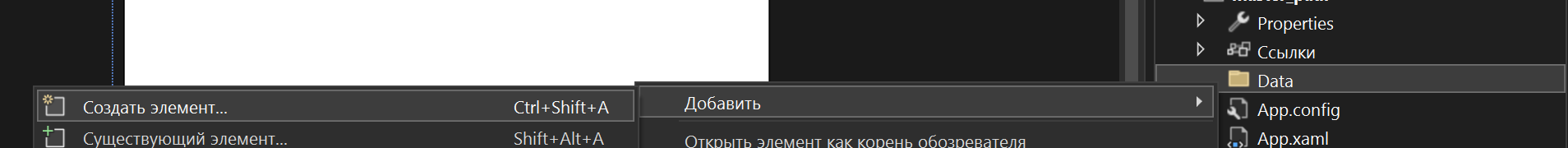
Изображение выглядит как снимок экрана, программное обеспечение, Мультимедийное программное обеспечение, текст

Автоматически созданное описание

Рисунок 22 – Создание папки «Data»

1. Создание модели данных:

* В папке проекта создайте новый элемент типа «Модель ADO.NET EDM».
* Переименуйте модель и нажмите «Добавить».



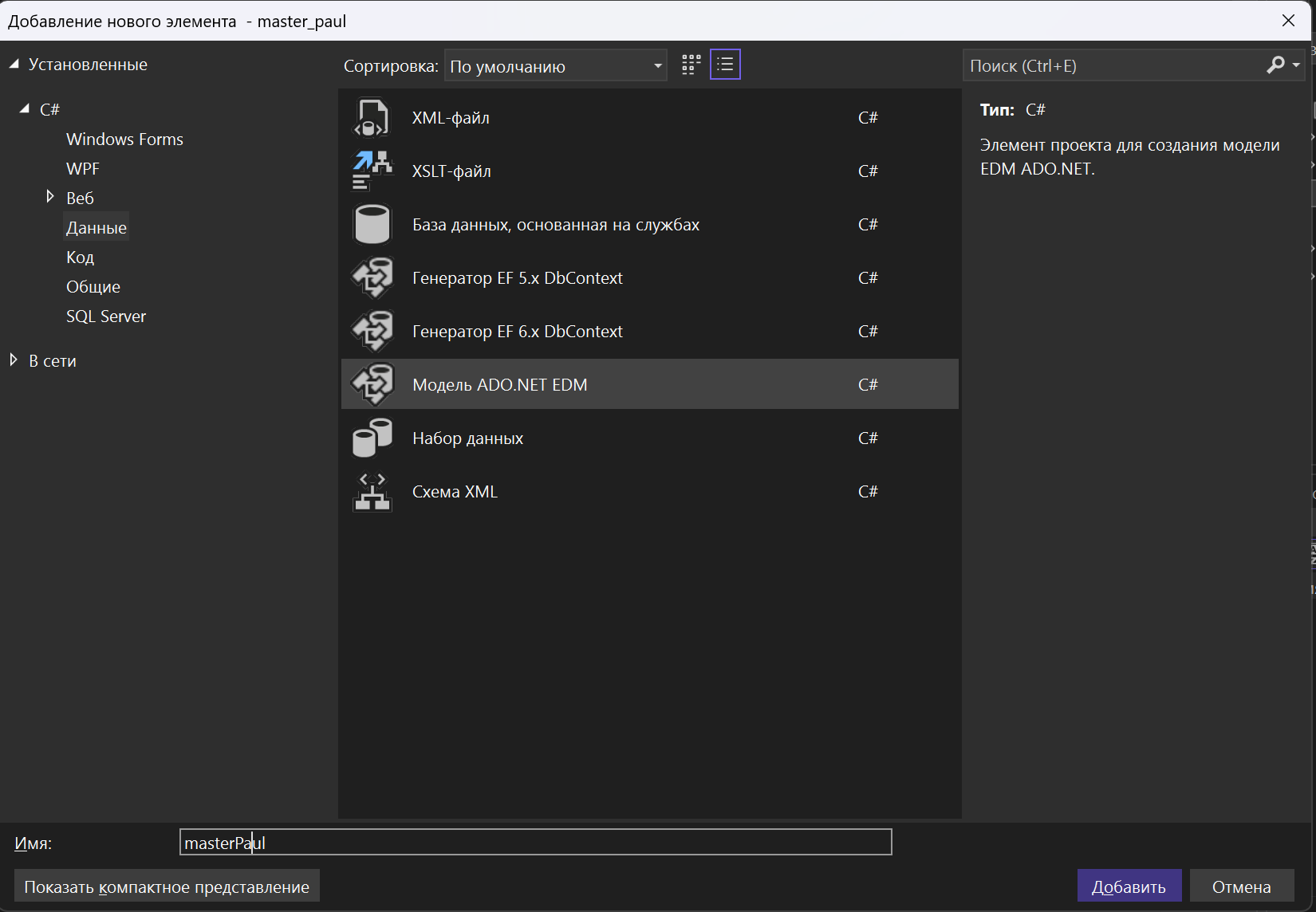


Рисунок 23 – Создание элемента типа «Модель ADO.NET EDM»

* Выберите «Конструктор EF из базы данных» в качестве источника модели.

Изображение выглядит как текст, снимок экрана, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 24 – Выбор «Конструктор EF из базы данных»

1. Создание подключения к базе данных:

* Если подключение еще не создано, нажмите «Создать подключение».
* Введите имя сервера, название базы данных и выберите тип проверки подлинности (SQL Server или Windows).
* Если выбрана «Проверка подлинности SQL Server», введите логин и пароль.
* После создания подключение появится в списке доступных подключений.
* Выберите, нужно ли сохранять логин и пароль в строке подключения. Для демонстрационного экзамена рекомендуется сохранить данные.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 25 – Создание подключения к БД

3. Выбор объектов базы данных:

* Выберите таблицы и хранимые процедуры, которые будут использоваться в модели.
* Нажмите «Готово».

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 26 – Выбор объектов базы данных

