**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Тема:** Создания информационной базы данных корпоративной сети

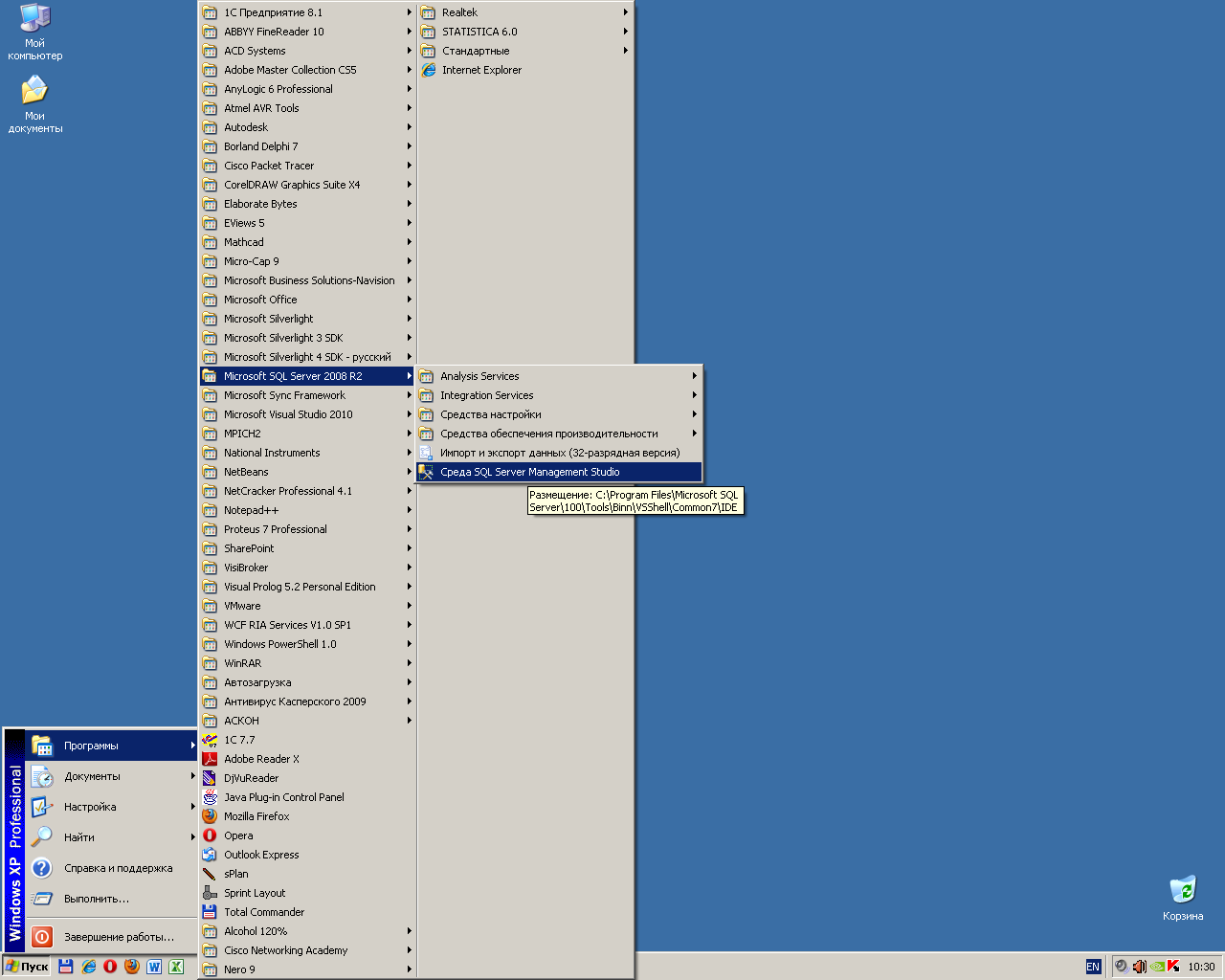
**Цель:** Получить навыки создания общей реляционной базы данных для использщования в сетевом пространстве.

ХОД РАБОТЫ

* Создайте структуру базы данных согласно вашему варианту и заполните каждую таблицу значениями (более 5 записей в каждой)

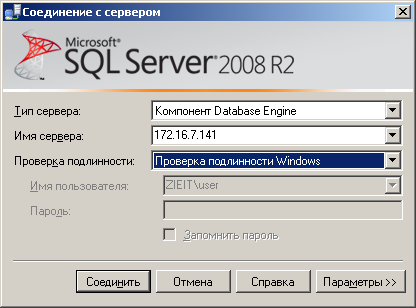
Для подключения к Серверу баз данных необходимо:

Запустить программу Среда SQL Server Management Studio (Пуск – Программы - Microsoft SQL Server 2008 R2)



После запуска ввести адресс сервера 172.16.7.141

Указать что будет произведена “Проверка подлинности Windows”   
(ДЛЯ ЭТОГО ВЫ ДОЛЖНЫ БЫТЬ СТУДЕНТАМИ ГРУППЫ КСИС ИЛИ ОКСИС, ЛИБО БЫТЬ ПОД УЧЕТНОЙ ЗАПИСЬЮ USER)



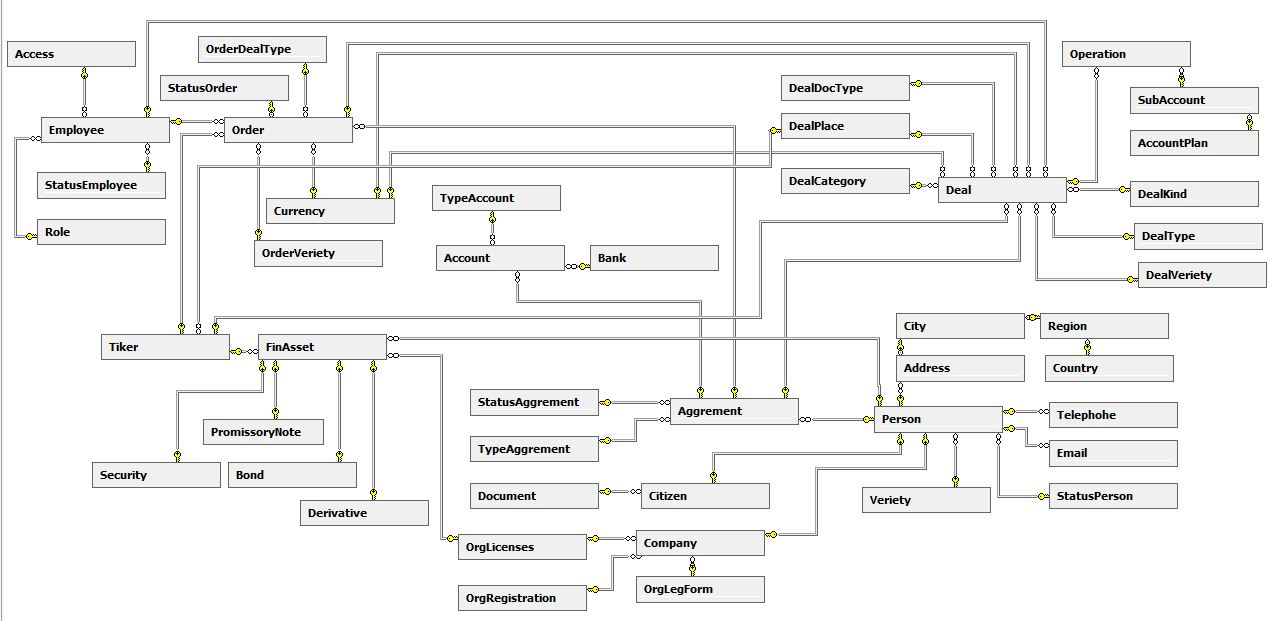
После входа в среду создайте личную базу данных (Не используйтеGlobalBase).

Как имя базы рекомендуется использовать Логин входа в систему (Ivanov\_ii)

* После создания личной базы данных создайте в ней схему таблиц И ИХ ВЗАИМОСВЯЗЕЙ согласно варианту задания.

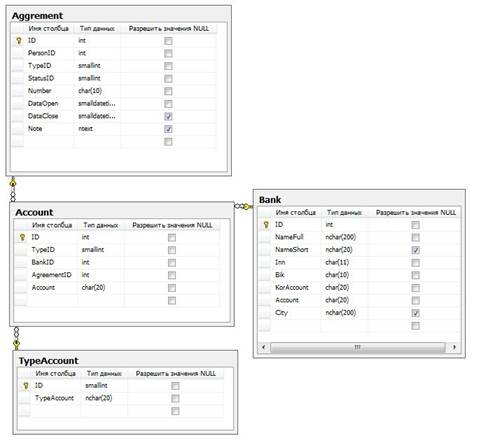
Для создания базы данных о пределения связей см. лекции по СУБД

ОБЩИЙ ВИД СИСТЕМЫ



**Вариант 1. Информационная подсистема ведения счетов клиентов**

Назначение подсистемы поддержание в актуальном состоянии инвестиционных счетов клиентов.



**Рис. l.2.**  Структура базы данных подсистемы ведения счетов клиентов

Назначение атрибутов таблицы Счет – Account:

* ID – суррогатный ключ;
* TypeID – внешний ключ для связи с таблицей Type;
* BankID – внешний ключ для связи с таблицей Bank;
* AgreementID – внешний ключ для связи с таблицей Agreement;
* Account – номер инвестиционного счета.

Назначение атрибутов таблицы Тип счета – TypeAccount:

* ID – суррогатный ключ;
* TypeAccount – тип счета.

Назначение атрибутов таблицы Банк – Bank:

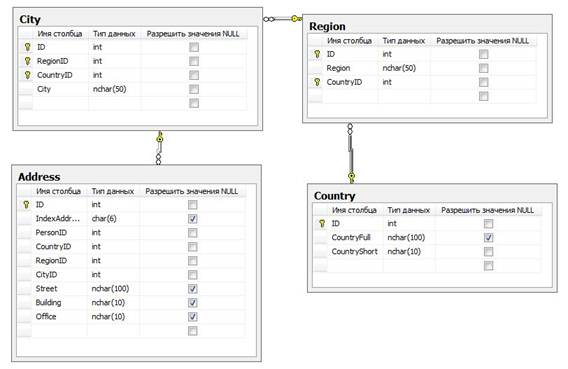
* ID – суррогатный ключ;
* NameFull – полное наименование банка;
* NameShort – краткое наименование банка;
* Inn – ИНН банка;
* Bik – БИК банка;
* CorAccount – номер корсчета;
* Account – номер счета;
* City – город.

Назначение атрибутов таблицы Договор –Agreement:

* ID – суррогатный ключ;
* PersonID – внешний ключ для связи с таблицей Person;
* TypeID – внешний ключ для связи с таблицей Type;
* StatusID – внешний ключ для связи с таблицей Status;
* Number – номер договора;
* DataOpen- - дата заключения договора;
* DataClouse – дата закрытия договора;
* Note – пояснения.

**Вариант 2. Информационная подсистема ведения адресов клиентов**

Назначение подсистемы поддержание в актуальном состоянии адресов клиентов.



**Рис. l.3.**  Структура базы данных подсистемы ведения адресов клиентов

Назначение атрибутов таблицы Адрес – Address:

* ID – суррогатный ключ;
* Index Address – адресный индекс;
* PersonID – внешний ключ для связи с таблицей Person;
* CountryID – внешний ключ для связи с таблицей Country;
* RegionID – внешний ключ для связи с таблицей Region;
* CityID – внешний ключ для связи с таблицей City;
* Street – наименование улицы;
* Bulding – номер строения, дома;
* Office – номер офиса.

Назначение атрибутов таблицы Город –City:

* ID – суррогатный ключ;
* RegionID – внешний ключ для связи с таблицей Region;
* CountryID – внешний ключ для связи с таблицей Country;
* City – город.

Назначение атрибутов таблицы Регион –Region:

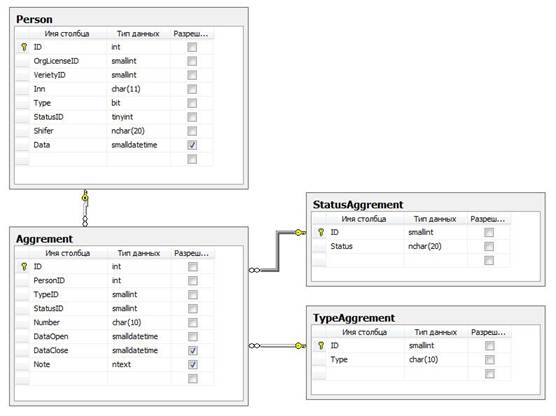
* ID – суррогатный ключ;
* CountryID – внешний ключ для связи с таблицей Country;
* Region – регион.

Назначение атрибутов таблицы Страна –Country:

* ID – суррогатный ключ;
* CountryFull – полное наименование страны;
* CountryShort – краткое наименование страны.

**Вариант 3. Информационная подсистема ведения договоров клиентов**

Назначение подсистемы поддержание в актуальном состоянии договоров клиентов.



**Рис. l.4.**  Структура базы данных подсистемы ведения договоров клиентов

Назначение атрибутов таблицы Договор –Agreement:

* ID – суррогатный ключ;
* PersonID – внешний ключ для связи с таблицей Person;
* TypeID – внешний ключ для связи с таблицей Type;
* StatusID – внешний ключ для связи с таблицей Status;
* Number – номер договора;
* DataOpen- - дата заключения договора;
* DataClouse – дата закрытия договора;
* Note – пояснения.

Назначение атрибутов таблицы Статус договор –StatusAggrement:

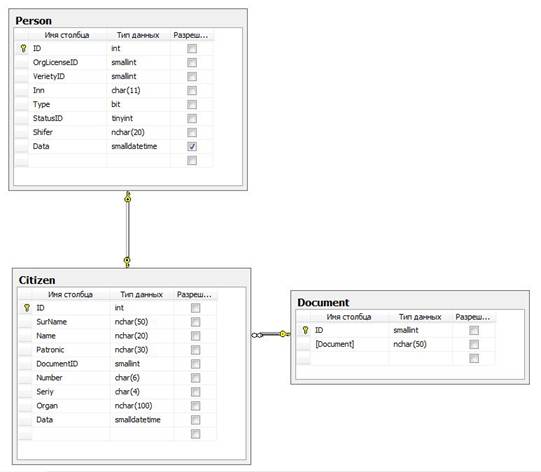
* ID – суррогатный ключ;
* Status – статус договора.

