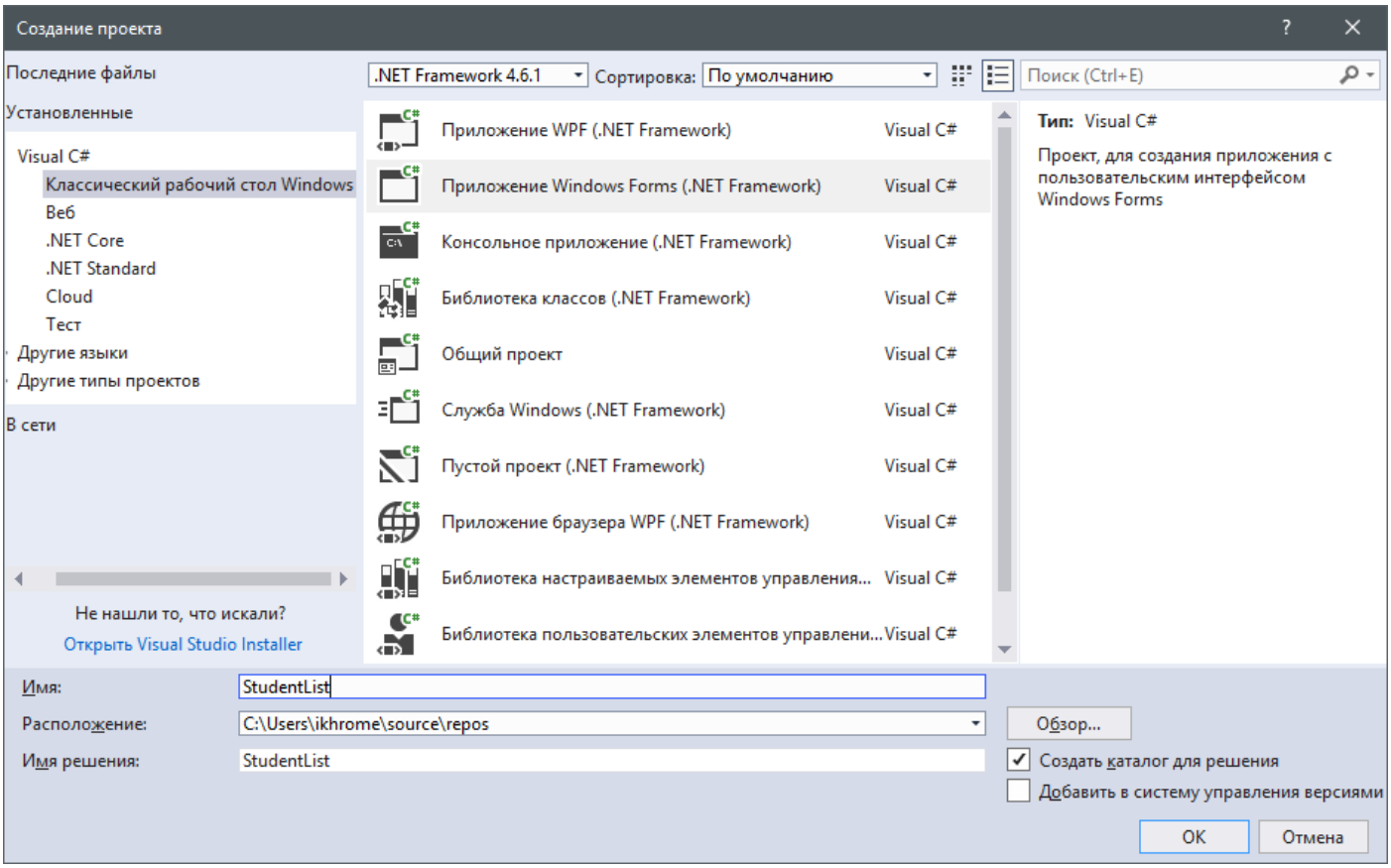
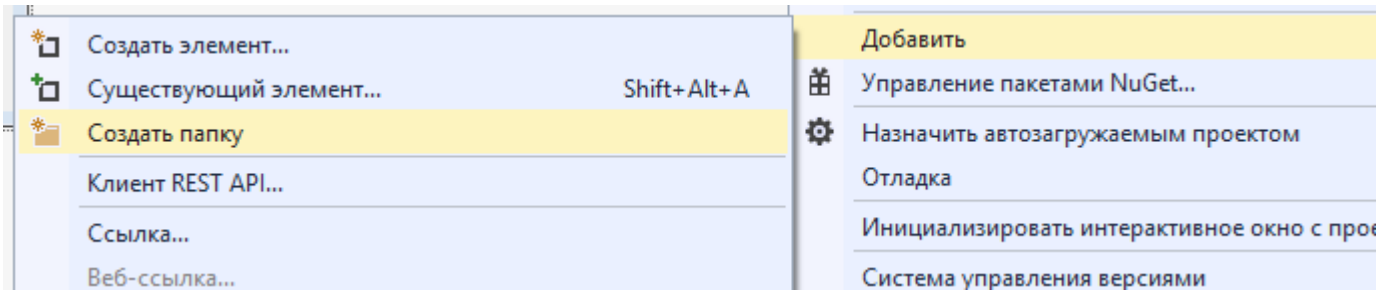