1. Создание модели:

* Мастер моделей закроется, и начнется подключение к базе данных.
* Подтвердите диалоговые окна, нажимая "ОК".
* После успешного подключения откроется окно с диаграммой базы данных в Visual Studio.
* В файле App.config появится строка подключения к БД.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 27 – Диаграмма БД

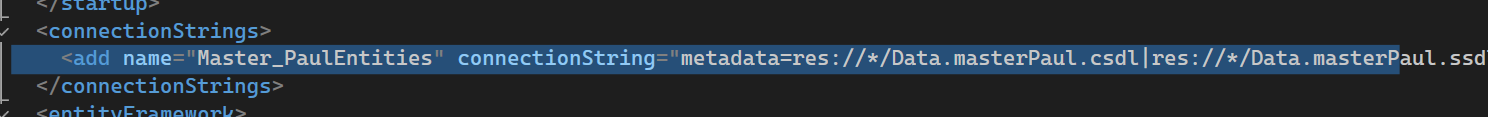


Рисунок 28 – App.config строка подключения к БД

1. Обновление модели:

* Если вы внесли изменения в структуру базы данных (переименовали поля, изменили типы данных, удалили связи), необходимо переподключиться к базе данных.
* Удалите строку подключения из App.config и файл с расширением .edmx из папки проекта.
* Повторите шаги 1-4, чтобы создать новое подключение и обновить модель.

**Модуль 3 (Проектирование интерфейса)**

Руководство по стилю

* Все экранные формы пользовательского интерфейса должны иметь заголовок.
* Кроме того, на главной форме должен быть установлен логотип (представлен в ресурсах). Логотип не искажать: не менять изображение, пропорции изображения, цвет.
* Для приложения должна быть установлена иконка (дана в ресурсах), если это реализуемо в рамках платформы.
* Использовать шрифт Segoe UI.
* В качестве основного фона используется белый цвет
* дополнительного фона используется цвет #F4E8D3.
* Для акцентирования внимания пользователя на целевом действии интерфейса используется цвет #67BA80.