Назначение атрибутов таблицы Клиент –Person:

* ID – суррогатный ключ;
* OrgLicenseID – внешний ключ для связи с таблицей OrgLicense;
* VerietyID – внешний ключ для связи с таблицей Veriety;
* StatusID – внешний ключ для связи с таблицей Status;
* Inn – ИНН клиента;
* Type – тип клиента;
* Shifer – шифр клиента;
* Data – дата регистрации клиента.

**Вариант 4. Информационная подсистема ведения клиентов – физических лиц**

Назначение подсистемы поддержание в актуальном состоянии информации по клиентам – физическим лицам.



**Рис. l.5.**  Структура базы данных подсистемы ведения клиентов – физических лиц

Назначение атрибутов таблицы Клиент –Person:

* ID – суррогатный ключ;
* OrgLicenseID – внешний ключ для связи с таблицей OrgLicense;
* VerietyID – внешний ключ для связи с таблицей Veriety;
* StatusID – внешний ключ для связи с таблицей Status;
* Inn – ИНН клиента;
* Type – тип клиента;
* Shifer – шифр клиента;
* Data – дата регистрации клиента.

Назначение атрибутов таблицы Физическое лицо –Citizen:

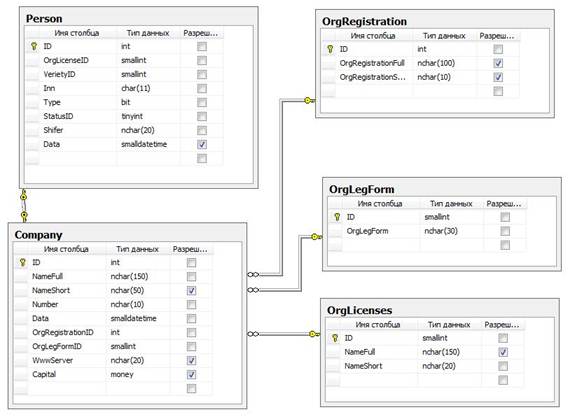
* ID – суррогатный ключ;
* DocumentID – внешний ключ для связи с таблицей Document;
* SurName – фамилия клиента;
* Name – имя клиента;
* Patronic – отчество клиента;
* Number – номер документа, удостоверяющего личность;
* Seriy – серия документа, удостоверяющего личность;
* Organ – орган, выдавший документ, удостоверяющий личность;
* Data – дата выдачи документа, удостоверяющий личность.

Назначение атрибутов таблицы Документ – Document:

* ID – суррогатный ключ;
* Document – наименование документа, удостоверяющего личность.

**Вариант 5. Информационная подсистема ведения клиентов – юридических лиц**

Назначение подсистемы поддержание в актуальном состоянии информации по клиентам – юридическим лицам.



**Рис. l.6.**  Структура базы данных подсистемы ведения клиентов – юридических лиц

Назначение атрибутов таблицы Клиент –Person:

* ID – суррогатный ключ;
* OrgLicenseID – внешний ключ для связи с таблицей OrgLicense;
* VerietyID – внешний ключ для связи с таблицей Veriety;
* StatusID – внешний ключ для связи с таблицей Status;
* Inn – ИНН клиента;
* Type – тип клиента;
* Shifer – шифр клиента;
* Data – дата регистрации клиента.

Назначение атрибутов таблицы Физическое лицо –Company:

* ID – суррогатный ключ;
* OgrRegistrationID – внешний ключ для связи с таблицей OgrRegistration;
* OrgLegFormID – внешний ключ для связи с таблицей OrgLegForm;
* OrgLecensesID – внешний ключ для связи с таблицей OrgLecenses;
* NameFull – полное наименование организации;
* NameShort – краткое наименование организации;
* Number – номер регистрации юридического лица;
* Data – дата регистрации юридического лица;
* WwwServer – WWW сервер организации;
* Capital – уставный капитал организации.

Назначение атрибутов таблицы Организационно-правовая форма –OrgLegForm:

* ID – суррогатный ключ;
* OrgLegForm – организационно-правовая форма организации.

Назначение атрибутов таблицы Орган лицензирования – OrgLecenses

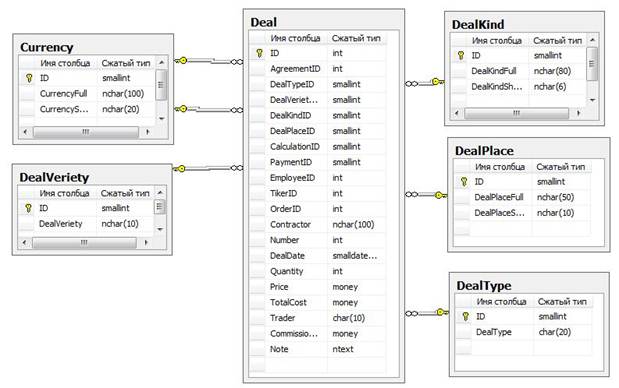
* ID – суррогатный ключ;
* NameFull – полное наименование органа лицензирования;
* NameShort – краткое наименование органа лицензирования;

Назначение атрибутов таблицы Орган регистрации – OrgRegistration

* ID – суррогатный ключ;
* NameFull – полное наименование органа регистрации;
* NameShort – краткое наименование органа регистрации;

**Вариант 6. Информационная подсистема учета сделок с ценными бумагами**

Назначение подсистемы поддержание в актуальном состоянии информации по сделкам с ценными бумагами клиентов.



**Рис. l.7.**  Структура базы данных подсистемы учета сделок с ценными бумагами

Назначение атрибутов таблицы Сделка – Deal:

* ID – суррогатный ключ;
* AgreementID – внешний ключ для связи с таблицей Agreement;
* DealTypeID – внешний ключ для связи с таблицей DealType;
* DealKindID – внешний ключ для связи с таблицей DealKind;
* DealPlaceID – внешний ключ для связи с таблицей DealPlace;
* CalculationID – внешний ключ для связи с таблицей Currency;
* PaymentID – внешний ключ для связи с таблицей Currency;
* EmployeeID – внешний ключ для связи с таблицей Employee;
* TikerID – внешний ключ для связи с таблицей Tiker;
* OrderID – внешний ключ для связи с таблицей Order;
* Contractor – контрагент по сделке;
* Number – номер сделки;
* DealDate – дата и время заключения сделки;
* Quantity – количество ценных бумаг по сделке;
* Price – цена по сделке;
* TotalCost – общая сумма сделки;
* Trader – код трейдера;
* CommissionExchange – комиссия торговой площадки;
* Note – примечание.

Назначение атрибутов таблицы Вид сделки –DealVerity:

* ID – суррогатный ключ;
* DealVerity – вид сделки.

Назначение атрибутов таблицы Тип сделки –DealType:

* ID – суррогатный ключ;
* DealType – тип сделки.

Назначение атрибутов таблицы Вид взаиморасчетов по сделке –DealKind:

* ID – суррогатный ключ;
* DealKindFull – полное наименование вида взаиморасчетов по сделке;
* DealKindShort – краткое наименование вида взаиморасчетов по сделке.

Назначение атрибутов таблицы Место проведения сделки –DealPlace:

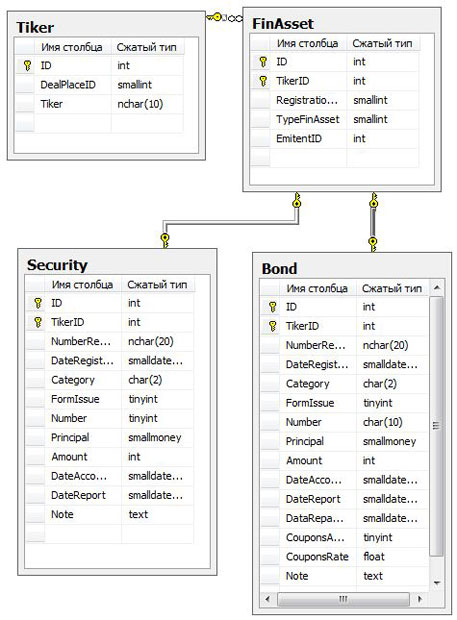
* ID – суррогатный ключ;
* DealPlaceFull – полное наименование места проведения сделки;
* DealPlaceShort – краткое наименование места проведения сделки.

Назначение атрибутов таблицы Валюта – Currency:

* ID – суррогатный ключ;
* CurrencyFull – полное наименование валюты;
* CurrencyShort – краткое наименование валюты.

**Вариант 7. Информационная подсистема учета ценных бумаг**

Назначение подсистемы поддержание в актуальном состоянии информации по торгуемым ценным бумагам.



**Рис. l.8.**  Структура базы данных подсистемы учета ценных бумаг

Назначение атрибутов таблицы Финансовый инструмент –FinAsset:

* ID – суррогатный ключ;
* TikerID – внешний ключ для связи с таблицей Tiker;
* RegistrationID – внешний ключ для связи с таблицей Registration;
* EmitentID – внешний ключ для связи с таблицей Emitent;
* TypeFinAsset - тип ценной бумаги.

Назначение атрибутов таблицы Тикер ценной бумаги–Tiker:

* ID – суррогатный ключ;
* DealPlaceID – внешний ключ для связи с таблицей DealPlace;
* TikerI – тикер ценной бумаги.

Назначение атрибутов таблицы Акция–Security:

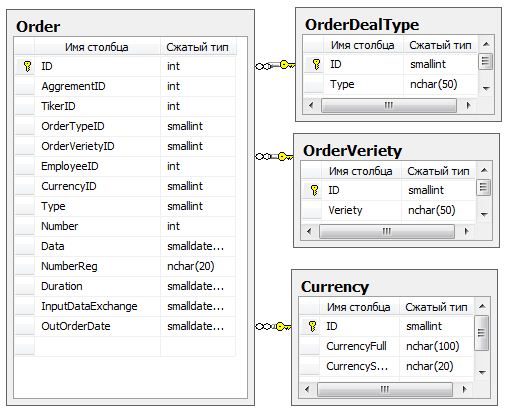
* ID – суррогатный ключ;
* TikerID – внешний ключ для связи с таблицей Tiker;
* NumberRegistration – Регистрационный номер ЦБ;
* DateRegistration – дата регистрации ЦБ;
* Category – категория ЦБ;
* FormIssue – форма выпуска;
* Number – порядковый номер выпуска;
* Principal – номинал;
* Amount – количество ЦБ в выпуске;
* DateAccommodation – дата размещения;
* DateReport – дата регистрации отчета о размещении;
* Note – примечания.

Назначение атрибутов таблицы Облигация – Bond:

* ID – суррогатный ключ;
* TikerID – внешний ключ для связи с таблицей Tiker;
* NumberRegistration – Регистрационный номер ЦБ;
* DateRegistration – дата регистрации ЦБ;
* Category – категория ЦБ;
* FormIssue – форма выпуска;
* Number – порядковый номер выпуска;
* Principal – номинал;
* Amount – количество ЦБ в выпуске;
* DateAccommodation – дата размещения;
* DateReport – дата регистрации отчета о размещении;
* DataRepayment – дата погашения;
* CouponsAmount – количество купонов;
* CouponsRate – размер купона;
* Note – примечания.

**Вариант 8. Информационная подсистема поручений по сделкам с ценными бумагами**

Назначение подсистемы поддержание в актуальном состоянии информации по поручениям клиентов.



**Рис. l.9.**  Структура базы данных подсистемы поручений по сделкам с ценными бумагами

Назначение атрибутов таблицы Поручение –Order:

* ID – суррогатный ключ;
* AgreementID
* TikerID – внешний ключ для связи с таблицей Tiker;
* OrderTypeID – внешний ключ для связи с таблицей OrderType;
* OrderVerietyID – внешний ключ для связи с таблицей OrderVeriety;
* EmployeeID – внешний ключ для связи с таблицей Employee;
* CurrencyID – внешний ключ для связи с таблицей CurrencyID;
* Type – тип поручения;
* Number – последовательный номер приказа;
* Data – дата и время получения заявки;
* NumberReg – регистрационный номер в системе внутреннего учета;
* Duration – срок действия приказа;
* InputDataExchange – время ввода заявки в торговую систему;
* OutOrderDate – время снятия заявки.

Назначение атрибутов таблицы Тип сделки – OrderDealType:

* ID – суррогатный ключ;
* Type – тип сделки по приказу.

Назначение атрибутов таблицы Вид сделки – OrderVeriety:

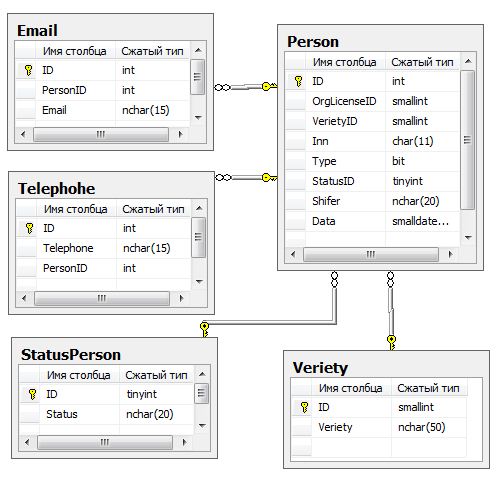
* ID – суррогатный ключ;
* Veriety – вид сделки по приказу.

Назначение атрибутов таблицы Валюта – Currency:

* ID – суррогатный ключ;
* CurrencyFull – полное наименование валюты;
* CurrencyShort – краткое наименование валюты.

**Вариант 9. Информационная подсистема учета клиентов компании**

Назначение подсистемы поддержание в актуальном состоянии информации по клиентам компании.



**Рис. l.10.**  Структура базы данных подсистемы учета клиентов компании

Назначение атрибутов таблицы Клиент –Person:

* ID – суррогатный ключ;
* OrgLicenseID – внешний ключ для связи с таблицей OrgLicense;
* VerietyID – внешний ключ для связи с таблицей Veriety;
* StatusID – внешний ключ для связи с таблицей Status;
* Inn – ИНН клиента;
* Type – тип клиента;
* Shifer – шифр клиента;
* Data – дата регистрации клиента.

Назначение атрибутов таблицы Адрес электронной почты –Email:

* ID – суррогатный ключ;
* PersonID – внешний ключ для связи с таблицей Person;
* Email – адрес электронной почты.

