

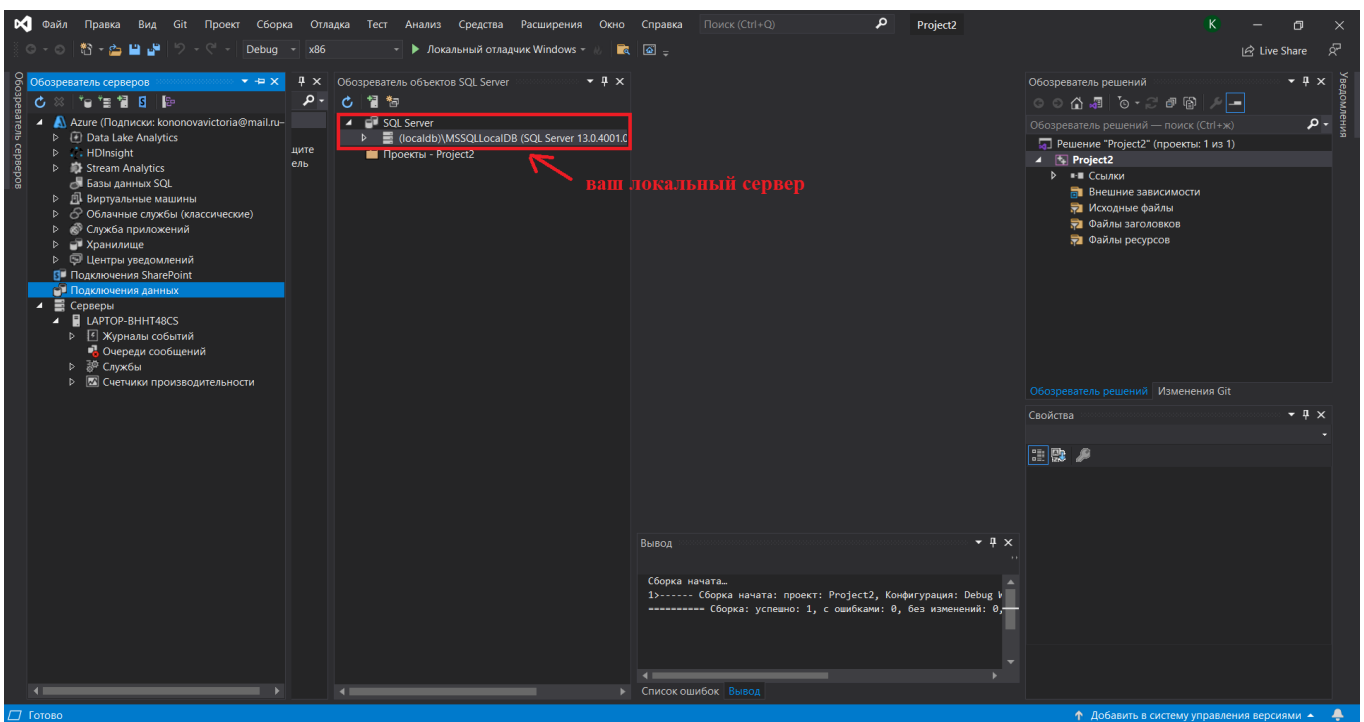
## Базы данных ADO.NET на языке C++\CLI (Автор этого материала Пахолков Юрий )

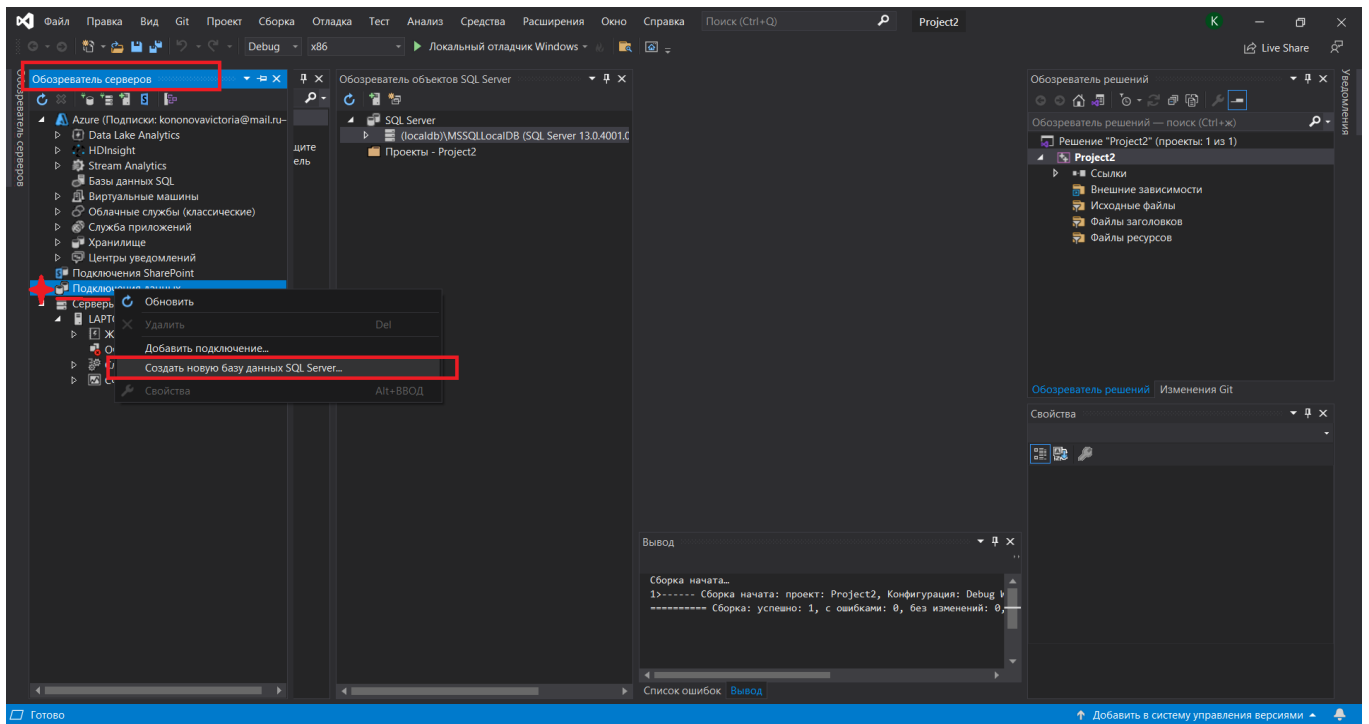
Материал взят из Интернет-источника url: <https://upread.ru/art.php?id=358>

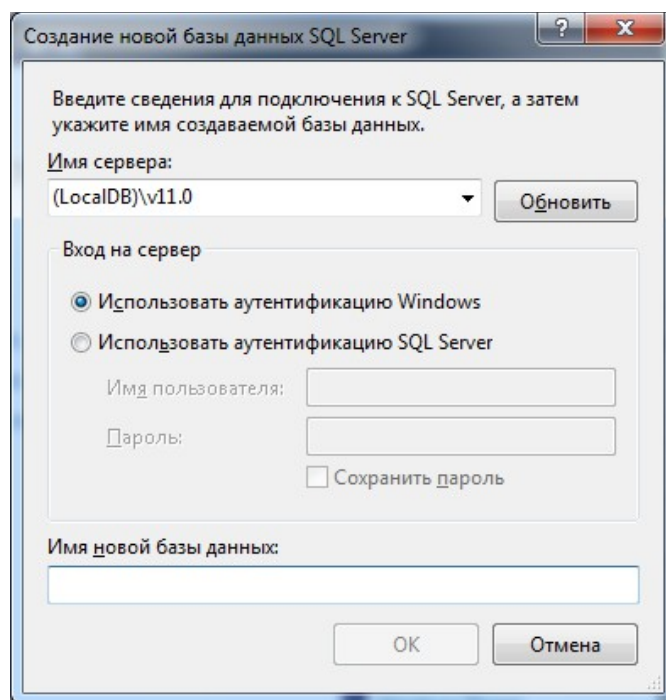
В материал добавлены скрины экрана Visual Studio 2019. Работоспособность инструкции проверена. Будьте внимательны с именами колонок!

Visual Studio работает с базами данных (БД) с помощью библиотеки ADO.NET, которая позволяет осуществлять доступ к различным типам серверов, одним из которых является Microsoft SQL Server (MSSQL). Он поставляется вместе с самой Visual Studio, и последняя обладает встроенной поддержкой для создания, удаления и редактирования БД на сервере MSSQL без использования дополнительных сторонних программ.

Первый шаг при разработке программы для работы с базами данных – это создание самой базы. Откройте любой проект. Выберите в меню Вид \ Другие окна \ Обозреватель серверов, в появившемся окне нажмите правой кнопкой мыши по «Подключения данных» и выберите «Создать новую базу данных SQL Server».







В качестве имени сервера введите (LocalDB)\v11.0.

**Вводите своё имя сервера, посмотрите как ваш сервер называется!**

MSSQL поддерживает 2 типа авторизации: по имени пользователя Windows, или по своему собственному списку пользователей. При использовании авторизации SQL Server Authentication необходимо ввести имя пользователя и пароль. Выберите «Использовать аутентификацию Windows» и укажите имя для новой базы данных.

Если не возникнет ошибок подключения или авторизации, то будет создана новая БД, и в окне **Обозреватель серверов** в разделе **Подключения данных** появится новая ветка с именем базы. Если ее раскрыть, то появятся записи для работы с этой базой.

Для примера создадим БД мобильных телефонов. Для хранения данных в базе используются таблицы, поэтому начнем с них. Разработка структуры будущей базы является важным этапом при создании приложений работы с данными, так четкая структура и наличие логических связей между таблицами упрощает приложение, и наоборот – лишние поля в таблицах, дублирование полей, недостаточные связи и т.п. могут существенно осложнить работу.

Сначала определим, какую информацию мы будем хранить о телефонах. В нашем случае, например, для упрощения будем учитывать 6 характеристик: производитель, модель, год выпуска, цвет, вес, изображение телефона. Если просто создать таблицу с заданными полями, то мы сможем хранить всю интересующую информацию, но это будет не оптимальное решение.

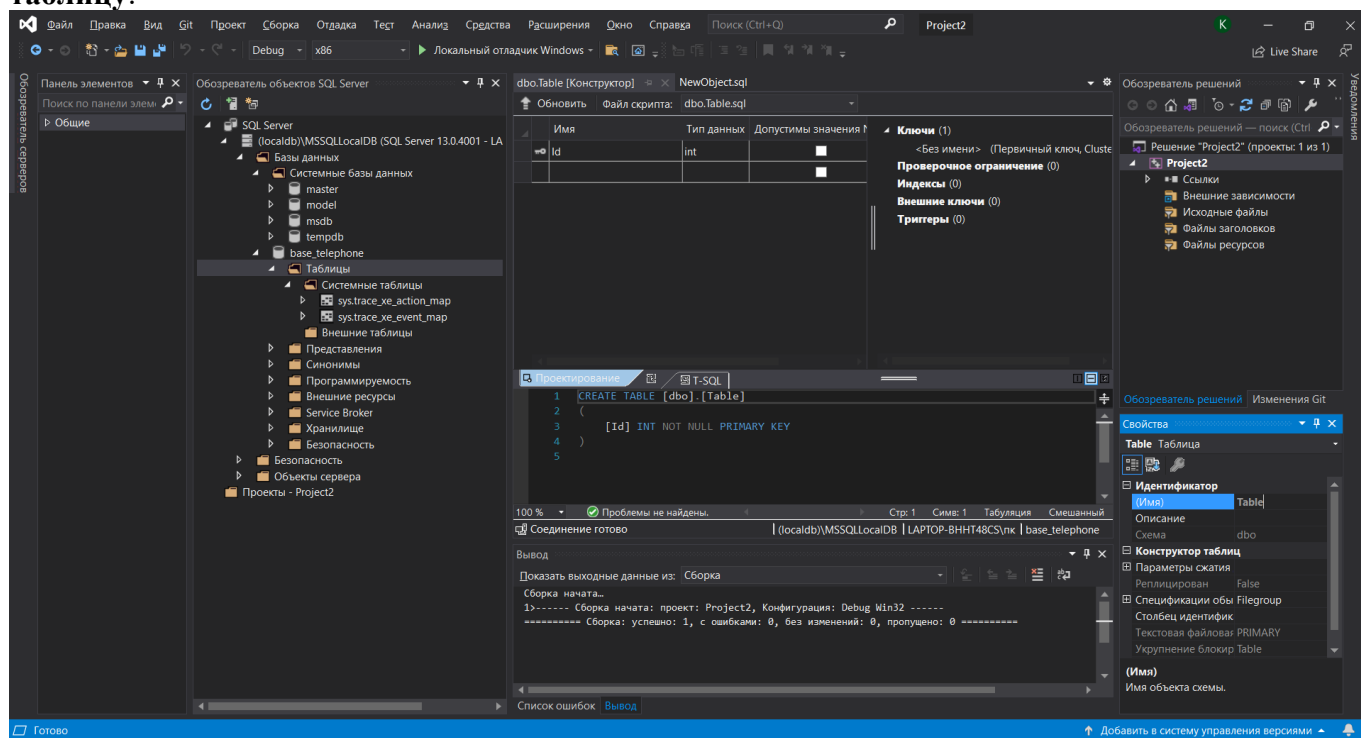
Все поля в общем случае будут иметь уникальное значение от телефона к телефону, за исключением производителя и года выпуска. Год можно кодировать целым числом типа int, поэтому данное поле не критично, но вот название производителя будет повторяться часто, тем более оно является строковым параметром, и его хранение в одной таблице с моделями телефонов нецелесообразно, так как попросту приведет к дублированию строк. А теперь представим, что производитель сменил свое наименование, а у нас в таблице уже содержатся тысячи моделей данной фирмы, которые потребуется все переименовывать!

