## **Платформы для проектирования и реализации баз данных**

Современные системы анализа и проектирования могут быть разделены на две большие категории. Первую составляют CASE- системы (как независимые (upper CASE), так и интегрированные с СУБД), обе системы анализа и проектированияечивающие проектирование БД и приложений в комплексе с интегрированными средствами разработки приложений "клиент-сервер" (например, Westmount I-CASE+Uniface, Designer/2000+Developer/2000). Их основное достоинство заключается в том, что они позволяют разрабатывать всю ИС целиком (функциональные системы анализа и проектирования спецификации, логику процессов, интерфейс с пользователем и базу данных), оставаясь в одной технологической среде. Инструменты этой категории, как правило, обладают существенной сложностью, широкой сферой применения и высокой гибкостью.

Вторую категорию составляют собственно средства проектирования БД, реализующие ту или иную методологию, как правило, "сущность-связь" ("entity-relationship") и рассматриваемые в комплексе со средствами разработки приложений. К средствам этой категории можно отнести такие, как SILVERRUN+JAM, ERwin/ERX+PowerBuilder и др.

Помимо указанных категорий, системы анализа и проектирования можно классифицировать по следующим признакам:

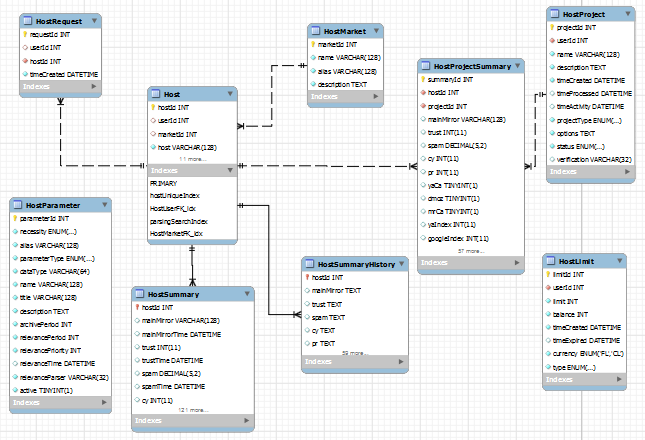
* степени интегрированности: (отдельные локальные средства, набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС и полностью интегрированные средства, связанные общей базой проектных данных - репозиторием);
* применяемым методологиям и моделям систем и БД;
* степени интегрированности с СУБД;
* степени открытости;
* доступным платформам.

В разряд системы анализа и проектирования попадают как относительно дешевые системы для персональных компьютеров (ПК) с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-систем, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.  
Применение системы анализа и проектирования требует от потенциальных пользователей системы анализа и проектирования специальной подготовки и обучения. Опыт показывает, что внедрение системы анализа и проектирования осуществляется медленно, однако по мере приобретения практических навыков и общей культуры проектирования эффективность применения этих средств резко возрастает, причем наибольшая потребность в использовании систем проектирования испытывается на начальных этапах разработки, а именно на этапах анализа и спецификации требований. Это объясняется тем, что цена ошибок, допущенных на начальных этапах, на несколько порядков превышает цену ошибок, выявленных на более поздних этапах разработки.



Практическое задание

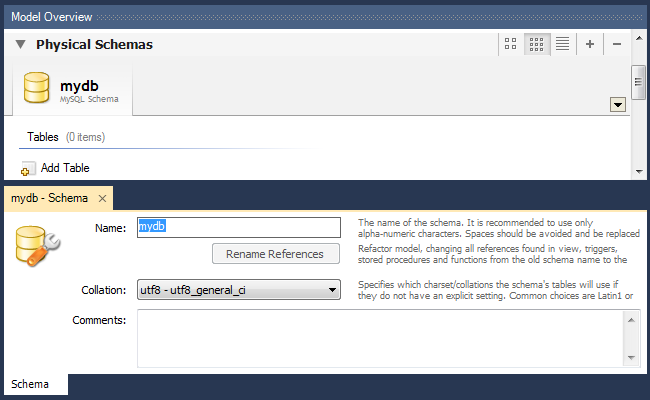
**MySQL Workbench** — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.



Программа позволяет быстро и с удовольствием накидывать **схемы данных проекта**, проектировать **сущности и связи** между ними, безболезненно **внедрять изменения** в схему и так же быстро и безболезненно **синхронизировать** её с удалённым сервером. А графический редактор **EER-диаграмм** позволяет увидеть общую картину модели данных.

## *Создание и редактирование модели данных*

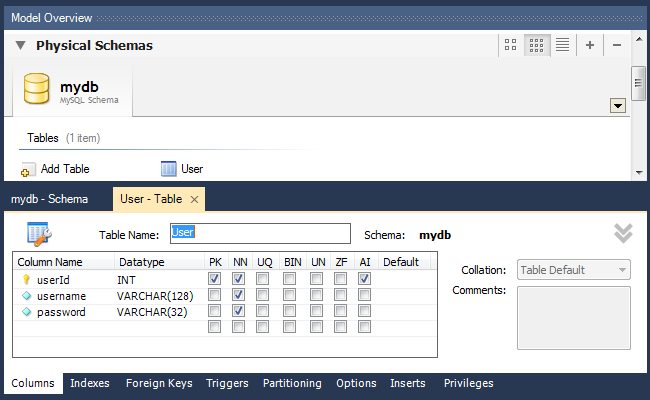
Для добавления модели нажимаем "Models" или выбираем *"File → New Model" (Ctrl + N):*



На этом экране вводим имя базы данных, выбираем кодировку по умолчанию и, если нужно, заполняем поле комментария. Можно приступать к созданию таблиц.

### *Добавление и редактирование таблицы*

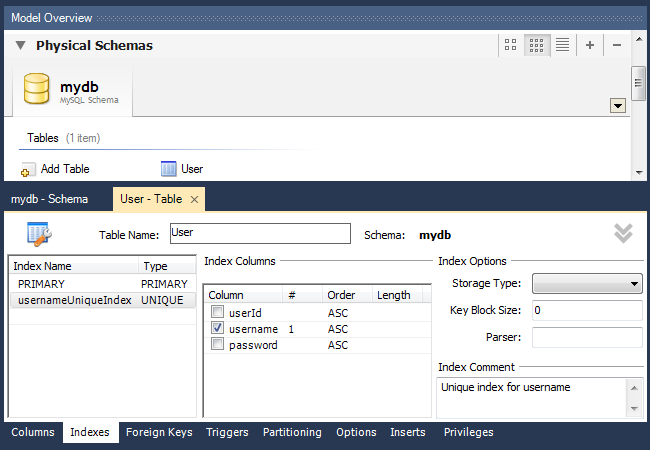
Список баз данных проекта и список таблиц в пределах базы данных будет располагаться во вкладке *"Physical Schemas".*Чтобы создать таблицу, дважды кликаем на *"+Add Table":*



Откроется удобный интерфейс для редактирования списка полей и их свойств. Здесь мы можем задать название поля, тип данных, а так же установить для полей различные атрибуты: назначить поле *первичным ключом (PK)*, пометить его *Not Null (NN)*, *бинарным (BIN)*, *уникальным (UQ)* и другие, установить для поля *авто-инкремирование (AI)* и *значение по умолчанию (Default)*.

### *Управление индексами*

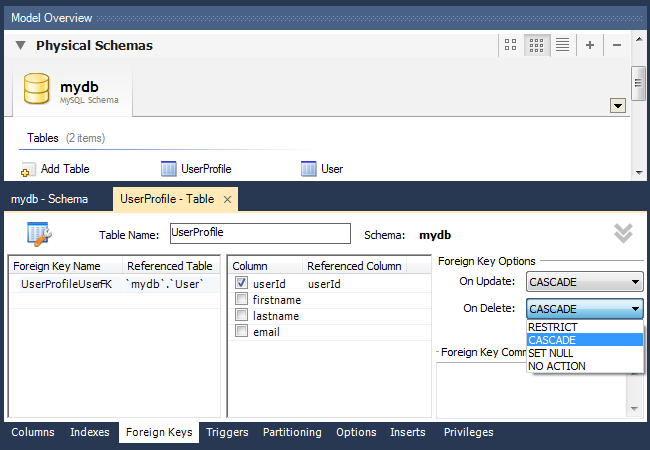
Добавлять, удалять и редактировать индексы таблиц можно во вкладке *"Indexes"* интерфейса управления таблицей:



Вводим название индекса, выбираем его тип, затем галочками помечаем в нужном порядке список полей, участвующих в данном индексе. Порядок полей будет соответствовать порядку, в котором были проставлены галочки. В данном примере добавлен уникальный индекс к полю *username*.