Назначение атрибутов таблицы Телефон – Telephone:

* ID – суррогатный ключ;
* PersonID – внешний ключ для связи с таблицей Person;
* Telephone – телефон клиента.

Назначение атрибутов таблицы Статус – Status:

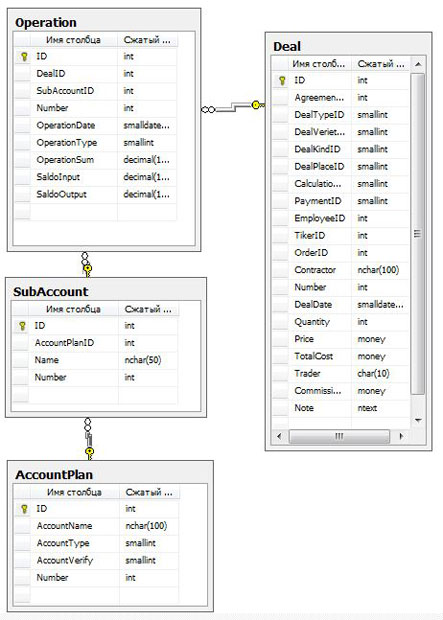
* ID – суррогатный ключ;
* Status – статус клиента.

Назначение атрибутов таблицы Вид клиента –Veriety:

* ID – суррогатный ключ;
* Veriety –вид клиента.

**Вариант 10. Информационная подсистема учета операций с ценными бумагами**

Назначение подсистемы поддержание в актуальном состоянии информации по операциям с ценными бумагами клиентов компании.



**Рис. l.11.**  Структура базы данных подсистемы учета операций с ценными бумагами

Назначение атрибутов таблицы Операция – Operation:

* ID – суррогатный ключ;
* DealID – внешний ключ для связи с таблицей Deal;
* SubAccountID – внешний ключ для связи с таблицей SubAccount;
* Number – номер операции;
* OperationDate – дата проведения операции;
* OperationType – тип операции;
* OperationSum – приход/расход по операции;
* SaldoInput – сальдо начальное;
* SaldoOutput – сальдо конечное.

Назначение атрибутов таблицы Субсчет –SubAccount:

* ID – суррогатный ключ;
* AccountPlanID – внешний ключ для связи с таблицей AccountPlan;
* Name – наименование субсчета;
* Number – номер субсчета.

Назначение атрибутов таблицы План счетов –AccountPlan:

* ID – суррогатный ключ;
* AccountName – наименование счета внутреннего учета;
* AccountType – тип счета;
* AccountVerify – вид счета;
* Number – номер операции;

Назначение атрибутов таблицы Сделка – Deal:

* ID – суррогатный ключ;
* AgreementID – внешний ключ для связи с таблицей Agreement;
* DealTypeID – внешний ключ для связи с таблицей DealType;
* DealKindID – внешний ключ для связи с таблицей DealKind;
* DealPlaceID – внешний ключ для связи с таблицей DealPlace;
* CalculationID – внешний ключ для связи с таблицей Currency;
* PaymentID – внешний ключ для связи с таблицей Currency;
* EmployeeID – внешний ключ для связи с таблицей Employee;
* TikerID – внешний ключ для связи с таблицей Tiker;
* OrderID – внешний ключ для связи с таблицей Order;
* Contractor – контрагент по сделке;
* Number – номер сделки;
* DealDate – дата и время заключения сделки;
* Quantity – количество ценных бумаг по сделке;
* Price – цена по сделке;
* TotalCost – общая сумма сделки;
* Trader – код трейдера;
* CommissionExchange – комиссия торговой площадки;
* Note – примечание.

**ЛАБОРАТОРНАЯ РАБОТА № 2**

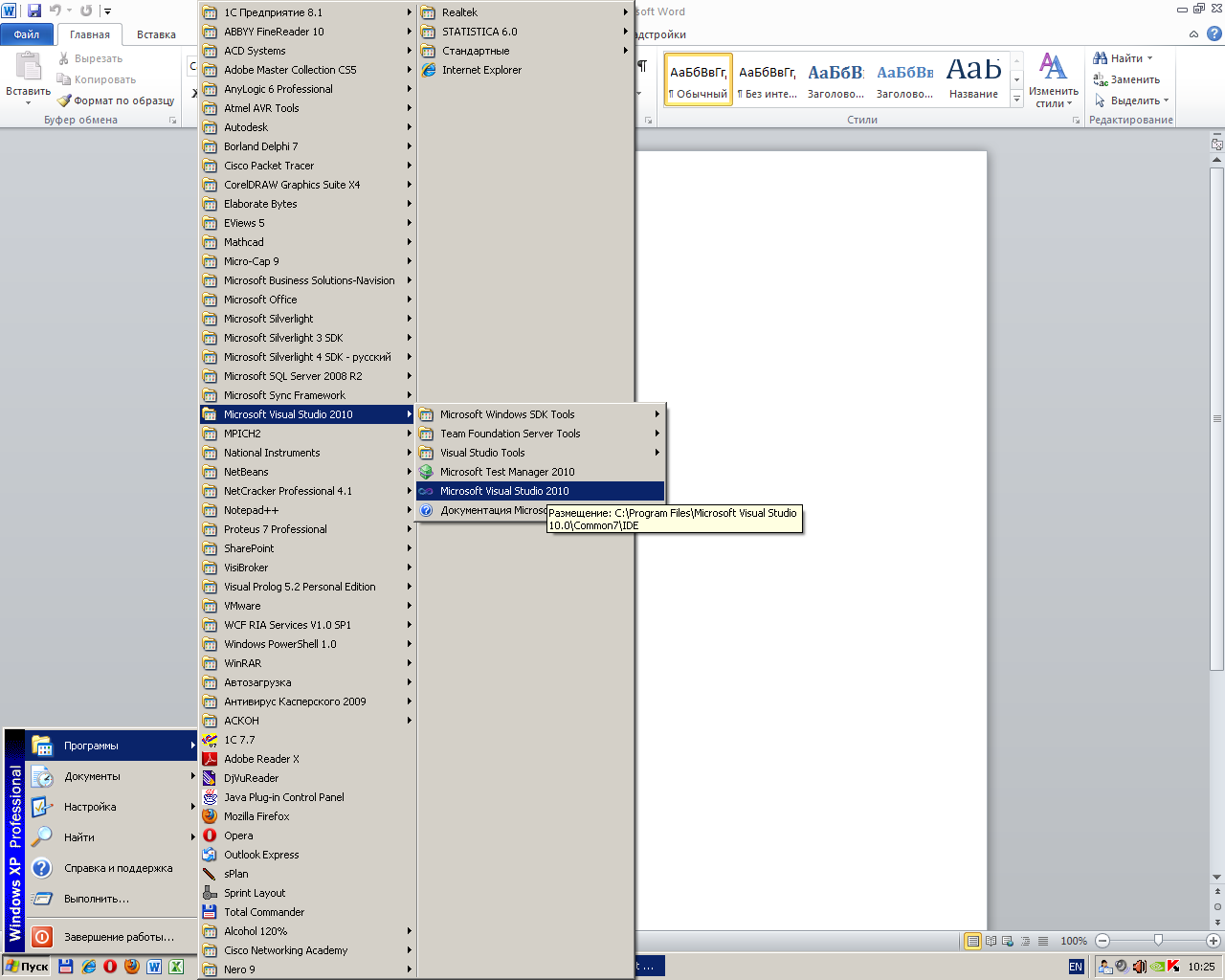
**Тема:** Создания монолитного приложения

**Цель:** Освоить принципы создания приложения для взаимодействия с базой на основе .NET технологий.

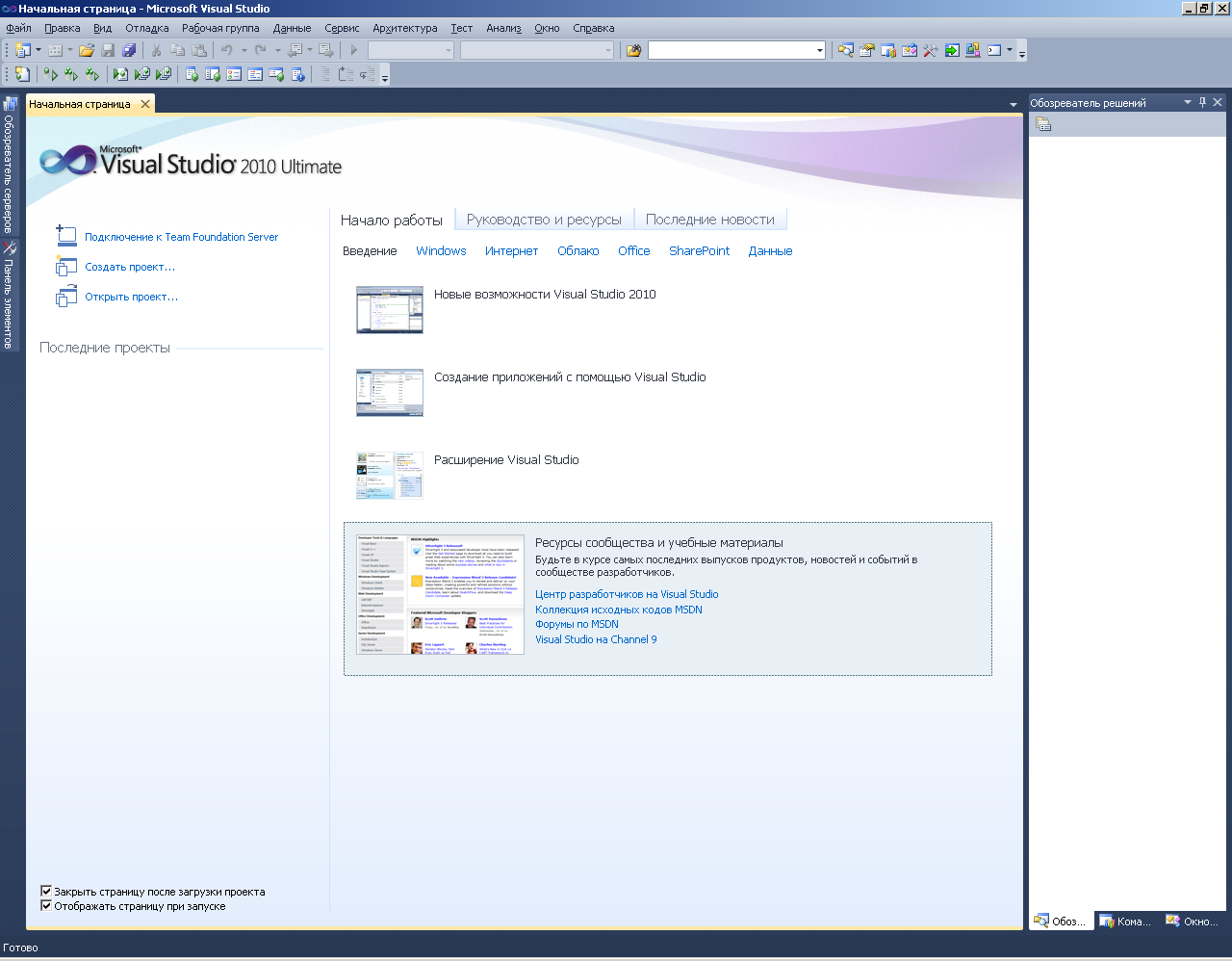
ХОД РАБОТЫ

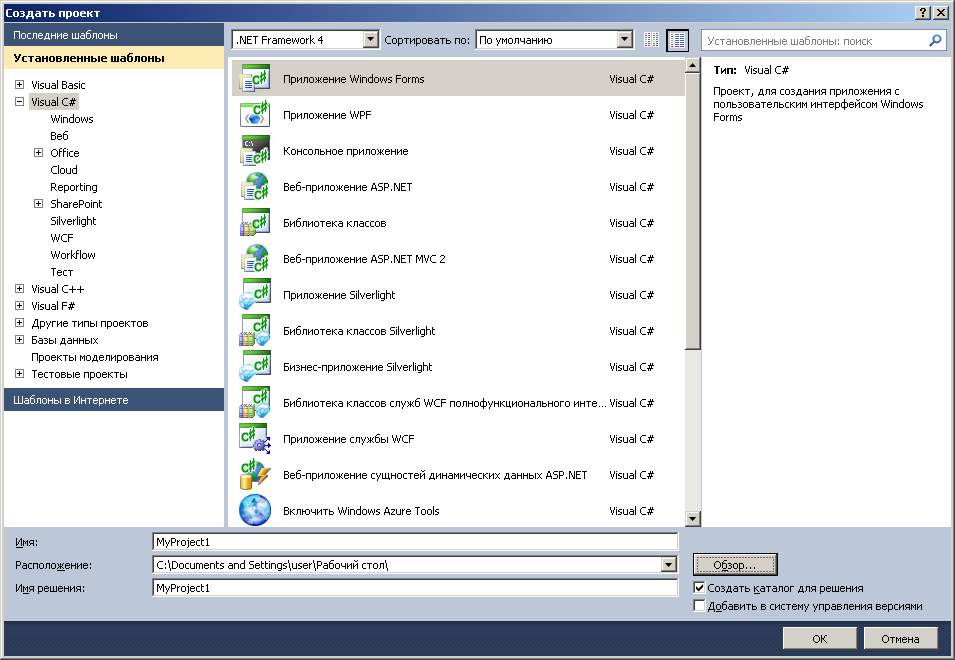
* Запустите среду Microsoft Visual Studio 2010