Для решения проблемы лучше будет хранить имена производителей в отдельной таблице, а в таблице моделей просто указывать его номер. Тогда для смены названия фирмы просто понадобится сменить имя в таблице производителей у одной записи, и все, ее номер не поменяется, и таблицу моделей редактировать не придется. К тому же когда нам понадобится вывести все модели телефонов заданной фирмы, поиск будет осуществляться по номеру, т.е. сравнением целых чисел, что намного быстрее, нежели сравнение строк в первом случае.

Еще одним приемом при проектировании БД, позволяющим упростить и ускорить работу с базой, является наличие уникального номера-идентификатора в таблицах (ключа), однозначно

определяющего строку таблицы. Гораздо удобнее и быстрее обращаться к записи в таблице в виде «номер 231», чем «фирма Nokia, модель ..., год ..., цвет ..., вес ..., ...».

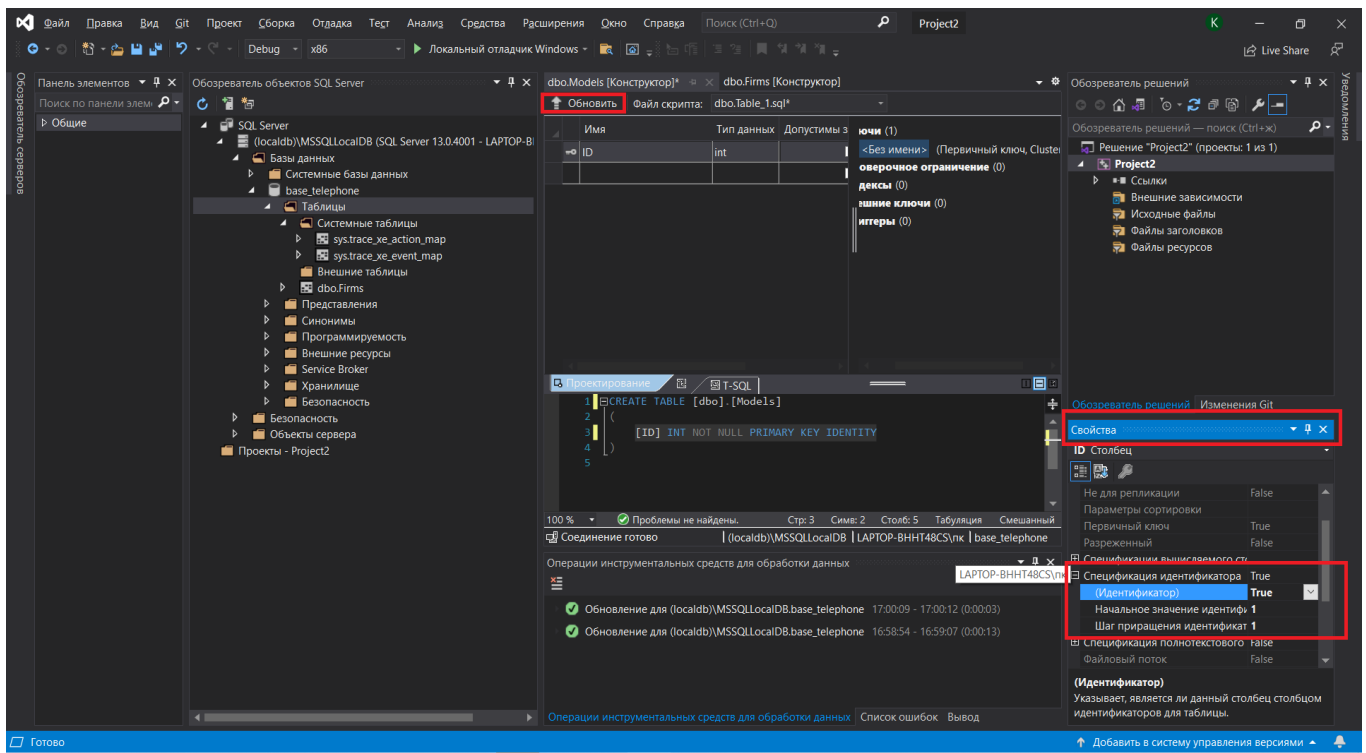
Начнем с создания таблицы производителей. Если раскрыть ветку с созданной базой данных в дереве в окне **Обозреватель серверов**, то появятся записи Таблицы, Представления, Типы и другие объекты базы. Нажмите правой кнопкой по **Таблицы** и выберите **Добавить новую таблицу**.



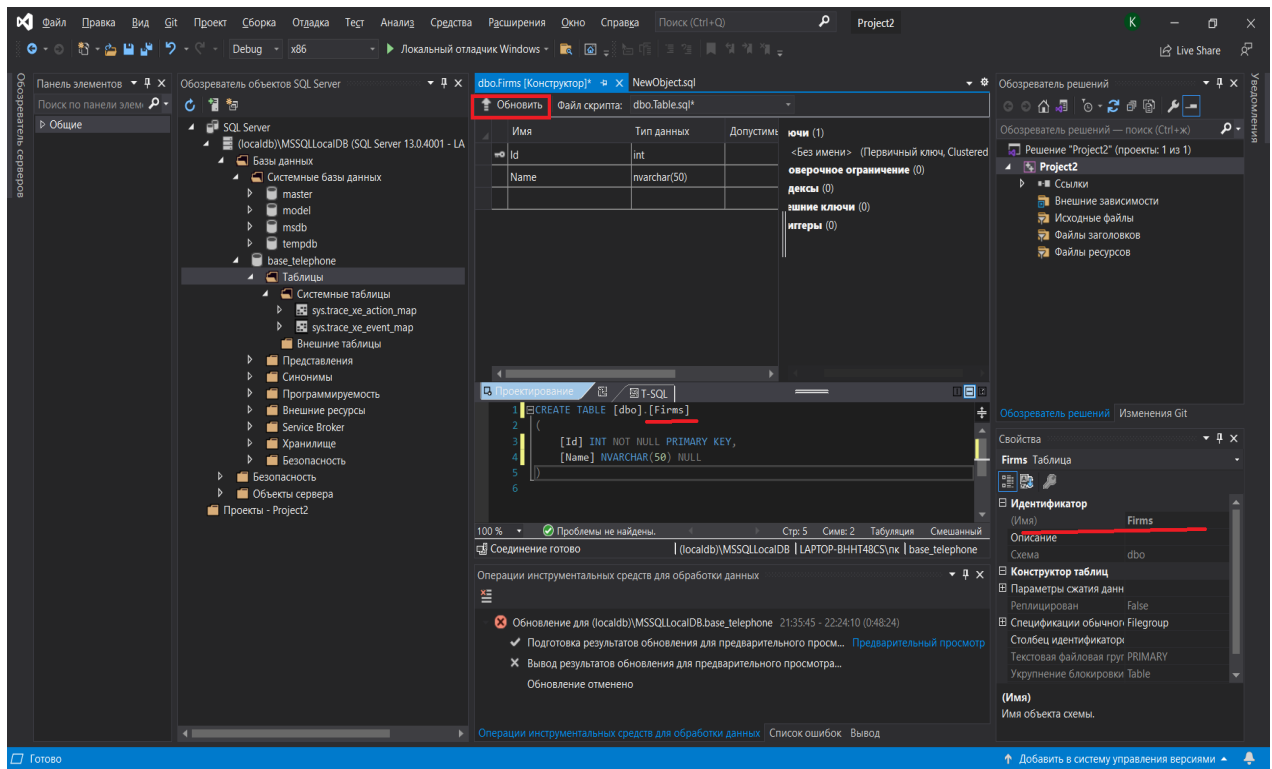
Появится окно ввода имени полей таблицы, их типов и «Допустимы значения Nulls» (может ли поле не содержать значения, т.е. быть пустым). Создайте следующие поля:

Имя поля	Тип	Allow Nulls
ID	int	
Name	nvarchar(20)	

- Поле ID – уникальный числовой идентификатор типа int. Чтобы сервер MSSQL сам назначал номера при добавлении новых записей в таблицу, в свойствах поля в списке выберите **Спецификация идентификатора** и установите значения True для (Идентификатор).



- Поле Name – название фирмы, nvarchar(20) означает строка длиной до 20 символов. Измените имя таблицы на Firms и нажмите кнопку «Обновить».



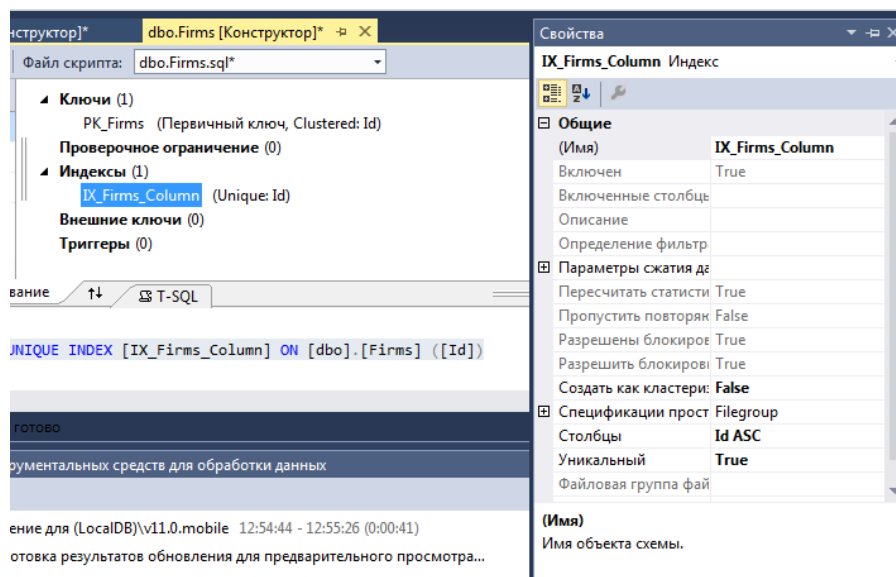
Теперь создайте еще одну таблицу Models для моделей телефонов с полями:

Имя поля	Тип	Allow Nulls
ID	int	
ID_Firm	int	
model	nvarchar(20)	+
year	int	+
color	nvarchar(15)	+
weight	int	+
photo	image	+

Названия полей говорят сами за себя, поле ID\_Firm будет содержать номер фирмы, т.е. значения из поля ID в таблице Firms.

Вы всегда можете поменять структуру таблицы, нажав правой кнопкой по ее имени в окне **Обозреватель серверов** и выбрав **Открыть определение таблицы**.

Вызовите таким образом на редактирование таблицу Firms, нажмите правой кнопкой на поле Индексы, и в появившемся меню выберите «Добавить новый индекс». В свойствах созданного индекса установите параметр **Уникальный** в значение True. В параметре **Столбцы** выберите поле ID. Нажмите правой кнопкой мыши на поле ID таблицы, и в появившемся меню выберите «Задать первичный ключ». Т.е. поле ID теперь является уникальным ключом (идентификатором) для записей в таблице. Обновите базу данных.



Расширения Окно Справка Поиск (Ctrl+Q) Project2

Панель инструментов: Live Share

dbo.Models [Конструктор] **dbo.Firms [Конструктор]**

Обновить Файл скрипта: dbo.Table.sql

Имя	Тип данных
ID	int
Name	nvarchar(50)

**Ключи (1)**  
PK\_Firms (Первичный ключ, Clustered: ID)

**Проверочное ограничение (0)**

**Индексы (1)**  
IX\_Firms\_Column (Unique: ID)

**Внешние ключи (0)**

**Триггеры (0)**

Проектирование T-SQL

```
7  
8 GO  
9  
10 CREATE UNIQUE INDEX [IX_Firms_Column] ON [dbo].[Firms] ([ID])  
11
```

100 % Проблемы не найдены. Стр: 10 Симв: 1 Табуляция Смешанный

Соединение готово | (localdb)\MSSQLLocalDB | LAPTOP-BHNT48CS\ПК | base\_telephone

Операции инструментальных средств для обработки данных

- ✓ Вывод результатов обновления для предварительного просмотра...
- ✓ Создание скрипта базы данных... [Просмотр скрипта](#)
- ✓ Выполнение скрипта обновления в базе данных "base\_telepho... [Просмотр результатов](#)

Обновление успешно завершено

✓ Обновление для (localdb)\MSSQLLocalDB.base\_telephone 17:16:43 - 17:16:48 (0:00:04)

Операции инструментальных средств для обработки данных Список ошибок Вывод

Обозреватель решений

Обозреватель решений — поиск (Ctrl+ж)

Решение "Project2" (проекты: 1 из 1)

Project2

- Ссылки
  - Внешние зависимости
  - Исходные файлы
  - Файлы заголовков
  - Файлы ресурсов

Обозреватель решений Изменения Git

Свойства IX\_Firms\_Column Индекс

Параметры сжатия данных

Пересчитать статистику	True
Пропустить повторяющиеся ключи	False
Разрешены блокировки строки	True
Разрешить блокировку страниц	True
Создать как кластеризованный	False

