Подключение через Entity Framework и Database First

Сначала, для выполнения задания нужно создать новый проект в Visual Studio, а именно проект Windows Forms (Майкрософт). Требуется именно .NET (Core) версия форм, а не .NetFramework, иначе подключение пакетов не сработает.
Для подключения требуются следующие пакеты
- Microsoft.EntityFrameworkCore.Design 9 версия
- Microsoft.EntityFrameworkCore.Tools 9 версия
Для импорта в проект

Pomelo.EntityFrameworkCore.MySQL
Провайдер

Дальше, требуется найти консоль диспетчера пакетов, если консоли не обнаружено, то следует нажать на вкладку:
Вид > Консоль Диспетчера пакетов.

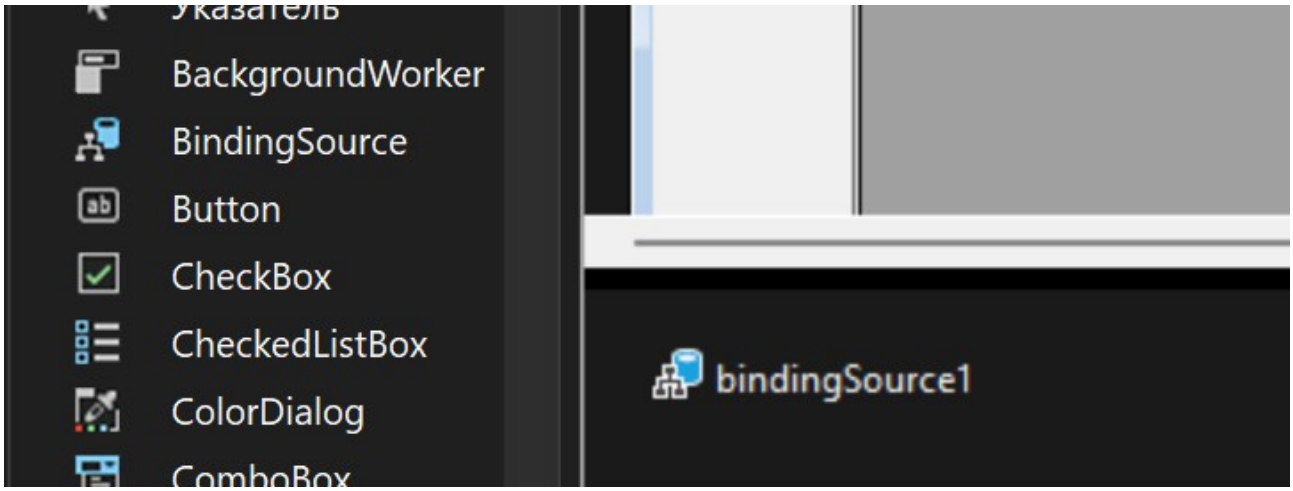
Далее, введите команду:
Scaffold-DbContext «Ваша строка подкоючения»
«Pomelo.EntityFrameworkCore.MySQL» 9 версия

Следующая команда: Scaffold-DbContext "server = cfif31.ru; username = username; password=youtpass; database=dbname" "Pomelo.EntityFrameworkCore.MySql" -OutputDir Models -f.

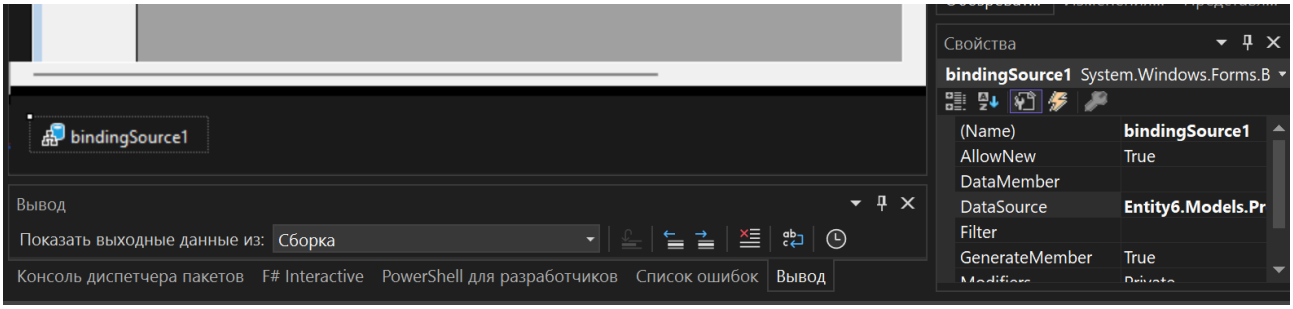
После успешного выполнения команды, выполните сборку проекта **Сборка > Собрать решение.** Это нужно, чтобы Visual Studio смогла получить информацию о типах данных, полученных при сборке.