Пуск – Программы - Microsoft Visual Studio 2010



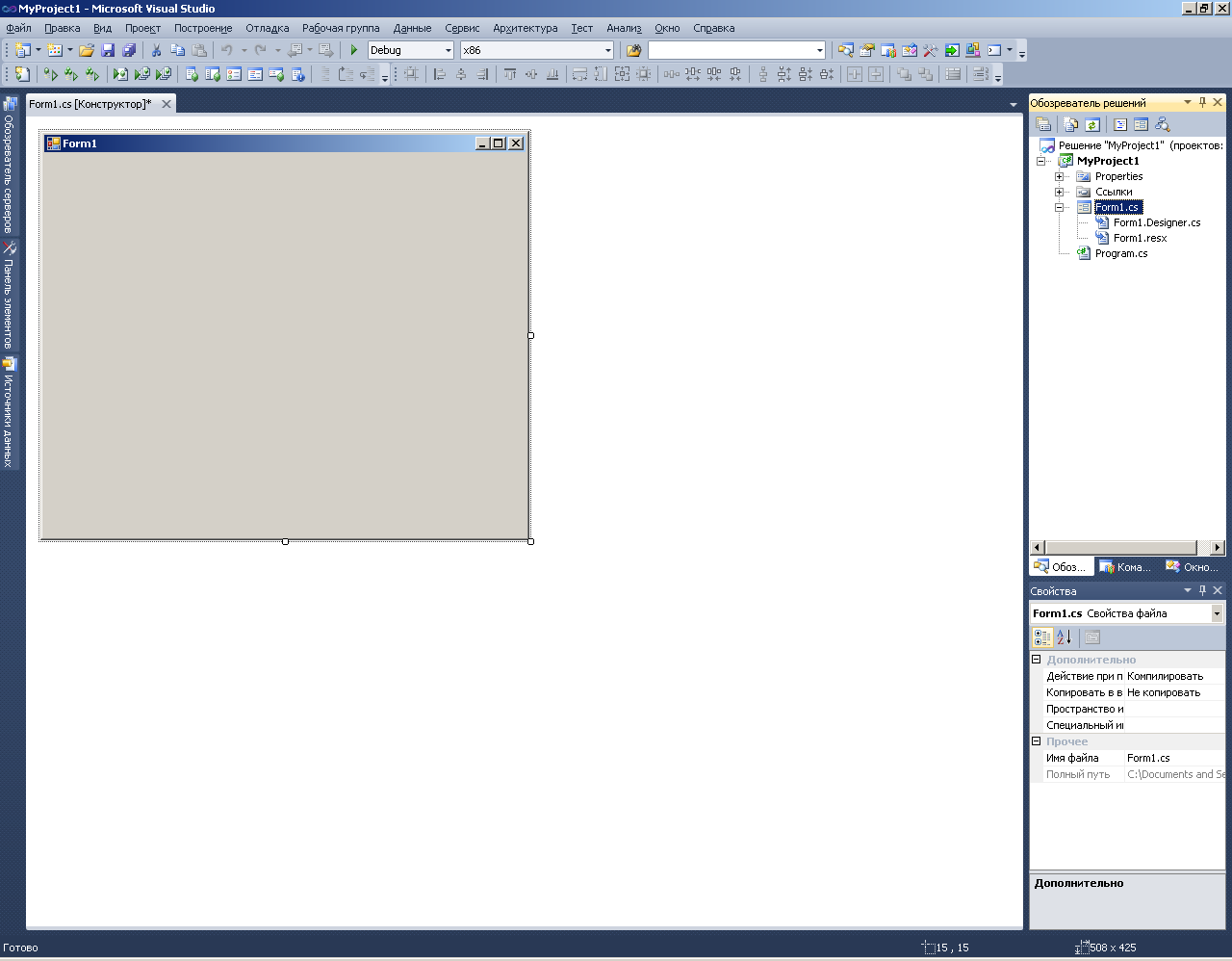
После загрузки среды разработки необходимо создать новый проект (На окне приветствия «Создать проект или Файл – Создать – проект»)



Как язык разработки рекомендуется выбрать С#. Тип приложения – Приложение Windows Forms 

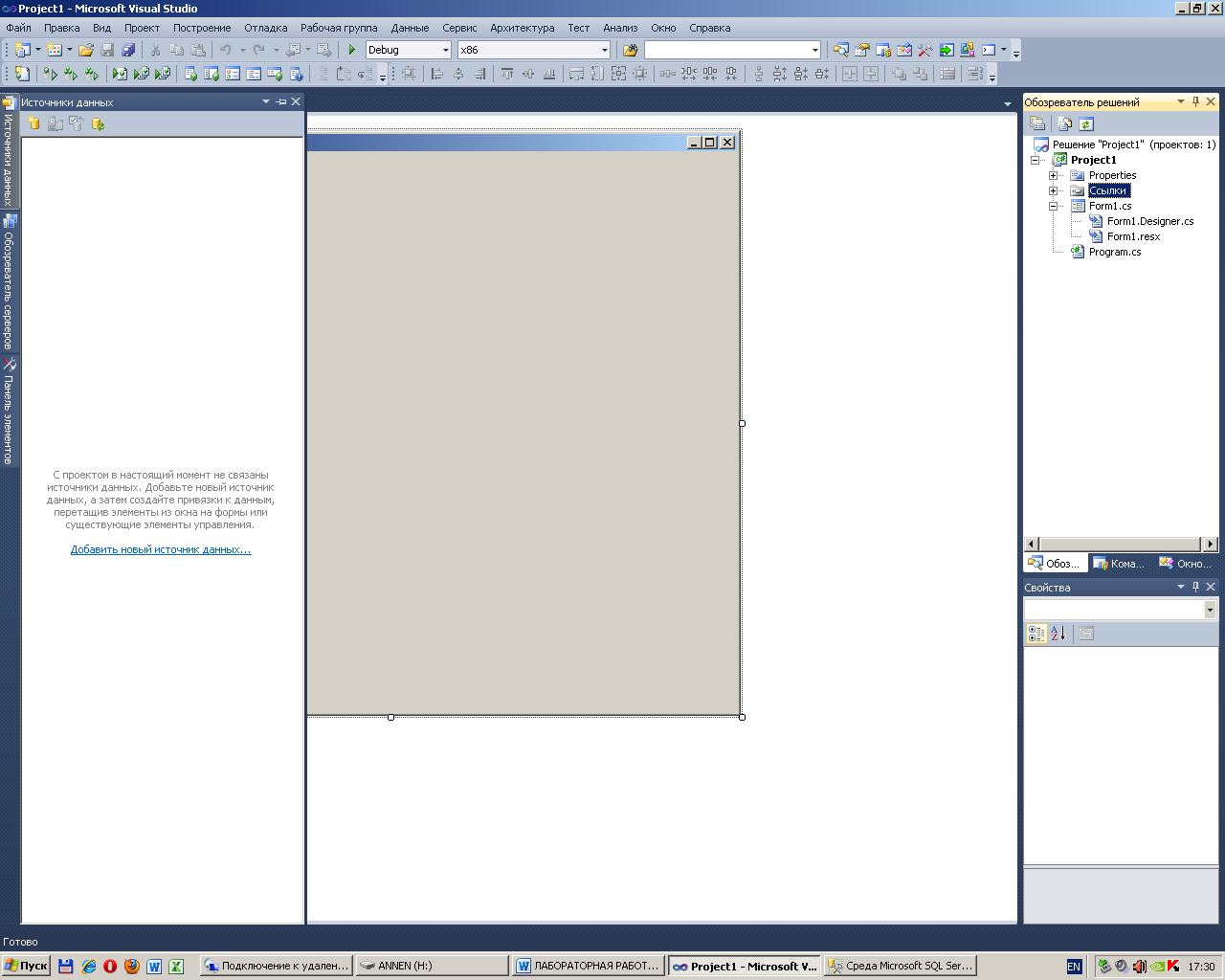
Укажите название проекта и место где он будет храниться, после чего нажмите ОК.

Откроется окно с внешним видом формы

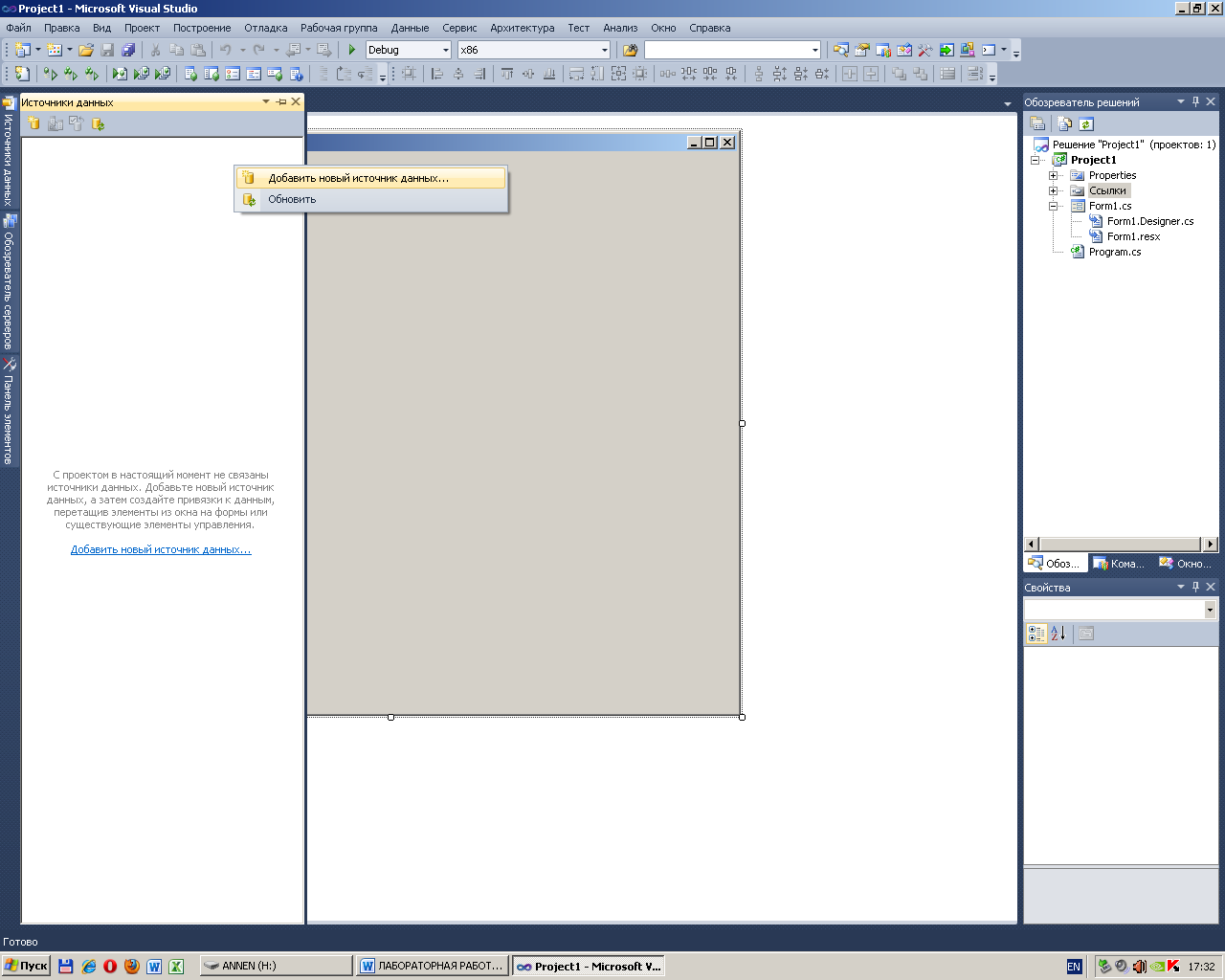


Для работы с базами данных в Visual Studio 2010 существует несколько компонентов доступа.

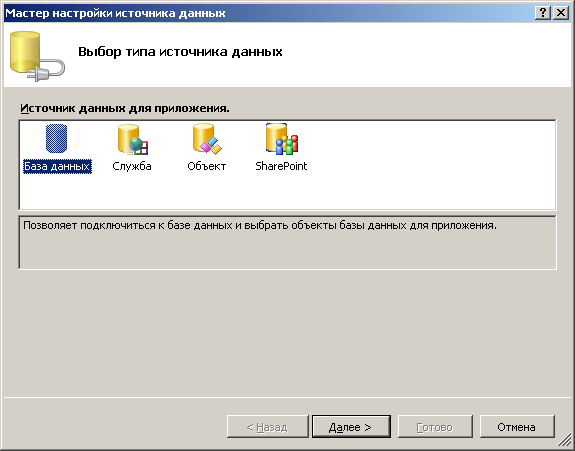
Для примера будем использовать: источники данных



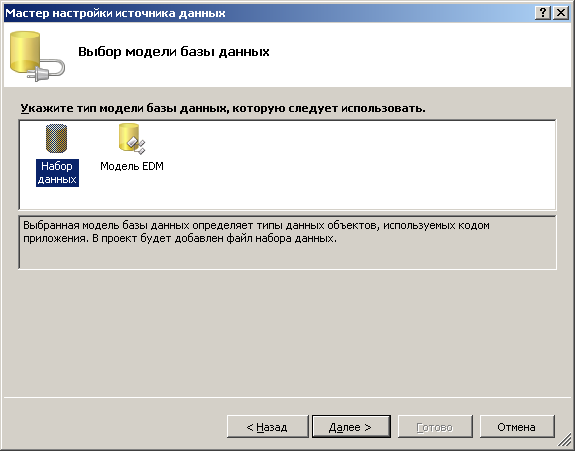
Создадим новый источник данных



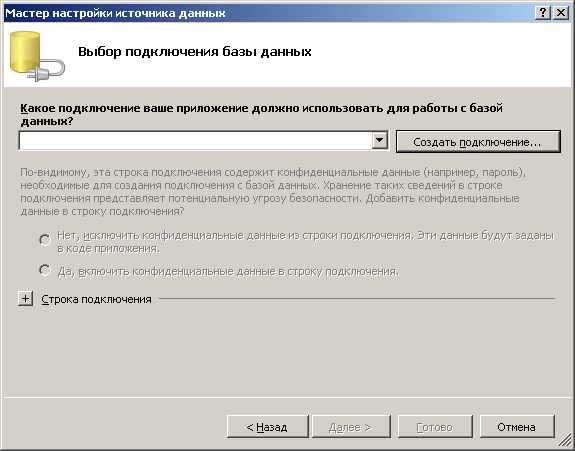
Выбираем источник данных как база данных