### *Связи между таблицами*

Установка внешних ключей и связывание таблиц возможно только для таблиц *InnoDB* (эта система хранения данных выбирается по умолчанию). Для управления связями в каждой таблице находится вкладка *"Foreign Keys":*



Для добавления связи открываем вкладку *"Foreign Keys"* **дочерней таблицы**, вводим имя внешнего ключа и выбираем **таблицу-родителя**. Далее в средней части вкладки в графе *Column* выбираем поле-ключ из дочерней таблицы, а в графе *Referenced Column* - соответствующее поле из родительской таблицы (тип полей должен совпадать). При создании внешних ключей **в дочерней таблице автоматически создаются соответствующие индексы**.

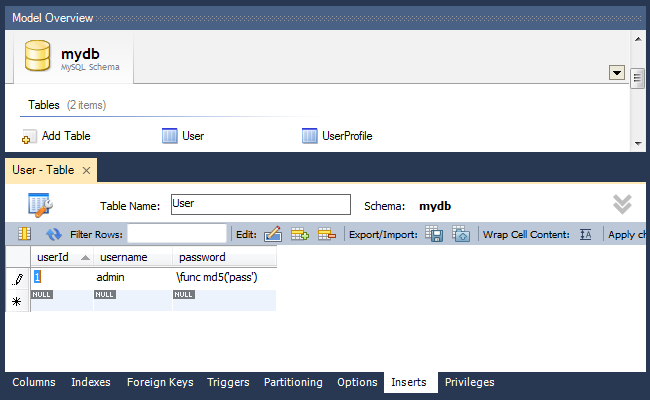
В разделе *"Foreign Key Options"* настраиваем поведение внешнего ключа при изменении соответствующего поля *(ON UPDATE)* и удалении *(ON DELETE)* родительской записи:

* *RESTRICT* - выдавать ошибку при изменении / удалении родительской записи
* *CASCADE* - обновлять внешний ключ при изменении родительской записи, удалять дочернюю запись при удалении родителя
* *SET NULL* - устанавливать значение внешнего ключа *NULL* при изменении / удалении родителя **(неприемлемо для полей, у которых установлен флаг***NOT NULL!***)**
* *NO ACTION* - не делать ничего, однако по факту эффект аналогичен RESTRICT

В приведённом примере добавили к дочерней таблице *UserProfile* внешний ключ для связи с родительской таблицей *User*. При редактировании поля *userId* и удалении позиций из таблицы *User* аналогичные изменения будут **автоматически** происходить и со связанными записями из таблицы *UserProfile*.

### *Наполнение таблицы базовыми данными*

При создании проекта в базу данных часто нужно добавлять стартовые данные. Это могут быть корневые категории, пользователи-администраторы и т.д. В управлении таблицами MySQL Workbench для этого существует вкладка *"Inserts":*

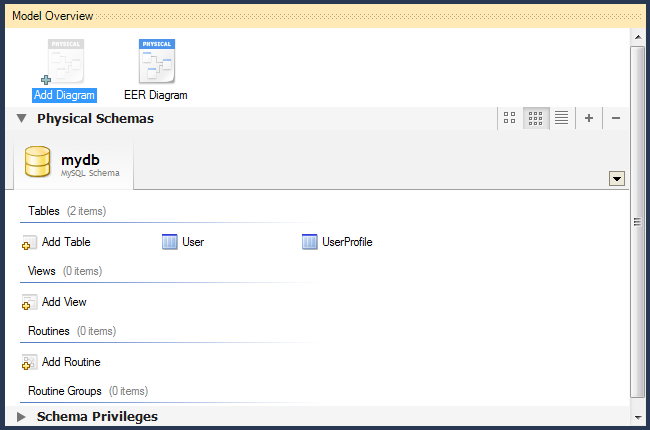


Как видно из примера, в случае, если перед записью в базу данных к данным нужно применить какую-то функцию MySQL, это делается с помощью синтаксиса *\func functionName('data')*, например, *\func md5('password')*.

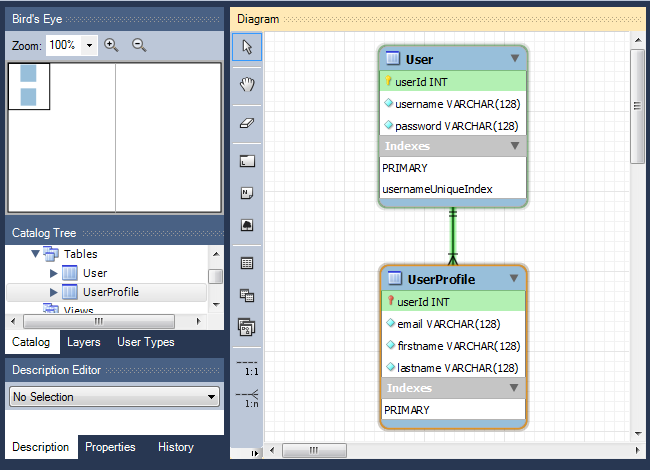
После ввода данных необходимо сохранить их в локальную базу данных нажатием на кнопку *"Apply Changes".*

## *Создание ER диаграммы (диаграммы "сущность-связь")*

Для представления схемы данных, сущностей и их связей в графическом виде в MySQL Workbench существует редактор EER-диаграмм. Для создания диаграммы в верхней части экрана управления базой данных дважды кликаем на иконку *"+Add Diagram":*



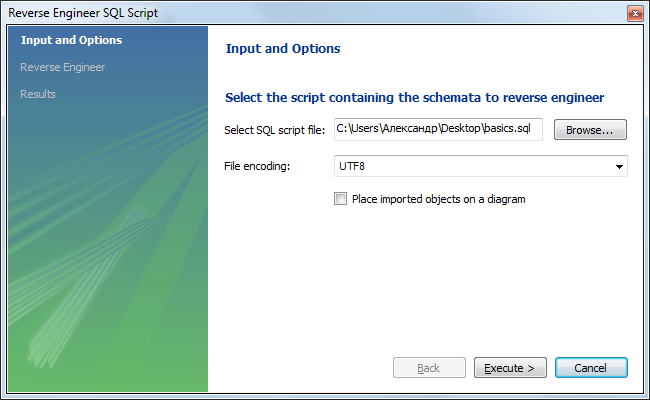
В его интерфейсе можно создавать и редактировать таблицы, добавлять между ними связи различных типов. Чтобы добавить уже существующую в схеме таблицу на диаграмму, просто перетащите её из панели *"Catalog Tree"*.



Для экспорта схемы данных в графический файл выберите *"File → Export"*, а затем один из вариантов *(PNG, SVG, PDF, PostScript File)*.

## *Импорт существующей схемы данных (из SQL дампа)*

Если уже есть схема данных, её можно без труда импортировать в MySQL Workbench для дальнейшей работы. Для импорта модели из SQL файла выбираем *"File → Import → Reverse Engineer MySQL Create Script..."*, после чего выбираем нужный SQL файл и жмём *"Execute >"*



В MySQL Workbench так же предусмотрен импорт и синхронизация модели данных нарямую с удалённым сервером.

Работа с MS SQL Server

Создание базы данных

Базу данных часто отождествляют с набором таблиц, которые хранят данные. Но это не совсем так. Лучше сказать, что база данных представляет хранилище объектов.