CRUD через datagridview и Binding Source

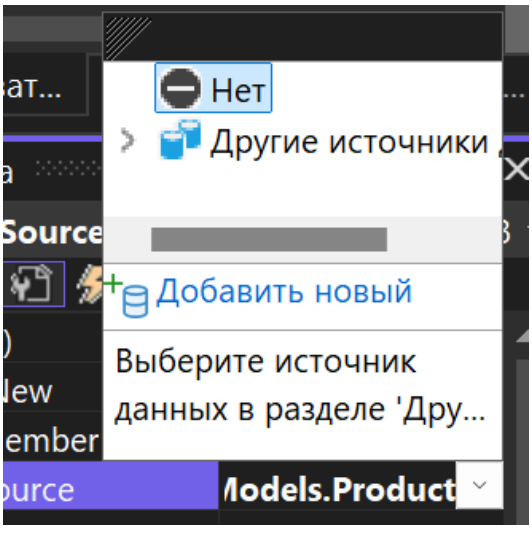
Добавьте на форму компонент BindingSource



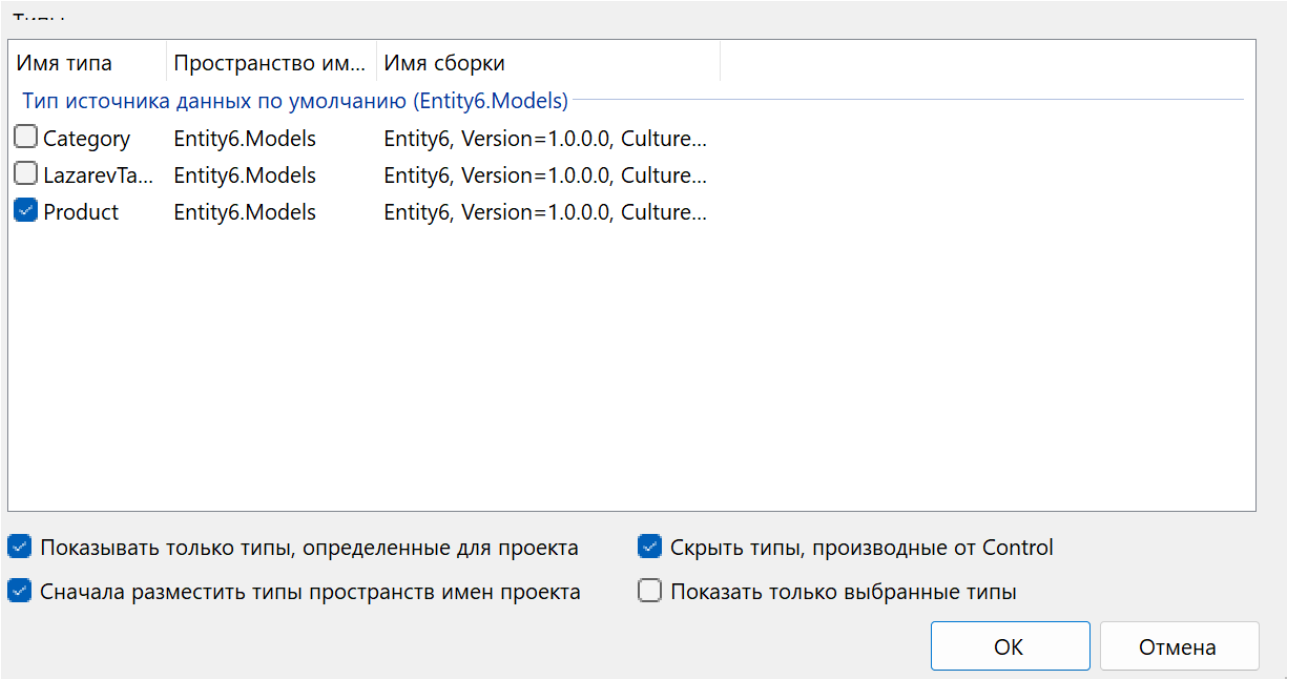
Далее, заходите в свойства у BindingSource



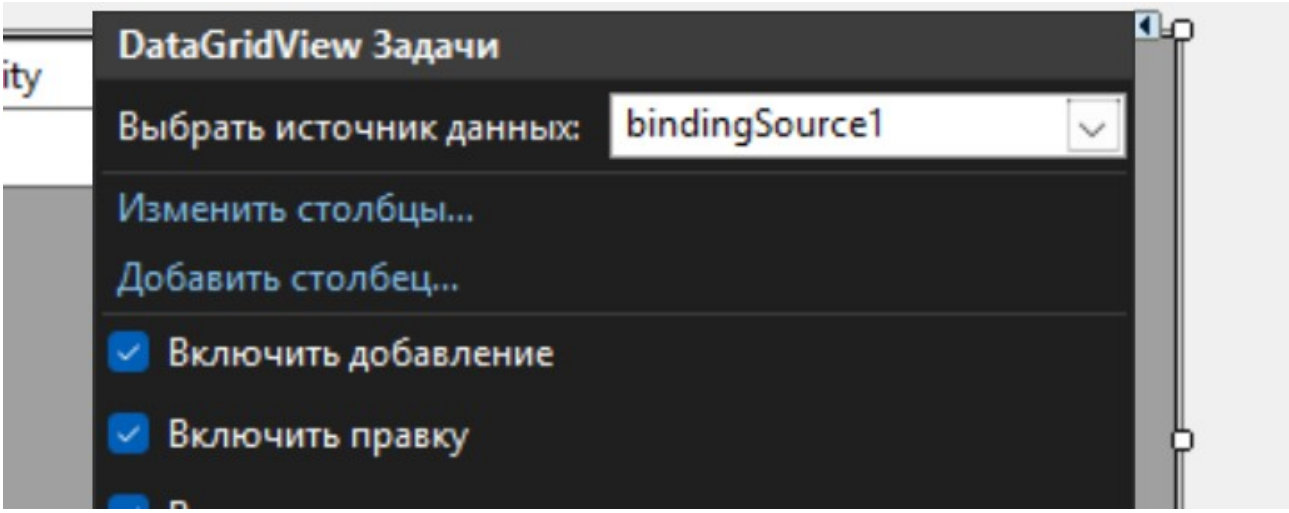
Затем требуется нажать на DataSource, а именно на стрелочку справа, чтобы раскрылось специальное меню



