

1. Работа с SSMS

1. Откроем SSMS и подключимся к серверу.
2. Зайдем в свою базу данных WFTutorial и создадим в ней новую таблицу DealSet, которую будем использовать для работы со сделками.
3. Зададим ключевое поле Id счетчиком (auto_increment).


	Имя столбца	Тип данных	Разрешить ...
	Id	int	<input type="checkbox"/>
	IdSupply	int	<input type="checkbox"/>
	IdDemand	int	<input type="checkbox"/>

Рисунок 1 – Таблица DealSet

4. Обратите внимание, что столбцы IdSupply, IdDemand являются внешними ключами к таблицам SupplySet, DemandSet.
5. Чтобы сделать связь между таблицами, необходимо нажать на любое поле в проекте таблицы и выбрать «Отношения...».

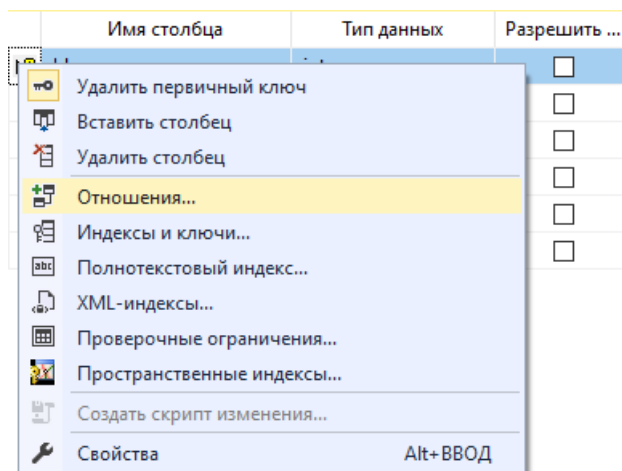


Рисунок 2 – Контекстное меню таблицы

6. В открывшемся окне нажимаем кнопку «Добавить»:

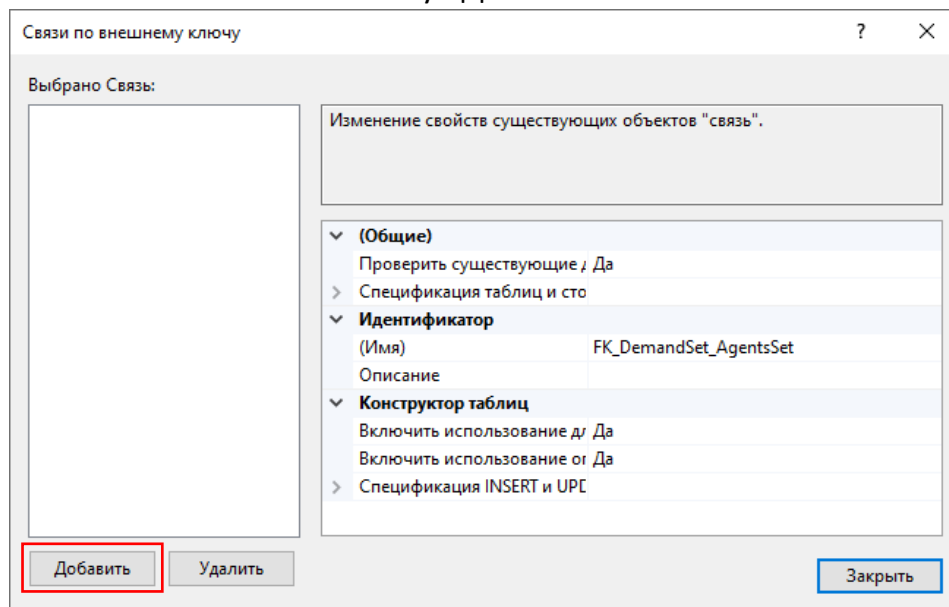


Рисунок 3 – Окно «Связи по внешнему ключу»

7. У появившейся связи выбираем «Спецификация таблиц и столбцов»:

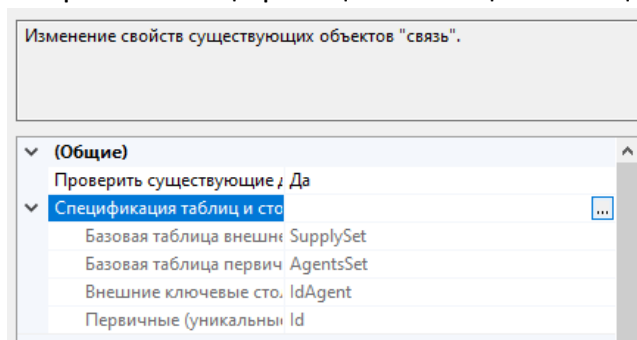


Рисунок 4 – Окно «Связи по внешнему ключу»