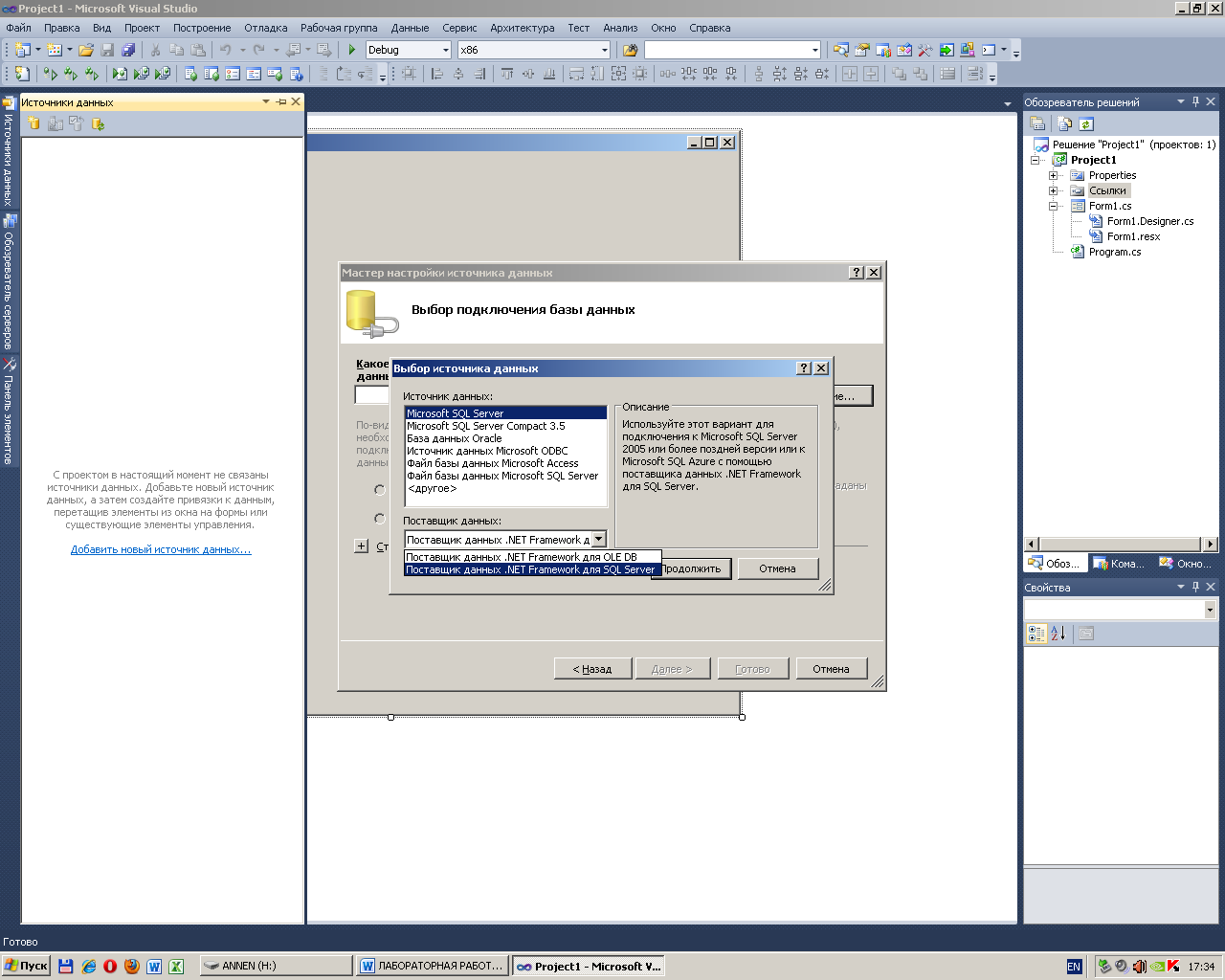
Определяем, что используем базу данных как набор данных



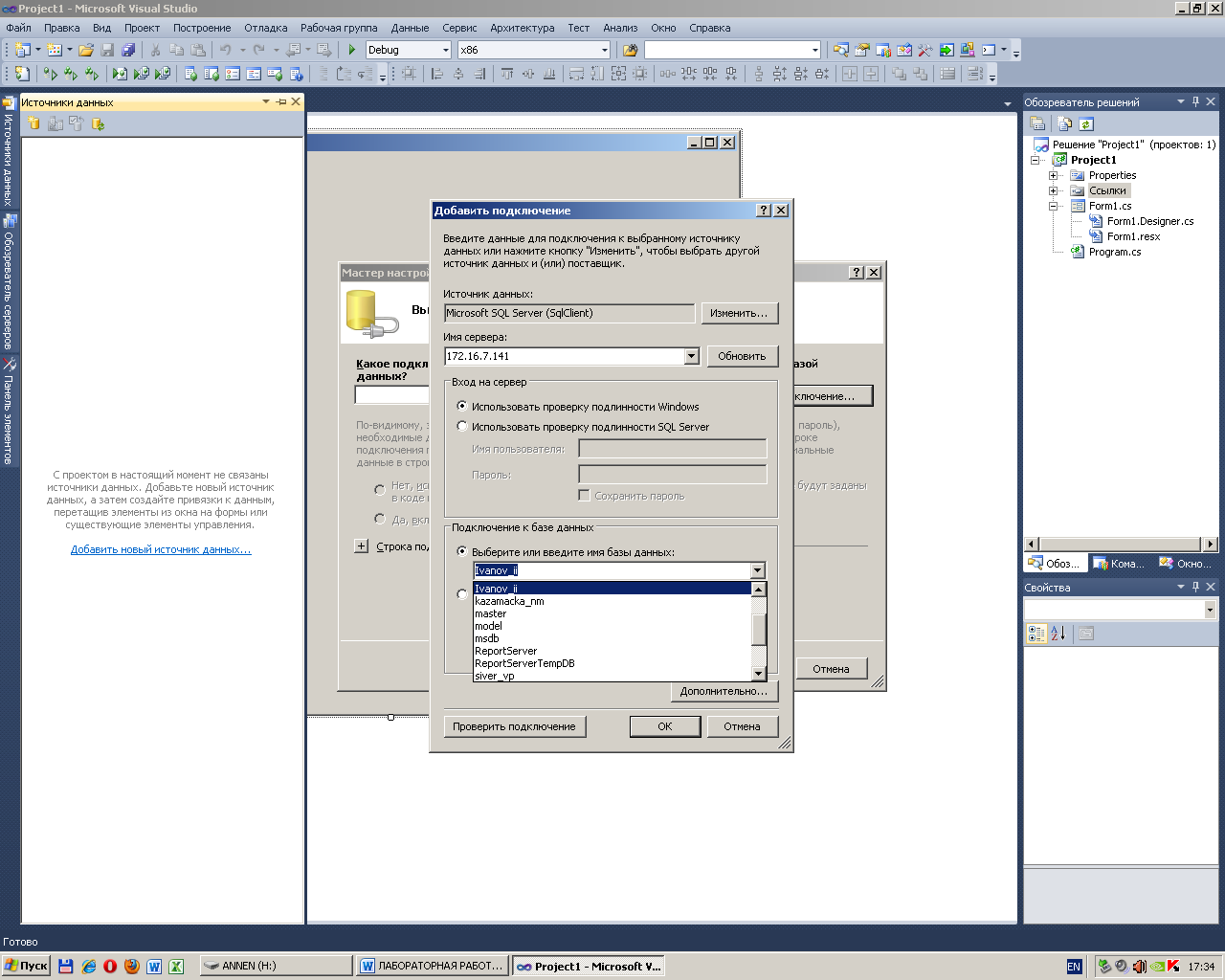
Для того чтобы знать с кем взаимодействовать создадим новое подключение



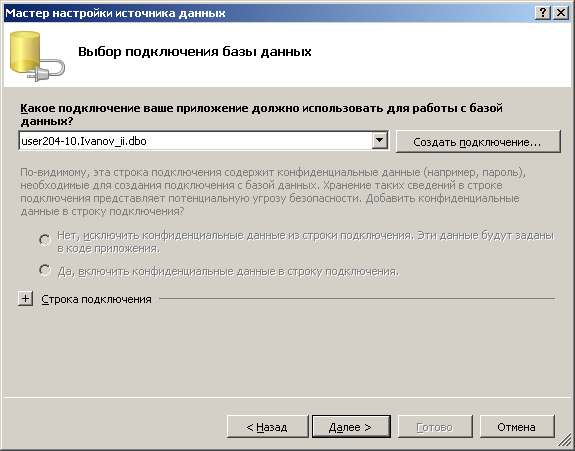
Укажем тип сервера и поставщика услуг



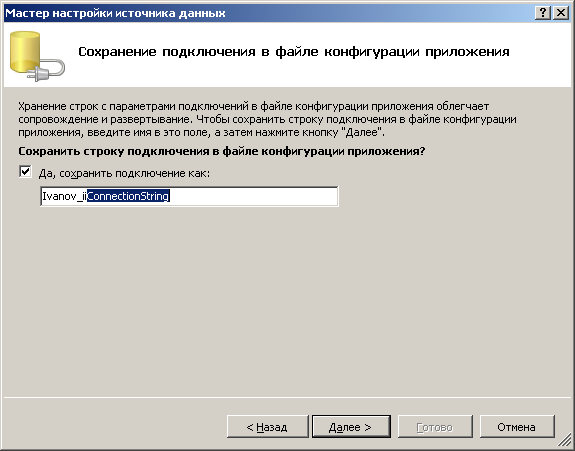
Укажите сервер подключения (по умолчанию 172.16.7.141) и базу, с которой будет связана программа



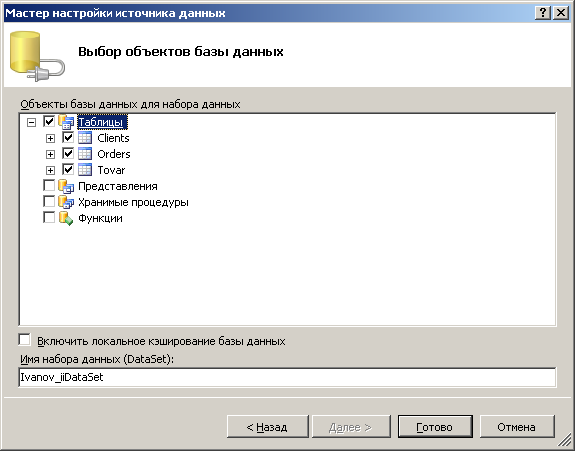
Подтверждаете создание соединения



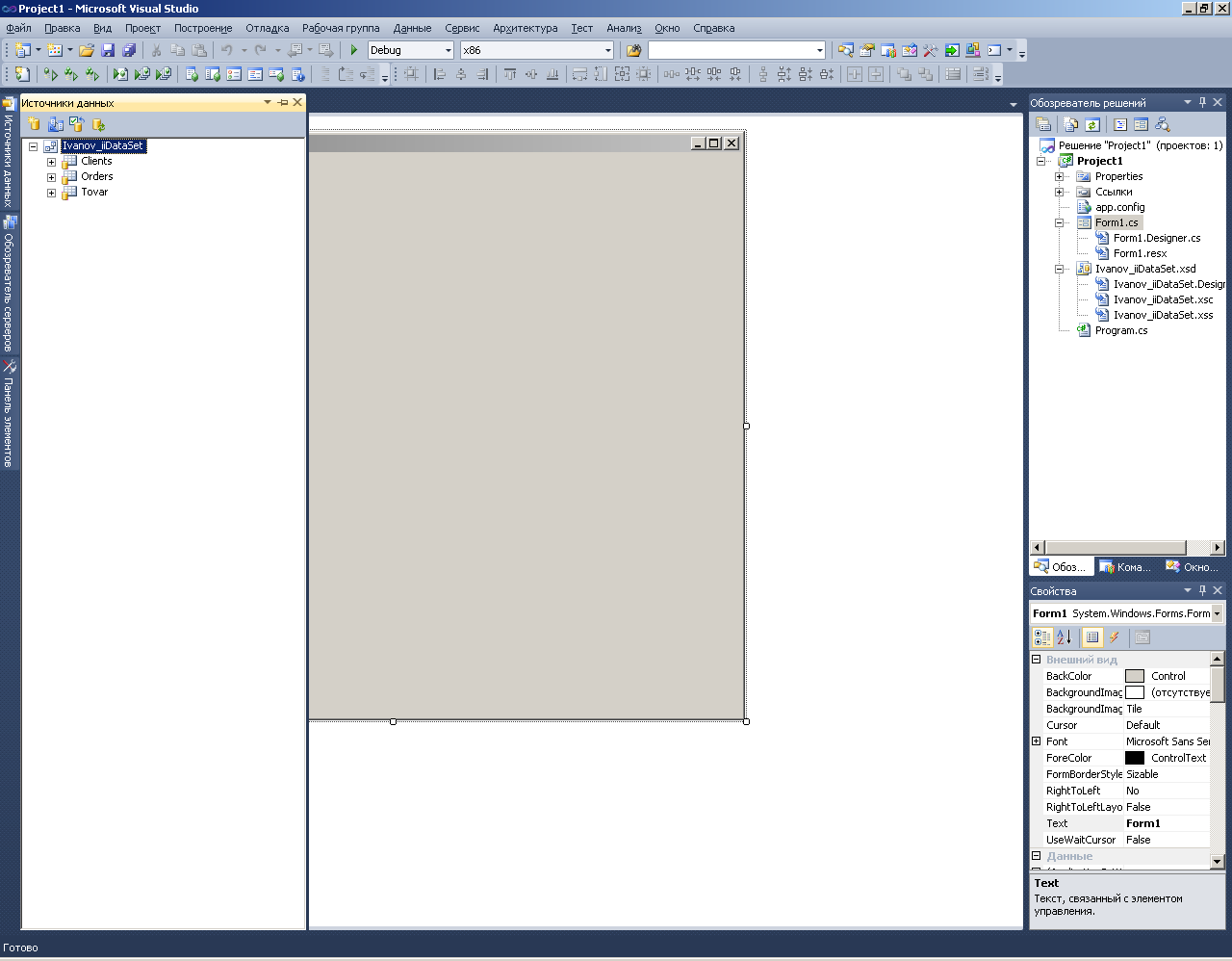
Сохраняете строку соединения



Последним шагом у вас является выбор объектов, которые будут использоваться соединением



По окончанию созданы связи с вашей базой данных



Обратите внимание – чтобы добавить элемент на поле можно просто перетащить – и будет создан соответствующий элемент по умолчанию

Можно выполнить создание интерфейса используя технологию Drag&Drop – перетягивая элементы с панели «Источники данных» а можно создать вручную.