Спецификации пространства лог. Filegroup

Столбцы	ID ASC
Уникальный	True

Файловая группа файлового потока

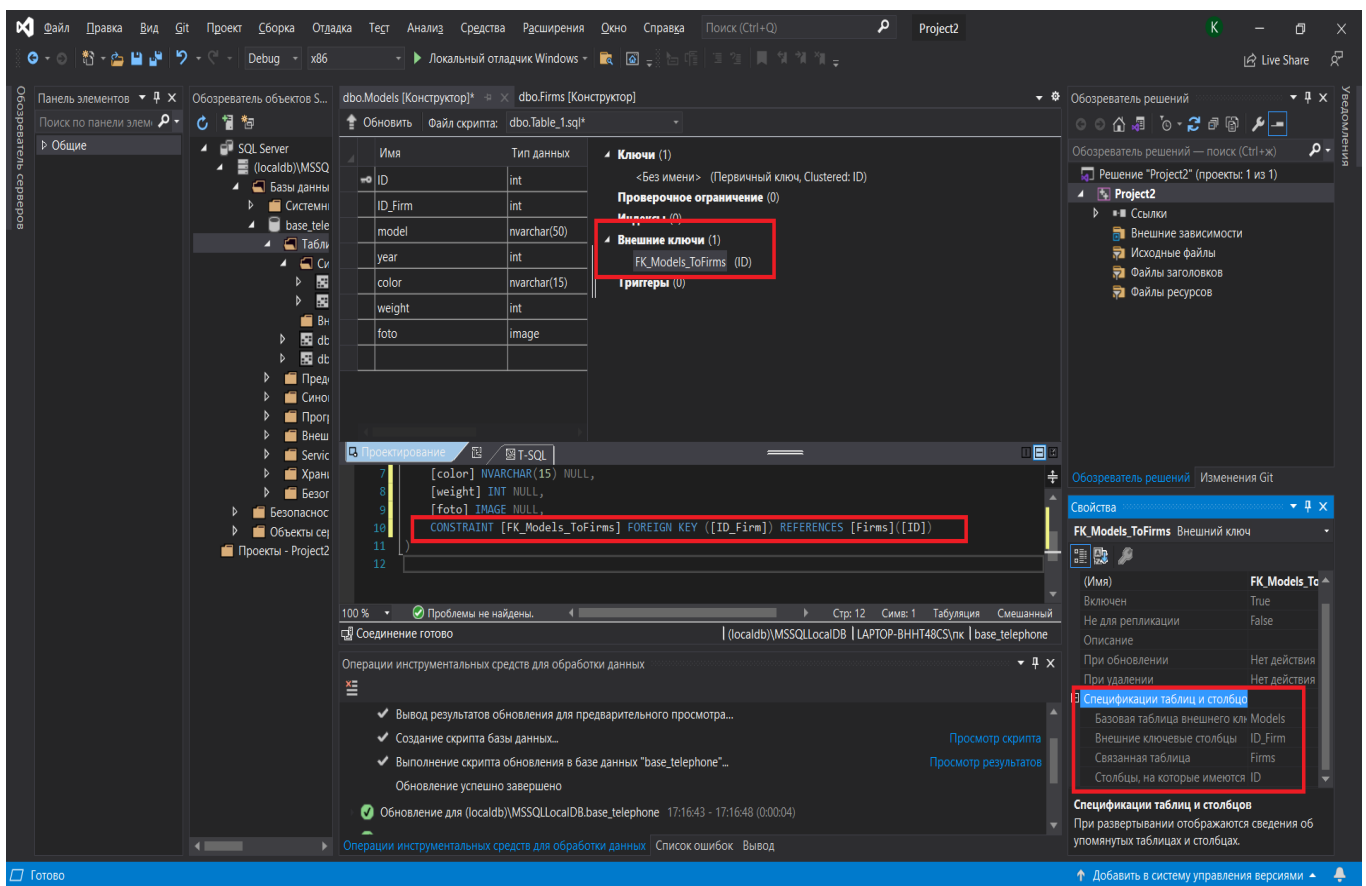
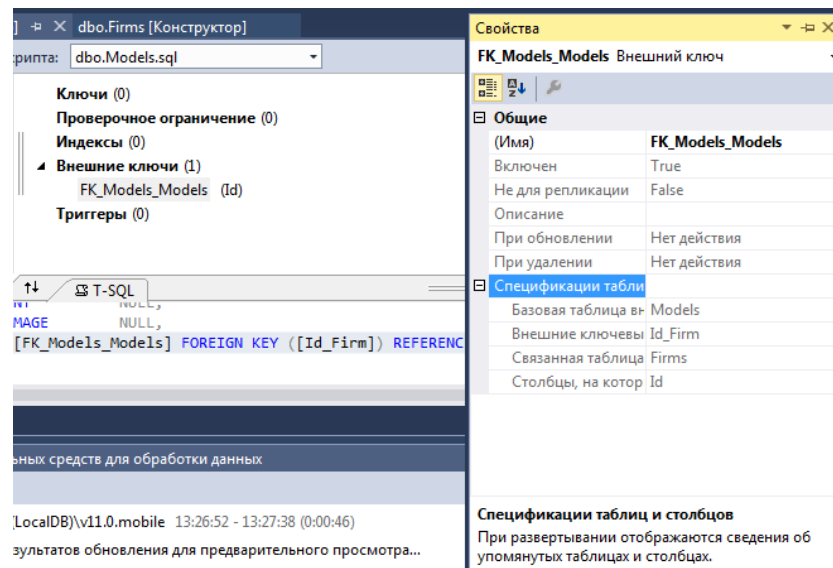
Характеристики заполнения

(Имя)  
Имя объекта схемы.

Добавить в систему управления версиями



Теперь отредактируем таблицу Models, по аналогии нажмите правой кнопкой на пункте «Внешние ключи», и в появившемся меню выберите «Добавить новый внешний ключ». Внешние ключи позволяют задавать связи между таблицами, в нашем случае надо сообщить серверу, что поле ID\_Firm связано с полем ID из таблицы Firms.



Обновите базу данных.

Теперь можно занести данные в таблицу. Нажмите правой кнопкой по имени таблицы, выберите меню **Показать таблицу данных**.

Для таблицы Firms:

	ID	Name
	1	Nokia
	2	Sony
	3	Samsung
►*	<a href="#">NULL</a>	NULL

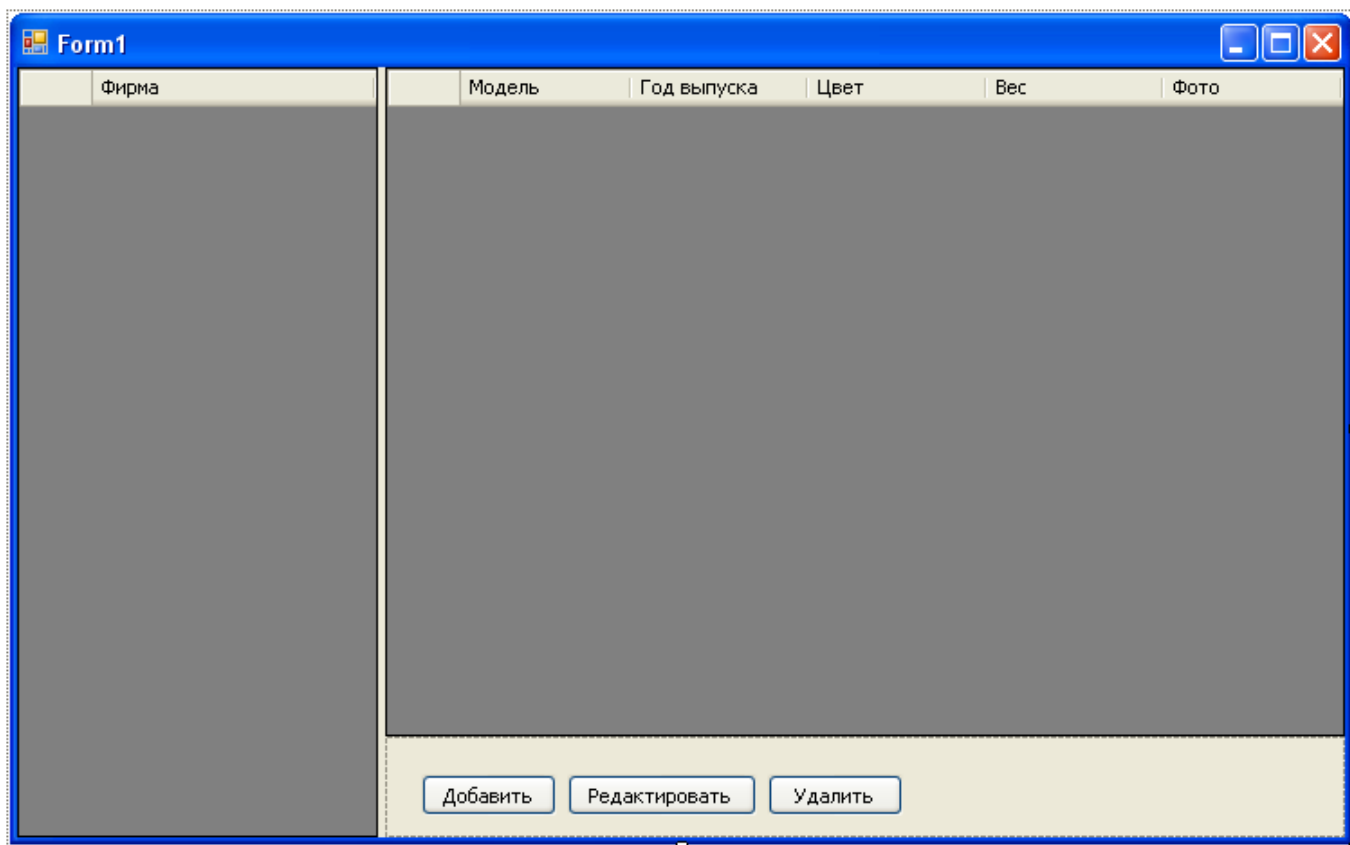
Для таблицы Models:

	ID	ID_Firm	model	year	color	weight	photo
	1	1	N80	2005	черный	134	NULL
	2	1	9300i	2006	серый	167	NULL
	3	2	CMD-J70	2006	белый	92	NULL
	4	3	SGH-D830	2004	черный	85	NULL
	5	3	SGH-E490	2005	красный	83	NULL
►*	<a href="#">NULL</a>	NULL	NULL	NULL	NULL	NULL	NULL

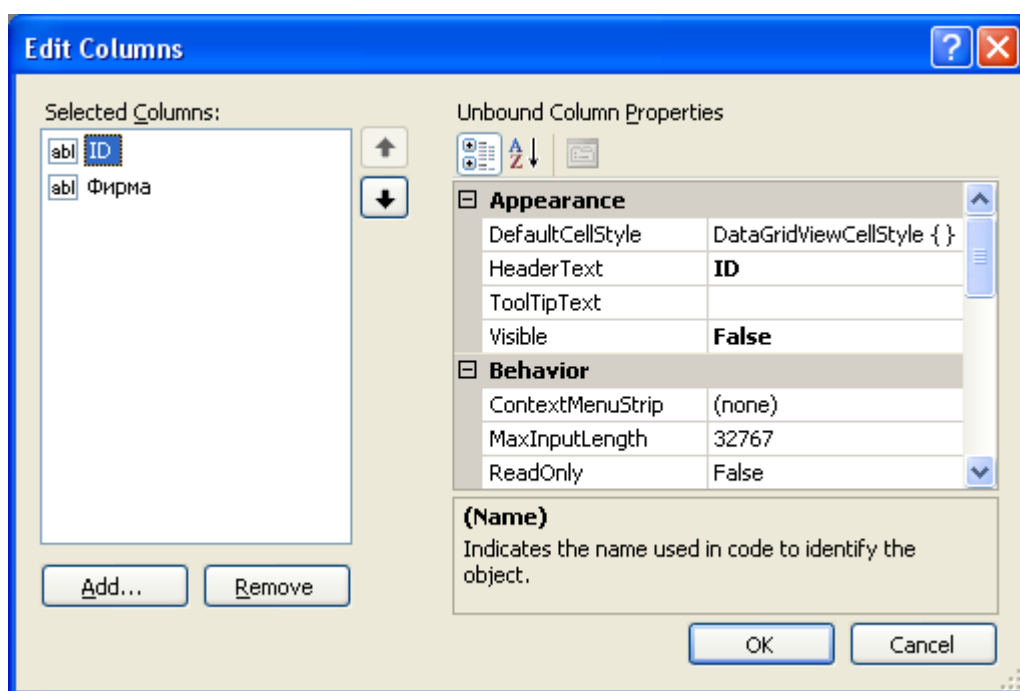
Вся занесенная информация является лишь примером, реальные характеристики телефонов можно найти в Интернете (например, на сайте <http://mobil.ru>).

Теперь, когда БД готова, можно приступить к написанию программы. Создайте проект Windows Forms, поместите на форму компонент SplitContainer. На его левую панель поместите DataGridView (поменяйте имя на dgvFirms, выравнивание Dock на Fill). На правую панель поместите компонент Panel с выравниваем по низу Dock = Bottom, на эту панель поместите 3 кнопки «Добавить», «Редактировать», «Удалить» с именами btnModelAdd, btnModelEdit и btnModelDel.

На оставшееся место в правой панели SplitContainer добавьте еще один компонент DataGridView с именем dgvModels и выравниваем Fill:

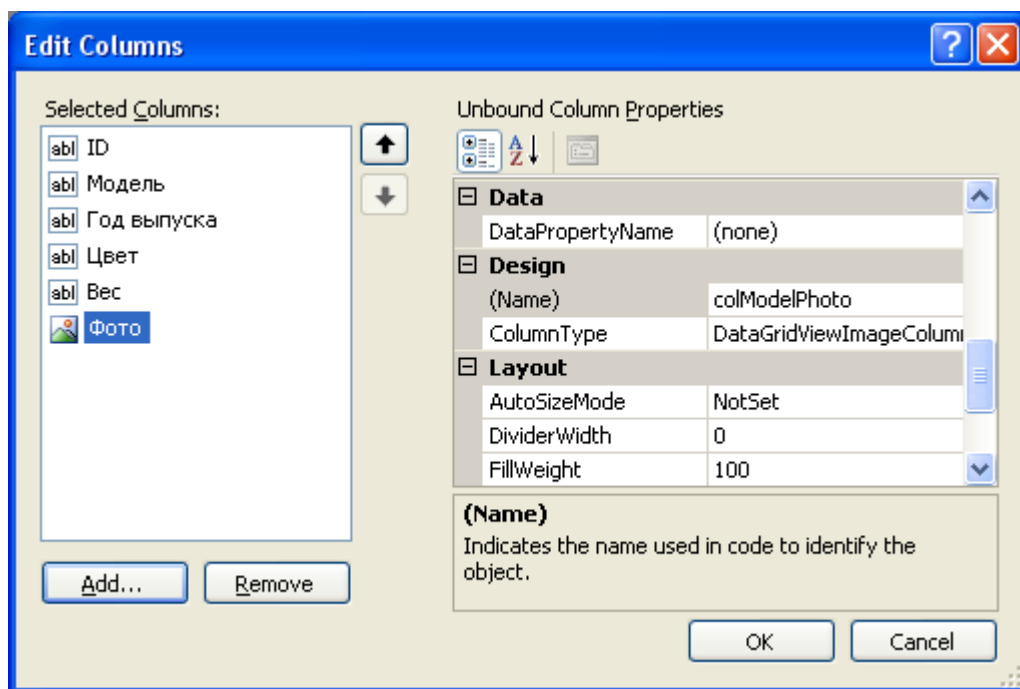


Чтобы появились заголовки в компонентах DataGridView (далее будем называть просто грид) как показано на рисунке, нажмите сначала правой кнопкой мыши на левом гриде, и выберите **Правка столбцов**:



Данный грид будет содержать имена фирм, поэтому добавьте в него 2 столбца с именами (св-во Name) colFirmID и colFirmName. Header text это то, что будет отображаться в заголовке столбца в самом гриде, поэтому желательно занести соответствующие русские названия. Столбец colFirmID будет хранить идентификатор записи и отображаться не будет, поэтому установите у него свойство Visible в false.