8. Настраиваем связь с таблицей SupplySet и нажимаем ОК:

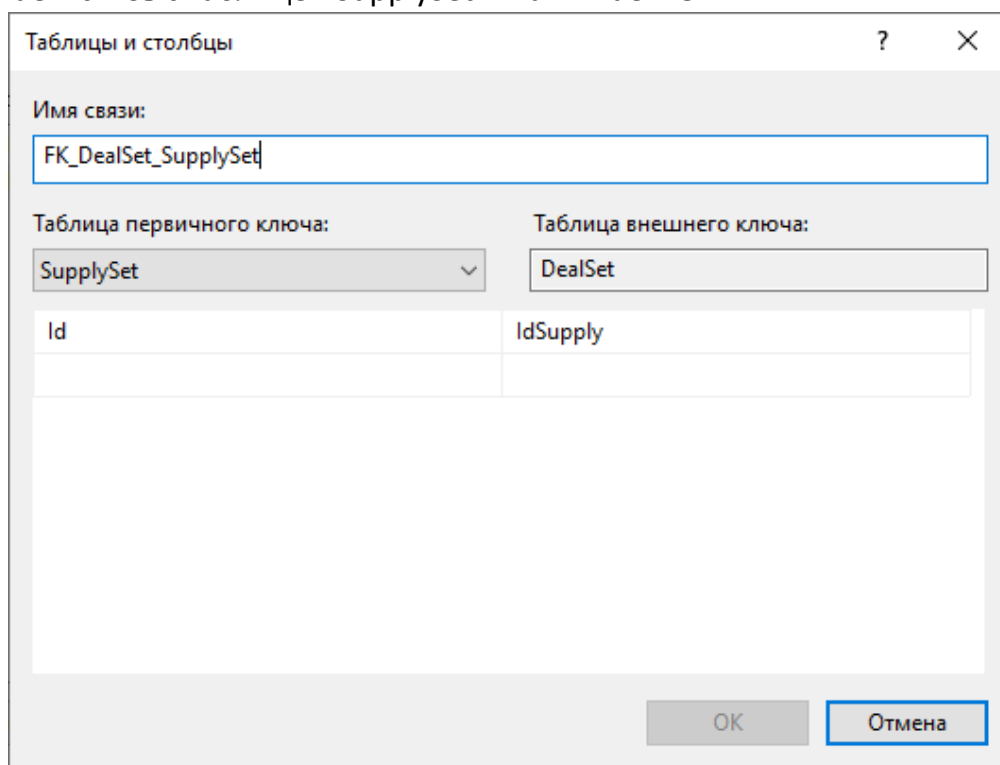


Рисунок 5 – Создание внешнего ключа к таблице SupplySet

9. Добавляем вторую связь и настраиваем внешний ключ к таблице DemandSet:

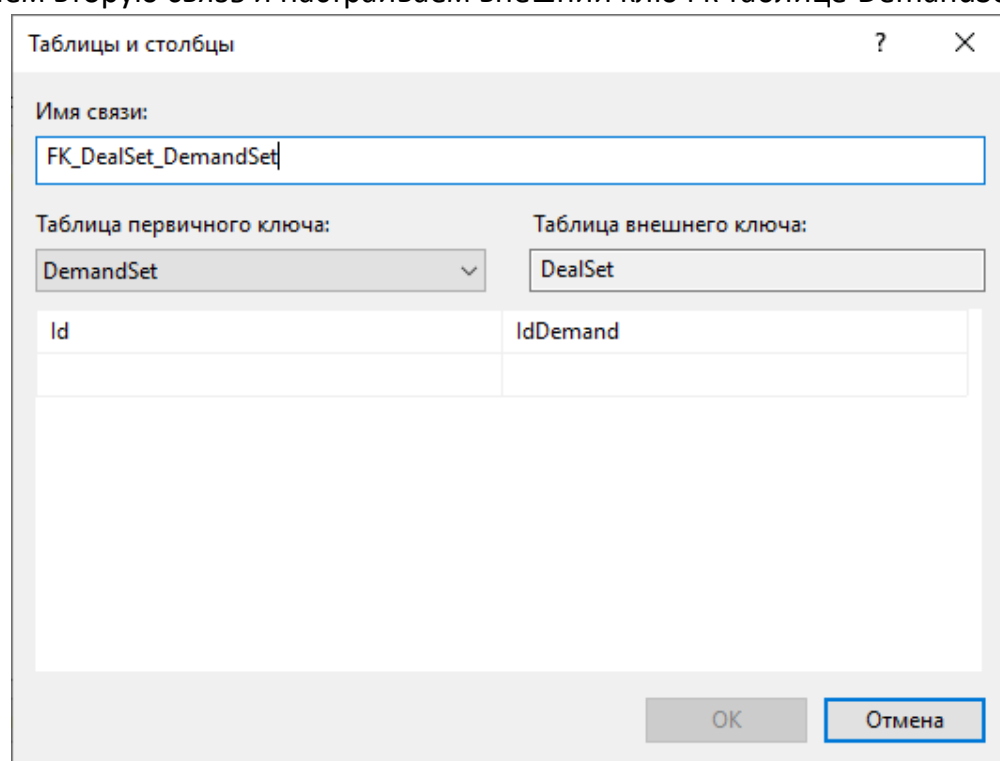


Рисунок 6 – Создание внешнего ключа к таблице DemandSet

10. Теперь окно «Связи по внешнему ключу» выглядит так:

Связи по внешнему ключу

Выбрано Связи:

FK_DealSet_DemandSet
FK_DealSet_SupplySet

Изменение свойств существующих объектов "связь".

Общие

Проверить существующие / Да

Спецификация таблиц и сто

Идентификатор

(Имя)FK_DealSet_SupplySet

Описание

Конструктор таблиц

Включить использование д/ Да

Включить использование от Да

Спецификация INSERT и UPC

Добавить

Удалить

Закреть

Рисунок 7 – Окно «Связи по внешнему ключу»

2. Работа с формой «Предложения»

11. Переходим в Visual Studio, откроем проект Esoft_Project.
12. Удалим уже созданную модель ModelTutorial.edmx и создадим ее снова (если вы не помните, как это сделать, обратитесь к документу https://vyatsu-my.sharepoint.com/:b:/g/personal/usr11033_vyatsu_ru/EYYIUb5DNWZBvycCVy3hvNwBw0II_BBRRhZJt11_FUqw_jg?e=cNdxB1).
13. Теперь в модели появились связи, и она выглядит так:

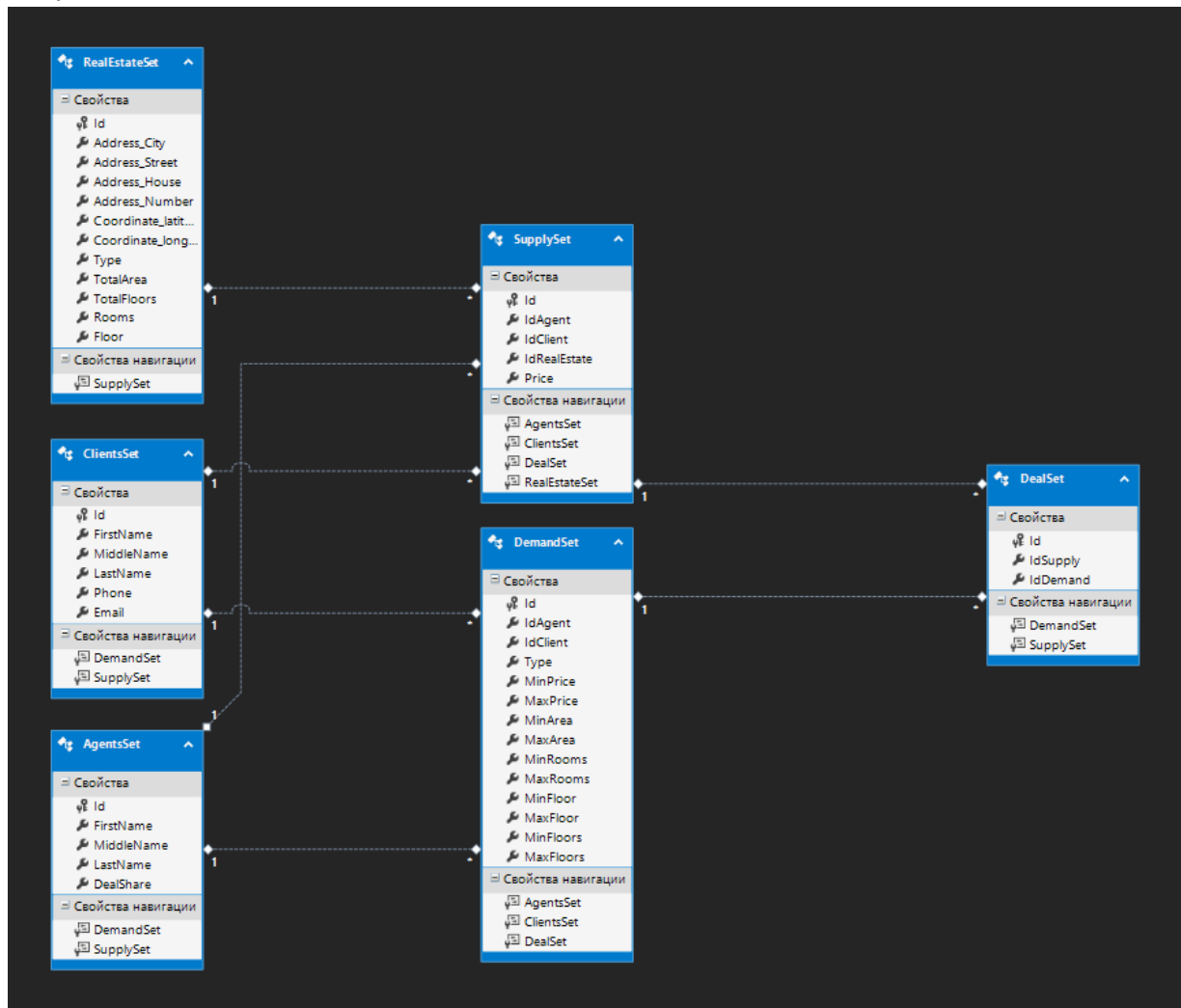


Рисунок 8 – Модель системы

14. Далее создаём форму, на которой будет располагаться интерфейс для сделок, для этого перейдём в вкладку «Проект», затем «Добавить форму». Зададим ей название FormDeal.
15. Сделаем ссылку на эту форму с кнопки «Сделки»:
16. Перейдем снова на форму «Сделки». Добавим необходимые элементы из «Панели элементов». Большинство элементов вам уже знакомы.

labelSupply

comboBoxSupply

labelDemand

comboBoxDemand

labelSellerCompanyDeductions

textBoxSellerCompanyDeductions

labelAgentSellerDeductions

textBoxAgentSellerDeductions

labelCustomerCompanyDeductions

textBoxCustomerCompanyDeductions

labelAgentCustomerDeductions

textBoxAgentCustomerDeductions

Сделки

Предложение

Потребность

Стоимость услуг для клиента-продавца

Отчисления риелтору клиента-продавца

Стоимость услуг для клиента-покупателя

Отчисления риелтору клиента-покупателя

Клиент-продавец

Риелтор клиента-продавца

Клиент-покупатель

Риелтор клиента-покупателя

Адрес объекта недвижимости

Стоимость

Создать

Изменить

Удалить

listViewDealSet

Рисунок 9 – Вид формы «Предложения»

17. Настроим вывод необходимой информации в comboBox-ax.
18. Создадим новый метод ShowSupply, который будет выводить информацию о предложениях.

```
void ShowSupply()
{
    //очищаем comboBox
    comboBoxSupply.Items.Clear();
    foreach (SupplySet supplySet in Program.wftDb.SupplySet)
    {
        //добавляем информацию, которую хотим видеть в строке comboBox-а
        //можно настроить по своему усмотрению, добавить/удалить некоторые элементы и сокращения
        //главное, не убирать Id, так как он нужен для дальнейшей работы
        string[] item = {supplySet.Id.ToString() + ". ", "Риелтор: "+supplySet.AgentsSet.LastName, "Клиент: "+supplySet.ClientsSet.LastName};
        comboBoxSupply.Items.Add(string.Join(" ", item));
    }
}
```

Рисунок 10 – Метод ShowSupply

Здесь можно изменять выводимую информацию в зависимости от ваших предпочтений.