|  |  |  |
| --- | --- | --- |
| Основной фон | Дополнительный фон | Акцентирование внимания |
| #FFFFFF | #F4E8D3 | #67BA80 |
|  |  |  |

Разработка интерфейса программного модуля для работы с партнерами

1. Интерфейс должен быть последовательным с возможностью переключения между окнами (например кнопка «Назад»)
2. Заголовок окна должен соответствовать назначению (На всех окнах title меняем на то, что там происходит)
3. Параметры должны быть грамотно названы и соответствовать стилю CamelCase
4. Вывод списка партнеров должен быть представлен согласно макету

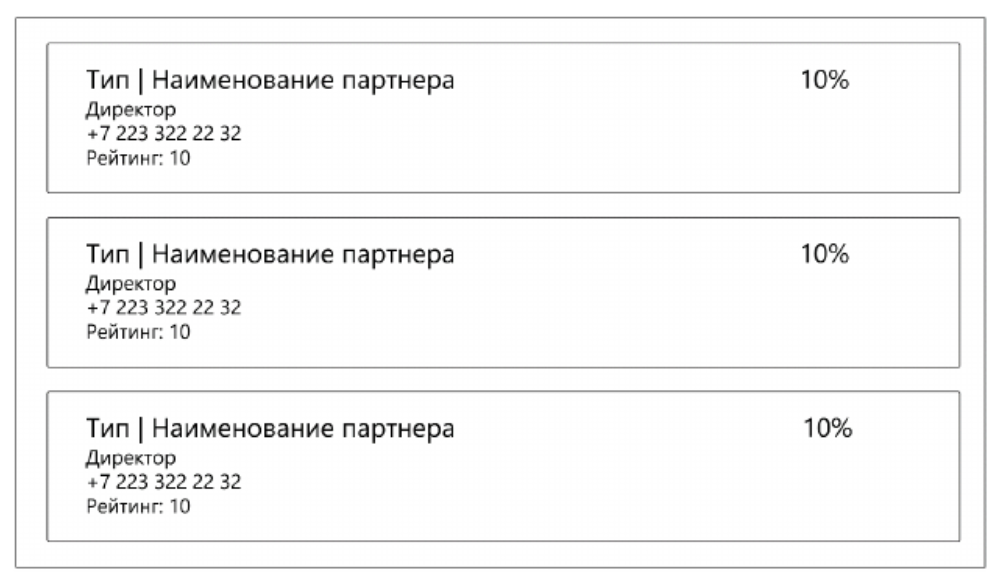


Рисунок 29 – Макет приложения

1. Установка иконки приложения

Для ее установки в обозревателе решений необходимо выбрать проект и открыть для него контекстное меню, в котором выбрать пункт «Свойства».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 30 – Выбор пункта «Свойства проекта»

После чего откроется окно со свойствами проекта. Нам нужен раздел «Приложение», в котором находим пункт «Значок и манифест», далее нажимаем кнопку «Обзор» и в диалоговом окне выбираем файл с расширением ico.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 31 – Выбор файла иконки приложения

Для установки иконки подходят только файлы с расширением ico.

Далее, чтобы поместить картинку в ресурсы, нужно выполнить следующие действия.

* Откроем свойства проекта и перейдем в раздел «Ресурсы».
* В разделе «Ресурсы» на панели инструментов выберем пункт «Добавить ресурс» > «Добавить существующий файл» и в открывшемся диалоговом окне выберем нужный(ые) файл(ы).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 32 – Добавление файла в ресурсы

* Чтобы наши файлы отображались во время работы приложения, нужно открыть свойства файла и для свойства «Действие при сборке» установить значение «Resource».

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 33 – Добавление свойства «Действие при сборке»

1. Шапка приложение и разметка Grid

Первое с чего стоит начать это установить размеры окна и дать форме название:

Title="Мастер Пол - Партнеры"

Height="450" Width="800"

MinHeight="400" MinWidth="600"

Далее необходимо выделить область для шапки (данная область выделяется внутри основного Grid):

Данный код разделяет Grid построчно, относительными размерами. ()

<Grid.RowDefinitions>

<RowDefinition Height="\*" MaxHeight="100"/> - 1/5 пространства

<RowDefinition Height="5\*"/> - 5 частей пространства = \*

</Grid.RowDefinitions>

Далее в контейнер DockPanel необходимо поместить элементы шапки: лого и название компании.

