

1. Работа с SSMS

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


	Имя столбца	Тип данных	Разрешить ...
	Id	int	<input type="checkbox"/>
	IdAgent	int	<input type="checkbox"/>
	IdClient	int	<input type="checkbox"/>
	IdRealEstate	int	<input type="checkbox"/>
	Price	bigint	<input type="checkbox"/>

Рисунок 1 – Таблица SupplySet

4. Обратите внимание, что столбцы IdAgent, IdClient, IdRealEstate являются внешними ключами к таблицам AgentsSet, ClientsSet, RealEstateSet.
5. Чтобы сделать связь между таблицами, необходимо нажать на любое поле в проекте таблицы и выбрать «Отношения...».

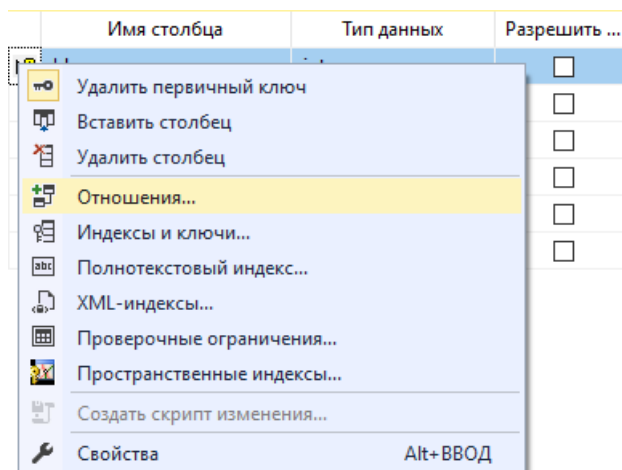


Рисунок 2 – Контекстное меню таблицы

6. В открывшемся окне нажимаем кнопку «Добавить»:

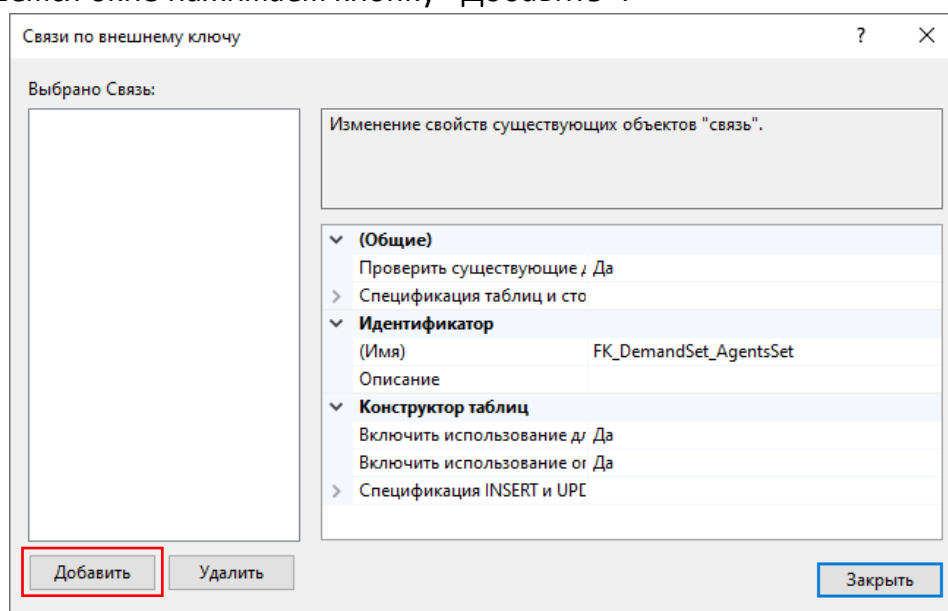


Рисунок 3 – Окно «Связи по внешнему ключу»