19. Следующий метод ShowDemand необходим для вывода информации о потребностях.

```
void ShowDemand()
{
    //очищаем comboBox
    comboBoxDemand.Items.Clear();
    foreach (DemandSet demandSet in Program.wftDb.DemandSet)
    {
        //добавляем информацию, которую хотим видеть в строке comboBox-а
        //можно настроить по своему усмотрению, добавить/удалить некоторые элементы и сокращения
        //главное, не убирать Id, так как он нужен для дальнейшей работы
        string[] item = { demandSet.Id.ToString() + ". ", "Риелтор: " + demandSet.AgentsSet.LastName, "Клиент: " + demandSet.ClientsSet.LastName };
        comboBoxDemand.Items.Add(string.Join(" ", item));
    }
}
```

Рисунок 11 – Метод ShowDemand

Метод ShowDemand тоже можно изменить.

20. Вызовем данные методы в инициализации формы:

```
public FormDeal()
{
    InitializeComponent();
    ShowSupply();
    ShowDemand();
}
```

Рисунок 12 – Вызов методов в инициализации формы

21. Запустите программу и протестируйте изменения. Если информация не появляется в comboBox-ax, обратитесь снова к коду и проверьте его правильность.
22. После настройки работы comboBox-ов, сделаем автоматический расчет комиссий компании и риелторам. Так как вычисления зависят от выбора предложения и потребности, то щелкнем по каждому из comboBox-ов и укажем там вызов метода Deductions, который будет рассчитывать комиссию.

```
ссылка 1
private void comboBoxSupply_SelectedIndexChanged(object sender, EventArgs e)
{
    Deductions();
}

ссылка 1
private void comboBoxDemand_SelectedIndexChanged(object sender, EventArgs e)
{
    Deductions();
}
```

Рисунок 13 – Вызов метода Deductions

23. После этого запишем сам метод Deductions. Первым делом выполним заполнение комиссии для компании и риелтора клиента-покупателя недвижимости. Обратите внимание, что для вывода результата используется шаблон "0.00".

```

void Deductions()
{
    if (comboBoxSupply.SelectedItem != null && comboBoxDemand.SelectedItem != null)
    {
        //находим в базе предложение и потребность с выбранными номерами
        SupplySet supplySet = Program.wftDb.SupplySet.Find(Convert.ToInt32(comboBoxSupply.SelectedItem.ToString().Split('.')[0]));
        DemandSet demandSet = Program.wftDb.DemandSet.Find(Convert.ToInt32(comboBoxDemand.SelectedItem.ToString().Split('.')[0]));
        //расчитываем отчисления компании для клиента-покупателя (3% от стоимости недвижимости), выводим в textCustomerCompanyDeductions
        double customerCompanyDeductions = supplySet.Price * 0.03;
        textCustomerCompanyDeductions.Text = customerCompanyDeductions.ToString("0.00");
        //расчитываем отчисления риелтору для клиента-покупателя (комиссия указана в таблице AgentsSet), выводим в textBoxAgentCustomerDeductions
        if (demandSet.AgentsSet.DealShare != null)
        {
            double agentCustomerDeductions = customerCompanyDeductions * Convert.ToDouble(demandSet.AgentsSet.DealShare) / 100.00;
            textBoxAgentCustomerDeductions.Text = agentCustomerDeductions.ToString("0.00");
        }
        else
        {
            //если комиссия не указана, то автоматически берется 45%
            double agentCustomerDeductions = customerCompanyDeductions * 0.45;
            textBoxAgentCustomerDeductions.Text = agentCustomerDeductions.ToString("0.00");
        }
    }
    else
    {
        textCustomerCompanyDeductions.Text = "";
        textBoxAgentCustomerDeductions.Text = "";
    }
}

```

Рисунок 14 –Метод Deductions (начало)