* Получения Доступа к данным вручную

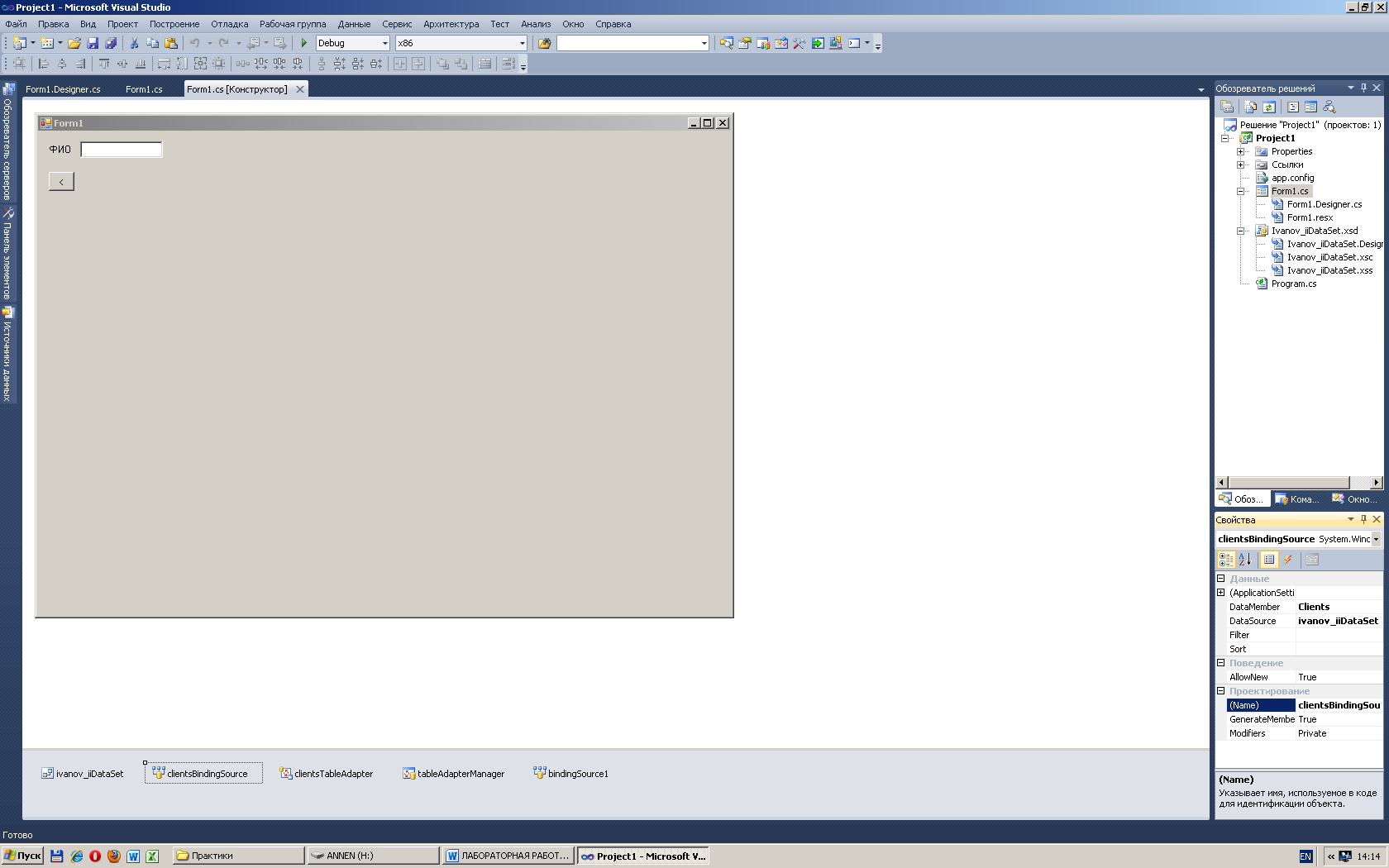
**Работа с данными на форме**

Во первых, необходимо создать элемент который будет связывать данные в хранилище с компонентами на форме. Процедура связи происходит следующим образом

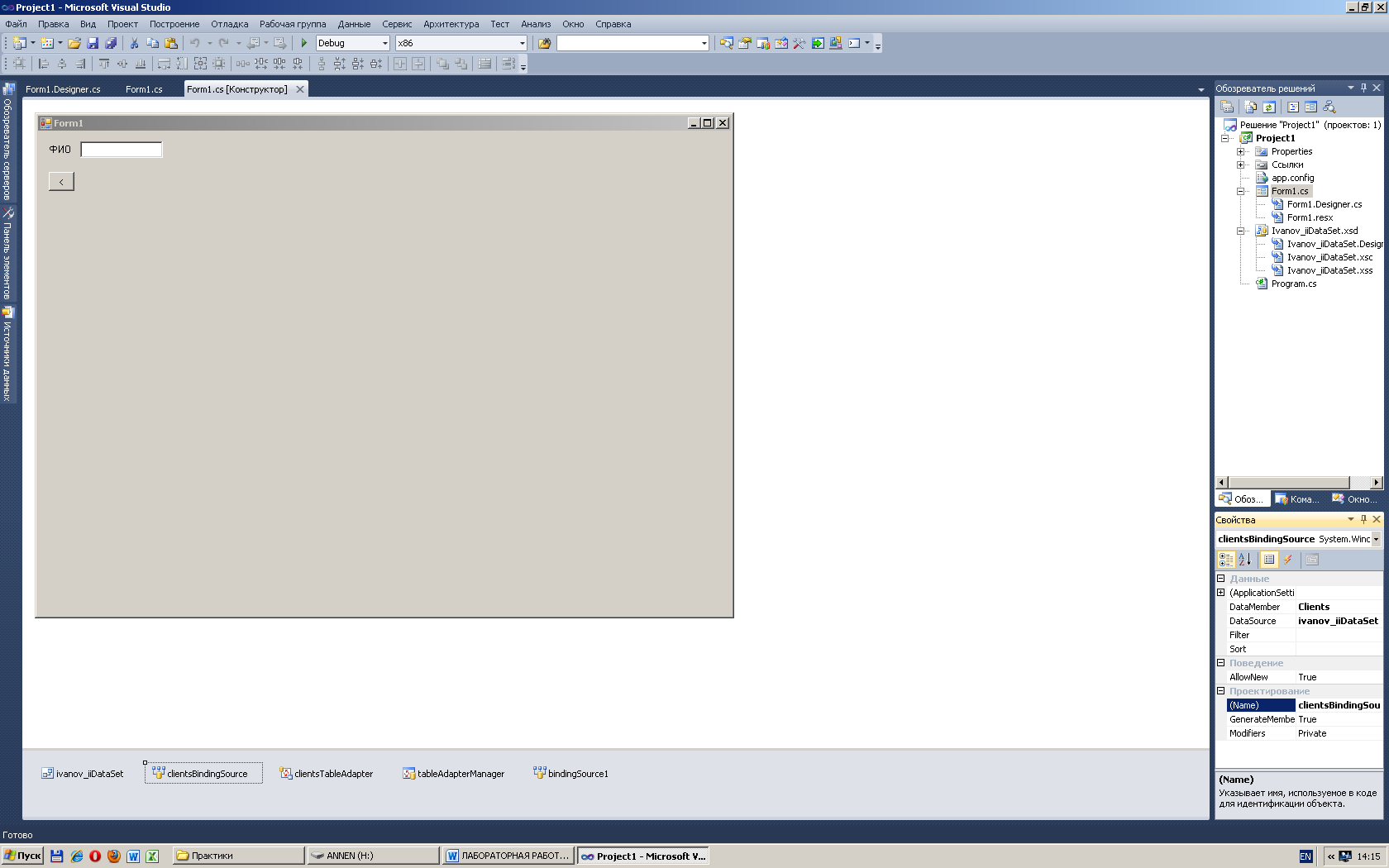
Компонеты

TableAdapter

BindingSource



DataSet



БД

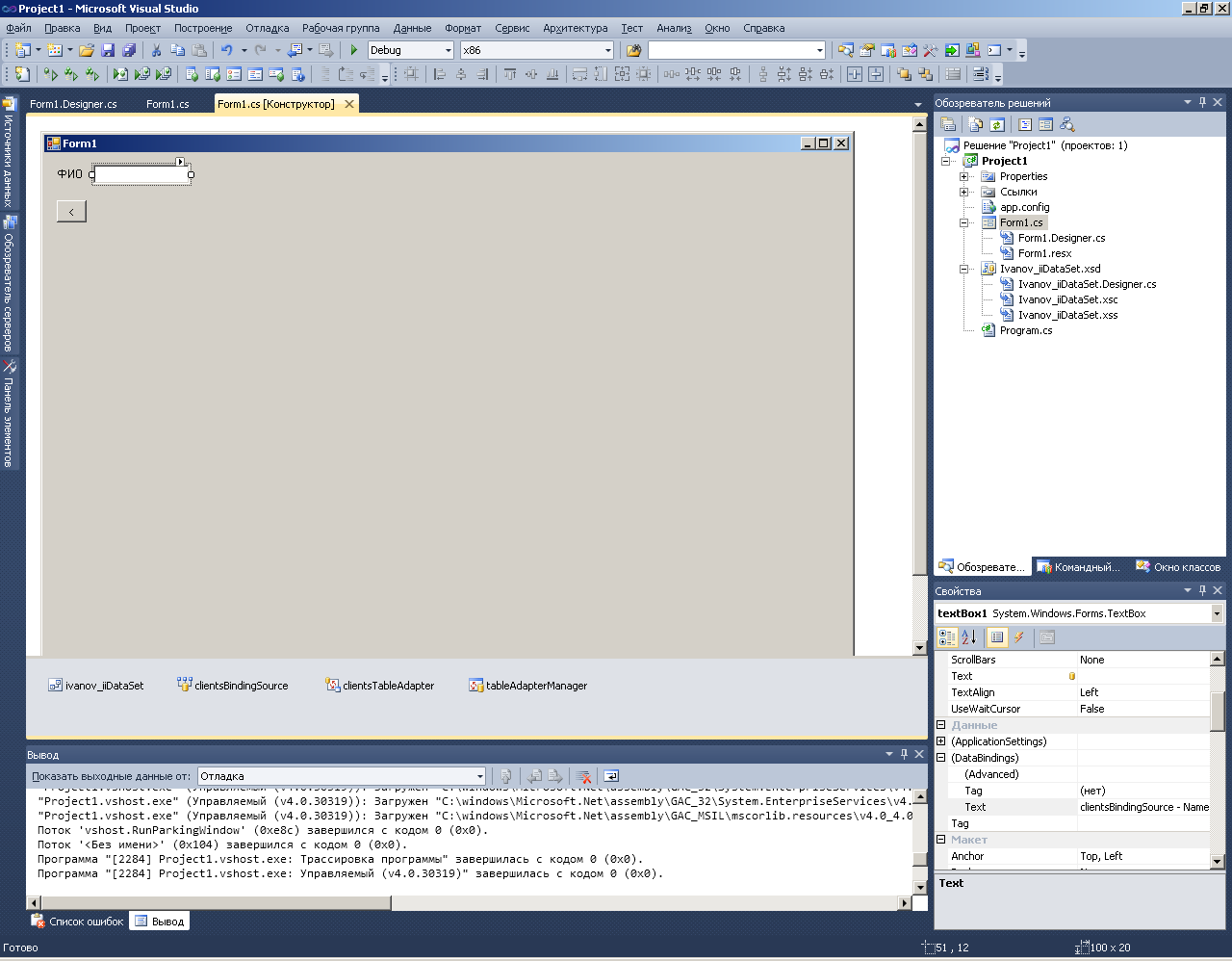
Выбор данных и управление ими на основе запросов

Отображения данных

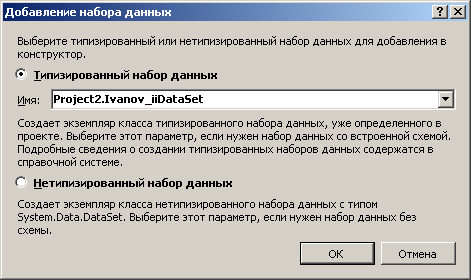
Компонент для того чтобы управлять связью с помощью компонентов