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

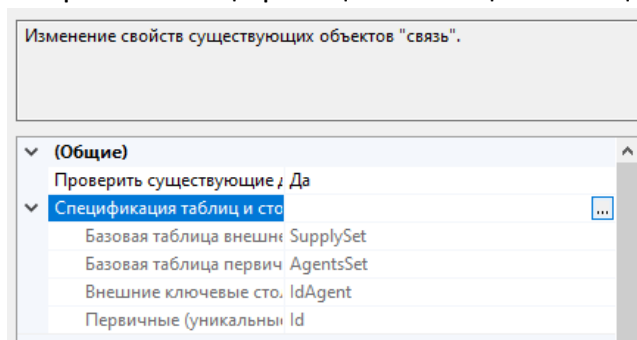


Рисунок 4 – Окно «Связи по внешнему ключу»

8. Настраиваем связь с таблицей AgentsSet и нажимаем ОК:

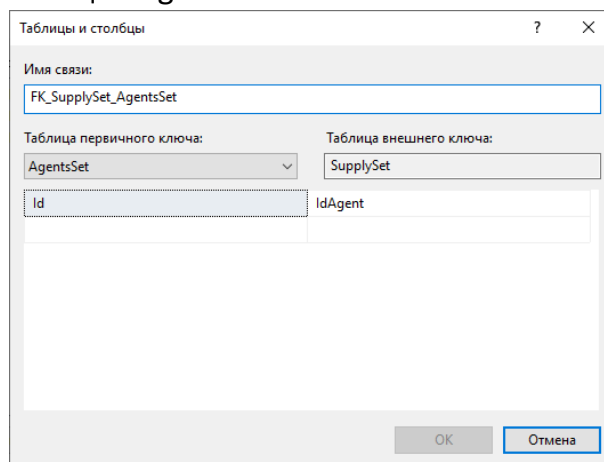


Рисунок 5 – Создание внешнего ключа к таблице AgentsSet

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

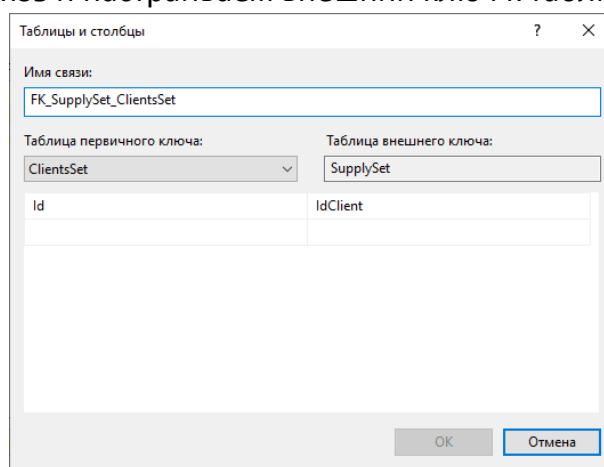


Рисунок 6 – Создание внешнего ключа к таблице ClientsSet

10. Добавляем третью связь и настраиваем внешний ключ к таблице RealEstateSet:

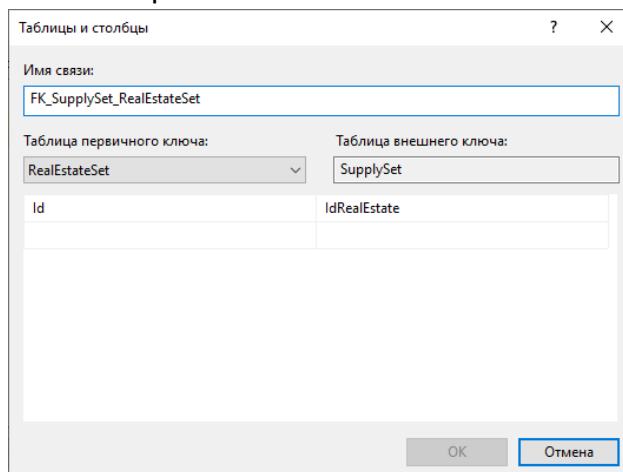


Рисунок 7 – Создание внешнего ключа к таблице RealEstateSet

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

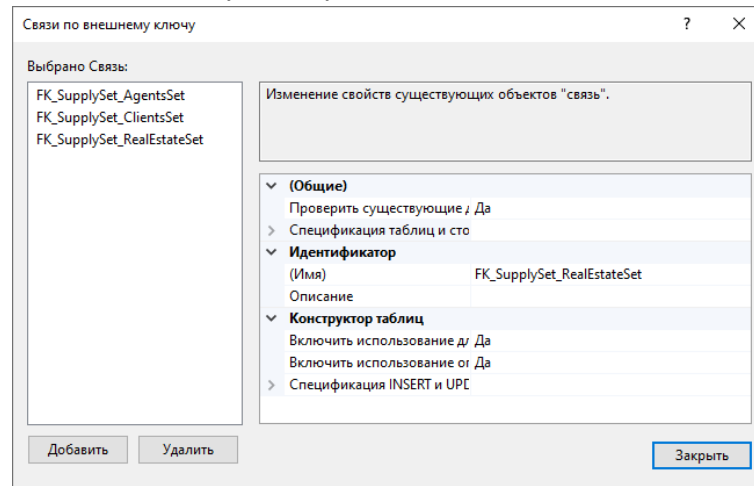


Рисунок 8 – Окно «Связи по внешнему ключу»

2. Работа с формой «Предложения»

12. Переходим в Visual Studio, откроем проект Esoft_Project.

13. 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).

14. Теперь в модели появились связи, и она выглядит так:

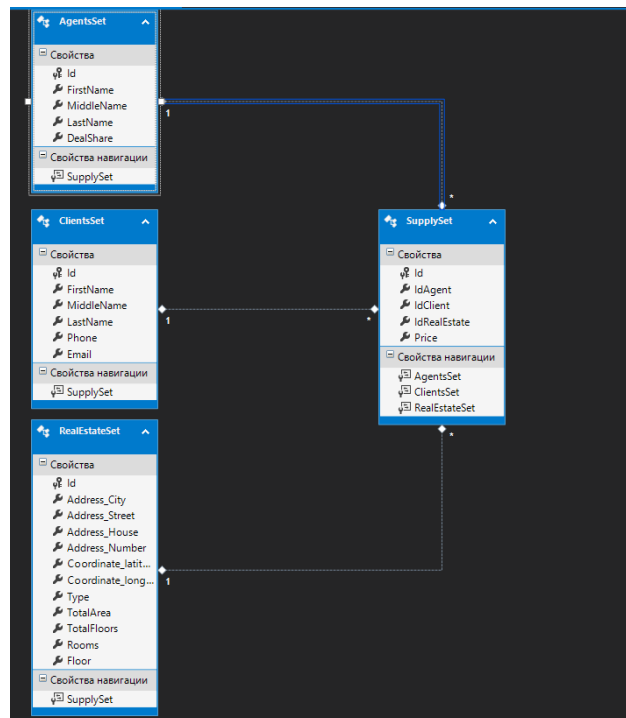


Рисунок 8 – Модель системы

15. Далее создаём форму, на которой будет располагаться интерфейс для предложений, для этого перейдём в вкладку «Проект», затем «Добавить форму». Зададим ей название FormSupply.

16. Сделаем ссылку на эту форму с кнопки «Предложения»:

- Перейдем в конструктор формы Menu.
- Два раза щелкнем правой кнопкой мыши по кнопке «Предложения» и добавим в метод вызова формы «Предложения».

```
private void buttonOpenSupplySet_Click(object sender, EventArgs e)
{
    Form formSupply = new FormSupply();
    formSupply.Show();
}
```

Рисунок 9 – Открытие формы «Предложения»

17. Перейдем снова на форму «Предложения». Добавим необходимые элементы из «Панели элементов».