24. Далее заполняем расчет комиссии для компании и риелтора клиента-продавца недвижимости.

```

if (comboBoxSupply.SelectedItem != null)
{
    //находим в базе предложение с выбранным номером
    SupplySet supplySet = Program.wftDb.SupplySet.Find(Convert.ToInt32(comboBoxSupply.SelectedItem.ToString().Split('.')[0]));
    //расчитываем отчисления компании для клиента-продавца
    //если продается квартира
    double sellerCompanyDeductions;
    if (supplySet.RealEstateSet.Type == 0)
    {
        sellerCompanyDeductions = 36000 + supplySet.Price * 0.01;
        textBoxSellerCompanyDeductions.Text = sellerCompanyDeductions.ToString("0.00");
    }
    //если продается дом
    else if (supplySet.RealEstateSet.Type == 1)
    {
        sellerCompanyDeductions = 30000 + supplySet.Price * 0.01;
        textBoxSellerCompanyDeductions.Text = sellerCompanyDeductions.ToString("0.00");
    }
    //если продается земля
    else
    {
        sellerCompanyDeductions = 30000 + supplySet.Price * 0.02;
        textBoxSellerCompanyDeductions.Text = sellerCompanyDeductions.ToString("0.00");
    }
    //расчитываем отчисления риелтору для клиента-покупателя (комиссия указана в таблице AgentsSet)
    if (supplySet.AgentsSet.DealShare != null)
    {
        double agentSellerDeductions = sellerCompanyDeductions * Convert.ToDouble(supplySet.AgentsSet.DealShare) / 100.00;
        textBoxAgentSellerDeductions.Text = agentSellerDeductions.ToString("0.00");
    }
    else
    {
        //если комиссия не указана, то автоматически берется 45%
        double agentSellerDeductions = sellerCompanyDeductions * 0.45;
        textBoxAgentSellerDeductions.Text = agentSellerDeductions.ToString("0.00");
    }
}
else
{
    textBoxSellerCompanyDeductions.Text = "";
    textBoxAgentSellerDeductions.Text = "";
    textCustomerCompanyDeductions.Text = "";
    textBoxAgentCustomerDeductions.Text = "";
}
}

```

Рисунок 15 –Метод Deductions (конец)

25. Протестируйте изменения. Проверьте правильность выполнения команд.

26. Для отображения сделок напишем следующий метод:

```
void ShowDealSet()
{
    //Предварительно очищаем все listView
    listViewDealSet.Items.Clear();
    //Проходим по коллекции сделок, которые находятся в базе с помощью foreach
    foreach (DealSet deal in Program.wftDb.DealSet)
    {
        //создадим новый элемент в listView с помощью массива строк
        ListViewItem item = new ListViewItem(new string[]
        {
            //указываем необходимые поля (можно добавить имена, отчества, информацию про объект недвижимости и т.д.)
            //Фамилия клиента-продавца
            deal.SupplySet.ClientsSet.LastName,
            //Фамилия риелтора клиента-продавца
            deal.SupplySet.AgentsSet.LastName,
            //Фамилия клиента-покупателя
            deal.DemandSet.ClientsSet.LastName,
            //Фамилия риелтора клиента-покупателя
            deal.DemandSet.AgentsSet.LastName,
            //Адрес объекта недвижимости
            "г. " + deal.SupplySet.RealEstateSet.Address_City + ", ул. " + deal.SupplySet.RealEstateSet.Address_Street + ", д. " +
            deal.SupplySet.RealEstateSet.Address_House + ", кв. " + deal.SupplySet.RealEstateSet.Address_Number,
            //Стоимость
            deal.SupplySet.Price.ToString()
        });
        //указываем по какому тегу выбраны элементы
        item.Tag = deal;
        //добавляем элементы в listViewRealEstateSet_Apartment для отображения
        listViewDealSet.Items.Add(item);
    }
}
```

Рисунок 16 – Метод отображения сделок в listView

27. Затем вызовем данный метод в инициализации формы.

```
public FormDeal()
{
    InitializeComponent();
    ShowSupply();
    ShowDemand();
    ShowDealSet();
}
```

Рисунок 17 – Вызов метода в инициализации формы

28. После настройки всех элементов нажимаем на форме на кнопку «Создать» левой кнопкой мыши два раза для открытия окна написания кода

```
private void buttonAdd_Click(object sender, EventArgs e)
{
}
}
```

Рисунок 18 – Метод кнопки Add

29. При добавлении нового предложения важно, чтобы все поля были заполнены, поэтому при написании кода учитываем этот момент.