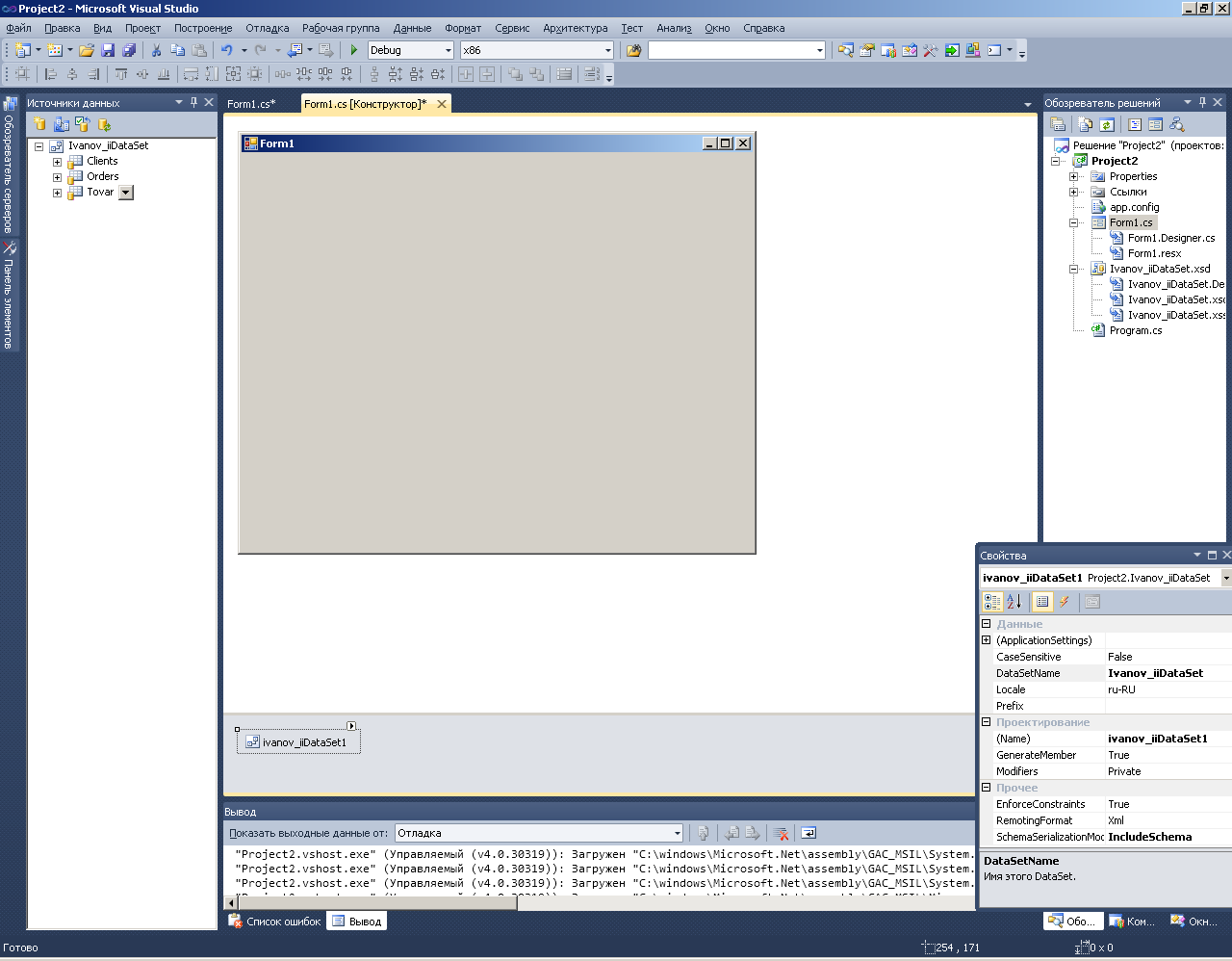
Компонент для связи с БД

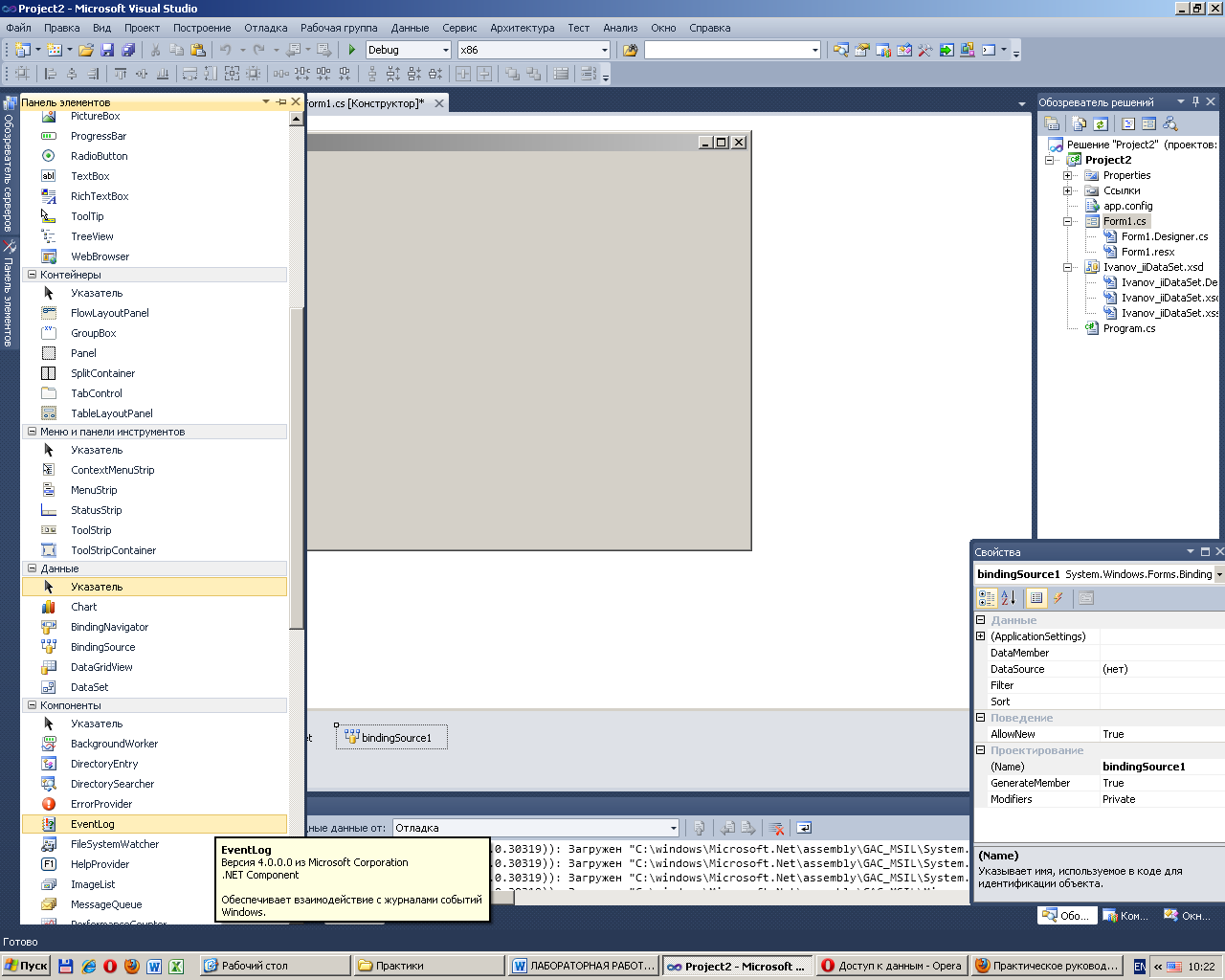
Сами данные

Для создания вручную необходимо создать элемент типа TextBox –  и установить связь между ним и данными, например

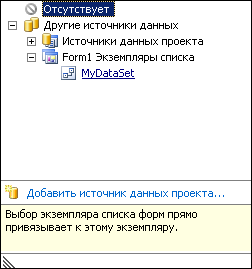
1. Создадим компонент управления DataSet – на панели элементов, вкладка ДАННЫЕ- компонент DataSet. И расположим его на форме. Он добавиться в список внизу….





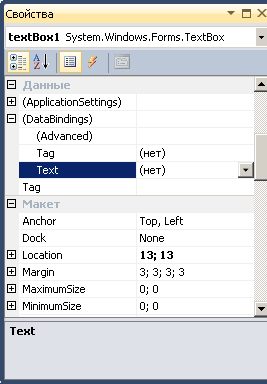
1. Изменим свойство Name на MyDataSet
2. Добавим компонент связи с данными bindingSource - из панели элементов, вкладка ДАННЫЕ- компонент bindingSource. Расположить его на форме.
3. Настроим его

Name = MySource

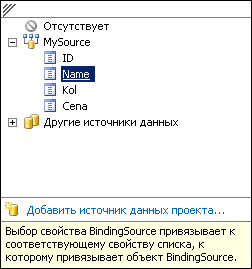
DataSource = MyDataSet 

DataMember = Tovar (Одной из ваших таблиц)

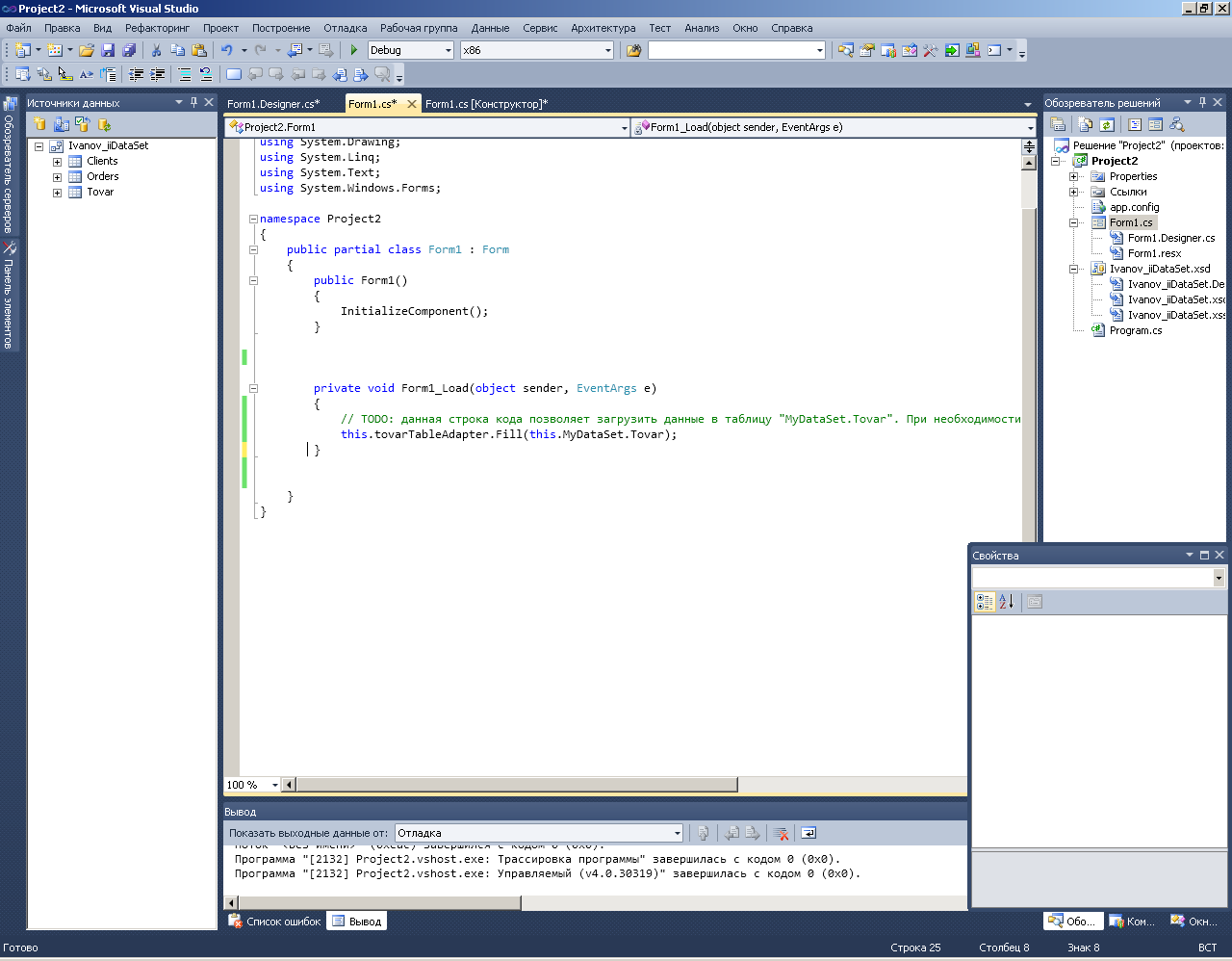
Обратите внимание что автоматически создался адаптер к данным tovarTableAdapter – который достает все данные из таблицы (Посмотрите его данные или создайте новый запрос на основе SQL)

1. Добавьте на форму несколько TextBox на форму
2. Настройте их: 

Свойства компонента – Данные – DataBindings – Text – MySource - Любое поле из таблицы



Обратите внимание, что в таблице должна быть строка -



процедуру, выполняемую над базой данных SQL Server.

**Перемещение по записям**

Создайте кнопку – NEXT – которая содержит код MySource.MoveNext();

Аналогичным образом создайте код для

MySource.MovePrevious();

MySource.MoveFirst();

MySource.MoveLast();

**Организация Добавления записи**

Ivanov\_iiDataSet.TovarRow s = MyDataSet.Tovar.NewTovarRow();

s.Name = "MyTovar";

s.Cena = 2.5;

s.Kol = 3;

MyDataSet.Tovar.Rows.Add(s);

Т.е. Создаем строку типа таблицы (DataRow) – её заполняем и добавляем в память базы DataSet

**Удаление записи**

Значение свойства RowState становится равным Deleted после использования метода Delete в существующем объекте DataRow. Значение Deleted сохраняется до вызова метода AcceptChanges. В этот момент объект DataRow удаляется из таблицы.

Удаленная строка может быть восстановлена с помощью вызова метода RejectChanges.

MyDataSet.Tovar.Rows[0].Delete();

MyDataSet.Tovar.Rows[1].Delete();

MyDataSet.Tovar.Rows[1].RejectChanges();

MyDataSet.Tovar.AcceptChanges();

**Работа с данными через базу**

Работа с данными в базе происходит через компонент SQLCommand, который использует SQLConnection

Например

НЕОБХОДИМО ПРОПИСАТЬ В USIGN

Using [System.Data.SqlClient](http://msdn.microsoft.com/ru-ru/library/system.data.sqlclient%28v=vs.90%29.aspx)

Далее создать соединение и запрос с обработкой данных

Например, выборка списка элементов (Последовательный доступ - потоковый)

// 1. Создаем соединение с базой Ivanov.ii

SqlConnection SC = new SqlConnection("Data Source=172.16.7.141;Initial Catalog=**Ivanov\_ii**;Integrated Security=True");

// 2. Открываем подключение

SC.Open();

// 3. Создаем запрос с выборкой данных

SqlCommand Sel = new SqlCommand("Select \* from tovar", SC);

// 4. Получаем читатель для данных

SqlDataReader SR = Sel.ExecuteReader();

// 5. Читаем до конца

while (SR.Read()) {

// 6. Обрабатываем данные

listBox1.Items.Add(SR.GetString(1));

}

// 7. Закрываем читатель

SR.Close();

По окончании использования объекта SqlReader всегда следует вызывать метод Close. Если SqlCommand содержит выходные данные, они будут недоступны до закрытия объекта DataReader.

Имейте в виду, что пока объект SqlReader открыт, соединение Connection используется исключительно этим объектом DataReader. Невозможно выполнять какие-либо команды для Connection, включая создание другого объекта DataReader, пока исходный объект DataReader не будет закрыт.

Использование SqlCommand для добавления данных в таблицу

// 1. Создаем строку запроса

string insertString = @"insert into Tovar (Name, Kol, Cena) values ('Kolbas’’OK', 5, 2.5)";

// 2. Создаем Команду и связываем её с существующим соединением

SqlCommand cmd = new SqlCommand(insertString, SC);

// 3. Вызываем ExecuteNonQuery и выполняем команду

cmd.ExecuteNonQuery();

Аналогичным образом осуществляется изменение и удаление данных.

При данном методе доступа – данные считываются одно за одним и невозможно переходить к предыдущим записям. Чтобы использовать

Для получения параллельного доступа к выбранным данным можно использовать таблицы с данными. Получение данных из запроса и отображение их на форме

SqlConnection SC = new SqlConnection("Data Source=172.16.7.141;Initial Catalog=Ivanov\_ii;Integrated Security=True");

SC.Open();

SqlDataAdapter SD = new SqlDataAdapter("Select \* from tovar", SC);

DataTable DT = new DataTable();

SD.Fill(DT);

textBox1.Text = DT.Rows[0]["Name"].ToString();

textBox2.Text = DT.Rows[0]["Kol"].ToString();

Изменение данных в базе с помощью запросов

SqlConnection SC = new SqlConnection("Data Source=172.16.7.141;Initial Catalog=Ivanov\_ii;Integrated Security=True");

SC.Open();

SqlCommand Sel = new SqlCommand("INSERT SQL TEXT", SC);

Sel.ExecuteNonQuery();

Итак подведем итоги:

DataSet - Представляет расположенный в памяти кэш данных.