18. В данном интерфейсе нам нужны: textBox (1), label (4), listView (1), button (3, их можно скопировать из созданных форм), comboBox (3). Перетаскиваем на форму и переименовываем каждый элемент для удобства.

Наименование (Name) элемента textBox:

- textBoxPrice

Наименование (Name) и текстовое отображение (Text) элементов label:

- labelAgent - Риелтор
- labelClient – Клиент
- labelRealEstate – Объект недвижимости
- labelPrice - Цена

Наименование (Name) элемента listView:

- listViewSupplySet

Наименование (Name) и текстовое отображение (Text) элементов button:

- buttonAdd – Создать
- buttonEdit – Изменить
- buttonDel – Удалить

Наименование (Name) элементов comboBox:

- comboBoxAgents
- comboBoxClients
- comboBoxRealEstate

Настройка элементов listView:

- Измените некоторые свойства объектов:

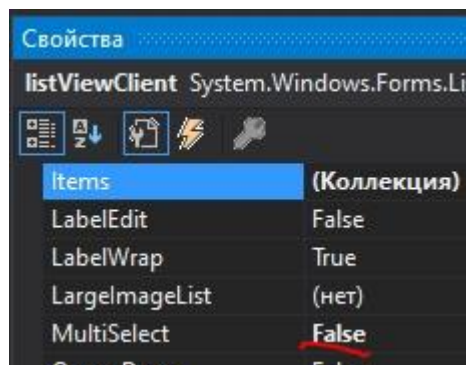


Рисунок 10 – Изменение свойств listView №1

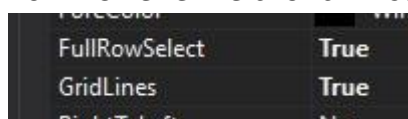


Рисунок 11 – Изменение свойств listView №2

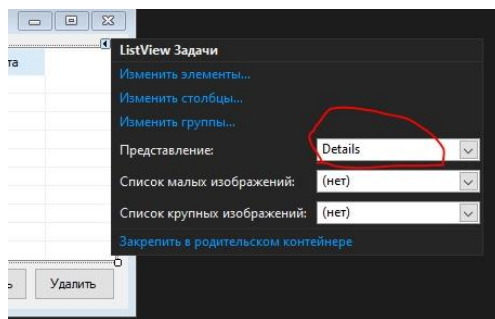


Рисунок 12 – Изменение представлений listView

- Для listView зададим необходимые столбцы:

Риелтор	Клиент	Объект недвижимости	Цена

Рисунок 13 – Столбцы listView

19. После настройки форма «Предложения» должна выглядеть примерно так:

Рисунок 14 – Вид формы «Предложения»

20. Настроим вывод необходимой информации в comboBox-ax.

21. Создадим новый метод ShowAgents, который будет выводить информацию об риелторах в comboBoxAgents.

```
void ShowAgents()
{
    //очищаем comboBox
    comboBoxAgents.Items.Clear();
    foreach (AgentsSet agentSet in Program.wftDb.AgentsSet)
    {
        //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 = {agentSet.Id.ToString()+".", agentSet.FirstName, agentSet.MiddleName, agentSet.LastName};
        comboBoxAgents.Items.Add(string.Join(" ", item));
    }
}
```

Рисунок 15 – Метод ShowAgents

Здесь можно изменять выводимую информацию в зависимости от ваших предпочтений (например, добавить комиссию или указывать инициалы риелтора, а не полные имя и отчество).

22. Следующий метод ShowClients необходим для вывода информации о клиентах в comboBoxClients.

```
void ShowClients()
{
    //очищаем comboBox
    comboBoxClients.Items.Clear();
    foreach (ClientsSet clientsSet in Program.wftDb.ClientsSet)
    {
        //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 = {clientsSet.Id.ToString()+".", clientsSet.FirstName, clientsSet.MiddleName, clientsSet.LastName};
        comboBoxClients.Items.Add(string.Join(" ", item));
    }
}
```

Рисунок 16 – Метод ShowClients

Метод ShowClients тоже нужно изменить.