<DockPanel Grid.Row="0" – начиная с первой выделенной части пространства

Background="AliceBlue">

<Image Source="Resources\Мастер пол.png"

DockPanel.Dock="Left" – картинку расположить слева

Margin="10"

VerticalAlignment="Center"/> - выровнять вертикально по центру

<TextBlock Text="Мастер пол"

FontSize="28"

FontWeight="Bold"

VerticalAlignment="Center"/>

</DockPanel>

Затем стоит разделить пространство которые мы выше обозначили как 5\*, то есть то, что находится ниже шапки.

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="15"/> <!--Отступ сверху и снизу, и всё пространство по середине-->

<RowDefinition Height="\*"/>

<RowDefinition Height="15"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="15"/> <!--Отступ справа и слева, и всё пространство по середине-->

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="15"/>

</Grid.ColumnDefinitions>

Изображение выглядит как снимок экрана, текст, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 34 – Шапка приложения

Далее идет добавление компонента ListView. Сначала определяем компонент, выводим его расположение относительно сделанной ранее сетки и задаем общий стиль для одного компонента списка.

<!--выбор контейнера {1}{1} строка/столбец (num = 0)-->

<ListView x:Name="ProductListView"

Grid.Row="1"

Grid.Column="1"

ScrollViewer.HorizontalScrollBarVisibility="Disabled">

<ListView.ItemContainerStyle> <!--Стиль для контейнера (1 партнер)-->

<Style TargetType="ListViewItem">

<Setter Property="HorizontalContentAlignment"

Value="Stretch"/> <!--Растянуть по горизонтали-->

<Setter Property="Margin" Value="0,0,0,10"/>

</Style>

</ListView.ItemContainerStyle>

Далее делаем внутреннюю разметку элемента списка, также выводим из базы поля (те, что указаны на макете). Но просто через XAML их вывести не получиться, нужно определить для них поля, внутри программы.

<ListView.ItemTemplate>

<DataTemplate>

<Border BorderThickness="1"

BorderBrush="Black">

<Grid Margin="10"

HorizontalAlignment="Stretch">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="4\*"/>

<ColumnDefinition Width="\*" MaxWidth="170" MinWidth="135"/>

</Grid.ColumnDefinitions>

<!--Заполнение колонок одного партнера-->

<StackPanel Grid.Column="0"

Margin="5, 0"

Orientation="Vertical"

VerticalAlignment="Stretch">

<TextBlock Text="{Binding TypePartners}"

TextWrapping="Wrap"/> <!--Перенос длинного текста на след строку-->

<TextBlock Text="{Binding Title}"/> - вывод поля БД

</StackPanel>

</Grid>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

Создание поля списка партнеров, получение полей БД и заполнение ListView данными.

public partial class MainWindow : Window

{

private IEnumerable<Partners> \_parnersList; //создаем поле списка партнеров

public MainWindow()

{

InitializeComponent();

// Получение партнеров.

\_parnersList = Core.GetContext().Partners.ToList();

// Заполнение ListView данными.

ProductListView.ItemsSource = \_parnersList;

}

}

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Рисунок 35 – Макет на начальном этапе

После проверки работоспособности приложения можно приступать к форматированию и редактированию элементов списка. Расположить внутри них текст так, как показано в макете. Для этого эффективнее всего воспользоваться вычислительными полями (создать поле внутри которого можно сложить несколько полей таблиц).

Для реализации вычислительных полей необходимо создать класс, описывающий их структуру. Для удобства организации кода рекомендуется создать папку «Classes» в проекте и поместить в нее созданный класс. (Файловая структура проекта оценивается на ДЭ).

Детали:

* Пространство имен: изменим пространство имен (namespace) нового файла на .Data, чтобы он находился в той же папке, где расположены все остальные классы моделей данных (правильное пространство имен можно посмотреть нажав CTRL при наведении на созданное вычислительное поле).
* Частичный класс: объявим класс Product как partial (частичный). Это позволит нам добавить новые поля и методы к существующему классу, определенному в файле, генерируемом при подключении к базе данных.