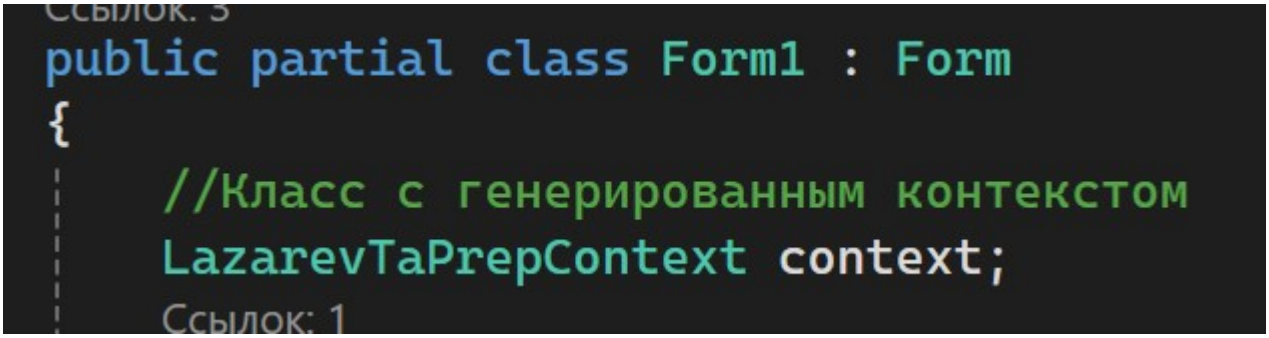
Затем выбрать ваши сущности, используемые для вывода



Требуется нажать ОК. Далее, разместите dataGridView на форме, нажмите стрелочку и выберите созданный источник данных:



Запишите созданный на предыдущих шагах класс контекста, состоящий из имени базы данных и слова Context



Затем требуется записать следующую логику в OnLoad().
В инициализации контекста требуется заменить контекст на импортированный.
В загрузке сущностей должен быть указан ваш сгенерированный класс.

```
Ссылок: 0
protected override void OnLoad(EventArgs e)
{
    base.OnLoad(e);

    //инициализация контекста
    context = new LazarevTaPrepContext();

    //Загрузка сущностей из базы
    context.Products.Load();

    //Проверка на создание базы
    context.Database.EnsureCreated();

    //привязка данных
    bindingSource1.DataSource = context.Products.Local.ToBindingList();
}
```

Затем требуется создать кнопку и оформить следующее событие щелчка.