Теперь создадим столбцы для правого грида, который будет содержать модели телефонов:



Имена для столбцов соответственно: colModelID, colModelName, colModelYear, colModelColor, colModelWeight, colModelPhoto. Последний столбец будет отображать фото телефона, поэтому поменяйте его тип на DataGridViewImageColumn. Первый столбец будет содержать идентификатор ID моделей, поэтому сделайте его невидимым Visible = false.

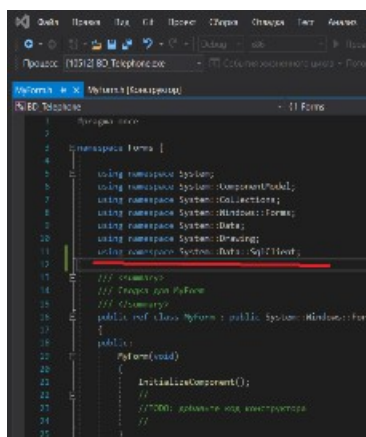
Установите у обоих гридов свойства AllowUserToAddRows = false, AllowUserToDeleteRows = false и ReadOnly = true, что отключит непосредственное редактирование в гридах (мы будем использовать отдельные кнопки). SelectionMode = FullRowSelect и ColumnHeadersHeightSizeMode = AutoSize, AutoSizeColumnsMode = Fill чтобы выбирать курсором строки целиком и растягивать столбцы по ширине грида соответственно.

Отдельно у правого грида dgvModels установите AutoSizeRowsMode = AllCellsExceptHeaders, чтобы высота строк в нем растягивалась по высоте изображения телефона.

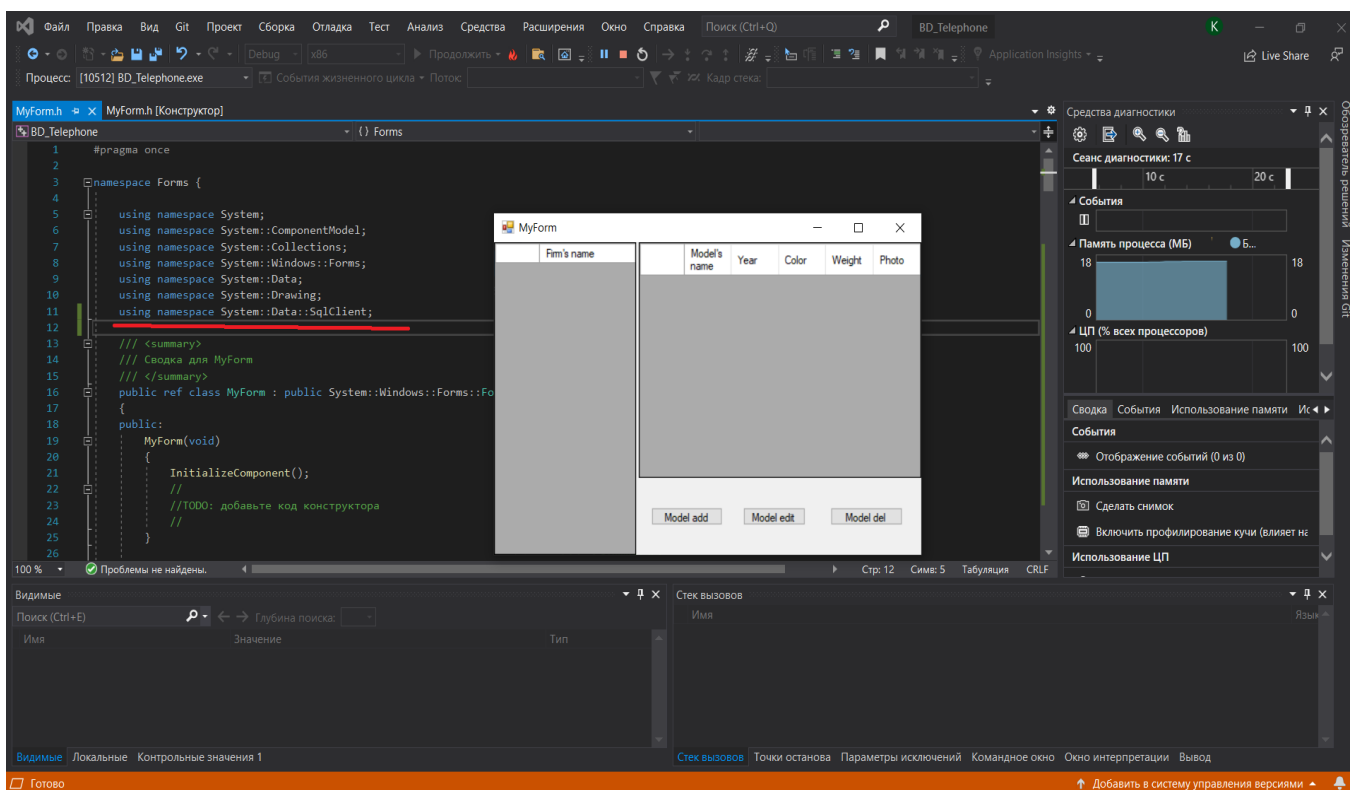
Всего компонент DataGridView содержит более 200 свойств и событий для настройки отображения данных, начиная со смены цвета фона ячеек до их ручной прорисовки с помощью библиотеки GDI+ и т.п., но это выходит за рамки рассмотрения данного раздела.

Все классы для работы с MSSQL находятся в соответствующем пространстве имен, поэтому добавьте к остальным директивам using в начале файла:

```
using namespace System::Data::SqlClient;
```



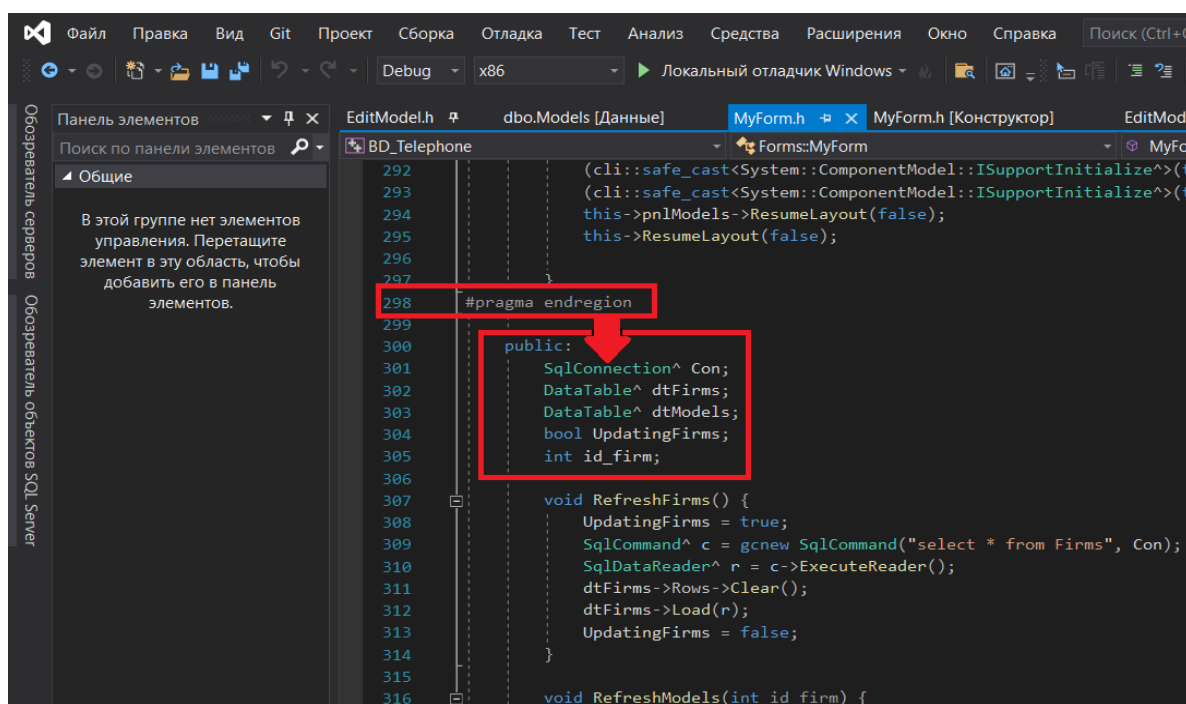
Запустите программу и проверьте на наличие ошибок.



Как уже можно было догадаться из проделанных действий, в левой части окна программы будет находиться перечень фирм-производителей телефонов, в правой модели телефонов для выбранной фирмы, кнопки внизу будут осуществлять манипуляцию данными.

Добавьте внутрь класса формы переменные:

```
public:
    SqlConnection^ Con;
    DataTable ^dtFirms;
    DataTable ^dtModels;
    bool UpdatingFirms;
    int id_firm;
```



Переменная `Con` будет хранить соединение к базе данных, объекты класса `DataTable` используются для данных таблиц, булева переменная-флаг `UpdateFirms` пригодится позже, переменная `id_firm` – идентификатор текущей выбранной фирмы.

Добавьте в конструктор формы код (отмечен жирным шрифтом):

```
public:
    Form1(void)
    {
        InitializeComponent();
        //
        //TODO: Add the constructor code here
        //
        try {
            Con = gcnew SqlConnection("Data Source=.\SQLEXPRESS;Initial
Catalog=mobile;User ID=sa;Password=111");
            Con->Open();
        } catch (SqlException ^e) {
            MessageBox::Show("Ошибка при подключении к базе данных: " +
e->Message);
            Application::Exit();
        }
        dtFirms = gcnew DataTable();
        dtModels = gcnew DataTable();

        dgvFirms->DataSource = dtFirms;
        dgvFirms->AutoGenerateColumns = false;
        colFirmID->DataPropertyName = "ID";
        colFirmName->DataPropertyName = "name";

        dgvModels->DataSource = dtModels;
        dgvModels->AutoGenerateColumns = false;
        colModelID->DataPropertyName = "ID";
        colModelName->DataPropertyName = "model";
        colModelYear->DataPropertyName = "year";
        colModelColor->DataPropertyName = "color";
        colModelWeight->DataPropertyName = "weight";
        colModelPhoto->DataPropertyName = "photo";
    }
```

!!! Здесь имена колонок для подключения к sql-базе! Проверьте имена колонок!

При создании соединения с БД используется строка соединения (Connection String), содержащая параметры подключения: имя сервера, базы, логин, пароль и др. Ее можно посмотреть, если нажать правой кнопкой в окне **Обозреватель серверов** на соединении с БД в ветке **Подключения данных** и выбрать **Свойства**. Хранение строки непосредственно в коде программы не лучшее решение, так как при смене настроек сервера БД, программу придется перекомпилировать, поэтому чаще ее хранят во внешних конфигурационных файлах.

Конструкция `try .. catch` используется для перехвата ошибок при исполнении программы. Если в коде, расположенном в блоке `try {}` произойдет ошибка (исключение `Exception`), то управление будет немедленно передано в блок `catch {}`. Исключение является классом, тип класса определяет вид произошедшей ошибки (в нашем случае ошибка класса `SqlException`), и объект данного класса будет помещен в переменную `e` для дальнейшей обработки программой.