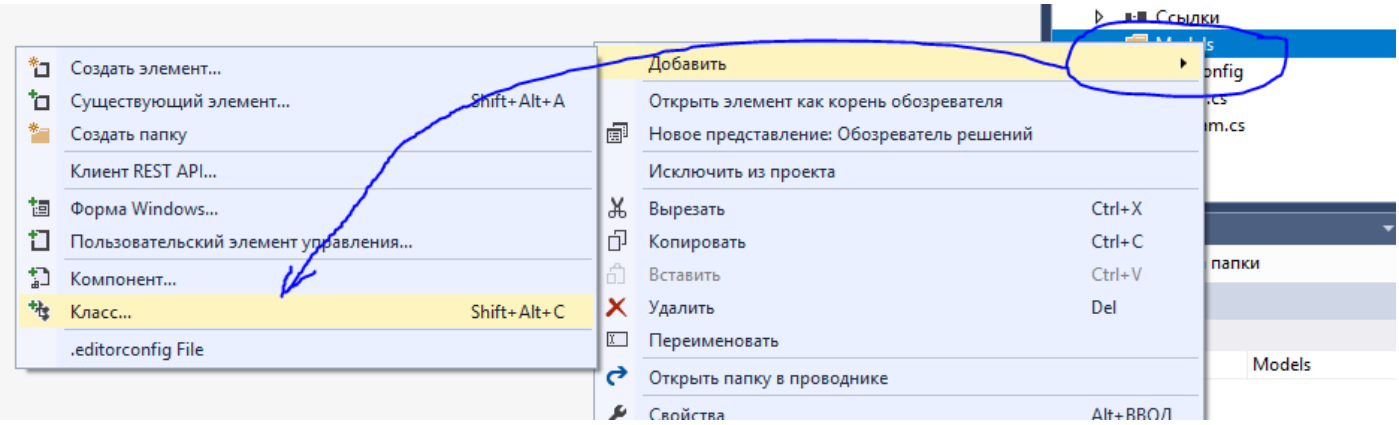
Создаем проект Win Forms:



ПКМ по проекту (значок с C# тоже подойдет) -> Добавить -> Создать папку

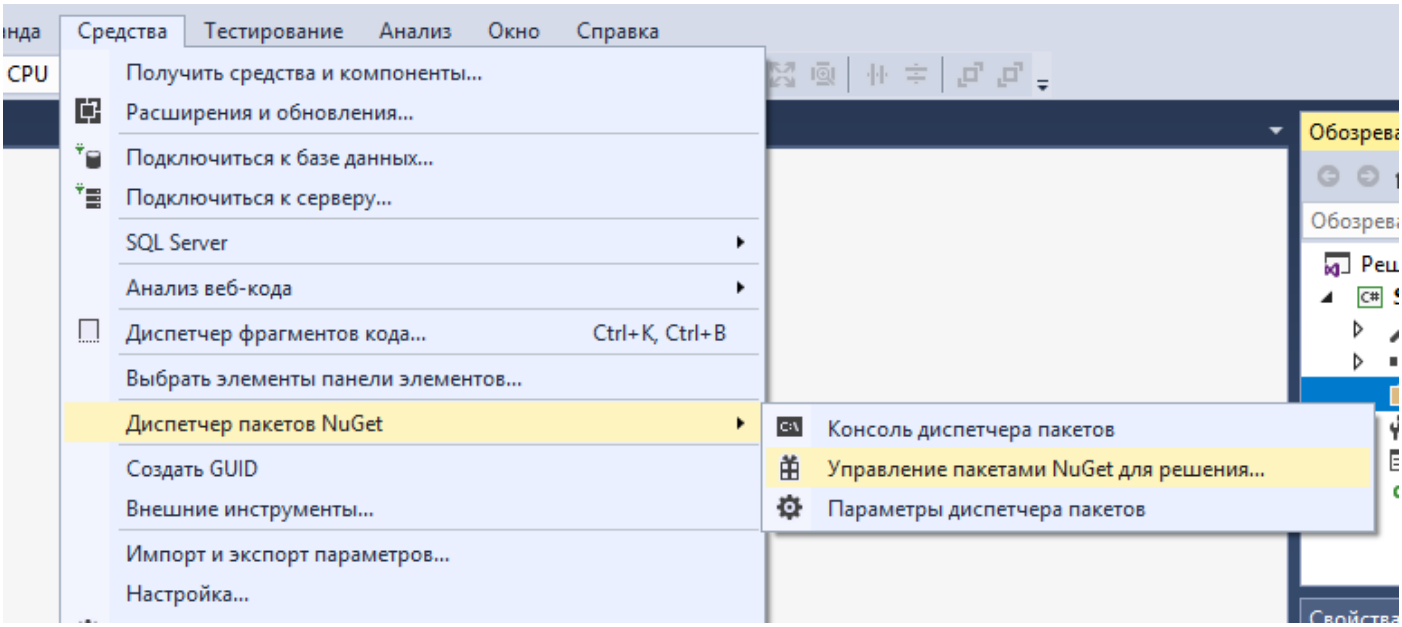


Назовите папку Models и следующим путём добавьте в нее 2 класса: AppDataContext и Student(например)

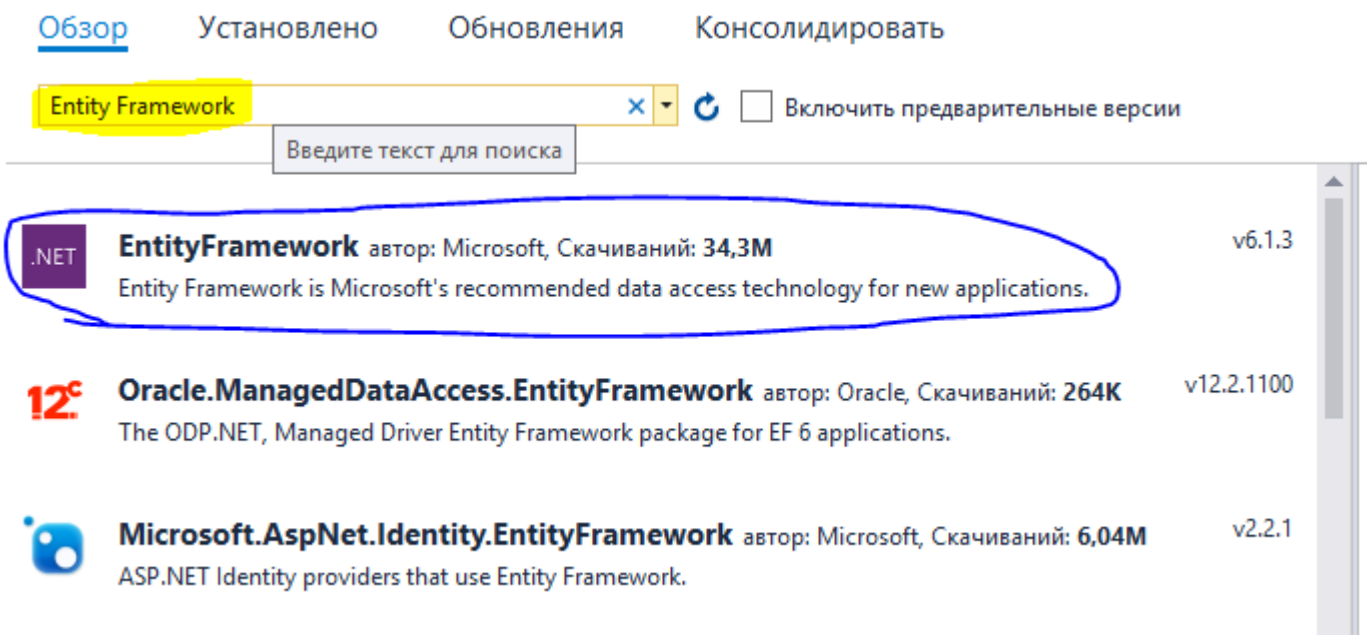


IDE сгенерирует для вас 2 файла – AppDataContext.cs и Student.cs

Добавим Microsoft® Entity Framework в проект. Запустим менеджер зависимостей(пакетов) NuGet:



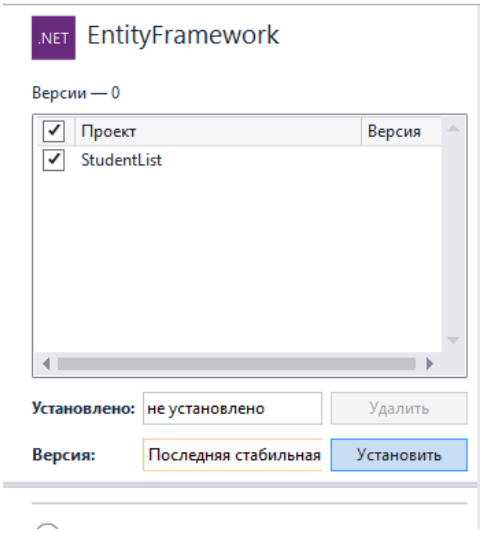
Кликаем на «Обзор» в открывшейся вкладке Visual Studio. Видим следующее, вводим «Entity Framework» в поле поиска:



В списке щелкаем по «EntityFramework». У кого-то это может быть «Microsoft.EntityFramework».

Выбираем проект. В моем случае это StudentList. Затем просто жмем «Установить».

Менеджер пакетов NuGet может выдать вам ошибку в том случае, если используется ОС ниже Windows 7(Vista, например), так как сам NuGet по сути набор командлетов узла PowerShell. На Windows 10 проблема с установкой может возникнуть, если используется аккаунт Windows без администраторских прав.



Можно это же самое проделать быстрее. Просто вместо «Управление пакетами…» откройте «Консоль…» и введите:

```
Install-Package EntityFramework -Version 6.1.3
```

И произойдет то же самое, что и в графическом установщике (везде одинаково):

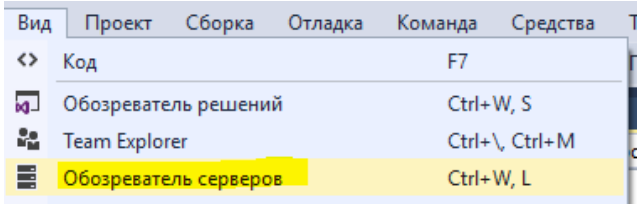
```
Вывод
Показать выходные данные из: Диспетчер пакетов
Выполнение файла сценария «C:\Users\ikhrome\source\repos\StudentList\packages\Ent

Type 'get-help EntityFramework' to see all available Entity Framework commands.
"EntityFramework 6.1.3" успешно установлено в StudentList
Выполнение действий NuGet заняло 28,92 sec
Прошло времени: 00:00:30.6638545
========== Готово ==========
```

Т.е. установится Entity Framework. Не забудьте принять лицензионное соглашение.

Продолжим создание схемы данных. Определим класс «Студент»

```
class Student
{
    public int Id { get; set; }
    public string FullName { get; set; }
    public DateTime BirthDate { get; set; }
    public string GroupName { get; set; }
    public string PhoneNumber { get; set; }
    public bool HaveDebts { get; set; } // долги по учебе
}
```

Обратите внимание. На лекции мы не проходили такое, но Entity Framework умеет распознавать такие типы SQL Server как datetime и date. Чуть позже мы добавим аннотацию, которая покажет серверу, что это просто дата, без времени. А пока создадим контекст данных и строку подключения. Тот способ, который я показывал работает эффективнее, т.к. IDE уже сгенерирует для вас строку подключения. Ее можно написать самостоятельно. Помните, что мой способ добавляет возможность просматривать SQL Server из Visual Studio без помощи SQL Server Management Studio. Это важно на слабых ПК. Так что для начала подключимся к серверу БД вот так →



Я нашел более простой способ создать БД для проекта, даже без использования SSMS (век живи – век учись).

Для этого нам нужно просто **знать имя компьютера**. Это можно сделать по нажатию Win+Pause в окне свойств ПК.