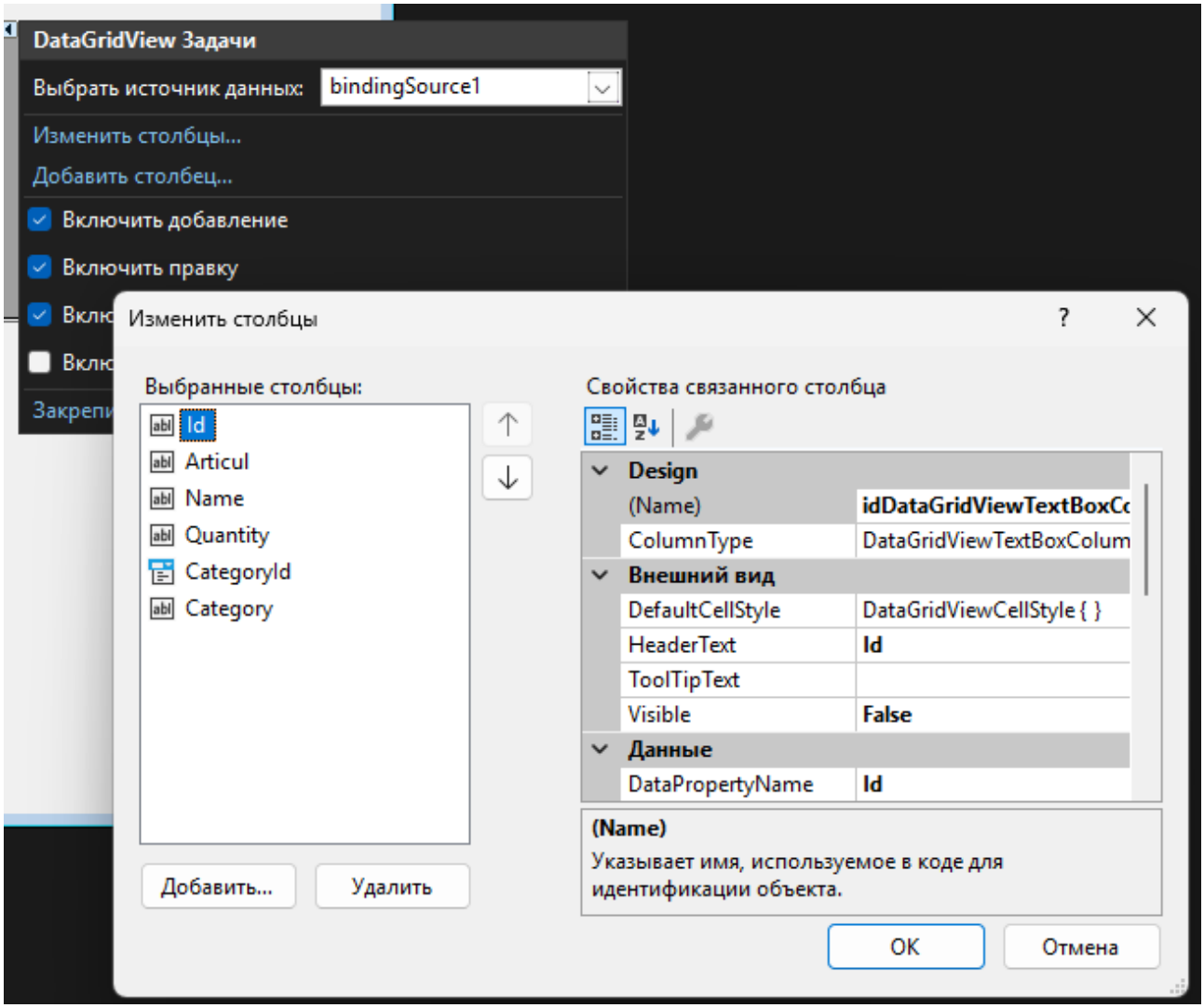
```
Ссылок: 1
private void saveButton_Click(object sender, EventArgs e)
{
    //Сохранение отслеживаемых сущностей в бд
    context.SaveChanges();
    dataGridView1.Refresh();
}
```

После сборки и запуска проекта можно убедиться, что записи в dataGridView можно добавлять, удалять, изменять без написания SQL запросов.

Редактирование внешнего вида колонок

Клиенту не нужно знать об идентификаторах, когда вы работаете с приложением.

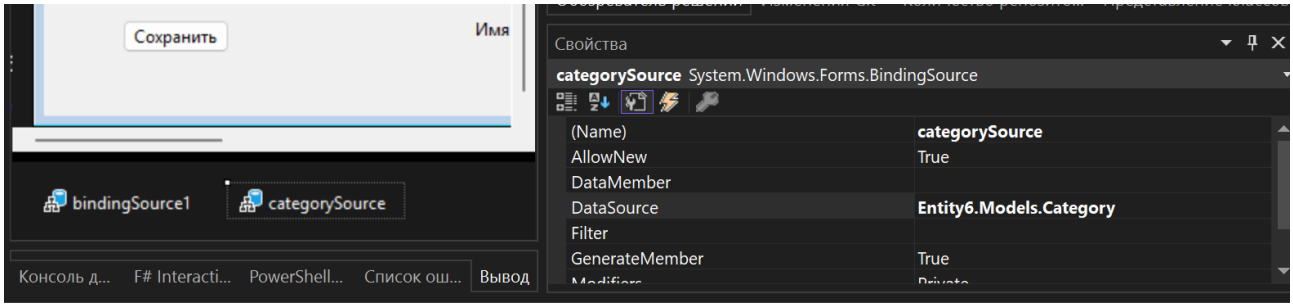
Чтобы бы убрать ту или иную колонку, нужно нажать на Изменить столбцы.



Здесь в категории Внешний вид, интересующими свойствами являются HeaderText и Visible, отвечающими соответственно за название колонки и видимость.