Основные из них:

* Таблицы: хранят собственно данные
* Представления (Views): выражения языка SQL, которые возвращают набор данных в виде таблицы
* Хранимые процедуры: выполняют код на языке SQL по отношению к данным к БД (например, получает данные или изменяет их)
* Функции: также код SQL, который выполняет определенную задачу

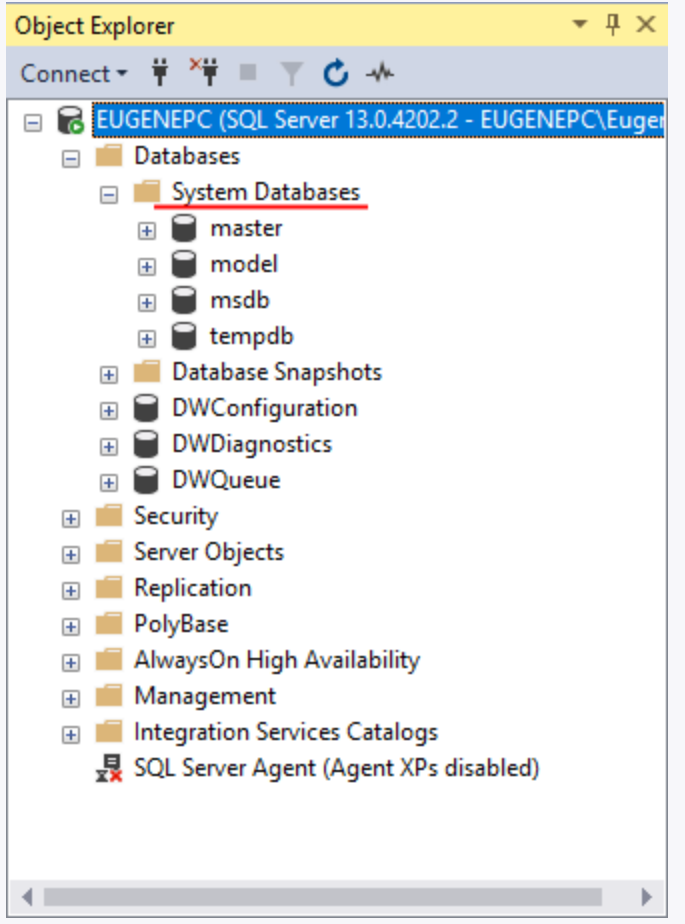
В SQL Server используется два типа баз данных: системные и пользовательские. Системные базы данных необходимы серверу SQL для корректной работы. А пользовательские базы данных создаются пользователями сервера и могут хранить любую произвольную информацию. Их можно изменять и удалять, создавать заново.

Системные базы данных

В MS SQL Server по умолчанию создается четыре системных баз данных:

* master: эта главная база данных сервера, в случае ее отсутствия или повреждения сервер не сможет работать. Она хранит все используемые логины пользователей сервера, их роли, различные конфигурационные настройки, имена и информацию о базах данных, которые хранятся на сервере, а также ряд другой информации.
* model: эта база данных представляет шаблон, на основе которого создаются другие базы данных. То есть когда мы создаем через SSMS свою БД, она создается как копия базы model.
* msdb: хранит информацию о работе, выполняемой таким компонентом как планировщик SQL. Также она хранит информацию о бэкапах баз данных.
* tempdb: эта база данных используется как хранилище для временных объектов. Она заново пересоздается при каждом запуске сервера.

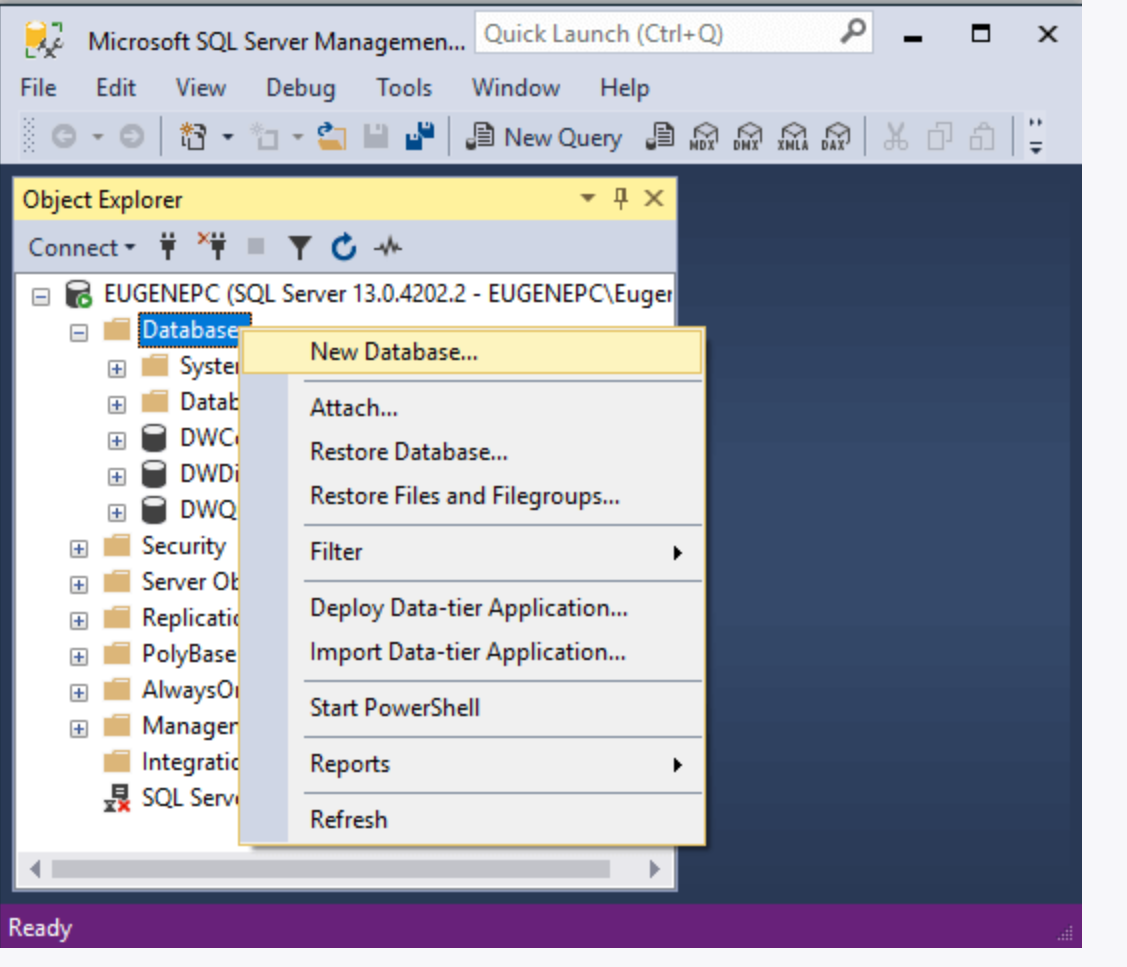
Все эти базы можно увидеть через SQL Server Management Studio в узле Databases -> System Databases:



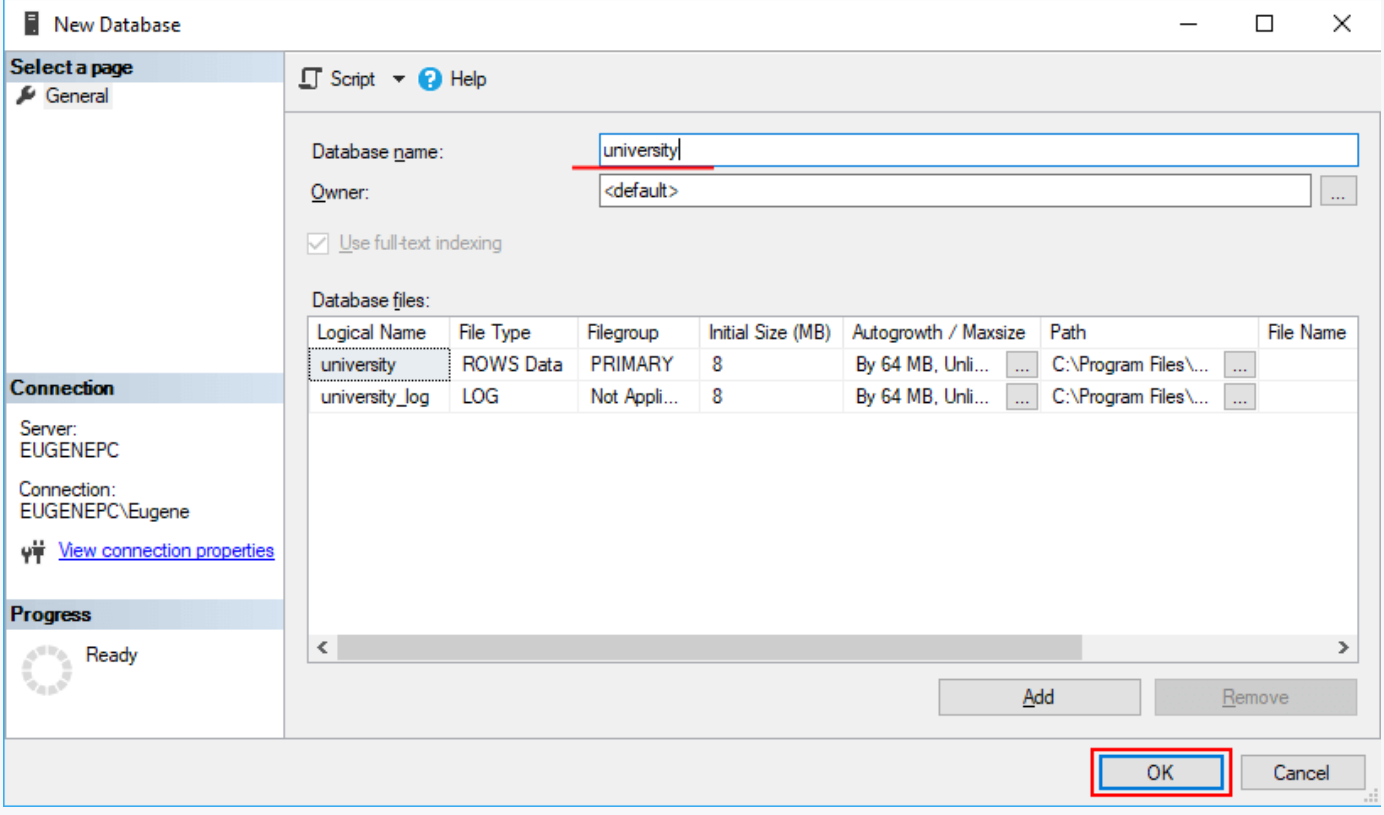
Эти базы данных не следует изменять.