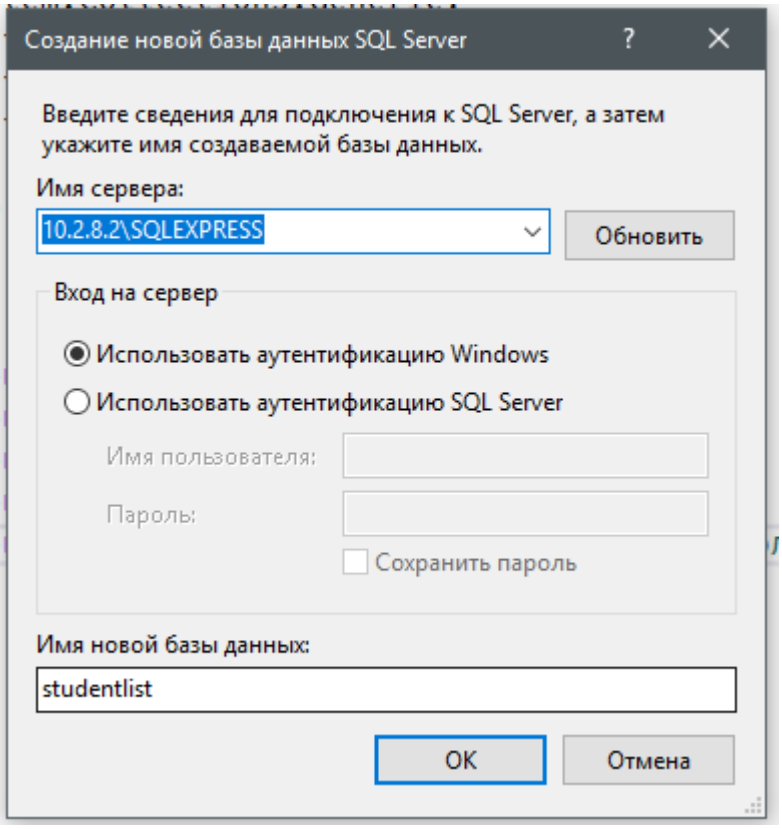
В «Обозреватель серверов» ПКМ по «Подключения данных», затем «Создать новую базу данных SQL Server».

В зависимости от настроек ОС могут сработать следующие имена серверов:

1. 192.168.0.<ЦИФРА> \SQLEXPRESS
2. 127.0.0.1\SQLEXPRESS
3. 10.2.8.2 – для локальной сети
4. <ИМЯ_ПК>\SQLEXPRESS
5. .\SQLEXPRESS
6. localhost

<ЦИФРА> заменяем на вывод в командной строке команды ipconfig:

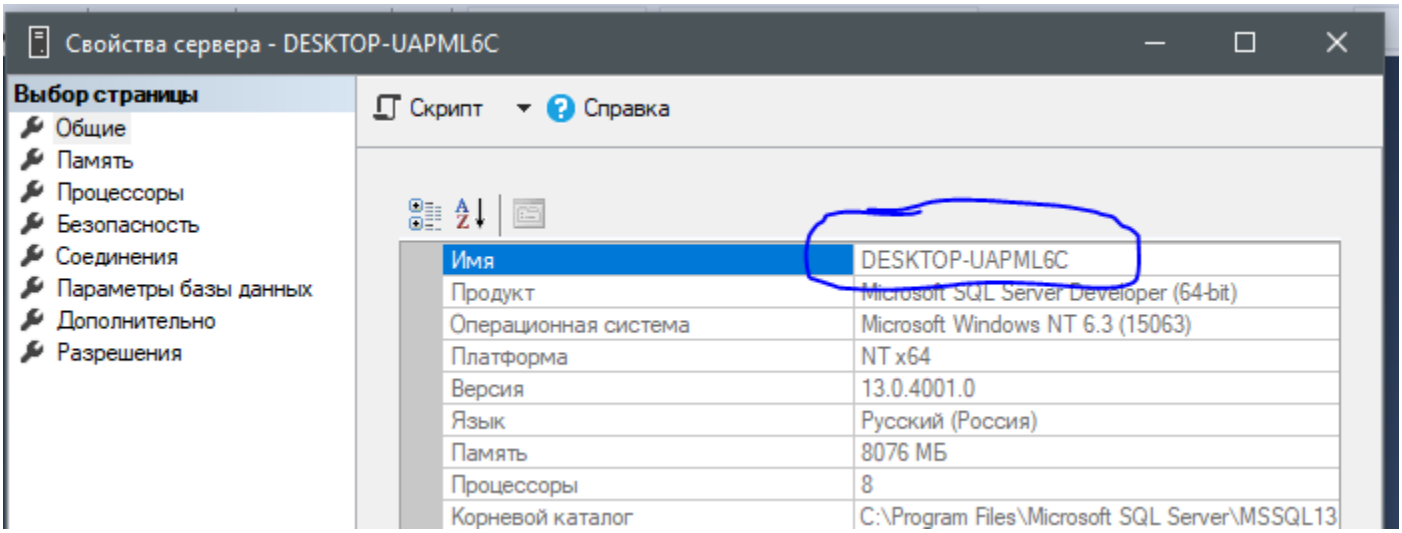
```
IPv4-адрес. . . . . . . . . . . . : 192.168.0.101
Маска подсети . . . . . . . . . . : 255.255.255.0
Основной шлюз. . . . . . . . . . : 192.168.0.1
```