23. Аналогично создаем метод ShowRealEstate для вывода информации об объектах недвижимости в comboBoxRealEstate. Здесь не стоит размещать всю информацию об объекте, чтобы не занимать слишком много места на форме.

```
void ShowRealEstate()
{
    //очищаем comboBox
    comboBoxRealEstate.Items.Clear();
    foreach (RealEstateSet realEstateSet in Program.wftDb.RealEstateSet)
    {
        //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 = { realEstateSet.Id.ToString() + ".", realEstateSet.Address_City + ", ", realEstateSet.Address_Street + ", ",
            "д. " + realEstateSet.Address_House + ", ", "кв. " + realEstateSet.Address_Number };
        comboBoxRealEstate.Items.Add(string.Join(" ", item));
    }
}
```

Рисунок 17 – Метод ShowRealEstate

Подумайте, как изменить и дополнить информацию об объекте недвижимости в comboBox.

24. Вызовем данные методы в инициализации формы:

```
public partial class FormSupply : Form
{
    ссылка 1
    public FormSupply()
    {
        InitializeComponent();
        ShowAgents();
        ShowClients();
        ShowRealEstate();
    }
}
```

Рисунок 18 – Вызов методов в инициализации формы

25. Запустите программу и протестируйте изменения. Если информация не появляется в comboBox-ах, обратитесь снова к коду и проверьте его правильность.
26. 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 19 – Метод кнопки Add

27. 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 (comboBoxAgents.SelectedItem!=null && comboBoxClients.SelectedItem!=null && comboBoxRealEstate!=null && textBoxPrice.Text!="")
    {
        //создаем новый экземпляр класса Предложение
        SupplySet supply = new SupplySet();
        //из выбранной строки в comboBoxAgents отделяем Id риелтора (он отделен точкой) и делаем ссылку supply.IdAgent
        supply.IdAgent = Convert.ToInt32(comboBoxAgents.SelectedItem.ToString().Split('.')[0]);
        //точно также отделяем и заносим Id клиента
        supply.IdClient = Convert.ToInt32(comboBoxClients.SelectedItem.ToString().Split('.')[0]);
        //отделяем и заносим Id объекта недвижимости
        supply.IdRealEstate = Convert.ToInt32(comboBoxRealEstate.SelectedItem.ToString().Split('.')[0]);
        //цена объекта недвижимости чаще всего превосходит миллион, поэтому используем Int64
        supply.Price = Convert.ToInt64(textBoxPrice.Text);
        //добавляем в таблицу SupplySet новый объект недвижимости supply
        Program.wftDb.SupplySet.Add(supply);
        //сохраняем изменения в модели wftDb
        Program.wftDb.SaveChanges();
    }
    else MessageBox.Show("Данные не выбраны", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Рисунок 20 – Метод создания предложения

28. Протестируйте программу. Проверьте в SSMS, добавились ли предложения в базу данных.
29. Для отображения списков предложений напишем следующий метод:

```
void ShowSupplySet()
{
    //Предварительно очищаем все listView
    listViewSupplySet.Items.Clear();
    //Проходим по коллекции клиентов, которые находятся в базе с помощью foreach
    foreach (SupplySet supply in Program.wftDb.SupplySet)
    {
        //создадим новый элемент в listView с помощью массива строк
        ListViewItem item = new ListViewItem(new string[]
        {
            //указываем необходимые поля
            supply.IdAgent.ToString(), supply.IdClient.ToString(), supply.IdRealEstate.ToString(), supply.Price.ToString()
        });
        //указываем по какому тегу выбраны элементы
        item.Tag = supply;
        //добавляем элементы в listViewRealEstateSet_Apartment для отображения
        listViewSupplySet.Items.Add(item);
    }
}
```