Создание базы данных в SQL Management Studio

Теперь создадим свою базу данных. Для этого мы можем использовать скрипт на языке SQL, либо все сделать с помощью графических средств в SQL Management Studio. В данном случае мы выберем второй способ. Для этого откроем SQL Server Management Studio и нажмем правой кнопкой мыши на узел Databases. Затем в появившемся контекстном меню выберем пункт New Database:



После этого нам открывается окно для создания базы данных:



В поле Database необходимо ввести название новой БД. Пусть у нас база данных называется university.

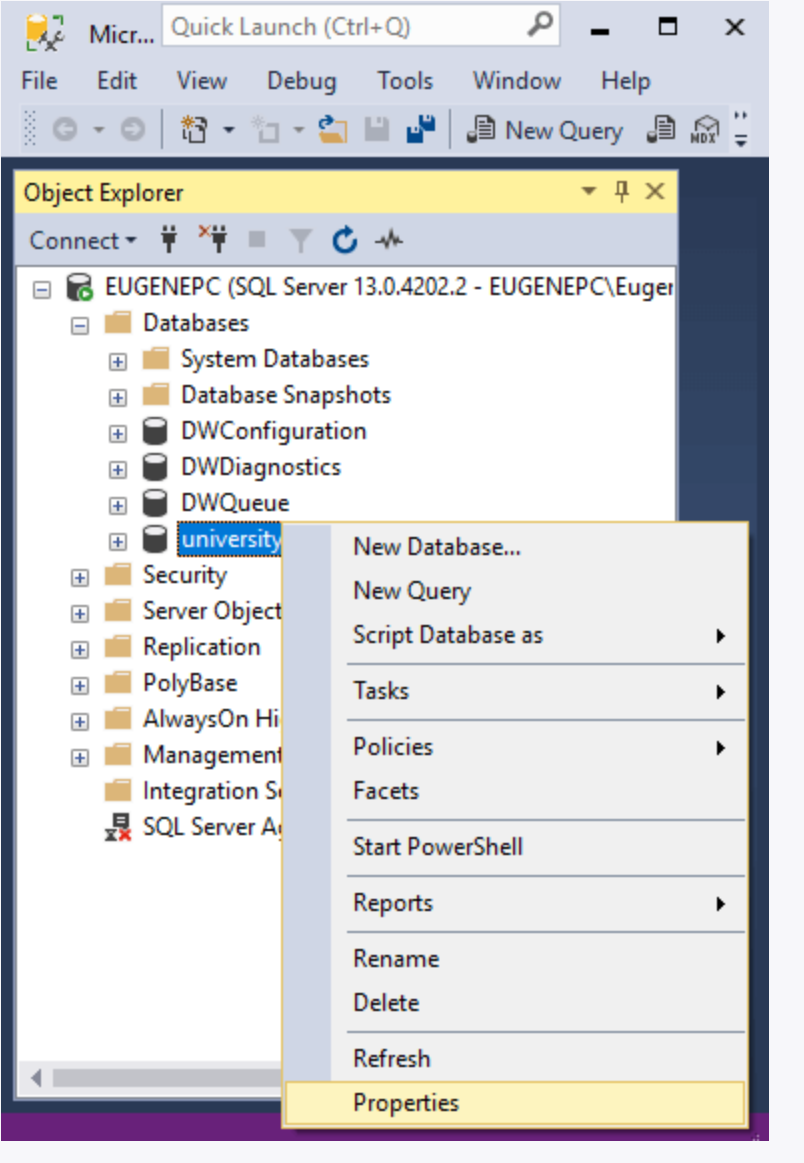
Следующее поле Owner задает владельца базы данных. По умолчанию оно имеет значение <defult>, то есть владельцем будет тот, кто создает эту базу данных. Оставим это поле без изменений.

Далее идет таблица для установки общих настроек базы данных. Она содержит две строки - первая для установки настроек для главного файла, где будут храниться данные, и вторая строка для конфигурации файла логгирования. В частности, мы можем установить следующие настройки:

* Logical Name: логическое имя, которое присваивается файлу базы данных.
* File Type: есть несколько типов файлов, но, как правило, основная работа ведется с файлами данных (ROWS Data) и файлом лога (LOG)
* Filegroup: обозначет группу файлов. Группа файлов может хранить множество файлов и может использоваться для разбиения базы данных на части для размещения в разных местах.
* Initial Size (MB): устанавливает начальный размер файлов при создании (фактический размер может отличаться от этого значения).
* Autogrowth/Maxsize: при достижении базой данных начального размера SQL Server использует это значение для увеличения файла.
* Path: каталог, где будут храниться базы данных.
* File Name: непосредственное имя физического файла. Если оно не указано, то применяется логическое имя.

После ввода названия базы данных нажмем на кнопку ОК, и БД будет создана.

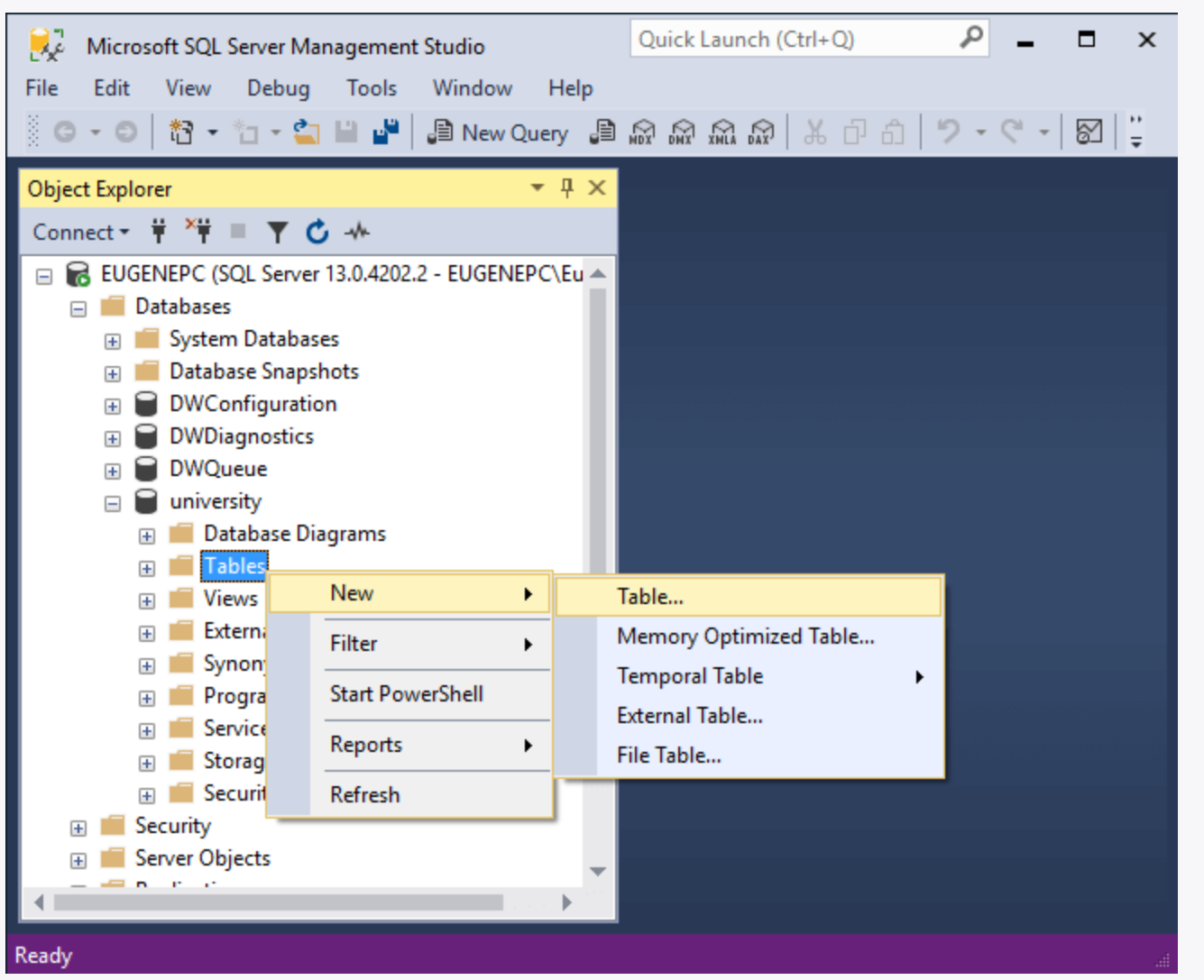
После этого она появится среди баз данных сервера. Если эта БД впоследствии не потребуется, то ее можно удалить, нажав на нее правой кнопкой мыши и выбрав в контекстном меню пункт Delete:



Ключевым объектом в базе данных являются таблицы. Таблицы состоят из строк и столбцов. Столбцы определяют тип информации, которая хранится, а строки содержат значения для этих столбцов.

Нами была создана база данных university. Теперь определим в ней первую таблицу. Опять же для создания таблицы в SQL Server Management Studio можно применить скрипт на языке SQL, либо воспользоваться графическим дизайнером. В данном случае выберем второе.

Для этого раскроем узел базы данных university в SQL Server Management Studio, нажмем на его подузел Tables правой кнопкой мыши и далее в контекстном меню выберем New -> Table

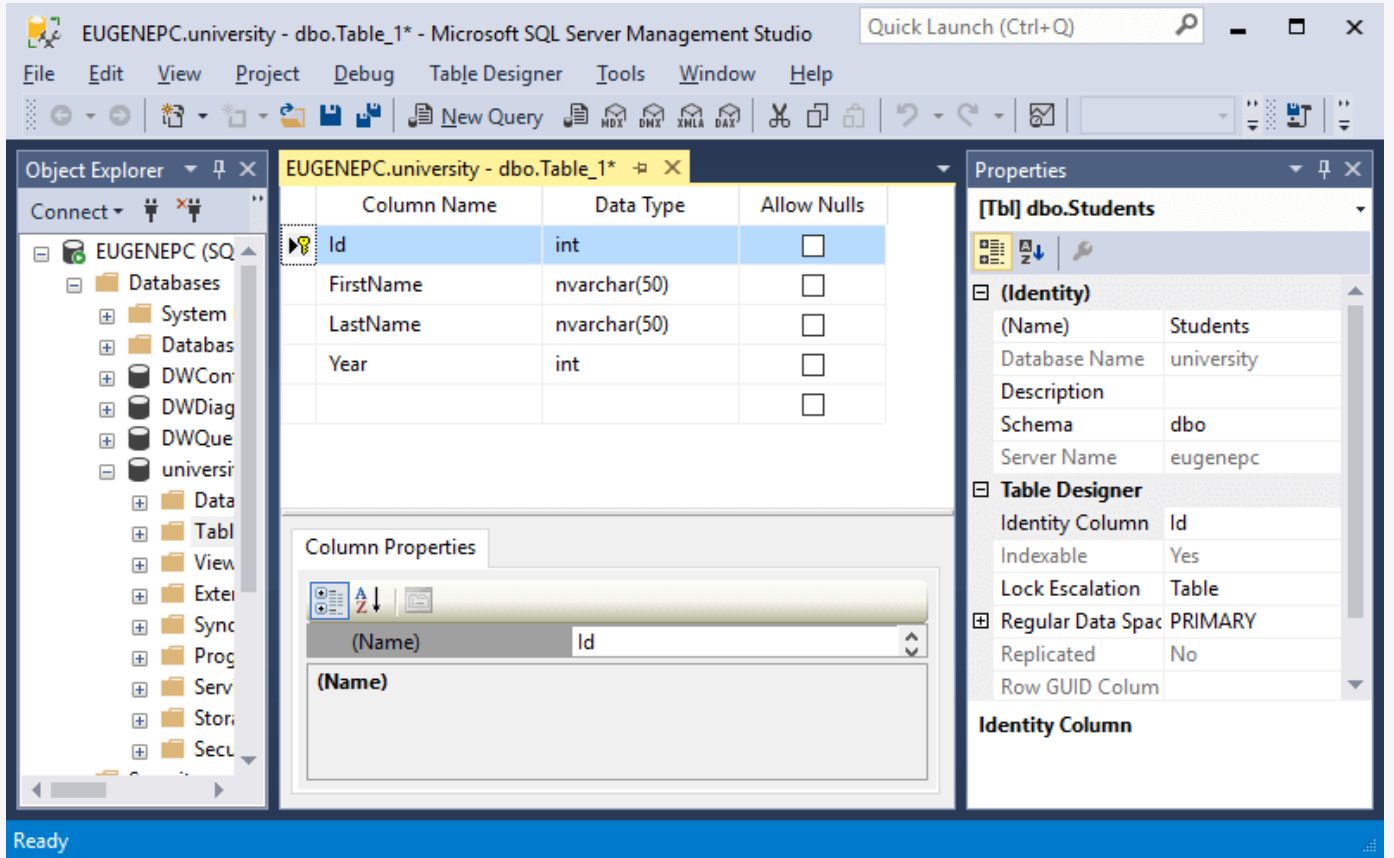


После этого нам откроется дизайнер таблицы. В центральной части в таблице необходимо ввести данные о столбцах таблицы. Дизайнер содержит три поля:

Column Name: имя столбца

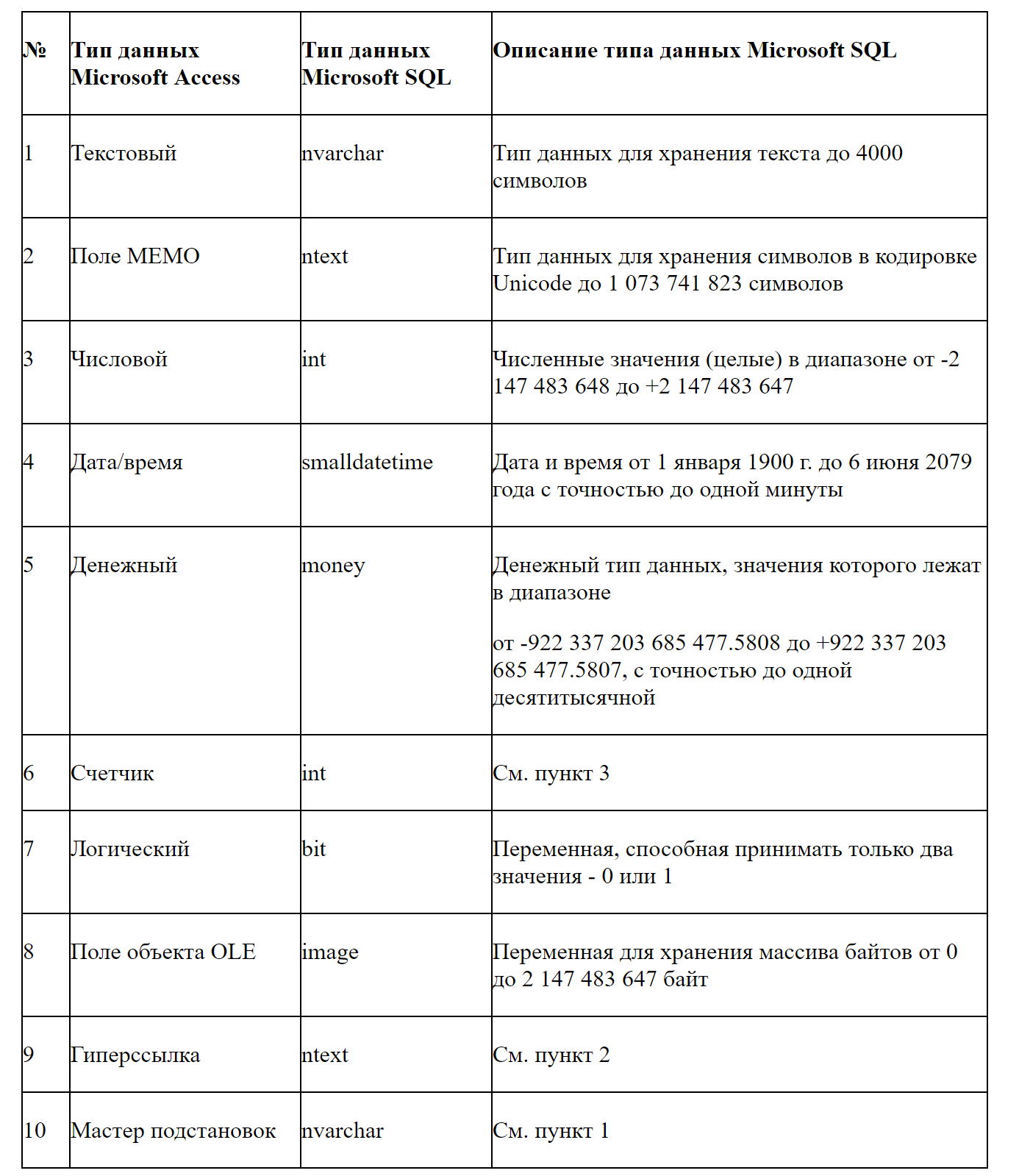
Data Type: тип данных столбца. Тип данных определяет, какие данные могут храниться в этом столбце. Например, если столбец представляет числовой тип, то он может хранить только числа.