Почему это важно?

Когда вы переподключаетесь к базе данных, файлы моделей данных пересоздаются. Если вы добавите новые поля и методы непосредственно в файл, сгенерированный при подключении, они будут удалены при переподключении. Создание отдельного файла с частичным классом позволяет вам добавлять функциональность к существующему классу без риска потери этих дополнений при обновлениях.

Создание поля «TypeAndName» представляет собой два совмещенных поля (Тип партнера и Название партнера), а также необходимо добавить поле «RatingString», для соответствия макету.

namespace master\_paul.Data

{

public partial class Partners

{

public string TypeAndName => TypePartners + " | " + Title; //нажать ctrl и зайти в глубь, там посмотреть namespace (важно!!!)

public string RatingString => "Рейтинг:" + Rating;

}

}

После того как вычислительные поля созданы, необходимо, изменить их в макете, а также дополнить недостающими.

<!--Заполнение колонок одного партнера-->

<StackPanel Grid.Column="0"

Margin="5, 0"

Orientation="Vertical"

VerticalAlignment="Stretch">

<TextBlock Text="{Binding TypeAndName}"

TextWrapping="Wrap"/>

<!--Перенос длинного текста на след строку-->

<TextBlock Text="{Binding Director}"

TextWrapping="Wrap"/>

<TextBlock Text="{Binding Phone}"

TextWrapping="Wrap"/>

<TextBlock Text="{Binding RatingString}"

TextWrapping="Wrap"/>

</StackPanel>

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 36 – Вывод полей партнера

Также в макете присутствует вывод скидки партнера. По ТЗ она рассчитывается так:

* Если количество продуктов меньше 10 000, то скидка равна 0%.
* Если количество продуктов от 10 000 до 49 999, то скидка равна 5%.
* Если количество продуктов от 50 000 до 299 999, то скидка равна 10%.
* Если количество продуктов больше или равно 300 000, то скидка равна 15%.

Для реализации данной функциональной возможности, необходимо добавить в разметку дополнительную колонку. И изменить соотношение колонок в Grid.

Добавление колонки:

<StackPanel Grid.Column="1"

Margin="5, 0"

Orientation="Vertical"

VerticalAlignment="Stretch">

<TextBlock Text="{Binding Discount}"

TextWrapping="Wrap"/>

</StackPanel>

Изменение Grid:

<Grid.ColumnDefinitions>

<ColumnDefinition Width="8\*"/>

<ColumnDefinition Width="\*" MaxWidth="170" MinWidth="142"/>

</Grid.ColumnDefinitions>

В созданную колонку было помещено поле «Discount», которое вычисляется следующими методами:

private static int GetProductCount(int partnerId)//получение кол-ва продуктов данного партнера

{

//ForEach - перебирает записи по порядку и смотрит совпадают ли ID (номер текущей записи и partnerId). дальше сумму считает

int sum = 0;//накопительная переменная

Core.GetContext().PartnerProducts.ToList().ForEach(product => {

if (product.PartnerID != partnerId)

return;//если не совпадает то пропускаем

sum += product.CountProduct.GetValueOrDefault(0);

//GetValueOrDefault - так как CountProduct(может быть null), и если все таки оно null, то мы считаем как 0.

});

return sum;

}

private static int GetDiscount(int partnerId) // метод расчета скидки

{

var productCount = GetProductCount(partnerId);//получение кол-ва продуктов данного партнера

if (productCount < 10000)//расчет скидки по ТЗ

{

return 0;

}

if(productCount > 10000 && productCount < 49999)

{

return 5;

}

if(productCount > 50000 && productCount < 300000)

{

return 10;

}

return 15;

}

public string Discount => GetDiscount(ID) + "%"; // форматирование вывода

Оба метода работают с данными о партнерах и их продуктах.

GetProductCount(int partnerId) - тот метод подсчитывает общее количество продуктов, связанных с определенным партнером, заданным идентификатором partnerId.