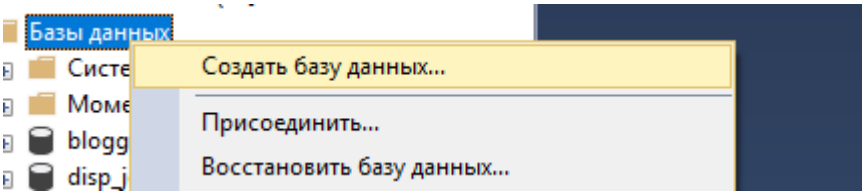
Вводим название БД и жмем «ОК», VS сообщит об успешности или неудаче операции.

В любом случае, разберитесь с этим.

Так же имя ПК можно посмотреть в SQL Server Management Studio, ПКМ по главному элементу дерева элементов(слева), затем «Свойства»:



Тут все гораздо проще – при работе в SSMS просто открываем дерево «Базы данных» и затем «Создать базу данных»:



Просто вводим ее название и Enter.

На этом готова сама БД. Далее вытащим строку подключения.

Сделать это можно двумя способами – самостоятельно или же при помощи Visual Studio.

Самостоятельно можно так:

1. Мы узнали имя сервера.
2. Мы знаем название БД

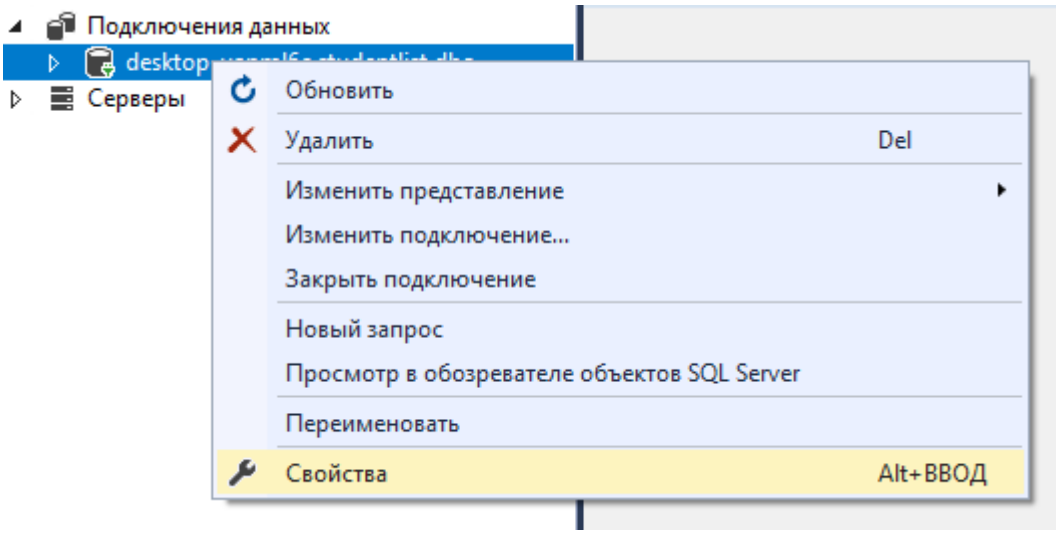
Просто составим это все вместе, добавив туда пару специфичных моментов.

```
Data Source=DESKTOP-UAPML6C;Initial Catalog=phonebook;Integrated Security=True
```