Allow Nulls: может ли отсутствовать значение у столбца, то есть может ли он быть пустым



Допустим, нам надо создать таблицу с данными учащихся в учебном заведении. Для этого в дизайнере таблицы четыре столбца: Id, FirstName, LastName и Year, которые будут представлять соответственно уникальный идентификатор пользователя, его имя, фамилию и год рождения. У первого и четвертого столбца надо указать тип int (то есть целочисленный), а у столбцов FirstName и LastName - тип nvarchar(50) (строковый).

Соответствие типов данных Microsoft Access и Microsoft SQL

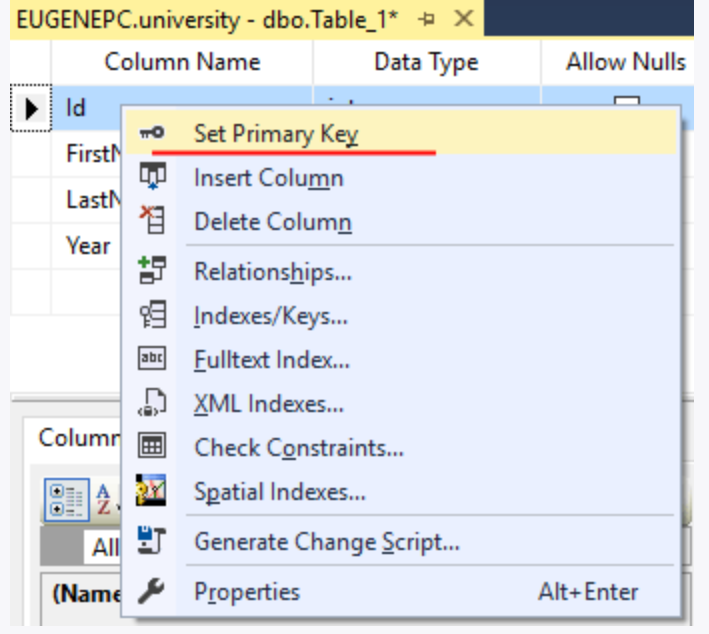


Затем в окне Properties, которая содержит свойства таблицы, в поле Name надо ввести имя таблицы - Students, а в поле Identity ввести Id, то есть тем самым указывая, что столбец Id будет идентификатором.

Имя таблицы должно быть уникальным в рамках базы данных. Как правило, название таблицы отражает название сущности, которая в ней хранится. Например, мы хотим сохранить студентов, поэтому таблица называется Students (слово студент во множественном числе на английском языке). Существуют разные мнения по поводу того, стоит использовать название сущности в единственном или множественном числе (Student или Students). В данном случае вопрос наименования таблицы всецело ложится на разработчика базы данных.

И в конце нам надо отметить, что столбец Id будет выполнять роль первичного ключа (primary key). Первичный ключ уникально идентифицирует каждую строку. В роли первичного ключа может выступать один столбец, а может и несколько.

Для установки первичного ключа нажмем на столбец Id правой кнопкой мыши и в появившемся меню выберем пункт Set Primary Key.

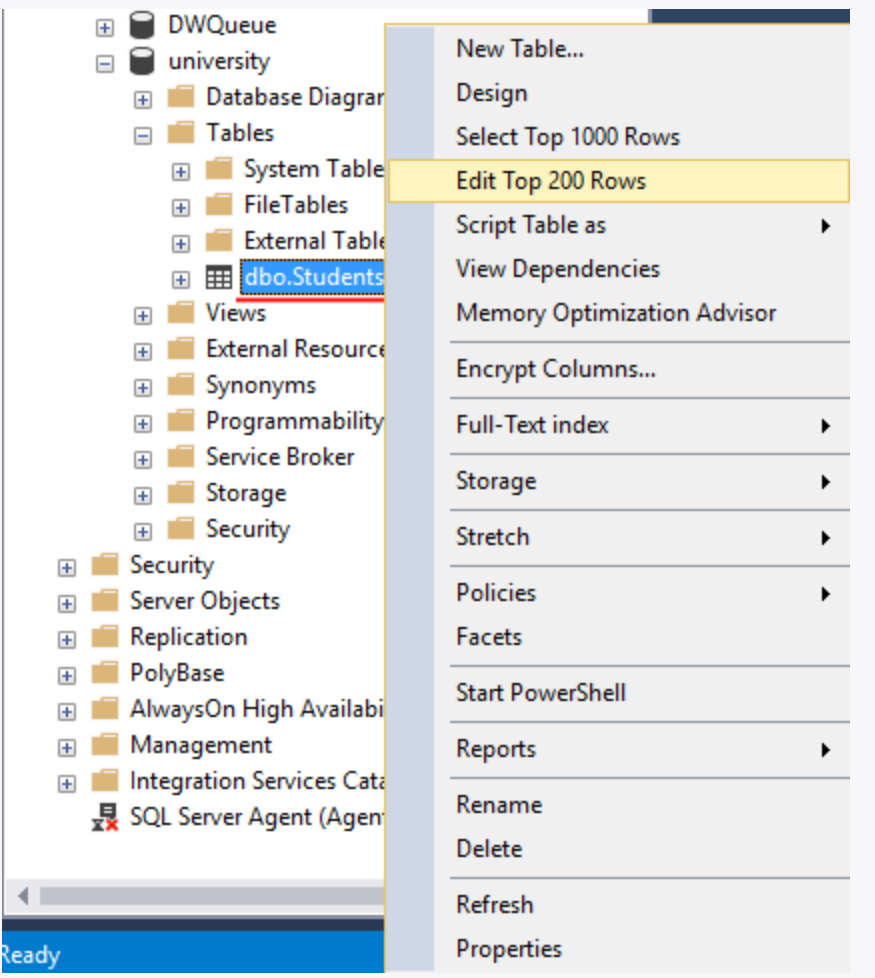


После этого напротив поля Id должен появиться золотой ключик. Этот ключик будет указывать, что столбец Id будет выполнять роль первичного ключа.