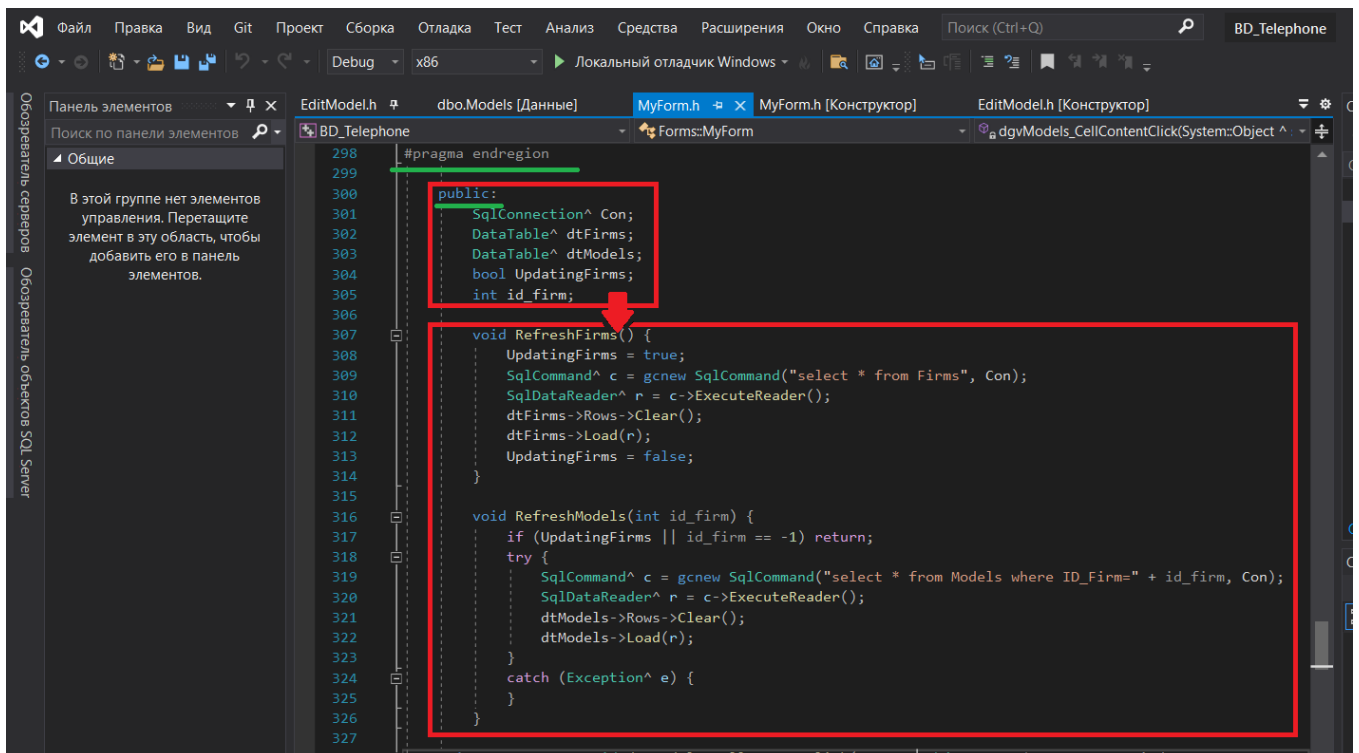
Так как операция подключения к серверу зависит от многих факторов и далеко не всегда будет успешной (нет сервера, не найден, нет базы, неверный логин и т.п.), то целесообразно проверять этот этап на ошибки. Далее в коде программы перехват ошибок встретится еще не раз.

После подключения создаются таблицы для данных, они назначаются гридам на форме в качестве источников для отображения информации, и идет присоединение столбцов к именам столбцов таблиц в БД. Свойство грида `AutoGenerateColumns` определяет, будет ли грид сам создавать столбцы. Так как мы их уже создали сами, то отключаем.

Добавьте в класс формы 2 функции:

```
void RefreshFirms() {
    UpdatingFirms = true;
    SqlCommand ^c = gcnew SqlCommand("select * from Firms", Con);
    SqlDataReader^ r = c->ExecuteReader();
    dtFirms->Rows->Clear();
    dtFirms->Load(r);
    UpdatingFirms = false;
}

void RefreshModels(int id_firm) {
    if (UpdatingFirms || id_firm == -1) return;
    try {
        SqlCommand ^c = gcnew SqlCommand("select * from Models where
ID_Firm=" + id_firm, Con);
        SqlDataReader^ r = c->ExecuteReader();
        dtModels->Rows->Clear();
        dtModels->Load(r);
    } catch (Exception ^e) {
    }
}
```



Первая делает запрос к БД для получения списка фирм из таблицы Firms и заносит данные в объект dtFirms. Вторая проделывает похожую операцию, но для моделей. При этом выбираются не все модели, а только те, которые относятся к текущей выбранной фирме (выборка select по идентификатору фирмы). Если фирма не выбрана (id\_firm == -1), или в данный момент выполняется ф-ция RefreshFirms(), то выборка не происходит. Дело в том, что класс SqlDataReader используемый при получении данных не может работать параллельно с другим таким же классом, поэтому для исключения такой вероятности используется флаг UpdatingFirms.

Назначьте для события SelectionChanged компонента dgvFirms обработчик:

```
private: System::Void dgvFirms_SelectionChanged(System::Object^ sender,
System::EventArgs^ e) {
    try {
        id_firm = Convert::ToInt32(dgvFirms->CurrentRow->Cells[0]-
>Value);
    } catch (Exception ^e) {
        id_firm = -1;
    }
    RefreshModels(id_firm);
}
```

Событие возникает, когда происходит перемещение курсора по фирмам, при этом нам необходимо получить ID выбранной фирмы (CurrentRow – выбранная строка в гриде, Cells[0] – первый столбец, содержащий идентификаторы), и обновить список моделей телефонов. Если при получении ID фирмы возникнет ошибка, то id\_firm = -1 и ф-ция RefreshModels обновлять список не будет.

Осталось заполнить гриды фирмами и моделями при первом запуске программы, для этого воспользуемся событием Shown формы:

```
private: System::Void Form1_Shown(System::Object^ sender, System::EventArgs^ e) {
    RefreshFirms();
    dgvFirms_SelectionChanged(nullptr, nullptr);
}
```

Запустите программу, попробуйте выбирать разные фирмы для смены моделей. Фактически мы добились просмотра нашей БД сотовых телефонов, пора приступить к редактированию, для чего уже заранее были созданы 3 кнопки в нижней части окна.

Переменная Con из статьи не отработала. Нужно заменить на свою!

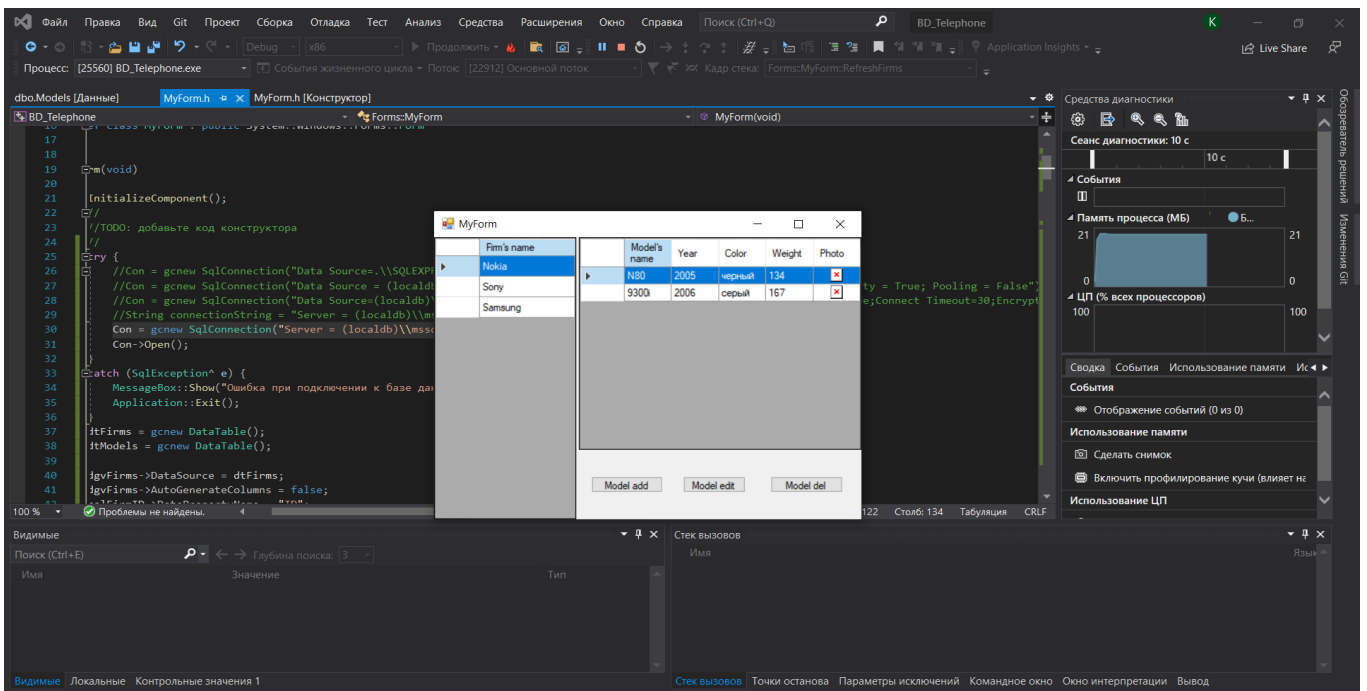
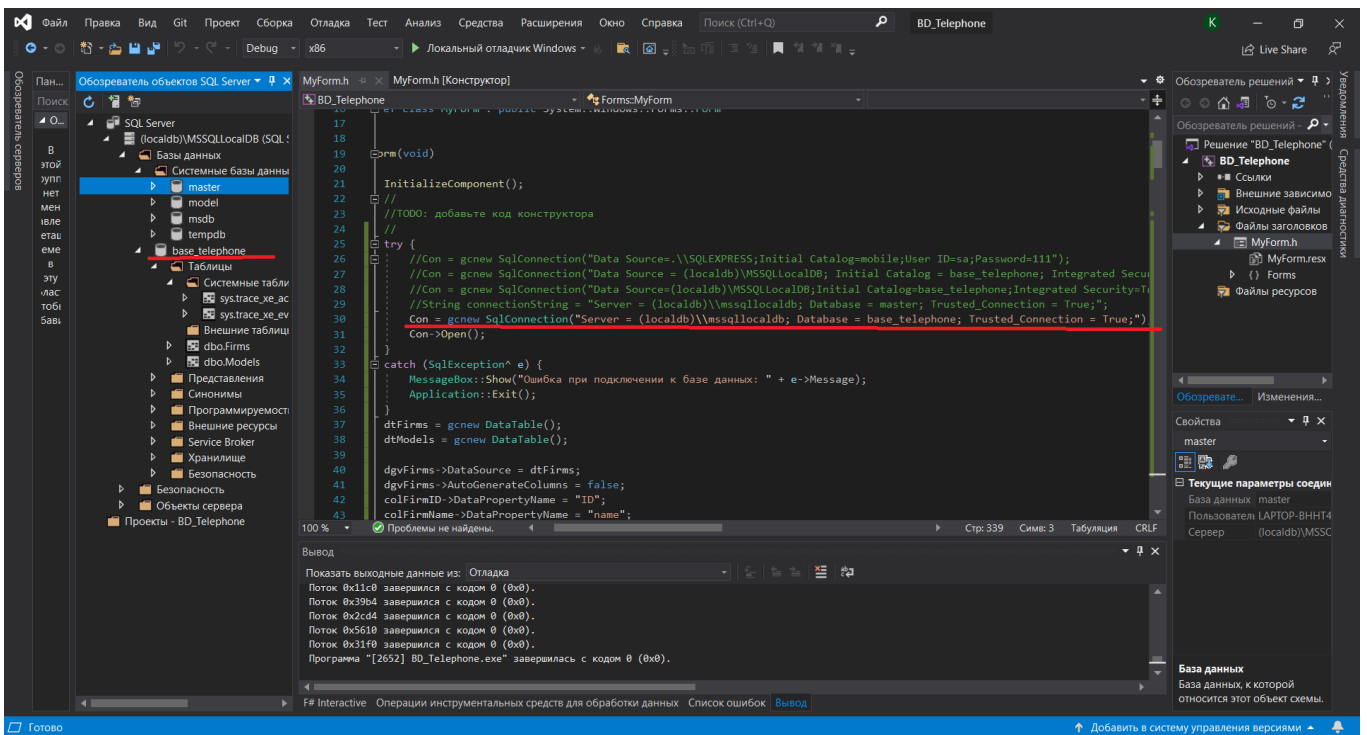
Попробуйте сначала подключиться к базе по умолчанию master с помощью этой строки кода

```
Con = gcnew SqlConnection("Server = (localdb)\\mssqllocaldb; Database = master;
Trusted_Connection = True;");
```

Если удачно, то посмотрите, как вы назвали свою базу на sql-сервере, в моем случае база имеет название base\_telephone и исправте переменную Con

```
Con = gcnew SqlConnection("Server = (localdb)\\mssqllocaldb; Database = base_telephone;
Trusted_Connection = True;");
```





Если все успешно, то запуститься окно программы.

Код для кнопки «Добавить»:

```
private: System::Void btnModelAdd_Click(System::Object^ sender, System::EventArgs^ e) {
    String ^sql = String::Format("insert into Models (ID_Firm, model) values ({0}, N'{1}']",
        id_firm, "Новая модель");
    SqlCommand ^c = gcnew SqlCommand(sql, Con);
    c->ExecuteNonQuery();
    RefreshModels(id_firm);
}
```

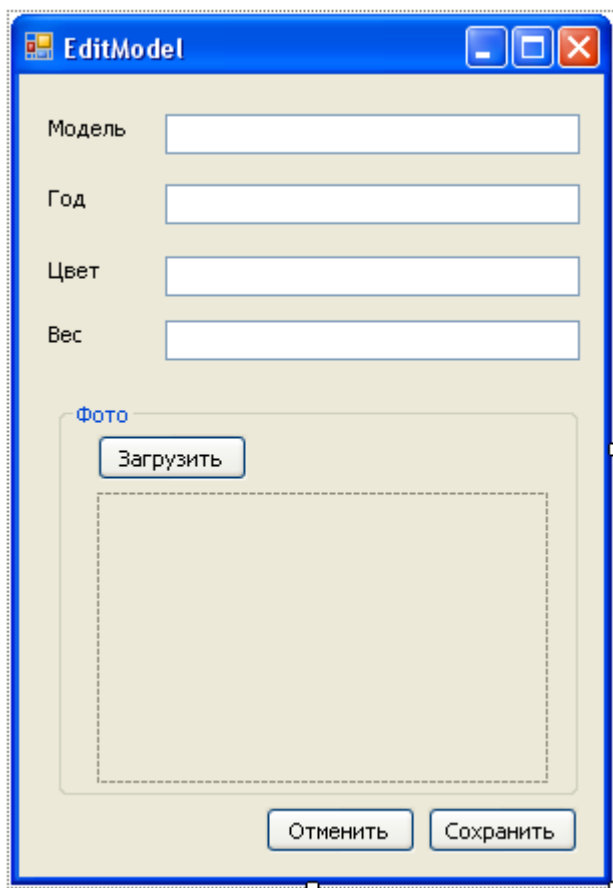
С помощью sql запроса insert добавляется новая запись в таблицу моделей с идентификатором текущей выбранной фирмы. Так как sql команда insert (как и все последующие) не возвращает данные в программу, то используется метод ExecuteNonQuery(). После добавления обновляем список моделей, чтобы изменения отразились в гриде.