Смежные таблицы

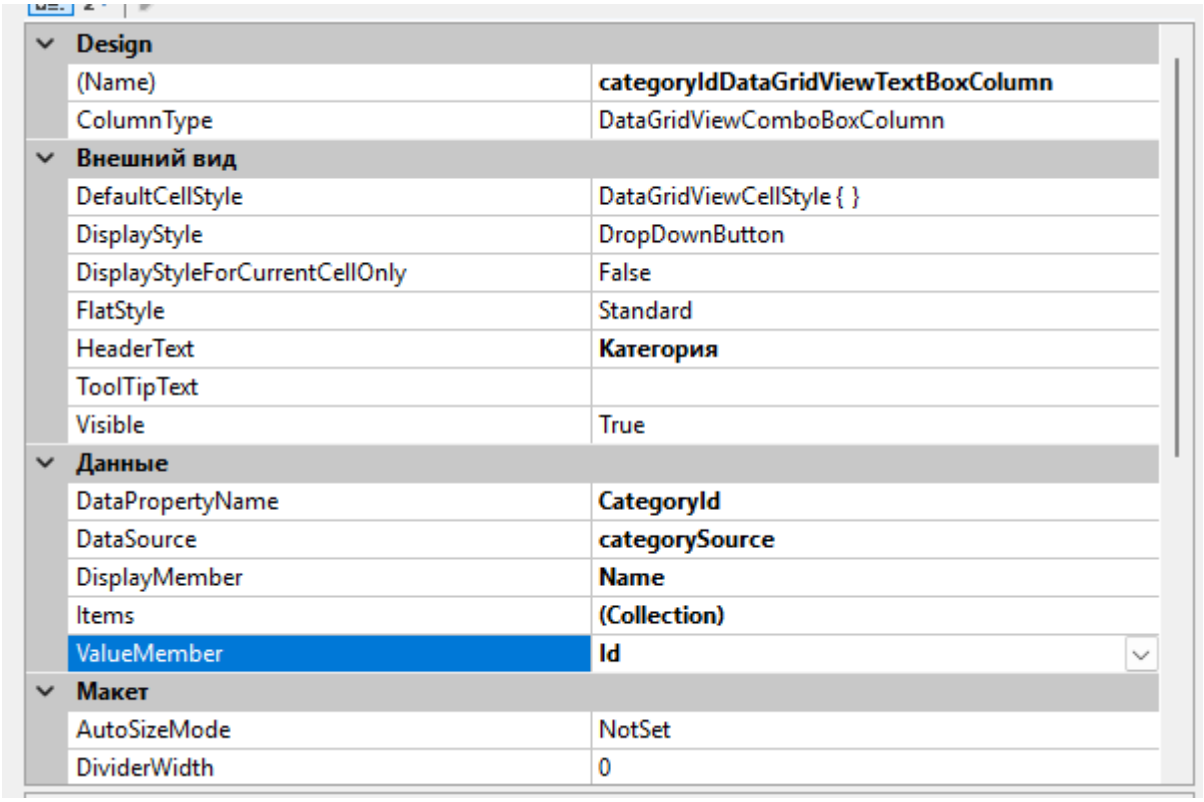
Требуется создать новый источник данных (bindingSource для категорий) и указать ваш класс категорий, который соответствует внешней, связанной сущности.



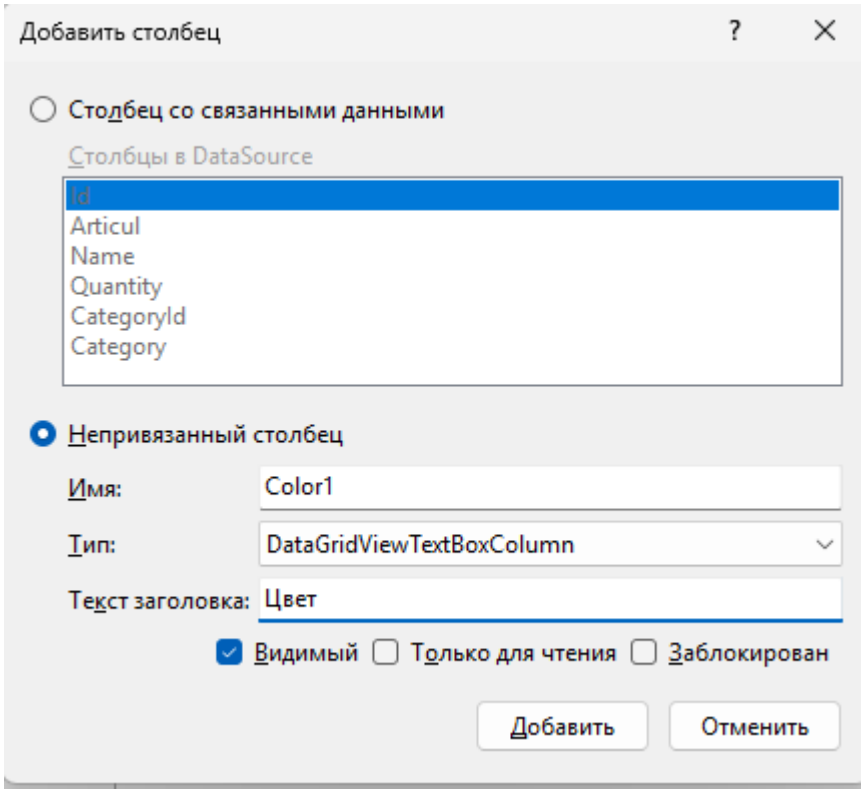
Следующая логика заполнения

```
{
    base.OnLoad(e);

    //инициализация контекста
    context = new LazarevTaPrepContext();

    //Загрузка сущностей из базы
    context.Products.Include(x => x.Category).Load();
    context.Categories.Load();

    //Проверка на создание базы
    context.Database.EnsureCreated();

    //привязка данных
    bindingSource1.DataSource = context.Products.Local.ToBindingList();
    categorySource.DataSource = context.Categories.Local.ToBindingList();
}
```