И после сохранения в базе данных university появится таблица Students:

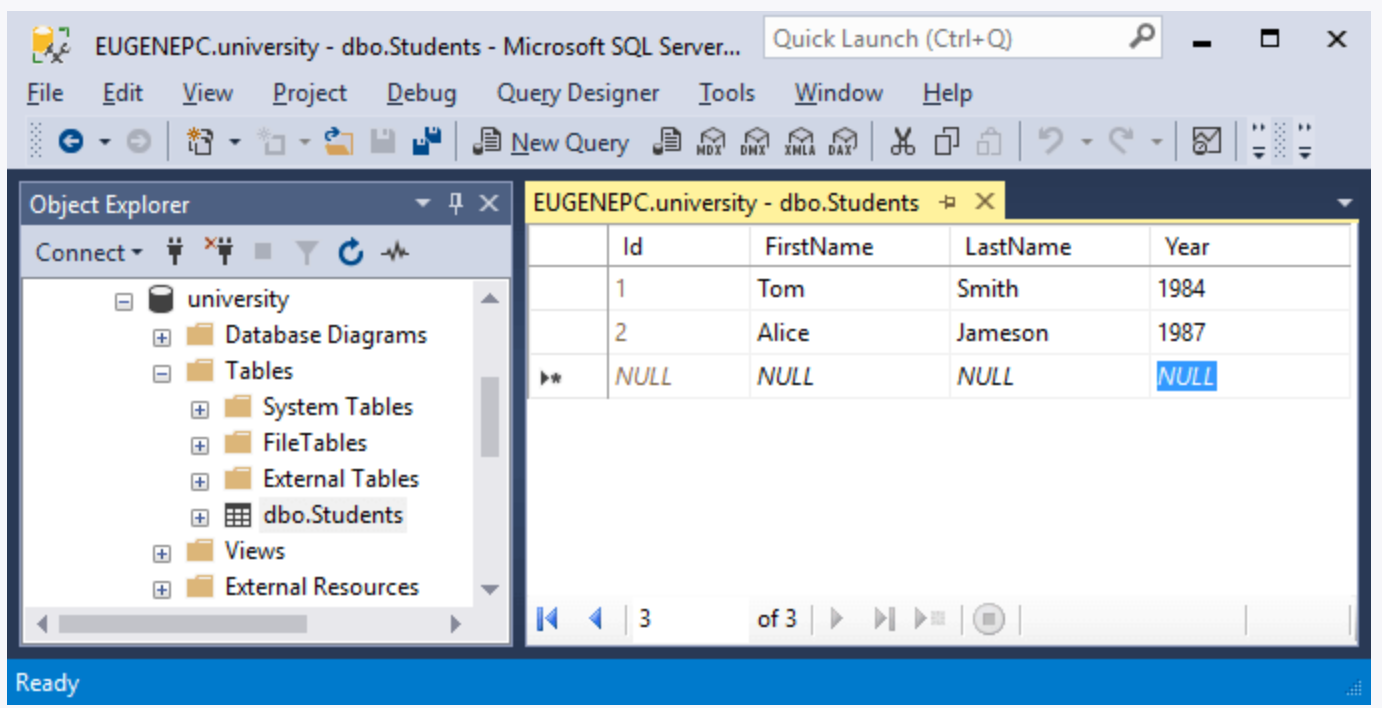
Мы можем заметить, что название таблицы на самом деле начинается с префикса dbo. Этот префикс представляет схему. Схема определяет контейнер, который хранит объекты. То есть схема логически разграничивает базы данных. Если схема явным образом не указывается при создании объекта, то объект принадлежит схеме по умолчанию - схеме dbo.

Нажмем правой кнопкой мыши на название таблицы, и нам отобразится контекстное меню с опциями:



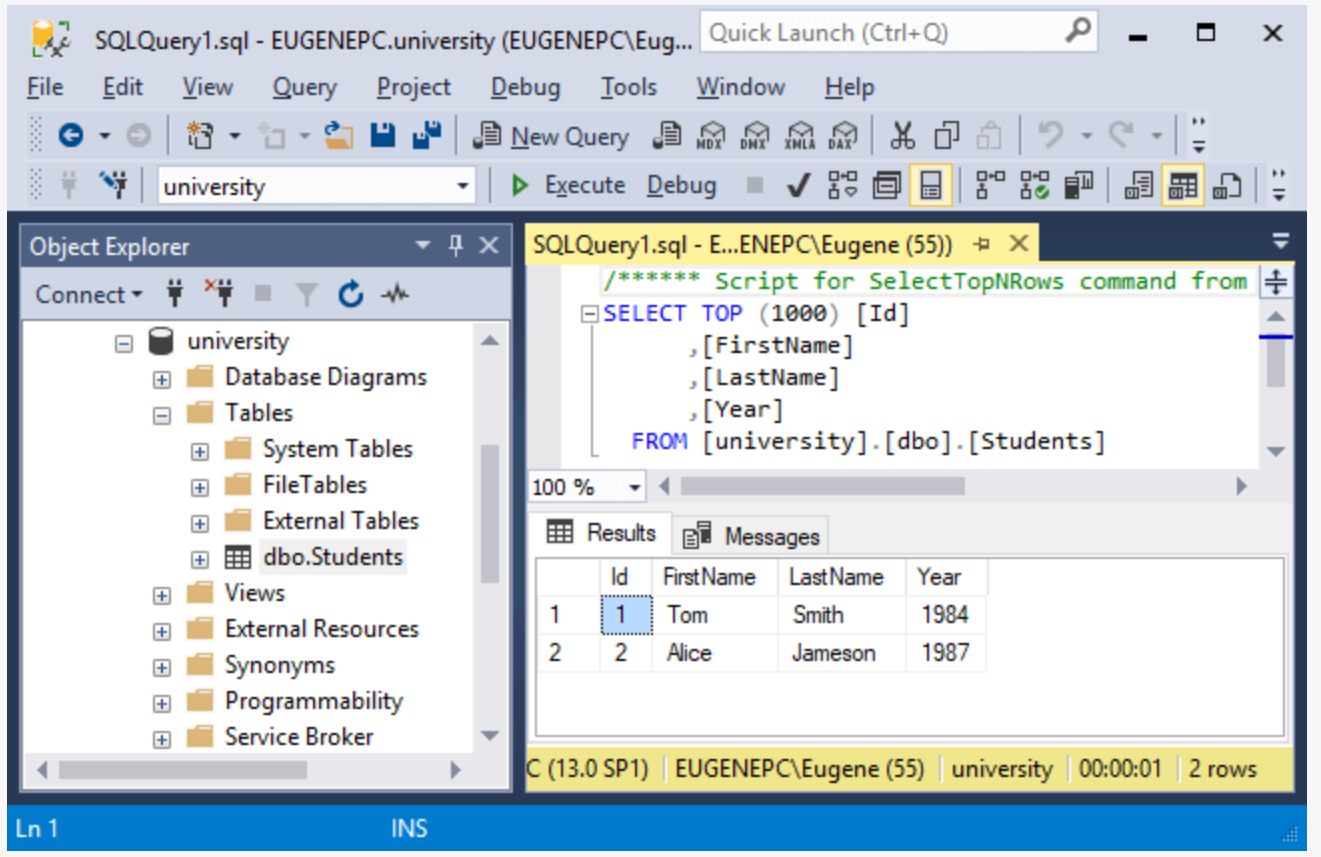
С помощью этих опций можно управлять таблицей. Так, опция Delete позволяет удалить таблицу. Опция Design откроет окно дизайнера таблицы, где мы можем при необходимости внести изменения в ее структуру.

Для добавления начальных данных можно выбрать опцию Edit Top 200 Rows. Она открывает в виде таблицы 200 первых строк и позволяет их изменить. Но так как у нас таблица только создана, то естественно в ней будет никаких данных. Введем пару строк - пару студентов, указав необходимые данные для столбцов:

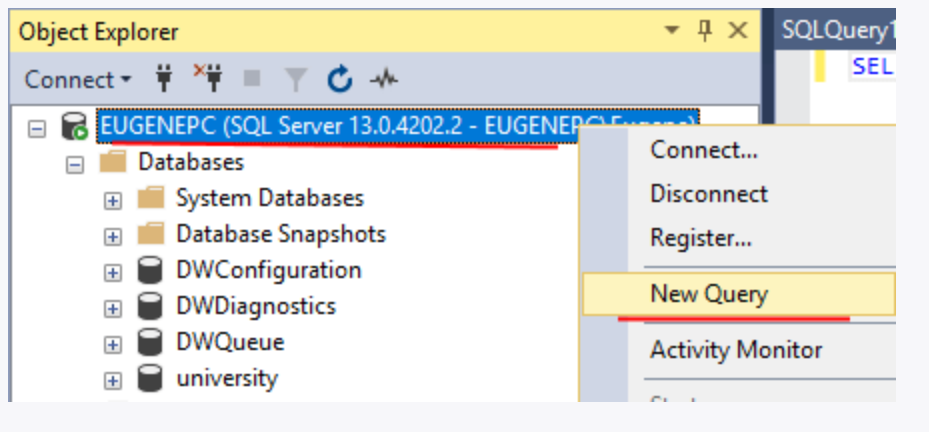


В данном случае мы добавили две строки.