Код для кнопки «Удалить»:

```
private: System::Void btnModelDel_Click(System::Object^ sender, System::EventArgs^ e) {
    if (dgvModels->RowCount == 0) return;
    int id = Convert::ToInt32(dgvModels->CurrentRow->Cells[0]->Value);
    String ^sql = "delete from Models where id=" + id;
    SqlCommand ^c = gcnew SqlCommand(sql, Con);
    c->ExecuteNonQuery();
    RefreshModels(id_firm);
}
```

Если в гриде моделей нет строк, то удалять нечего и сразу выходим. Иначе получаем идентификатор выбранной модели и удаляем запись с помощью sql запроса delete.

Для редактирования моделей потребуется создать новую форму, назовите ее EditModel:



Присвойте текстовым полям имена tbModel, tbYear, tbColor и tbWeight. Поле под кнопкой «Загрузить» – компонент PictureBox, задайте для него имя pbPhoto, свойство SizeMode = CenterImage. Для кнопок загрузить, отменить, сохранить – имена btnLoadPhoto, btnCancel и btnSave соответственно.

Для свойства формы CancelButton установите кнопку btnCancel, что позволит ей закрывать окно без написания какого-либо кода.

Так как в этой форме будем работать с БД, то добавьте в начало файла `using namespace System::Data::SqlClient;`

В класс формы добавьте переменные:

```
public:
    SqlConnection ^con;
    DataGridView ^dgv;
    int id;
```

Через них будет передавать в эту форму из главной объект SqlConnection чтобы иметь возможность выполнять sql команды, и грид моделей чтобы заполнить поля формы информацией о модели. Переменная id для хранения идентификатора модели.

Чтобы при появлении формы на экране в полях уже содержалась информация, создадим событие Shown:

```
private: System::Void EditModel_Shown(System::Object^ sender, System::EventArgs^ e)
{
    id = Convert::ToInt32(dgv->CurrentRow->Cells[0]->Value);
    tbModel->Text = Convert::ToString(dgv->CurrentRow->Cells[1]-
>Value);
    tbYear->Text = Convert::ToString(dgv->CurrentRow->Cells[2]-
>Value);
    tbColor->Text = Convert::ToString(dgv->CurrentRow->Cells[3]-
>Value);
    tbWeight->Text = Convert::ToString(dgv->CurrentRow->Cells[4]-
>Value);

    using namespace System::Drawing;
    using namespace System::IO;
    try {
        array<Byte> ^b = (array<Byte>^)dgv->CurrentRow->Cells[5]-
>Value;

        MemoryStream ^ms = gcnew MemoryStream(b->Length);
        ms->Write(b, 0, b->Length);
        pbPhoto->Image = Image::FromStream(ms);
        pbPhoto->Refresh();
    } catch (Exception ^e) {
    }
}
```

**!!! В этом куске кода возникло исключение. Исправлено введением if (b)**

```
try {
    array<Byte>^ b = (array<Byte>^)dgv->CurrentRow->Cells[5]->Value;
    if (b) {
        MemoryStream ^ms = gcnew MemoryStream(b->Length);
        ms->Write(b, 0, b->Length);
        pbPhoto->Image = Image::FromStream(ms);
        pbPhoto->Refresh();
    }
} catch (Exception^ e) {
}
```

Получение текстовой информации о цвете, модели и др. не представляет труда, они хранятся в столбцах грида по порядку. С полем фото сложнее, так как оно представляет собой двоичную информацию, т.е. массив байт. Вдобавок к этому, объект Image в PictureBox может считывать информацию из файла, потока, или другого такого же объекта, но не из массива. Поэтому чтобы представить массив байт в виде изображения, приходится предварительно записывать его в MemoryStream – поток, хранящий данные в памяти.

Код для кнопки «Загрузить»:

```
private: System::Void btnLoadPhoto_Click(System::Object^ sender, System::EventArgs^ e) {
    OpenFileDialog ^d = gcnew OpenFileDialog();
    d->Filter = "JPEG файлы (*.jpg)|*.jpg|Bitmap файлы (*.bmp)|*.bmp|
Все файлы (*.*)|*.*";
    if (d->ShowDialog() == Windows::Forms::DialogResult::OK)
        pbPhoto->ImageLocation = d->FileName;
}
```

Ничего особенного нет, занесение имени выбранного изображения в PictureBox.

Код для кнопки «Сохранить»:

```
private: System::Void btnSave_Click(System::Object^ sender, System::EventArgs^ e) {
    String ^sql = "update Models set model=N'{0}', year='{1}', "
        "color=N'{2}', weight='{3}' where ID=" + id;
    sql = String::Format(sql, tbModel->Text, tbYear->Text,
        tbColor->Text, tbWeight->Text);
    SqlCommand ^c = gcnew SqlCommand(sql, con);
    c->ExecuteNonQuery();

    if (pbPhoto->ImageLocation) {
        using namespace System::IO;
        FileStream ^fs = File::OpenRead(pbPhoto->ImageLocation);
        array<Byte> ^b = gcnew array<Byte>(fs->Length);
        fs->Read(b, 0, fs->Length);
        fs->Close();
        sql = "update Models set photo = @photo where ID=" + id;
        SqlCommand ^c2 = gcnew SqlCommand(sql, con);
        c2->Parameters->Add(gcnew SqlParameter("photo",
SqlDbType::Image));

        c2->Parameters["photo"]->Value = b;
        c2->ExecuteNonQuery();
    }
    DialogResult = ::DialogResult::OK;
}
```

Сохранение информации начинается с текстовых полей. Используется sql команда update, но стоит отметить, что, несмотря на то, что год и вес являются числами, внутри запроса их значения все равно помещаются внутри одиночных кавычек ‘’. Если бы их не было, то в случае, например, если после редактирования значение года выпуска окажется пустым, в sql запросе будет year= что вызовет ошибку на сервере. С кавычками же year='', сервер сам выполняет преобразование типов, и пустая строка будет расценена как 0, и ошибки не возникнет.

К сожалению, сохранить подобным образом изображение не получится, так как представить в виде строки массив байт длиной в десятки, сотни килобайт или больше, достаточно сложно. Для таких случаев в классе SqlCommand предусмотрены «параметры» - именованные переменные, начинающиеся в sql запросе с символа @, значения которых можно задавать практически любым способом, включая и просто двоичные массивы. Для получения массива байт изображения из файла используется файловый поток FileStream.

В конце ф-ции закрываем окно с помощью DialogResult, значение ОК сообщит основному окну программы, что запись была изменена, и надо обновить грид моделей.

Теперь, когда форма редактирования готова, можно вернуться к основному окну, и добавить код для последней оставшейся кнопки:

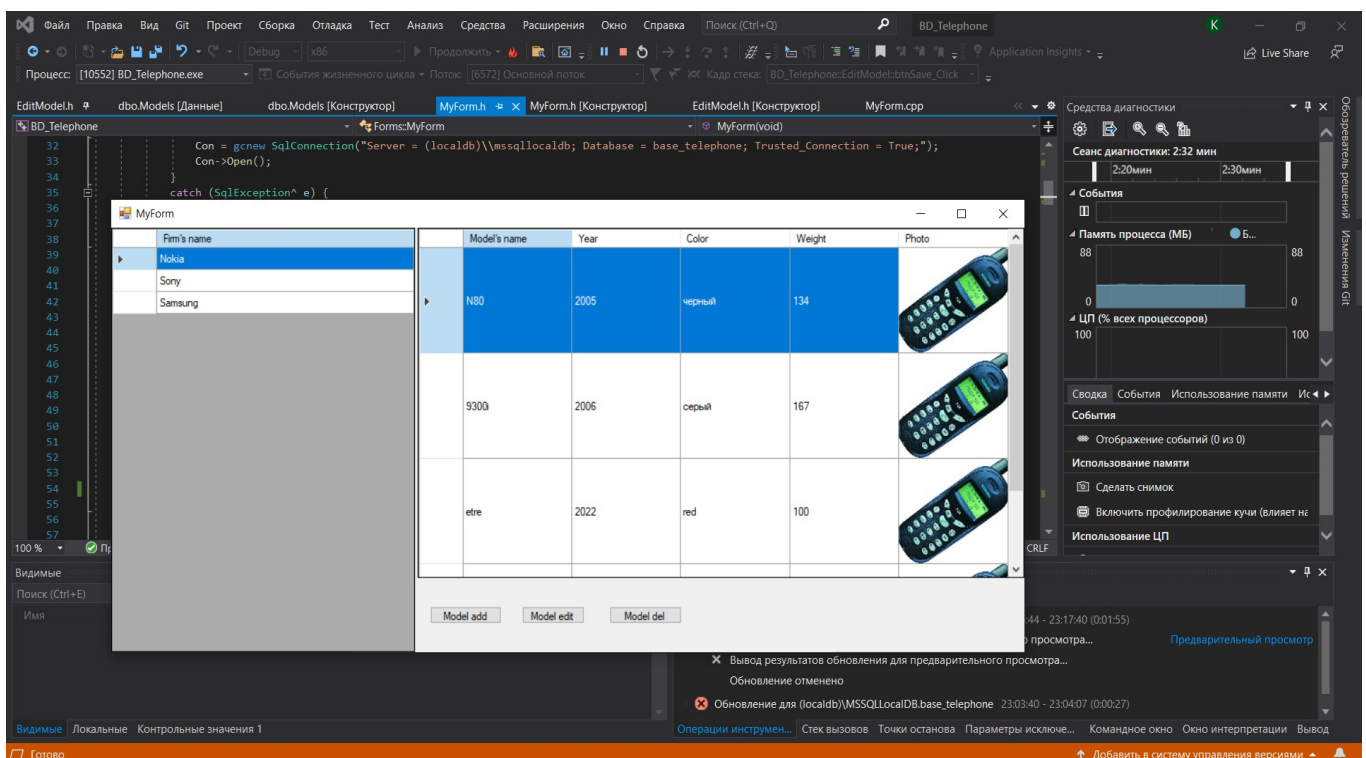
```
private: System::Void btnModelEdit_Click(System::Object^ sender, System::EventArgs^ e) {
    if (dgvModels->RowCount == 0) return;
    EditModel ^form = gcnew EditModel();
    form->con = Con;
    form->dgv = dgvModels;
    if (form->ShowDialog() == ::DialogResult::OK)
        RefreshModels(id_firm);
}
```

Не забудьте про `#include "EditModel.h"` в начале файла.

Запустите программу и проверьте на наличие ошибок. Теперь можно редактировать модели, загружать фото телефонов в базу (несколько изображений для примера прилагается), понажимать на заголовки столбцов в гриде для сортировки (встроенная возможность DataGridView). Можно и дальше дорабатывать программу, расширяя ее функциональность, например:

- Редактирование фирм – можно сделать аналогично моделям
- Поиск – при заполнении гридов запросом select добавить в строку запроса условия where для заданных критериев поиска
- Добавить проверку вводимых данных для исключения ошибок
- Печать отчетов
- Экспорт данных
- И многое другое...

Тем не менее, написанный код уже дает достаточное представление о принципах работы с ADO.NET на языке C++\CLI.

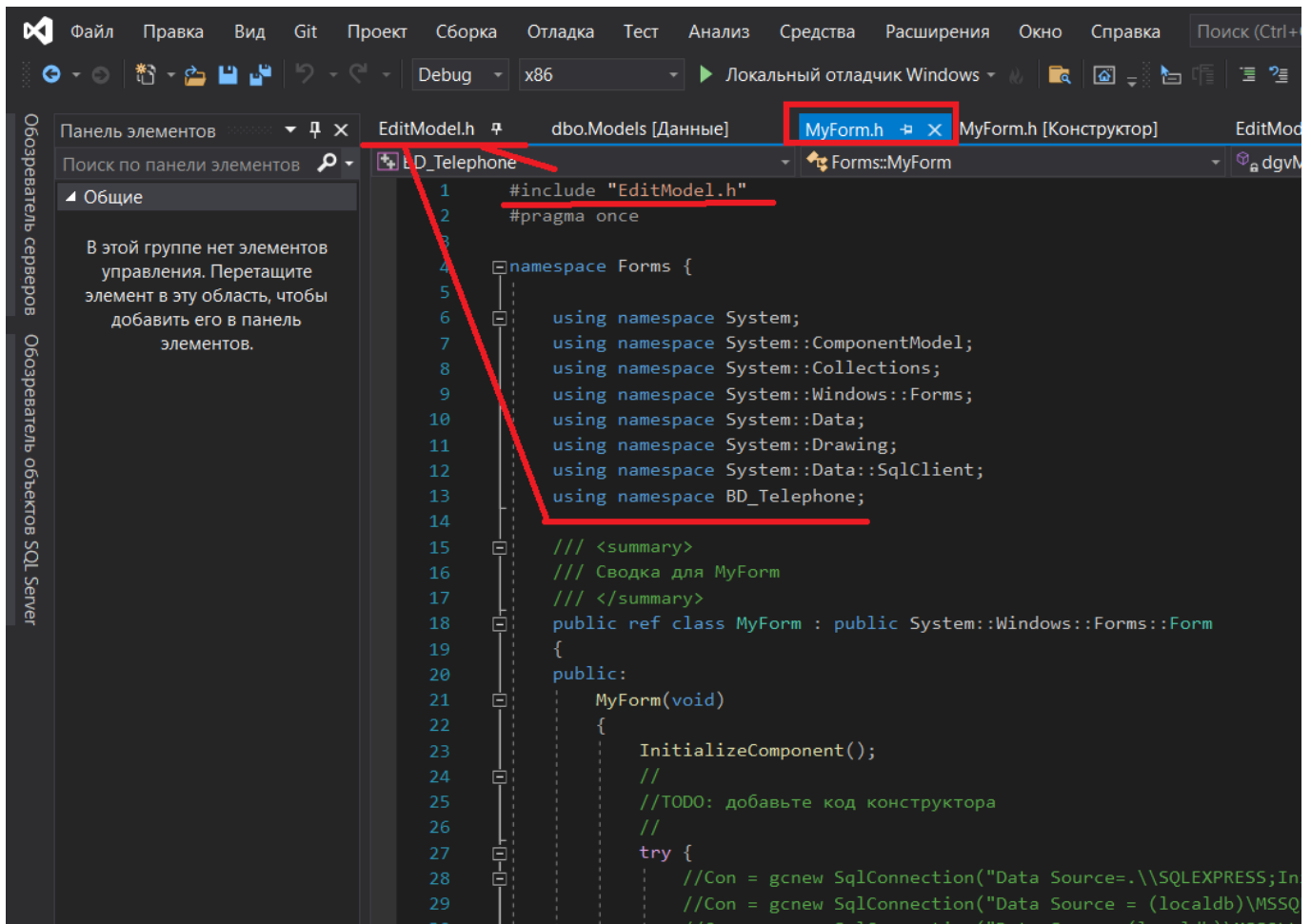


### Возможны проблемы:

1. Имена колонок таблиц в sql-базе не соответствуют именам, указанным в тексте программы для подключения. Будьте внимательны с именами столбцов! (Возникнет исключение «инвалид колонки ...»)
2. забыли `#include "EditModel.h"`
3. после п.2 не хочет видеть `EditModel ^form = gcnew EditModel();` нужно добавить к списку namespace, тот который указан в файле `EditModel.h` в моем случае  
`using namespace BD_Telephone`