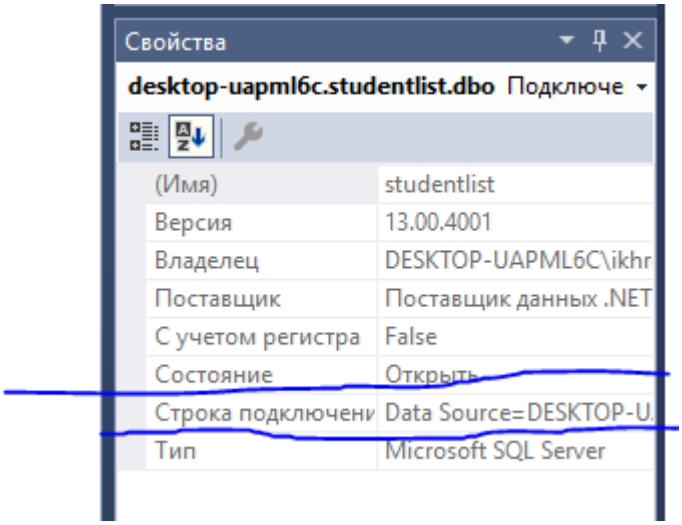
Здесь видим обычные пары «параметр=значение».

1. **`Data Source`** – наш сервер
2. **`Initial Catalog`** – название БД
3. **`Integrated Security`** – использовать аутентификацию Windows

Мы же скопируем строку из VS, тем более при использовании первого способа создания БД у вас автоматически добавится источник данных:



Нам нужны «Свойства». У вас по умолчанию эта панель может быть закрыта.



Нам нужен параметр «Строка подключения».

Скопируйте его значение.

В случае, если вы создали БД через SSMS – вам нужно лишь добавить подключение к ней.

Проделайте те же самые шаги, но только нам нужно добавить базу данных, ввести имя сервера.

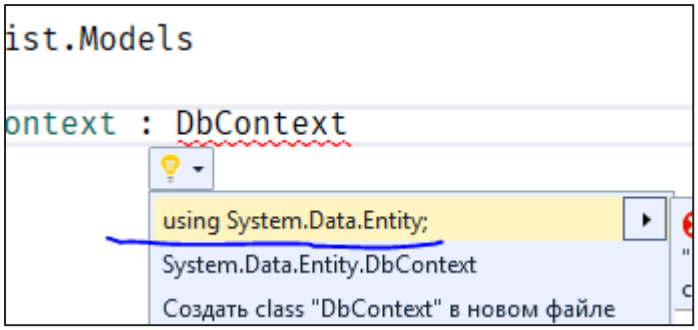
В любых шагах выбираем тип «Microsoft SQL Server»!

Теперь все просто – открываем App.config, и после </entityFramework> прописываем нашу строку подключения:

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="<тут она родимая!>"
        providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Все. Теперь добавим на форму компонент DataGridView. Он находится на панели компонентов в разделе «Данные». Изменяем размер переименуем в **`StudentData`**.

Сделаем так, чтобы контекст наших данных читался Entity Framework в дальнейшем. Унаследуем его от DbContext и добавим сет из данных класса «Student». Воспользуемся помощью Visual Studio и импортируем нужный нам класс:



Подводим к проблемному участку кода, видим лампочку, подводим на нее курсор и кликаем. Из списка выбираем **`using <Имя класса>`**

Это IntelliSense, система подсказок и автодополнения кода. Скажу сразу, в IntellijIDEA она на порядок лучше и надежнее, чем в VS, так что будьте предельно внимательны!

В данном случае подсказка верна.

Создадим таки конструктор и одно поле в этом классе:

```
namespace StudentList.Models
{
    class AppDataContext : DbContext
    {
        public AppDataContext() : base("DefaultConnection") { }

        public DbSet<Student> Students { get; set; }
    }
}
```

<u>ВАЖНАЯ ДЕТАЛЬ</u>: для удобства мы называем коллекции, заполненные объектами моделей во множественном числе.

Щелкаем двойным кликом по форме и для нас создали событие. И мы просто пишем код. Причем, используя выше описанные способы импорта классов.

Весь код события Load для формы:

```
private void StudentList_Main_Load(object sender, EventArgs e)
{
    using (var dbx = new AppDataContext())
    {
        dbx.Students.Load();

        StudentData.DataSource = dbx.Students.Local.ToBindingList();
    }
}
```

Классы StudentList.Models и System.Data.Entity VS поможет вам импортировать. Помним, что по умолчанию метода Load у нас нет, нужно импортировать System.Data.Entity.

Ну и теперь открываем консоль диспетчера пакетов NuGet и выполняем последовательно команды:

`Enable-Migrations` – вы должны увидеть сообщение на подобии «Code First Migrations enabled for project

StudentList.»