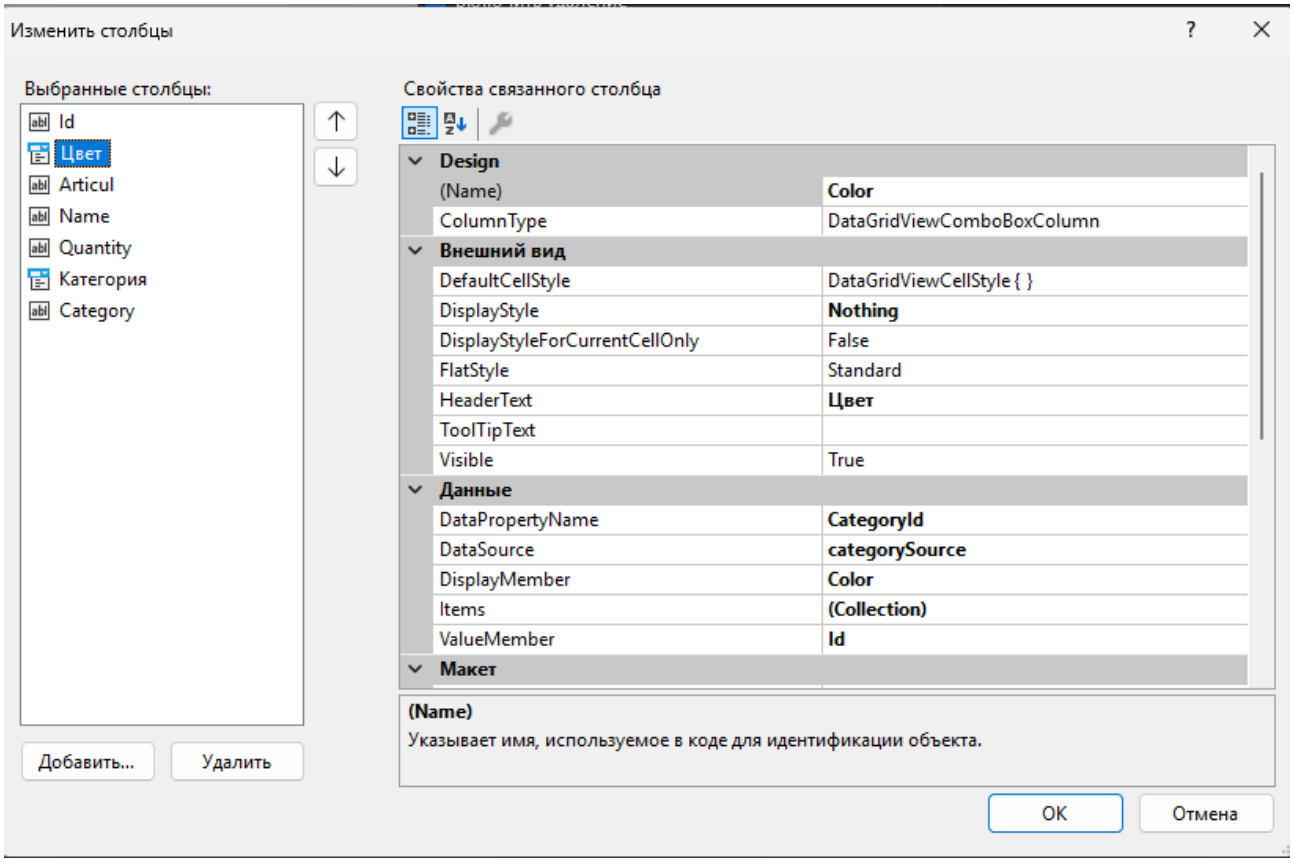
Далее перейдите в изменение колонок, выберите колонку идентификатором вашей связанной таблицы и измените columnType на указанный в скриншоте и измените колонку для того, чтобы вместо идентификатора показывалось имя товара.



Для просмотра полей в связанной таблице, вы сначала должны добавить несвязанный столбец



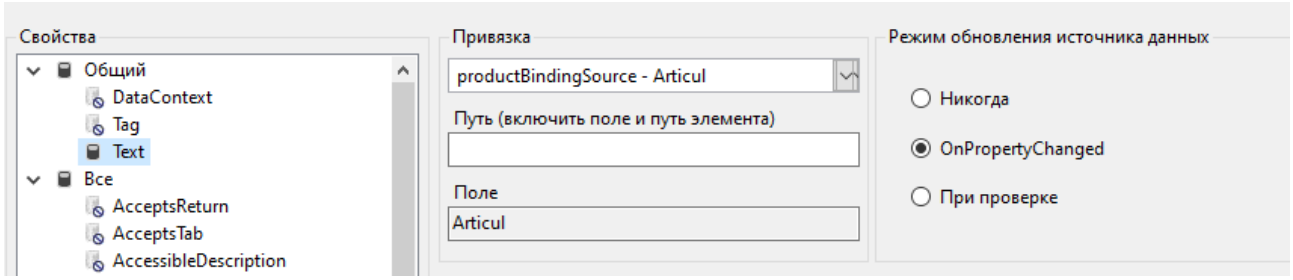
Далее в ColumnType Нужно подставить DataGridViewComboBoxColumn
В настройках столбцах нужно указать в DataPropertyName идентификатор внешней сущности. В dataSource указать источник таблицы. В DisplayMember нужно подставить интересующую колонку. В ValueMember также указать её идентификатор.