4. не нравится текст `if (form->ShowDialog() == ::DialogResult::OK)`, нужно добавить `System::Windows::Forms::`

`if (form2->ShowDialog() == System::Windows::Forms::DialogResult::OK)`



При выполнении домашнего задания может возникнуть необходимость скопировать базу данных MS SQL со своего домашнего компьютера на компьютер в дисплейном классе. Может быть полезно посмотреть видео.

1. Перенос баз данных MS SQL Server с одного ПК на другой.

<https://www.youtube.com/watch?v=HiueGyHPCCg&t=594s>

Дата обращения: 25.12.2022

2. SQL Создание бэкапа - копии базы данных - резервное копирование БД

<https://www.youtube.com/watch?v=afc47dq34QE>

Дата обращения: 25.12.2022

3. Копирование базы данных в MS SQL Server

<https://www.youtube.com/watch?v=CLH7oVnhlWQ>

Дата обращения: 25.12.2022

## Текст файла MyForm.cpp

```
#include "MyForm.h"
//#include "EditModel.h"

using namespace System;
using namespace System::Windows::Forms;
using namespace Forms;

[STAThread]
void main(array<String^>^ arg) {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);

    Forms::MyForm form;    //Forms - name of your project
    Application::Run(% form);
}
```

## Текст файла MyForm.h

```
#include "EditModel.h"
#pragma once

namespace Forms {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;
    using namespace BD_Telephone;

    /// <summary>
    /// Сводка для MyForm
    /// </summary>
    public ref class MyForm : public System::Windows::Forms::Form
    {
    public:
        MyForm(void)
        {
            InitializeComponent();
            //
            //TODO: добавьте код конструктора
            //
            try {
                //Con = gcnew SqlConnection("Data Source=.\SQLEXPRESS;Initial
Catalog=mobile;User ID=sa;Password=111");
                //Con = gcnew SqlConnection("Data Source = (localdb)\MSSQLLocalDB;
Initial Catalog = base_telephone; Integrated Security = True; Pooling = False");
                //Con = gcnew SqlConnection("Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=base_telephone;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFail
over=False");
                //String connectionString = "Server = (localdb)\\mssqllocaldb;
Database = master; Trusted_Connection = True;";
                Con = gcnew SqlConnection("Server = (localdb)\\mssqllocaldb; Database
= base_telephone; Trusted_Connection = True;");
                Con->Open();
            }
            catch (SqlException^ e) {
                MessageBox::Show("Ошибка при подключении к базе данных: " + e-
>Message);

                Application::Exit();
            }
            dtFirms = gcnew DataTable();
            dtModels = gcnew DataTable();
        }
    };
}
```

```

        dgvFirms->DataSource = dtFirms;
        dgvFirms->AutoGenerateColumns = false;
        colFirmID->DataPropertyName = "ID";
        colFirmName->DataPropertyName = "name";

        dgvModels->DataSource = dtModels;
        dgvModels->AutoGenerateColumns = false;
        colModelID->DataPropertyName = "ID";
        colModelName->DataPropertyName = "model";
        colModelYear->DataPropertyName = "year";
        colModelColor->DataPropertyName = "color";
        colModelWeight->DataPropertyName = "weight";
        colModelPhoto->DataPropertyName = "foto";
    }

protected:
    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    ~MyForm()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::SplitContainer^ splitContainer1;
private: System::Windows::Forms::DataGridView^ dgvFirms;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colFirmID;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colFirmName;
private: System::Windows::Forms::DataGridView^ dgvModels;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colModelID;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colModelName;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colModelYear;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colModelColor;
private: System::Windows::Forms::DataGridViewTextBoxColumn^ colModelWeight;
private: System::Windows::Forms::DataGridViewImageColumn^ colModelPhoto;
private: System::Windows::Forms::Panel^ pnlModels;
private: System::Windows::Forms::Button^ btnModelDel;
private: System::Windows::Forms::Button^ btnModelEdit;
private: System::Windows::Forms::Button^ btnModelAdd;
protected:

private:
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Требуемый метод для поддержки конструктора – не изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    void InitializeComponent(void)
    {
        this->splitContainer1 = (gcnew System::Windows::Forms::SplitContainer());
        this->dgvFirms = (gcnew System::Windows::Forms::DataGridView());
        this->colFirmID = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colFirmName = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->dgvModels = (gcnew System::Windows::Forms::DataGridView());
        this->colModelID = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelName = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelYear = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelColor = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelWeight = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelPhoto = (gcnew
System::Windows::Forms::DataGridViewImageColumn());
        this->pnlModels = (gcnew System::Windows::Forms::Panel());
        this->btnModelDel = (gcnew System::Windows::Forms::Button());
        this->btnModelEdit = (gcnew System::Windows::Forms::Button());
        this->btnModelAdd = (gcnew System::Windows::Forms::Button());
    }

```



```

        this->colModelColor = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelWeight = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
        this->colModelPhoto = (gcnew
System::Windows::Forms::DataGridViewImageColumn());
        this->pnlModels = (gcnew System::Windows::Forms::Panel());
        this->btnModelDel = (gcnew System::Windows::Forms::Button());
        this->btnModelEdit = (gcnew System::Windows::Forms::Button());
        this->btnModelAdd = (gcnew System::Windows::Forms::Button());
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>splitContainer1))->BeginInit();
        this->splitContainer1->Panel1->SuspendLayout();
        this->splitContainer1->Panel2->SuspendLayout();
        this->splitContainer1->SuspendLayout();
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dgvFirms))->BeginInit();
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dgvModels))->BeginInit();
        this->pnlModels->SuspendLayout();
        this->SuspendLayout();
        //
        // splitContainer1
        //
        this->splitContainer1->Dock = System::Windows::Forms::DockStyle::Fill;
        this->splitContainer1->Location = System::Drawing::Point(0, 0);
        this->splitContainer1->Name = L"splitContainer1";
        //
        // splitContainer1.Panel1
        //
        this->splitContainer1->Panel1->Controls->Add(this->dgvFirms);
        //
        // splitContainer1.Panel2
        //
        this->splitContainer1->Panel2->Controls->Add(this->dgvModels);
        this->splitContainer1->Panel2->Controls->Add(this->pnlModels);
        this->splitContainer1->Size = System::Drawing::Size(646, 405);
        this->splitContainer1->SplitterDistance = 215;
        this->splitContainer1->TabIndex = 0;
        //
        // dgvFirms
        //
        this->dgvFirms->AllowUserToAddRows = false;
        this->dgvFirms->AllowUserToDeleteRows = false;
        this->dgvFirms->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;
        this->dgvFirms->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
        this->dgvFirms->Columns->AddRange(gcnew cli::array<
System::Windows::Forms::DataGridViewColumn^ >(2) { this->colFirmID, this->colFirmName });
        this->dgvFirms->Dock = System::Windows::Forms::DockStyle::Fill;
        this->dgvFirms->Location = System::Drawing::Point(0, 0);
        this->dgvFirms->Name = L"dgvFirms";
        this->dgvFirms->ReadOnly = true;
        this->dgvFirms->RowHeadersWidth = 51;
        this->dgvFirms->RowTemplate->Height = 24;
        this->dgvFirms->SelectionMode =
System::Windows::Forms::DataGridViewSelectionMode::FullRowSelect;
        this->dgvFirms->Size = System::Drawing::Size(215, 405);
        this->dgvFirms->TabIndex = 0;
        this->dgvFirms->SelectionChanged += gcnew System::EventHandler(this,
&MyForm::dgvFirms_SelectionChanged);
        //
        // colFirmID
        //
        this->colFirmID->HeaderText = L"ID Firm";
        this->colFirmID->MinimumWidth = 6;
        this->colFirmID->Name = L"colFirmID";
        this->colFirmID->ReadOnly = true;
        this->colFirmID->Visible = false;

```

```

//
// colFirmName
//
this->colFirmName->HeaderText = L"Firm\'s name";
this->colFirmName->MinimumWidth = 6;
this->colFirmName->Name = L"colFirmName";
this->colFirmName->ReadOnly = true;
//
// dgvModels
//
this->dgvModels->AllowUserToAddRows = false;
this->dgvModels->AllowUserToDeleteRows = false;
this->dgvModels->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;
this->dgvModels->AutoSizeRowsMode =
System::Windows::Forms::DataGridViewAutoSizeRowsMode::AllCellsExceptHeaders;
this->dgvModels->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
this->dgvModels->Columns->AddRange(gcnew cli::array<
System::Windows::Forms::DataGridViewColumn^ >(6) {
    this->colModelID,
        this->colModelName, this->colModelYear, this->colModelColor,
this->colModelWeight, this->colModelPhoto
});
this->dgvModels->Dock = System::Windows::Forms::DockStyle::Fill;
this->dgvModels->Location = System::Drawing::Point(0, 0);
this->dgvModels->Name = L"dgvModels";
this->dgvModels->ReadOnly = true;
this->dgvModels->RowHeadersWidth = 51;
this->dgvModels->RowTemplate->Height = 24;
this->dgvModels->SelectionMode =
System::Windows::Forms::DataGridViewSelectionMode::FullRowSelect;
this->dgvModels->Size = System::Drawing::Size(427, 305);
this->dgvModels->TabIndex = 1;
this->dgvModels->CellContentClick += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this, &MyForm::dgvModels_CellContentClick);
//
// colModelID
//
this->colModelID->HeaderText = L"Model ID";
this->colModelID->MinimumWidth = 6;
this->colModelID->Name = L"colModelID";
this->colModelID->ReadOnly = true;
this->colModelID->Visible = false;
//
// colModelName
//
this->colModelName->HeaderText = L"Model\'s name";
this->colModelName->MinimumWidth = 6;
this->colModelName->Name = L"colModelName";
this->colModelName->ReadOnly = true;
//
// colModelYear
//
this->colModelYear->HeaderText = L"Year";
this->colModelYear->MinimumWidth = 6;
this->colModelYear->Name = L"colModelYear";
this->colModelYear->ReadOnly = true;
//
// colModelColor
//
this->colModelColor->HeaderText = L"Color";
this->colModelColor->MinimumWidth = 6;
this->colModelColor->Name = L"colModelColor";
this->colModelColor->ReadOnly = true;
//
// colModelWeight
//
this->colModelWeight->HeaderText = L"Weight";
this->colModelWeight->MinimumWidth = 6;

```

```

this->colModelWeight->Name = L"colModelWeight";
this->colModelWeight->ReadOnly = true;
//
// colModelPhoto
//
this->colModelPhoto->HeaderText = L"Photo";
this->colModelPhoto->MinimumWidth = 6;
this->colModelPhoto->Name = L"colModelPhoto";
this->colModelPhoto->ReadOnly = true;
//
// pnlModels
//
this->pnlModels->Controls->Add(this->btnModelDel);
this->pnlModels->Controls->Add(this->btnModelEdit);
this->pnlModels->Controls->Add(this->btnModelAdd);
this->pnlModels->Dock = System::Windows::Forms::DockStyle::Bottom;
this->pnlModels->Location = System::Drawing::Point(0, 305);
this->pnlModels->Name = L"pnlModels";
this->pnlModels->Size = System::Drawing::Size(427, 100);
this->pnlModels->TabIndex = 0;
//
// btnModelDel
//
this->btnModelDel->Location = System::Drawing::Point(289, 38);
this->btnModelDel->Name = L"btnModelDel";
this->btnModelDel->Size = System::Drawing::Size(109, 23);
this->btnModelDel->TabIndex = 2;
this->btnModelDel->Text = L"Model del";
this->btnModelDel->UseVisualStyleBackColor = true;
this->btnModelDel->Click += gcnew System::EventHandler(this,
&MyForm::btnModelDel_Click);
//
// btnModelEdit
//
this->btnModelEdit->Location = System::Drawing::Point(157, 38);
this->btnModelEdit->Name = L"btnModelEdit";
this->btnModelEdit->Size = System::Drawing::Size(95, 23);
this->btnModelEdit->TabIndex = 1;
this->btnModelEdit->Text = L"Model edit\r\n";
this->btnModelEdit->UseVisualStyleBackColor = true;
this->btnModelEdit->Click += gcnew System::EventHandler(this,
&MyForm::btnModelEdit_Click);
//
// btnModelAdd
//
this->btnModelAdd->Location = System::Drawing::Point(19, 38);
this->btnModelAdd->Name = L"btnModelAdd";
this->btnModelAdd->Size = System::Drawing::Size(108, 23);
this->btnModelAdd->TabIndex = 0;
this->btnModelAdd->Text = L"Model add";
this->btnModelAdd->UseVisualStyleBackColor = true;
this->btnModelAdd->Click += gcnew System::EventHandler(this,
&MyForm::btnModelAdd_Click);
//
// MyForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(646, 405);
this->Controls->Add(this->splitContainer1);
this->Name = L"MyForm";
this->Text = L"MyForm";
this->Shown += gcnew System::EventHandler(this, &MyForm::MyForm_Shown);
this->splitContainer1->Panel1->ResumeLayout(false);
this->splitContainer1->Panel2->ResumeLayout(false);
(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>splitContainer1))->EndInit();
this->splitContainer1->ResumeLayout(false);
(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dgvFirms))->EndInit();