Рисунок 21 – Метод отображения предложений в listView

30. Затем вызовем данный метод в инициализации формы и в методе buttonAdd_Click.

```
public partial class FormSupply : Form
{
    ссылка 1
    public FormSupply()
    {
        InitializeComponent();
        ShowAgents();
        ShowClients();
        ShowRealEstate();
        ShowSupplySet();
    }
}
```

Рисунок 22 – Вызов метода в инициализации формы

```
Program.wftDb.SupplySet.Add(supply);
//сохраняем изменения в модели wftDb
Program.wftDb.SaveChanges();
ShowSupplySet();
}
else MessageBox.Show("Данные не выбраны", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Рисунок 23 – Вызов метода в buttonAdd_Click

31. Протестируйте изменения. Проверьте правильность выполнения команд.
32. 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 (listViewSupplySet.SelectedItems.Count == 1)
    {
        //ищем элемент из таблицы по тегу
        SupplySet supply = listViewSupplySet.SelectedItems[0].Tag as SupplySet;
        //указываем, что может быть изменено
        supply.IdAgent = Convert.ToInt32(comboBoxAgents.SelectedItem.ToString().Split('.')[0]);
        supply.IdClient = Convert.ToInt32(comboBoxClients.SelectedItem.ToString().Split('.')[0]);
        supply.IdRealEstate = Convert.ToInt32(comboBoxRealEstate.SelectedItem.ToString().Split('.')[0]);
        supply.Price = Convert.ToInt64(textBoxPrice.Text);
        //Сохраняем изменения в модели wftDb
        Program.wftDb.SaveChanges();
        ShowSupplySet();
    }
}
```