Редактирование записи

Создайте новую форму и добавьте поля. Далее зайдите в один из текстбоксов и найдите категорию Данные, а в неё (DataBindings).

Затем нужно выбрать меню в поле расширенный. Если кликнуть на пустой слот, то появится троеточие, на которое нужно, чтобы открылось следующее меню.

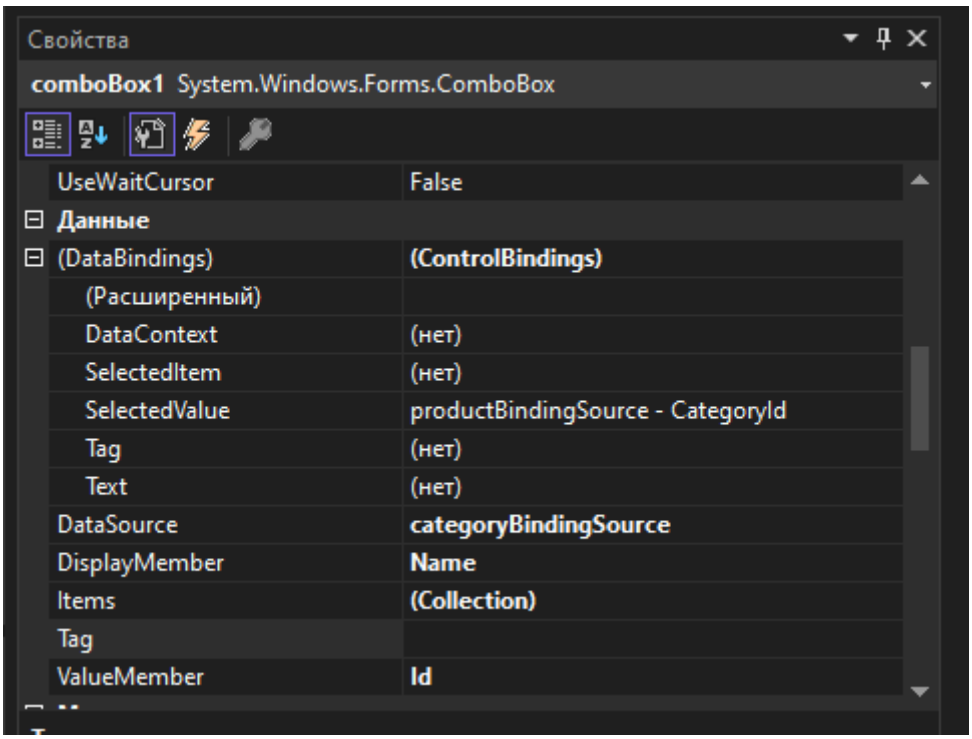


Можно выбрать источник данных для каждого свойства слева, для вывода информации в текстбоксе следует выбрать Text

Если вы не создали источник данных для выводимой сущности, то можете выбрать его из существующих классов, а именно из **других источников данных**, где вы можете выбрать подходящий класс.

Режим обновления источника данных следует поставить OnPropertyChanged.

Для редактирования связанных сущностей следует воспользоваться отдельными источниками данных. Для настройки привязки требуется создать поле комбобокс и зайти в свойство Данные:



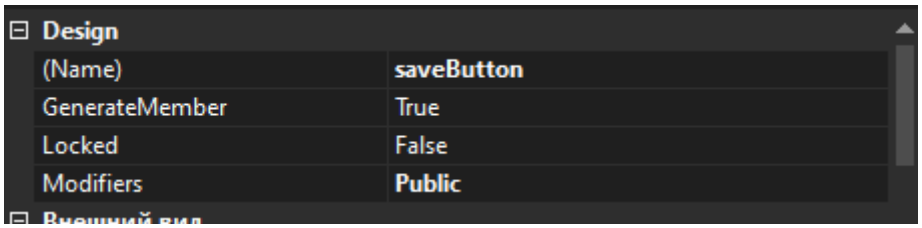
Внутри (DataBindings) поле Selected Value должно соответствовать ключу связанной сущности внутри редактируемой сущности.

DataSource соответствует источнику данных, из которого можно будет выбирать категории.

В DisplayMember можно выбрать атрибут у сущности по которому в комбобоксе будет выводится текст.

В ValueMember находится первичный ключ сущности, соответствующий внешнему ключу у редактируемой сущности.