Принцип работы:

1. Получает список всех продуктов из базы данных (через Core.GetContext().PartnerProducts.ToList()).
2. Перебирает каждый продукт в списке.
3. Если идентификатор продукта (product.PartnerID) совпадает с partnerId, то:
4. Извлекает значение product.CountProduct.
5. Если CountProduct не равно null, то прибавляет его к накопительной переменной sum.
6. Если CountProduct равно null, то прибавляет 0 к sum.
7. Возвращает накопленное значение sum – общее количество продуктов партнера.

GetDiscount(int partnerId) – этот метод определяет скидку, которую получает партнер в зависимости от количества его продуктов.

Принцип работы:

1. Сначала вызывает метод GetProductCount(partnerId), чтобы получить количество продуктов партнера.
2. Затем выполняет проверку количества продуктов и вычисляет скидку.

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 37 – Вывод полной информации о партнере

Следующим шагом является добавление в интерфейс кнопок «Добавить и Редактировать». Кнопка «Добавить» в главной форме отображается всегда, а кнопка «Редактировать», появляется при выборе партнера.

Для начала необходимо задать общие стили приложения, чтобы не описывать стили для каждого элемента по несколько раз. Описываются они в файле «App.xml».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 38 – Общие стили приложения

Теперь можно приступать к добавлению кнопок. Обе кнопки располагаются в шапке приложения.

У кнопки редактирования установлено свойство Collapsed, которое указывает на то что, элемент не виден, но участвует в компоновке.

<StackPanel HorizontalAlignment="Right" Orientation="Horizontal">

<Button x:Name="AddBtn"

Width="180"

Content="Добавить продукт"

Margin="10,10"

Click="AddBtn\_Click"/>

<!--Collapsed - элемент не виден и не участвует в компоновке.-->

<Button x:Name="EditBtn"

Grid.Column="2"

Content="Редактировать"

Margin="0,0,15,0"

Padding="0"

Visibility="Collapsed"

HorizontalAlignment="Right"

Click="EditBtn\_Click"/>

</StackPanel>

Также необходимо прописать кнопке редактирования метод, который будет выводить её при выборе партнера.

private void PartnerListView\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ProductListView.SelectedItems.Count > 0)

{

EditBtn.Visibility = Visibility.Visible;

}

}

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 39 – Готовый макет главной формы

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рисунок 40 – Готовый макет главной формы с кнопкой редактирования

Создание формы «Добавления/Редактирования»

Для начала необходимо определить как должна выглядеть форма и в чем различие её функциональных возможностей при открытии через кнопку «Добавление» и при открытии через кнопку «Редактирование».

При открытии формы через кнопку «Добавление», открывается форма с НЕ заполненными полями с возможностью сохранения и добавления данных в БД, а при открытии через кнопку «Редактирование», открывается форма с уже заполненными полями выбранного партнера с возможностью сохранения изменений в БД.

Для сохранения файловой структуры сначала необходимо создать папку «Windows», где будут храниться окна проекта, а затем поместить в эту папку окно «Добавления/Редактирования», присвоив ему название «Add\_Edit».

Сначала создадим новое окно, которое будет отвечать за добавление и редактирование продуктов. Укажем название и минимальные размеры окна. копируем разметку «шапки» приложения из главного окна.

Title="Партнеры - добавление и редактирование"

Height="450" Width="500"

MinHeight="400" MinWidth="500">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*" MaxHeight="100"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<DockPanel Grid.Row="0"

Background="#67BA80">

<Image Source="D:\it\4 курс\демка\ДЭ25\master\_paul\master\_paul\Resources\Мастер пол.png"

DockPanel.Dock="Left"

Margin="10"

VerticalAlignment="Center"/>

<TextBlock Text="Мастер пол"

FontSize="28"

FontWeight="Bold"

VerticalAlignment="Center"/>

</DockPanel>

</Grid>

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 41 – Макет формы «Добавления/Редактирования»

Теперь необходимо добавить поля для ввода информации:

Поля:

* Наименование
* Тип партнера (Выпадающий список)
* Рейтинг (int > 0) компании
* Адрес компании
* ФИО директора
* Телефон компании
* Email компании
* ИНН

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 42 – Оформление СomboBox

Данные в выпадающий список не берутся из базы, а прописываются по следующим причинам:

1. Возможно повторение типов: при добавлении новых записей в таблицу, в списке будут дублироваться типы партнеров, например «ЗАО». Это затрудняет выбор нужного типа и ухудшает визуальное восприятие списка.
2. Возможно ограничение на создание новых типов: если в базе данных отсутствует запись о партнере с определенным типом, например, «ОАО», в дальнейшем станет невозможно создавать партнеров с этим типом.

Код для вывода полей:

<!--Колонка с полями для заполнения-->

<StackPanel Grid.Row="1"

Margin="15">

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="Наименование"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding Title}" MaxLength="10"/>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="Тип партнера"

Margin="0,0,5,0"

Width="160"/>

<ComboBox x:Name="TypeComboBox"

SelectedItem="{Binding TypePartners}">

<ComboBoxItem>ЗАО</ComboBoxItem>

<ComboBoxItem>ООО</ComboBoxItem>

<ComboBoxItem>ПАО</ComboBoxItem>

<ComboBoxItem>ОАО</ComboBoxItem>

</ComboBox>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="Рейтинг компании"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding Rating}" MaxLength="10"/>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="Арес компании"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding LegalAddress}" MaxLength="10"/>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="ФИО Директора"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding Director}" MaxLength="10"/>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="Телефон компании"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding Phone}" MaxLength="10"/>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="Email компании"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding Email}" MaxLength="10"/>

</DockPanel>

<DockPanel Margin="0,10,0,0">

<TextBlock DockPanel.Dock="Left"

Text="ИНН"

Margin="0,0,5,0"

Width="160"/>

<TextBox Text="{Binding INN}" MaxLength="10"/>

</DockPanel>

Код для вывода кнопок очистки полей, добавления партнера и перемещения на главную форму:

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Right"

Margin="0,20,15,0">

<Button x:Name="DeleteBtn"

Width="90"

Content="Очистить"

Visibility="Visible"

Margin="0,0,10,0"

Click="DeleteBtn\_Click"/>

<Button x:Name="SaveBtn"

Width="100"

Content="Сохранить"

Click="SaveBtn\_Click"/>

<Button x:Name="backBtn"

Width="100"

Content="Назад"

Margin="10,0,0,0"

Click="SaveBtn\_Click"/>

</StackPanel>

</StackPanel>

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 42 – Макет формы «Добавления/Редактирования»

На кнопку «Назад», необходимо повесить обработчик закрытия окна:

private void BackBtn\_Click(object sender, RoutedEventArgs e)

{

Close();

}

**Модуль 4 (Написание логики программы)**

Сформировать алгоритм подсистемы для работы с партнерами. (Дополнительно реализовать алгоритм функции расчёта индивидуальной скидки для клиента)

Пункты, которые необходимо выполнить:

1. Подключить БД к приложению
2. Использовать модальные окна с заголовком и иконкой для отображения информации об ошибках и успешных действиях
3. Добавить подсказки в поля ввода (например маску ввода)
4. Позаботиться о наличии комментариев
5. Реализовать обработку try-catch (удобнее всего сделать при добавлении в базу нового партнера)
6. Главной должна быть форма списка партнеров. На ней должна быть кнопка «Добавить нового партнера». При нажатии на конкретный элемент должна быть кнопка «Редактировать данные партнера»
7. Расчет скидки партнера для вывода на главную форму (в зависимости от количества реализованной продукции):
   * до 10 000–0%
   * от 10 000 до 50 000–5%
   * от 50 000 до 300 000–10%
   * от 300 000–15%
8. Создать новое окно (страницу) для добавления/редактирования данных о партнере:
   1. Поля:

* Наименование
* Тип партнера (Выпадающий список)
* Рейтинг (int > 0) компании
* Адрес компании
* ФИО директора
* Телефон компании
* Email компании
* ИНН