Рисунок 24 – Метод изменения информации о предложении

33. 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 listViewSupplySet_SelectedIndexChanged(object sender, EventArgs e)
{
    //если в listView выбран элемент
    if (listViewSupplySet.SelectedItems.Count == 1)
    {
        //ищем элемент из таблицы по тегу
        SupplySet supply = listViewSupplySet.SelectedItems[0].Tag as SupplySet;
        //указываем, что может быть изменено
        //находим в comboBoxAgents необходимую строку по Id риелтора из supply.IdAgent и задаем ее отображение comboBox-y
        comboBoxAgents.SelectedIndex = comboBoxAgents.FindString(supply.IdAgent.ToString());
        //точно также поступаем с comboBoxClients и comboBoxRealEstate
        comboBoxClients.SelectedIndex = comboBoxClients.FindString(supply.IdClient.ToString());
        comboBoxRealEstate.SelectedIndex = comboBoxRealEstate.FindString(supply.IdRealEstate.ToString());
        textBoxPrice.Text = supply.Price.ToString();
    }
    else
    {
        //если не выбран ни один элемент, задаем пустые элементы
        comboBoxAgents.SelectedItem = null;
        comboBoxClients.SelectedItem = null;
        comboBoxRealEstate.SelectedItem = null;
        textBoxPrice.Text = "";
    }
}
```

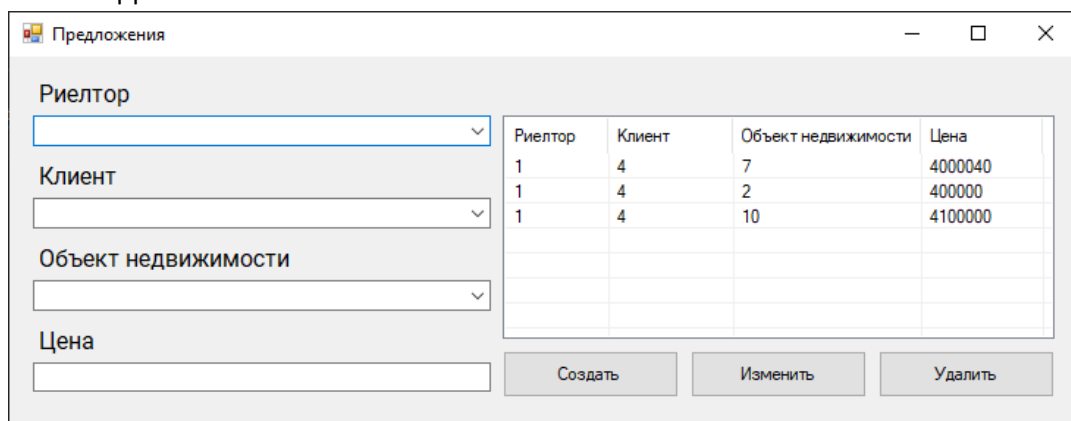
Рисунок 25 – Метод выбора элементов listView

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

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

Рисунок 29 – Метод удаления предложения

36. Протестируйте изменения. Проверьте правильность выполнения команд.
37. При проведении тестирования, вы могли заметить, что данные, отображаемые в listView не очень понятны, так как там используются только идентификаторы риелторов, клиентов и объектов недвижимости.



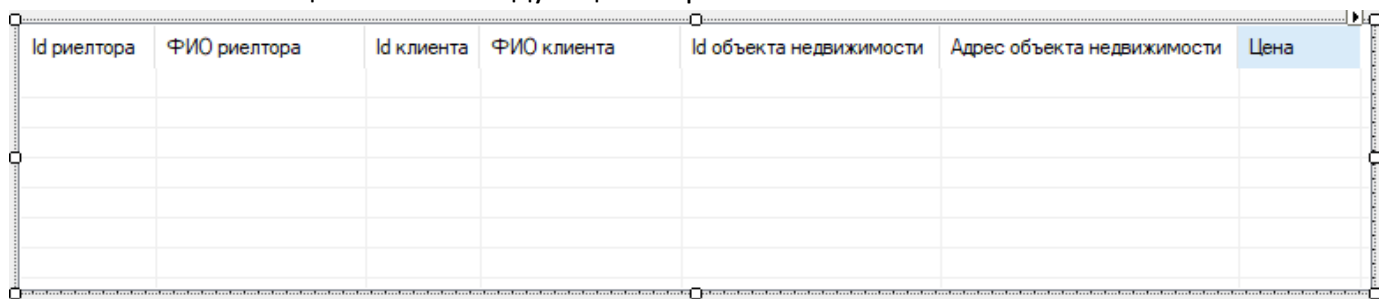
Риелтор	Клиент	Объект недвижимости	Цена
1	4	7	4000040
1	4	2	400000
1	4	10	4100000

Рисунок 30 – Вид формы «Предложения»

38. Сделаем таблицу более понятной.

Вы можете изменить столбцы в listView и код к его отображению по своему усмотрению.

39. Изменим столбцы listView следующим образом:



Id риелтора	ФИО риелтора	Id клиента	ФИО клиента	Id объекта недвижимости	Адрес объекта недвижимости	Цена

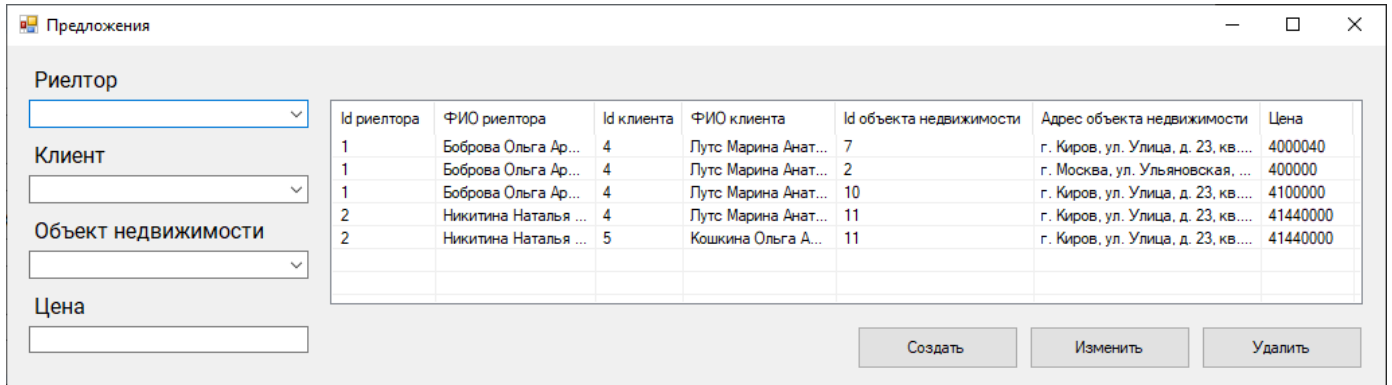
Рисунок 31 – Измененный listViewSupplySet

40. Воспользуемся связями между таблицами и дополним код в методе ShowSupplySet:

```
void ShowSupplySet()
{
    //Предварительно очищаем все listView
    listViewSupplySet.Items.Clear();
    //Проходим по коллекции клиентов, которые находятся в базе с помощью foreach
    foreach (SupplySet supply in Program.wftDb.SupplySet)
    {
        //создадим новый элемент в listView с помощью массива строк
        ListViewItem item = new ListViewItem(new string[]
        {
            //указываем необходимые поля
            //Id риелтора
            supply.IdAgent.ToString(),
            //ФИО риелтора (фамилия+имя+отчество)
            supply.AgentsSet.LastName+" "+supply.AgentsSet.FirstName+" "+supply.AgentsSet.MiddleName,
            //Id клиента
            supply.IdClient.ToString(),
            //ФИО клиента (фамилия+имя+отчество)
            supply.ClientsSet.LastName+" "+supply.ClientsSet.FirstName+" "+supply.ClientsSet.MiddleName,
            //Id объекта недвижимости
            supply.IdRealEstate.ToString(),
            //Адрес объекта недвижимости
            "г. "+supply.RealEstateSet.Address_City+", ул. "+supply.RealEstateSet.Address_Street+", д. "+
            supply.RealEstateSet.Address_House+", кв. "+supply.RealEstateSet.Address_Number,
            //Цена
            supply.Price.ToString()
        });
        //указываем по какому тегу выбраны элементы
        item.Tag = supply;
        //добавляем элементы в listViewRealEstateSet_Apartment для отображения
        listViewSupplySet.Items.Add(item);
    }
}
```

Рисунок 32 – Измененный метод ShowSupplySet

41. Теперь форма выглядит понятнее:



Id риелтора	ФИО риелтора	Id клиента	ФИО клиента	Id объекта недвижимости	Адрес объекта недвижимости	Цена
1	Боброва Ольга Ар...	4	Лутс Марина Анат...	7	г. Киров, ул. Улица, д. 23, кв....	4000040
1	Боброва Ольга Ар...	4	Лутс Марина Анат...	2	г. Москва, ул. Ульяновская, ...	400000
1	Боброва Ольга Ар...	4	Лутс Марина Анат...	10	г. Киров, ул. Улица, д. 23, кв....	4100000
2	Никитина Наталья ...	4	Лутс Марина Анат...	11	г. Киров, ул. Улица, д. 23, кв....	41440000
2	Никитина Наталья ...	5	Кошкина Ольга А...	11	г. Киров, ул. Улица, д. 23, кв....	41440000

Рисунок 30 – Новый вид формы «Предложения»

42. На этом написание кода закончено.

43. Протестируйте изменения. Проверьте правильность выполнения команд.

44. Теперь поработаем немного над оформлением:

В свойствах формы изменим некоторые параметры:

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

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

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