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    //проверяем, что все поля (раскрывающихся списков и текстового поля) были заполнены
    if (comboBoxDemand.SelectedItem != null && comboBoxSupply.SelectedItem != null)
    {
        //создаем новый экземпляр класса Сделка
        DealSet deal = new DealSet();
        //из выбранной строки отделяем Id предложения (он отделен точкой) и делаем ссылку
        deal.IdSupply = Convert.ToInt32(comboBoxSupply.SelectedItem.ToString().Split('.')[0]);
        //из выбранной строки отделяем Id потребности (он отделен точкой) и делаем ссылку
        deal.IdDemand = Convert.ToInt32(comboBoxDemand.SelectedItem.ToString().Split('.')[0]);
        //добавляем в таблицу DealSet новую сделку
        Program.wftDb.DealSet.Add(deal);
        //сохраняем изменения в модели wftDb
        Program.wftDb.SaveChanges();
        ShowDealSet();
    }
    else MessageBox.Show("Данные не выбраны", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Рисунок 19 – Метод создания сделки

30. Протестируйте программу. Проверьте в SSMS, добавились ли предложения в базу данных.
31. Далее перейдём к написанию кода на изменение информации об объекте недвижимости. Для этого нажмем двойным щелчком на кнопку «Изменить» и напишем следующий метод.

```
private void buttonEdit_Click(object sender, EventArgs e)
{
    //если в listView выбран элемент
    if (listViewDealSet.SelectedItems.Count == 1)
    {
        //ищем элемент из таблицы по тегу
        DealSet deal = listViewDealSet.SelectedItems[0].Tag as DealSet;
        //указываем, что может быть изменено
        deal.IdSupply = Convert.ToInt32(comboBoxSupply.SelectedItem.ToString().Split('.')[0]);
        deal.IdDemand = Convert.ToInt32(comboBoxDemand.SelectedItem.ToString().Split('.')[0]);
        //Сохраняем изменения в модели wftDb
        Program.wftDb.SaveChanges();
        ShowDealSet();
    }
}
```

Рисунок 20 – Метод изменения информации о сделке

32. Затем напишем ещё один метод, который будет осуществлять отображение выбранного элемента, для этого на форме щелкнем два раза по listView и напишем следующее:

```
private void listViewDealSet_SelectedIndexChanged(object sender, EventArgs e)
{
    //если в listView выбран элемент
    if (listViewDealSet.SelectedItems.Count == 1)
    {
        //ищем элемент из таблицы по тегу
        DealSet deal = listViewDealSet.SelectedItems[0].Tag as DealSet;
        comboBoxSupply.SelectedIndex = comboBoxSupply.FindString(deal.IdSupply.ToString());
        comboBoxDemand.SelectedIndex = comboBoxDemand.FindString(deal.IdDemand.ToString());
    }
    else
    {
        //если не выбран ни один элемент, задаем пустые элементы
        comboBoxSupply.SelectedItem = null;
        comboBoxDemand.SelectedItem = null;
    }
}
```

Рисунок 21 – Метод выбора элементов listView

33. Протестируйте изменения. Проверьте правильность выполнения команд.
34. Далее напишем метод для удаления сделок из базы данных:

```
private void buttonDel_Click(object sender, EventArgs e)
{
    //попробуем совершить действие
    try
    {
        //если в listView выбран элемент
        if (listViewDealSet.SelectedItems.Count == 1)
        {
            //ищем элемент из таблицы по тегу
            DealSet deal = listViewDealSet.SelectedItems[0].Tag as DealSet;
            //удаляем из модели базы данных
            Program.wftDb.DealSet.Remove(deal);
            //сохраняем изменения
            Program.wftDb.SaveChanges();
            //отображаем обновленный список
            ShowDealSet();
        }
        //очищаем все поля
        comboBoxSupply.SelectedItem = null;
        comboBoxDemand.SelectedItem = null;
    }
    //если возникает какая-то ошибка
    catch
    {
        MessageBox.Show("Невозможно удалить, эта запись используется", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Рисунок 22 – Метод удаления сделки

35. Протестируйте изменения. Проверьте правильность выполнения команд.
36. На этом написание кода закончено.

37. Теперь поработаем немного над оформлением:

В свойствах формы изменим некоторые параметры:

- Текст в строке заголовка (Text) – Объекты недвижимости.
- Стартовая позиция (StartPosition) – CenterScreen.

38. На форму добавим изображение (элемент PictureBox) – логотип компании Esoft.

39. Если вы решили выполнить дополнительные задания под *, вам необходимо добавить нужные элементы и прописать условия вывода в них информации.

Цвет, положение, шрифт объектов и самой формы необходимо будет изменить в соответствии с **Руководством по стилю!**