```

```

        (cli::safe_cast<System::ComponentModel::ISupportInitialize>)(this-
>dgvModels))->EndInit();
        this->pn1Models->ResumeLayout(false);
        this->ResumeLayout(false);

    }
#pragma endregion

    public:
        SqlConnection^ Con;
        DataTable^ dtFirms;
        DataTable^ dtModels;
        bool UpdatingFirms;
        int id_firm;

        void RefreshFirms() {
            UpdatingFirms = true;
            SqlCommand^ c = gcnew SqlCommand("select * from Firms", Con);
            SqlDataReader^ r = c->ExecuteReader();
            dtFirms->Rows->Clear();
            dtFirms->Load(r);
            UpdatingFirms = false;
        }

        void RefreshModels(int id_firm) {
            if (UpdatingFirms || id_firm == -1) return;
            try {
                SqlCommand^ c = gcnew SqlCommand("select * from Models where ID_Firm="
+ id_firm, Con);

                SqlDataReader^ r = c->ExecuteReader();
                dtModels->Rows->Clear();
                dtModels->Load(r);
            }
            catch (Exception^ e) {
            }
        }

    private: System::Void dgvModels_CellContentClick(System::Object^ sender,
System::Windows::Forms::DataGridViewCellEventArgs^ e) {
    }
    private: System::Void dgvFirms_SelectionChanged(System::Object^ sender, System::EventArgs^
e) {
        try {
            id_firm = Convert::ToInt32(dgvFirms->CurrentRow->Cells[0]->Value);
        }
        catch (Exception^ e) {
            id_firm = -1;
        }
        RefreshModels(id_firm);
    }

    private: System::Void MyForm_Shown(System::Object^ sender, System::EventArgs^ e) {
        RefreshFirms();
        dgvFirms_SelectionChanged(nullptr, nullptr);
    }
    private: System::Void btnModelAdd_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ sql = String::Format("insert into Models (ID_Firm, model) values ({0},
N'{1}']", id_firm, "Новая модель");
        SqlCommand^ c = gcnew SqlCommand(sql, Con);
        c->ExecuteNonQuery();
        RefreshModels(id_firm);
    }
    private: System::Void btnModelDel_Click(System::Object^ sender, System::EventArgs^ e) {
        if (dgvModels->RowCount == 0) return;
        int id = Convert::ToInt32(dgvModels->CurrentRow->Cells[0]->Value);
        String^ sql = "delete from Models where id=" + id;
        SqlCommand^ c = gcnew SqlCommand(sql, Con);
        c->ExecuteNonQuery();
        RefreshModels(id_firm);
    }
}

```

```

private: System::Void btnModelEdit_Click(System::Object^ sender, System::EventArgs^ e) {
    if (dgvModels->RowCount == 0) return;
    EditModel^ form2 = gcnew EditModel();
    form2->con = Con;
    form2->dgv = dgvModels;
    if (form2->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        RefreshModels(id_firm);
}
};
}

```

## Текст файла EditModel.cpp

```
#include "EditModel.h"
```

## Текст файла EditModel.h

```
#pragma once
```

```

namespace BD_Telephone {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;

    /// <summary>
    /// Сводка для EditModel
    /// </summary>
    public ref class EditModel : public System::Windows::Forms::Form
    {
    public:
        EditModel(void)
        {
            InitializeComponent();
            //
            //TODO: добавьте код конструктора
            //
        }

    protected:
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~EditModel()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^ label1;
    protected:
    private: System::Windows::Forms::TextBox^ tbModel;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::TextBox^ tbYear;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::TextBox^ tbColor;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::TextBox^ tbWeight;
    private: System::Windows::Forms::GroupBox^ groupBox1;
    private: System::Windows::Forms::PictureBox^ pbPhoto;
    private: System::Windows::Forms::Button^ btnLoadPhoto;
    private: System::Windows::Forms::Button^ btnSave;

    private: System::Windows::Forms::Button^ bntCancel;
}

```

```

private:
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Требуемый метод для поддержки конструктора — не изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    void InitializeComponent(void)
    {
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->tbModel = (gcnew System::Windows::Forms::TextBox());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->tbYear = (gcnew System::Windows::Forms::TextBox());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->tbColor = (gcnew System::Windows::Forms::TextBox());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->tbWeight = (gcnew System::Windows::Forms::TextBox());
        this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
        this->pbPhoto = (gcnew System::Windows::Forms::PictureBox());
        this->btnLoadPhoto = (gcnew System::Windows::Forms::Button());
        this->btnSave = (gcnew System::Windows::Forms::Button());
        this->btnCancel = (gcnew System::Windows::Forms::Button());
        this->groupBox1->SuspendLayout();
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pbPhoto))-
>BeginInit();

        this->SuspendLayout();
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Location = System::Drawing::Point(26, 30);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(58, 17);
        this->label1->TabIndex = 0;
        this->label1->Text = L"Модель";
        //
        // tbModel
        //
        this->tbModel->Location = System::Drawing::Point(89, 27);
        this->tbModel->Name = L"tbModel";
        this->tbModel->Size = System::Drawing::Size(100, 22);
        this->tbModel->TabIndex = 1;
        //
        // label2
        //
        this->label2->AutoSize = true;
        this->label2->Location = System::Drawing::Point(26, 74);
        this->label2->Name = L"label2";
        this->label2->Size = System::Drawing::Size(32, 17);
        this->label2->TabIndex = 2;
        this->label2->Text = L"Год";
        //
        // tbYear
        //
        this->tbYear->Location = System::Drawing::Point(89, 71);
        this->tbYear->Name = L"tbYear";
        this->tbYear->Size = System::Drawing::Size(100, 22);
        this->tbYear->TabIndex = 3;
        //
        // label3
        //
        this->label3->AutoSize = true;

```

```

this->label3->Location = System::Drawing::Point(26, 120);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(41, 17);
this->label3->TabIndex = 4;
this->label3->Text = L"Цвет";
//
// tbColor
//
this->tbColor->Location = System::Drawing::Point(89, 117);
this->tbColor->Name = L"tbColor";
this->tbColor->Size = System::Drawing::Size(100, 22);
this->tbColor->TabIndex = 5;
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(26, 168);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(32, 17);
this->label4->TabIndex = 6;
this->label4->Text = L"Бес";
//
// tbWeight
//
this->tbWeight->Location = System::Drawing::Point(89, 165);
this->tbWeight->Name = L"tbWeight";
this->tbWeight->Size = System::Drawing::Size(100, 22);
this->tbWeight->TabIndex = 7;
//
// groupBox1
//
this->groupBox1->Controls->Add(this->pbPhoto);
this->groupBox1->Controls->Add(this->btnLoadPhoto);
this->groupBox1->Location = System::Drawing::Point(29, 219);
this->groupBox1->Name = L"groupBox1";
this->groupBox1->Size = System::Drawing::Size(230, 253);
this->groupBox1->TabIndex = 8;
this->groupBox1->TabStop = false;
this->groupBox1->Text = L"Фото";
//
// pbPhoto
//
this->pbPhoto->Location = System::Drawing::Point(21, 69);
this->pbPhoto->Name = L"pbPhoto";
this->pbPhoto->Size = System::Drawing::Size(186, 160);
this->pbPhoto->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::CenterImage;
this->pbPhoto->TabIndex = 1;
this->pbPhoto->TabStop = false;
//
// btnLoadPhoto
//
this->btnLoadPhoto->Location = System::Drawing::Point(21, 21);
this->btnLoadPhoto->Name = L"btnLoadPhoto";
this->btnLoadPhoto->Size = System::Drawing::Size(92, 26);
this->btnLoadPhoto->TabIndex = 0;
this->btnLoadPhoto->Text = L"Загрузить";
this->btnLoadPhoto->UseVisualStyleBackColor = true;
this->btnLoadPhoto->Click += gcnew System::EventHandler(this,
&EditModel::btnLoadPhoto_Click);
//
// btnSave
//
this->btnSave->Location = System::Drawing::Point(29, 479);
this->btnSave->Name = L"btnSave";
this->btnSave->Size = System::Drawing::Size(91, 37);
this->btnSave->TabIndex = 9;
this->btnSave->Text = L"Сохранить";
this->btnSave->UseVisualStyleBackColor = true;

```

```

        this->btnSave->Click += gcnew System::EventHandler(this,
&EditModel::btnSave_Click);
        //
        // bntCancel
        //
        this->bntCancel->DialogResult = System::Windows::Forms::DialogResult::Cancel;
        this->bntCancel->Location = System::Drawing::Point(168, 479);
        this->bntCancel->Name = L"bntCancel";
        this->bntCancel->Size = System::Drawing::Size(91, 37);
        this->bntCancel->TabIndex = 10;
        this->bntCancel->Text = L"Отменить";
        this->bntCancel->UseVisualStyleBackColor = true;
        //
        // EditModel
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->CancelButton = this->bntCancel;
        this->ClientSize = System::Drawing::Size(282, 528);
        this->Controls->Add(this->bntCancel);
        this->Controls->Add(this->btnSave);
        this->Controls->Add(this->groupBox1);
        this->Controls->Add(this->tbWeight);
        this->Controls->Add(this->label4);
        this->Controls->Add(this->tbColor);
        this->Controls->Add(this->label3);
        this->Controls->Add(this->tbYear);
        this->Controls->Add(this->label2);
        this->Controls->Add(this->tbModel);
        this->Controls->Add(this->label1);
        this->Name = L"EditModel";
        this->Text = L"EditModel";
        this->Shown += gcnew System::EventHandler(this, &EditModel::EditModel_Shown);
        this->groupBox1->ResumeLayout(false);
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pbPhoto))-
>EndInit();

        this->ResumeLayout(false);
        this->PerformLayout();

    }
#pragma endregion

    public:
        SqlConnection^ con;
        DataGridView^ dgv;
        int id;

    private: System::Void EditModel_Shown(System::Object^ sender, System::EventArgs^ e) {

        id = Convert::ToInt32(dgv->CurrentRow->Cells[0]->Value);
        tbModel->Text = Convert::ToString(dgv->CurrentRow->Cells[1]->Value);
        tbYear->Text = Convert::ToString(dgv->CurrentRow->Cells[2]->Value);
        tbColor->Text = Convert::ToString(dgv->CurrentRow->Cells[3]->Value);
        tbWeight->Text = Convert::ToString(dgv->CurrentRow->Cells[4]->Value);

        using namespace System::Drawing;
        using namespace System::IO;
        try {
            array<Byte>^ b = (array<Byte>^)dgv->CurrentRow->Cells[5]->Value;
            if (b) {
                MemoryStream ^ ms = gcnew MemoryStream(b->Length);
                ms->Write(b, 0, b->Length);
                pbPhoto->Image = Image::FromStream(ms);
                pbPhoto->Refresh();
            }
        }
        catch (Exception^ e) {

        }
        catch (NullReferenceException^ e)

```



```

    {
    }
    catch (...)
    {
    }
}

private: System::Void btnLoadPhoto_Click(System::Object^ sender, System::EventArgs^ e) {

    OpenFileDialog^ d = gcnew OpenFileDialog();
    d->Filter = "JPEG файлы (*.jpg)|*.jpg|Bitmap файлы (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
    if (d->ShowDialog() == Windows::Forms::DialogResult::OK)
        pbPhoto->ImageLocation = d->FileName;
}

private: System::Void btnSave_Click(System::Object^ sender, System::EventArgs^ e) {
    String^ sql = "update Models set model=N'{0}', year='{1}', "
        "color=N'{2}', weight='{3}' where ID=" + id;
    sql = String::Format(sql, tbModel->Text, tbYear->Text,
        tbColor->Text, tbWeight->Text);
    SqlCommand^ c = gcnew SqlCommand(sql, con);
    c->ExecuteNonQuery();

    if (pbPhoto->ImageLocation) {
        using namespace System::IO;
        FileStream^ fs = File::OpenRead(pbPhoto->ImageLocation);
        array<Byte>^ b = gcnew array<Byte>(fs->Length);
        fs->Read(b, 0, fs->Length);
        fs->Close();
        sql = "update Models set foto = @foto where ID=" + id;
        //sql = "update Models set photo = N'{0}' where ID=" + id;

        //sql = String::Format(sql, b);
        SqlCommand^ c2 = gcnew SqlCommand(sql, con);
        c2->Parameters->Add(gcnew SqlParameter("foto", SqlDbType::Image));
        c2->Parameters["foto"]->Value = b;
        c2->ExecuteNonQuery();
    }
    DialogResult = System::Windows::Forms::DialogResult::OK;
}
};
}

```