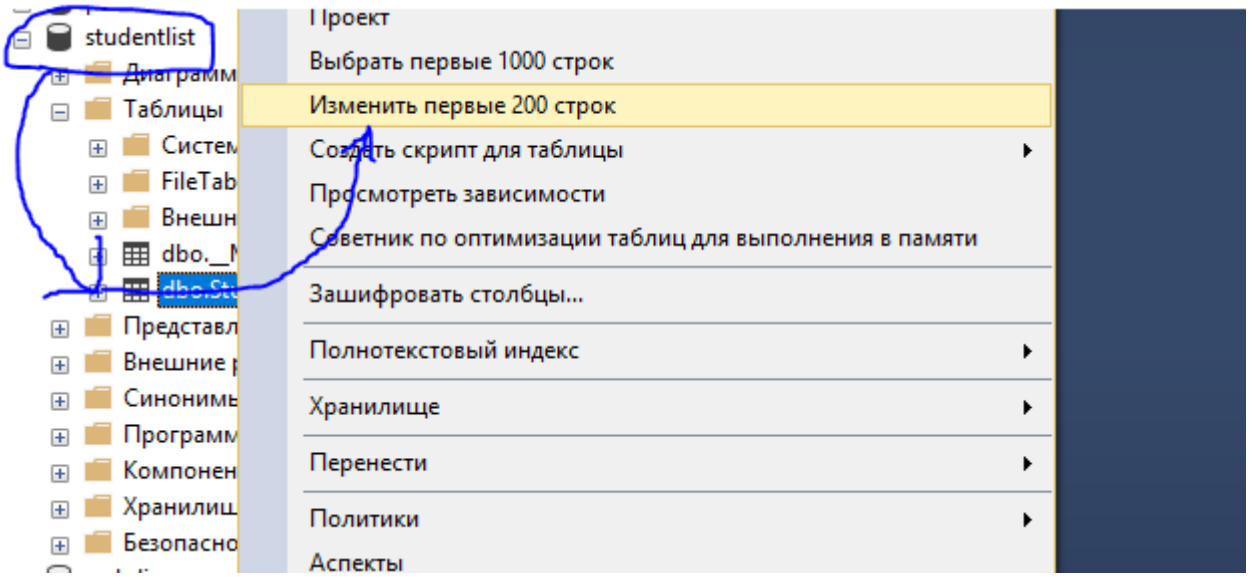
Add-Migration Init – должен открыться файл .cs с кодом миграции.

```
public partial class Init : DbMigration
{
    public override void Up()
    {
        CreateTable(
            "dbo.Students",
            c => new
                {
                    Id = c.Int(nullable: false, identity:
                    FullName = c.String(),
                    BirthDate = c.DateTime(nullable: fals
                    GroupName = c.String(),
                    PhoneNumber = c.String(),
                    HaveDebts = c.Boolean(nullable: false
                })
            .PrimaryKey(t => t.Id);

    }
```

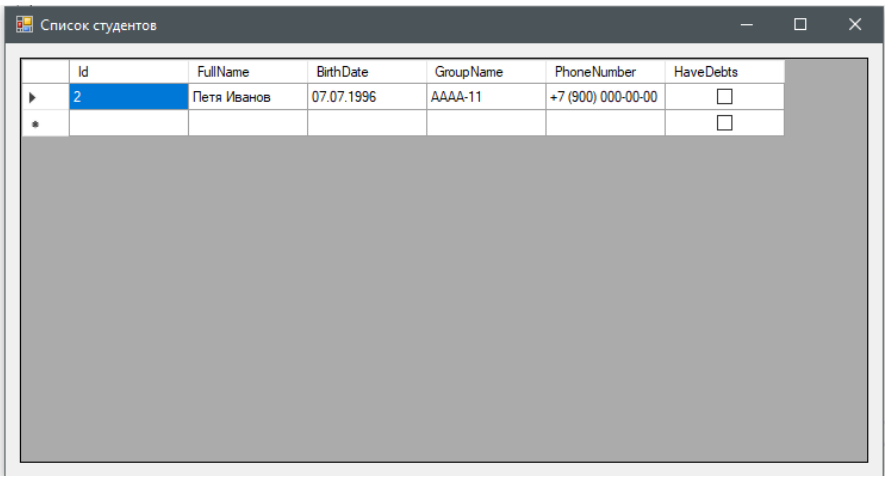
Update-Database – сообщения об ошибках вы точно не пропустите. У некоторых есть код, их дофига.

Погуглите, если не выйдет – разберемся на паре.

В SSMS можно добавить данные в таблицу. Следуем схеме:



И можете писать. Не пугаемся восклицательного знака! Это означает лишь что поле изменено.



Вот что получится в конце. Ура!