DataTable - Представляет одну таблицу с данными в памяти.

DataRow - Представляет одну строку данных в [DataTable](http://msdn.microsoft.com/ru-ru/library/system.data.datatable%28v=vs.90%29.aspx).

SqlCommand - Представляет инструкцию Transact-SQL или сохраненную

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Тема:** Получение ответа от сервера с указаним протокола и порта.

**Цель:** Получить навыки обмена информацией с сервером на основании протокола HTTP.

ХОД РАБОТЫ

Создать приложение которое подключиться к серверам

«Server:8080» - получит главную страницу

«center.zieit/sTest.html» - получит страницу после авторизации пользователя

«Google.com.ua» - отправит запрос получит результаты запроса и выведет их пользователю.

Например, чтобы получить заголовок страницы mail.ru были использованы следующие объекты.

Нужно добавить using System.Net;

WebClient wc = new WebClient();

wc.Proxy = new WebProxy("172.16.7.40", 3128);

wc.Proxy.Credentials = new NetworkCredential("login", "password");

string s = wc.DownloadString("http://www.mail.ru/");

s=s.Substring(s.IndexOf("<title>")+9);

s = s.Remove(s.IndexOf("</ti"));

textBox1.Text = s;

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**Тема:** Создание приложения для подключения к серверам HTTP.

**Цель:** Получить навыки работы с WebServer на основе объекта Socket.

ХОД РАБОТЫ

Создать консольное приложение:

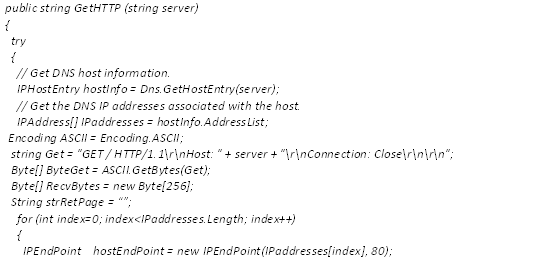
Которое по введенному адресу и порту подключиться к ВЕБ серверу и выведет содержимое страницы по умолчанию

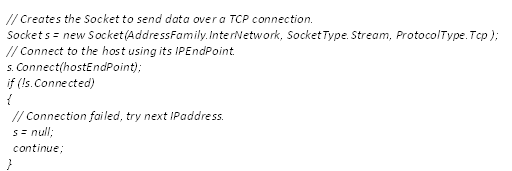
Сервера для тестирования

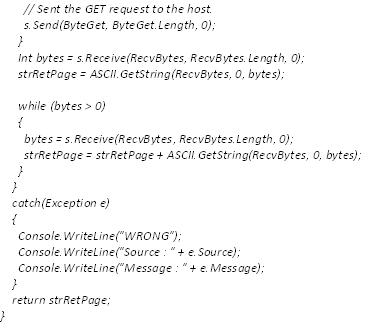
center.zieit

server:8080

Пример метода







**ЛАБОРАТОРНАЯ РАБОТА № 5**

**Тема:** Создание Клиент-серверного приложения для обмена данными.

**Цель:** Получить навыки обмена информацией с сервером на основании собственного протокола передачи данных через протоколы TCP или UPD.

ХОД РАБОТЫ

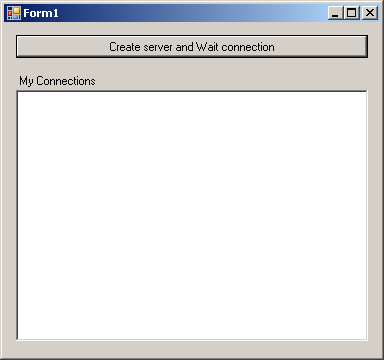
Создать два приложения:

1 – Сервер – подключенный к базе данных и имеет возможность обмениваться SQL сообщениями c базой данных

2 – Клиент – отправляет запросы в серверное приложение, получает ответ – и предоставляет его пользователю.

Например, чтобы получить с сервера список продуктов, которые запросит с фильтром пользователь можно реализовать следующие приложения.

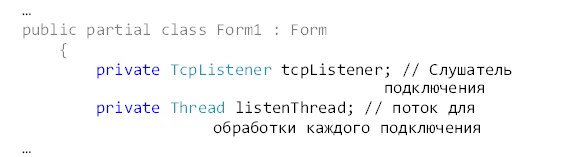
* Создать приложение имеющее подключение к БД.
* Добавить кнопку «создания сервера и ожидания подключений» и добавить для мониторинга списки обслуживаемых клиентов



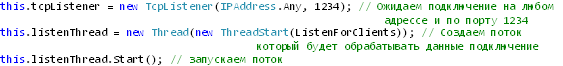
* В коде добавить ссылки на пакеты:



* Добавить две переменные для взаимодействия

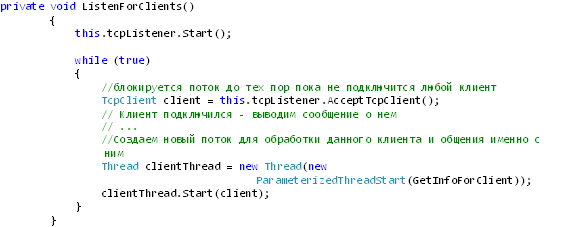


* В кнопке создания реализуем код:

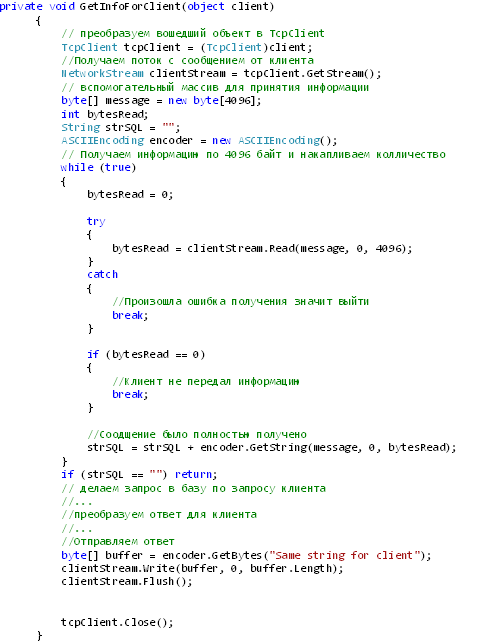


* И добавляем в код несколько методов для обработки подключений и принятие информации

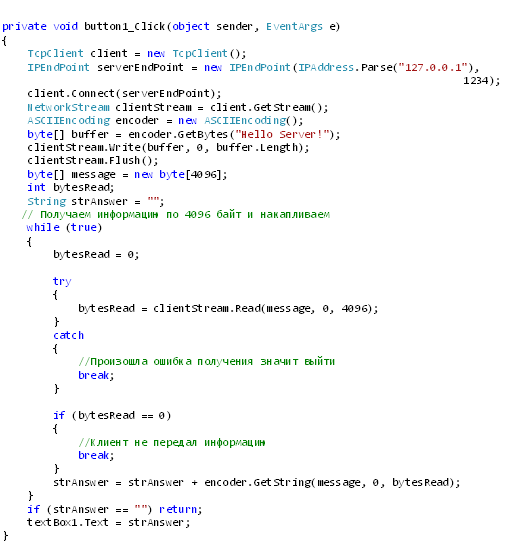
Метод ожидания клиентов



Метод обработки каждого клиента



Для подключения к Серверу необходимо реализовать в клиенте:



**ЛАБОРАТОРНАЯ РАБОТА № 5**

**Тема:** Создание Клиент-серверного приложения для обмена данными.

**Цель:** Получить навыки обмена информацией с сервером на основании собственного протокола передачи данных через протоколы TCP или UPD.

ХОД РАБОТЫ

Создать два приложения:

1 – Сервер – подключенный к базе данных и имеет возможность

**ЛАБОРАТОРНАЯ РАБОТА № 7**

**Тема:** Разработка Веб-сервисов

**Цель:** Получить навыки создания веб сервисов для обработки информации на другом сервере.

ХОД РАБОТЫ

Создать веб-сервис который:

1 – Примет запрос на выбор данных с фильтрацией для определенной таблицы, сформирует ответ и отправит обратно

2 – Примет запрос добавление, удаление или обновление данных и отправит информацию об успешном изменении

Веб-служба это код, к которому можно обратиться через HTTP.

Так как все это реализовано через HTTP, то язык на котором будет строиться веб служба не имеет значения. В качестве клиентов, используются как оконные так и веб приложения. Для вызова методов типов веб служб можно обойтись HTTP запросами на основе REST протокола, но так же можно воспользоваться XML и SOAP.

Веб-служба строится из тех же компонентов .NET (Классы, интерфейсы, сборки) и выполняет роль черного ящика для клиента, возвращая ответ на запрос. Для того чтобы Windows и консольные приложения могли работать с веб-сервисами, в .NET существуют средства для генерации прокси сборок, котоорые перенаправляют запросы от приложений на веб службу через HTTP и SOAP

Веб-службе (как и любому приложению ASP.NET) соответствует виртуальный каталог IIS. Однако также необходимо реализовать поддержку инфраструктуры

* Протокол подключения (HTTP-Rest,HTTP-JSON,SOAP)
* Служба описания - Description service (Чтобы клиент мог получить информацию о том, что веб-служба делает)
* Служба обнаружения - Discovery service (Чтобы клиент мог получить информацию о том что веб служба существует)

Стоит ориентироваться на SOAP, так как с его помощью можно передавать не только стандартные типы данных, такие как int, string и тд , но и сложные пользовательские классы, dataSet, массивы объектов и т.д.

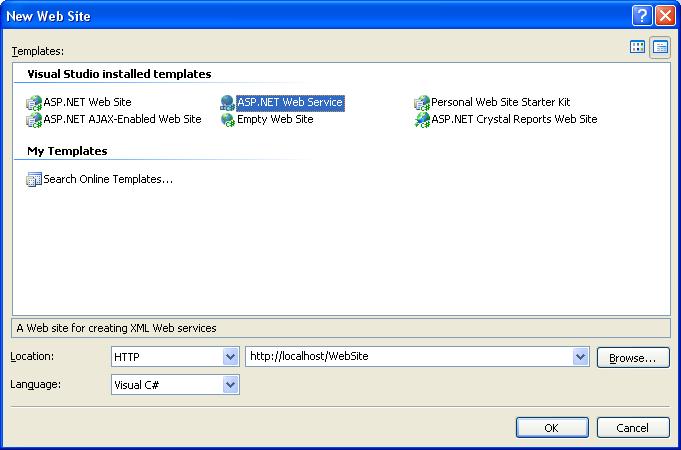
При обращении к удаленной web-службе клиент должен обладать полной информацией о членах типов веб-службы. Клиент должен иметь информацию какие методы может вызывать и с какими параметрами, а также что возвращает данный метод. За предоставление данной информации отвечает служба описания (description service). Информация о веб-службе предоставляется в XML формате Данная XML схема называется WSDL

Служба обнаружения позволяет клиентуобнаруживать веб-службы по адресу URL. Для этой службы используется файл .disco (тоже в XML)

Все типы для работы с инфраструктурой и самими веб сервисами собраны в следующих пространствах имен

* System.Web.Services - минимальный набор типов для построения веб служб
* System.Web.Services.Description - Набор типов для взаимодействия с WSDL
* System.Web.Services.Discovery - позволяет клиенту находить службу
* System.Web.Services.Discovery - Определены типы для работы с протоколами HTTP, REST, SOAP

Для создания веб службы создаем новый проект и выбираем тип проекта ASP.NET WebService



Cоздался класс наследник Webservice

Внутри этого класса и будут описываться наши веб методы

|  |
| --- |
| public class Service : System.Web.Services.WebService  {   public Service () {   //реализовать процедуры и их код  //InitializeComponent();   }  } |

**Реализация методов Web службы**

Создадим метод который будет возвращать время со смещение переданным от клиента. Для того чтобы вебслужба предоставила доступ к этому методу через HTTP его необходимо обозначить атрибутом [WebMethod]

Веб сервис принял вид

|  |
| --- |
| public class Service : System.Web.Services.WebService  {      public Service () {           //Uncomment the following line if using designed components           //InitializeComponent();       }       [WebMethod]      public DateTime GetClientTime(int hour) {          return DateTime.Now.AddHours(hour);      }        } |

http://simple-cs.ru/store/Webservice/9/

http://www.realcoding.net/article/view/2519

*public string GetHTTP (string server)*

*{*

*try*

*{*

*// Get DNS host information.*

*IPHostEntry hostInfo = Dns.GetHostEntry(server);*

*// Get the DNS IP addresses associated with the host.*

*IPAddress[] IPaddresses = hostInfo.AddressList;*

*Encoding ASCII = Encoding.ASCII;*

*string Get = "GET / HTTP/1.1\r\nHost: " + server + "\r\nConnection: Close\r\n\r\n";*

*Byte[] ByteGet = ASCII.GetBytes(Get);*

*Byte[] RecvBytes = new Byte[256];*

*String strRetPage = “”;*

*for (int index=0; index<IPaddresses.Length; index++)*

*{*

*IPEndPoint hostEndPoint = new IPEndPoint(IPaddresses[index], 80);*

*// Creates the Socket to send data over a TCP connection.*

*Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp );*

*// Connect to the host using its IPEndPoint.*

*s.Connect(hostEndPoint);*

*if (!s.Connected)*

*{*

*// Connection failed, try next IPaddress.*

*s = null;*

*continue;*

*}*

*// Sent the GET request to the host.*

*s.Send(ByteGet, ByteGet.Length, 0);*

*}*

*Int bytes = s.Receive(RecvBytes, RecvBytes.Length, 0);*

*strRetPage = ASCII.GetString(RecvBytes, 0, bytes);*

*while (bytes > 0)*

*{*

*bytes = s.Receive(RecvBytes, RecvBytes.Length, 0);*

*strRetPage = strRetPage + ASCII.GetString(RecvBytes, 0, bytes);*

*}*

*}*

*catch(Exception e)*

*{*

*Console.WriteLine("WRONG");*

*Console.WriteLine("Source : " + e.Source);*

*Console.WriteLine("Message : " + e.Message);*

*}*

*return strRetPage;*

*}*