1. Сделать кнопку для вывода истории реализации продукции для каждого партнера
2. Создать новое окно, где будет указано какого товара, продали, сколько товара продали и когда его продали (списком как на главном окне с др. данными)
3. Выполнить отладку и тестирование модуля
4. Разработать метод расчета количества материала для производства:
   1. Метод получает следующие данные
      1. Тип продукции (целое число)
      2. Тип материала (целое число)
      3. Количество реализованной продукции (целое число)
      4. 2 параметра продукции (вещественные положительные числа числа)
   2. Метод возвращает следующие данные
      1. Целое число - количество материала с учетом возможного брака.
      2. Если не существует тип продукции, тип материала или некорректно получены др. значения, то метод возвращает -1
   3. Расчет количества материала
      1. парам\_продукции\_1 \* парам\_продукции\_2 \* коэф\_типа\_продукции \* (1 + процент\_брака) \* количество\_продукции
5. Загрузить исходный код метода в отдельный репозиторий с названием проекта

Объяснение метода:

Представьте, что вы работаете на фабрике, которая производит разные вещи, например стулья и столы. Вам нужно знать, сколько материала (например, дерева) потребуется для изготовления определенного количества продукции.

Вот как работает метод расчета:

1. Ввод данных: Вам нужно ввести информацию о том, что вы хотите произвести:

- Тип продукции: Стул или стол?

- Тип материала: Дерево, металл или пластик?

- Количество продукции: Сколько стульев или столов вам нужно сделать?

- Параметры продукции: например, для стула - высота сиденья и ширина спинки, для стола - длина и ширина столешницы.

2. Проверка данных: Метод проверяет, чтобы введенная информация была корректной. Например, если вы выбрали "дерево" как тип материала, но ввели отрицательную ширину столешницы, метод покажет ошибку (вернет -1).

3. Расчет материала на одну единицу продукции: Метод использует формулу:

- Параметры продукции (длина ширина) коэффициент типа продукции.

- Коэффициент типа продукции учитывает, сколько материала нужно для изготовления одной единицы продукции (например, для стула коэффициент может быть 2, а для стола - 3).

4. Учет брака: Метод учитывает, что часть материала может быть испорчена во время производства. Процент брака зависит от типа материала:

- Например, для дерева процент брака может быть 5%, а для металла - 10%.

- Метод увеличивает количество материала на этот процент, чтобы компенсировать потери.

5. Расчет общего количества материала:

- Метод умножает количество материала на одну единицу продукции на количество продукции.

- Например, если для изготовления одного стула нужно 5 метров дерева, а вам нужно сделать 10 стульев, то общее количество материала будет 50 метров.

6. Результат: Метод возвращает целое число - общее количество материала с учетом возможного брака.

Пример:

Допустим, вы хотите сделать 20 столов из дерева, длина столешницы 1.5 метра, ширина 0.8 метра. Коэффициент для стола - 3, процент брака для дерева - 5%.

Расчет:

1. Материал на один стол: 1.5 метра 0.8 метра 3 = 3.6 метра дерева.

2. Учет брака: 3.6 метра + (3.6 метра \* 0.05) = 3.78 метра дерева на один стол.

3. Общее количество: 3.78 метра \* 20 столов = 75.6 метра дерева.

Результат: Метод вернет число 76 (округляем в большую сторону), т.е. вам потребуется 76 метров дерева, чтобы сделать 20 столов.

public partial class Partners

{

private static int GetDiscount(int partnerId)- метод

{

int sum = 0;

Core.GetContext().PartnerProducts.ToList().ForEach(product => {

if (product.ID != partnerId)

return;

sum += product.CountProduct.GetValueOrDefault(0);

});

if(sum < 10000)

{

return 0;

}

if(sum < 50000)

{

return 5;

}

if(sum < 300000)

{

return 10;

}

return 15;

}

public string Discount => GetDiscount(ID) + "%";

Итог:

1. Исходный код приложения (структура с файлами, не архив)
2. Исполняемые файлы (.exe)
3. Файл скрипта БД
4. Прочие графические, текстовые файлы (блок-схема, если будем делать и pdf с er-диаграммой)