Далее требуется создать на кнопку с модификатором public, чтобы можно было привязаться к её событиям из другой формы.



Модификатор Public, таким же образом, нужно выставить для всех источников данных в проекте

Затем перейдите на исходную форму, где отображаются все данные и добавьте логику для кнопки редактирования данных.

```
//обработка редактирования
Ссылок: 1
private void editButton_Click(object sender, EventArgs e)
{
    var form = new Form2();
    //Пробрасывание источника данных категорий в форму для редактирования
    form.categoryBindingSource.DataSource = categoryBindingSource.DataSource;
    //Пробрасывание выбранного элемента в источнике данных в форму
    form.productBindingSource.DataSource = bindingSource1.Current;
    //Переиспользование логики сохранения
    form.saveButton.Click += saveButton_Click;
    form.Show();
}

Ссылок: 2
private void saveButton_Click(object sender, EventArgs e)
{
    context.SaveChanges();
    dataGridView1.Refresh();
}
```

Добавление и удаление

Логика добавления и удаления следующая

```
Ссылок: 1
private void add_Click(object sender, EventArgs e)
{
    var form = new Form2();
    form.categoryBindingSource.DataSource = categoryBindingSource.DataSource;
    //одновременно добавляет элемент в источник данных и на форму
    form.productBindingSource.DataSource = bindingSource1.AddNew();
    form.saveButton.Click += saveButton_Click;
    form.Show();
}

Ссылок: 1
private void delete_Click(object sender, EventArgs e)
{
   //удаляет текущий элемент
   bindingSource1.RemoveCurrent();
   context.SaveChanges();
   dataGridView1.Refresh();
}
```

Валидация сущностей

Валидация происходит с использованием интерфейса IEditableObject, отвечающий за транзационное добавление сущностей в базу. Эту абстракцию использует BindingSource для добавления данных. В ней же можно и проводить валидацию сущностей.

Такой класс может выглядеть следующим образом:

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
namespace WinFormsAppEntity.Models;

Ссылок: 10
public partial class Product : IEditableObject
{
    LazarevTaPrepContext context;

    Ссылок: 0
    public void BeginEdit()
    {
        context = new();
        context.Update(this);
    }

    Ссылок: 0
    public void CancelEdit()
    {

        var entry = context.Entry(this);
        if (entry.State == Microsoft.EntityFrameworkCore.EntityState.Modified)
        {
            entry.Reload();
        }

    }

    Ссылок: 0
    public void EndEdit()
    {
        //Логика валидации
        if (string.IsNullOrEmpty(Name))
        {
            throw new ValidationException("Имя не должно быть пустым");
        }
        //Сохраняет измененную сущность
        context.SaveChanges();
    }
}
```

В BeginEdit инициализируется ваш контекст, который ничего не знает о текущем объекте, поэтому нужен context.Update, чтобы EF стал отслеживать его.

В CancelEdit присутствует Reload для загрузки сущности из баз данных.

CancelEdit вызывается напрямую из кода и из datagridview, когда строка с проблемой выходит из фокуса

```
Ссылок: 2
private void saveButton_Click(object sender, EventArgs e)
{
    try
    {
        bindingSource1.EndEdit();
    }
    catch (ValidationException error)
    {
        MessageBox.Show(error.Message);
        bindingSource1.CancelEdit();
    }


    dataGridView1.Refresh();
}

Ссылок: 1
private void add_Click(object sender, EventArgs e)
{
    var form = new Form2();
    form.categoryBindingSource.DataSource = categoryBindingSource.DataSource;
    form.productBindingSource.DataSource = bindingSource1.AddNew();
    form.saveButton.Click += (o, e) =>
    {
        try
        {
            bindingSource1.EndEdit();
        }
        catch (ValidationException error)
        {
            MessageBox.Show(error.Message);
        }
    };
    ;
    form.Show();
}
```

Если при добавлении выйдет исключение, то сущность не будет добавляться как в BindingSource, так и в базу данных

Авторизация

Для следующего примера требуется самостоятельно создать окно авторизации и привязать поля, как в примере с редактированием. Для авторизации может использоваться следующий код:

```
using System.Data;
using WinFormsAppEntity.Models;

namespace WinFormsAppEntity
{
    Ссылок: 3
    public partial class Auth : Form
    {
        User user = new();
        Ссылок: 1
        public Auth()
        {
            InitializeComponent();
        }

        Ссылок: 0
        protected override void OnLoad(EventArgs e)
        {
            base.OnLoad(e);
            //нам не нужно обращаться к textbox
            userBindingSource.DataSource = user;
        }

        Ссылок: 1
        private void enter_Click(object sender, EventArgs e)
        {
            var context = new LazarevTaPrepContext();
            //лямбда выражение для поиска
            var u = context.Users.Where(x => x.Login == user.Login && x.Pwd == user.Pwd).FirstOrDefault();
            if (u != null)
            {
                Hide();
                new Form1().Show();
            }
            else
            {
                MessageBox.Show("Пользователь не найден");
            }
        }
    }
}
```