Затем опять же по клику на таблицу правой кнопкой мыши мы можем выбрать в контекстном меню пункт Select To 1000 Rows, и будет запущен скрипт, который отобразит первые 1000 строк из таблицы:



Теперь определим и выполним первый SQL-запрос. Для этого откроем SQL Management Studio, нажмем правой кнопкой мыши на элемент самого верхнего уровня в Object Explorer (название сервера) и в появившемся контекстном меню выберем пункт New Query:

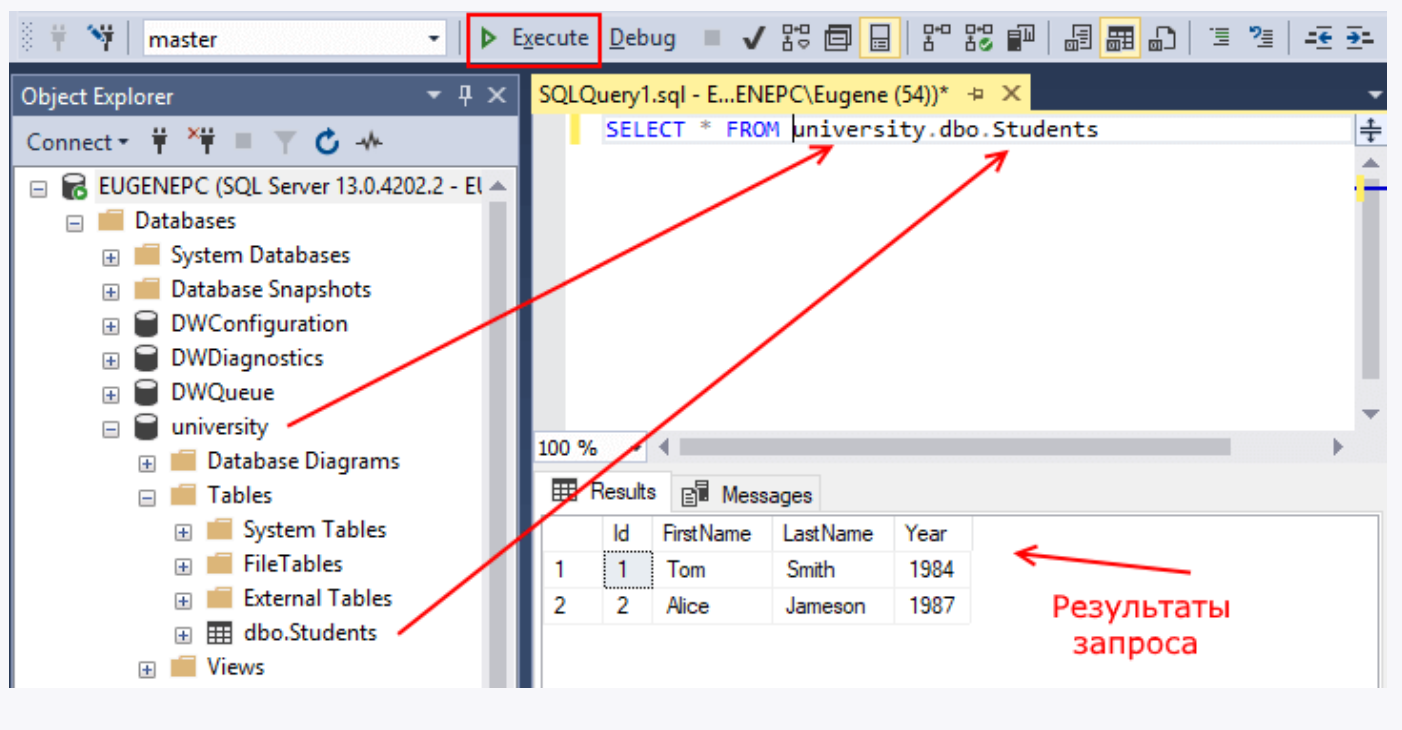


После этого в центральной части программы откроется окно для ввода команд языка SQL.

Выполним запрос к таблице, которая была создана в прошлой теме, в частности, получим все данные из нее. База данных у нас называется university, а таблица - dbo.Students, поэтому для получения данных из таблицы введем следующий запрос:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM university.dbo.Students |

Оператор SELECT позволяет выбирать данные. FROM указывает источник, откуда брать данные. Фактически этим запросом мы говорим "ВЫБРАТЬ все ИЗ таблицы university.dbo.Students". Стоит отметить, что для названия таблицы используется полный ее путь с указанием базы данных и схемы.



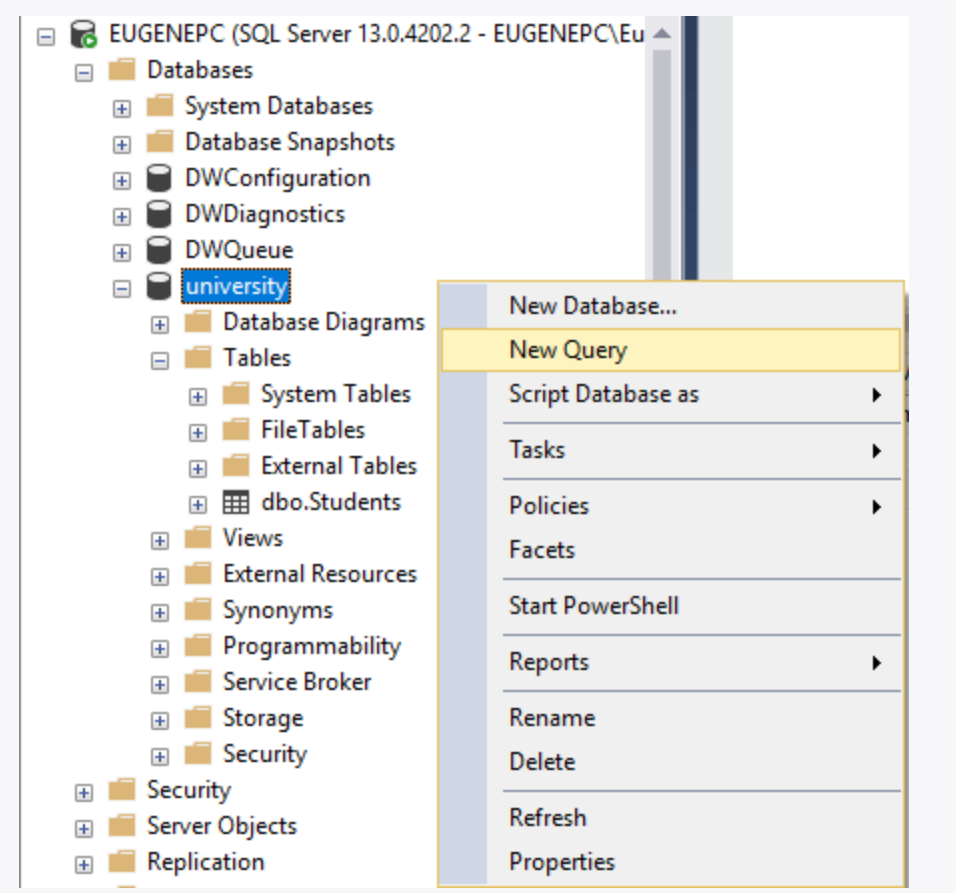
После ввода запроса нажмем на панели инструментов на кнопку Execute, либо можно нажать на клавишу F5.

В результате выполнения запроса в нижней части программы появится небольшая таблица, которая отобразит результаты запроса - то есть все данные из таблицы Students.

Если необходимо совершить несколько запросов к одной и той же базе данных, то мы можем использовать команду USE, чтобы зафиксировать базу данных. В этом случае при запросах к таблицам достаточно указать их имя без имени бд и схемы:

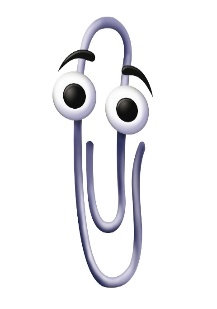
|  |  |
| --- | --- |
| 1  2 | USE university  SELECT \* FROM Students |

В данном случае мы выполняем запрос в целом для сервера, мы можем обратиться к любой базе данных на сервере. Но также мы можем выполнять запросы только в рамках конкретной базы данных. Для этого необходимо нажать правой кнопкой мыши на нужную БД и в контекстном меню выбрать пункт New Query:



Если в этом случае мы захотим выполнить запрос к выше использованной таблице Students, то нам не пришлось бы указывать в запросе название базы данных и схему, так как эти значения итак уже были бы понятны:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM Students |



Для выполнения практического задания можно ознакомиться с обучающими видео



Задание:

1. разработать ER-диаграмму к информационной системе на основании описания бизнес-процесса (<https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-lkn>);
2. разработать элемент рабочей тетради по теме “UML-диаграммы’’(действующие лица, варианты использования, инструментальные средства и т.д.) .

# МОДУЛЬ КОМПЕТЕНЦИИ 2: «РАЗРАБОТКА ОКОННЫХ ПРИЛОЖЕНИЙ»

## Платформы для разработки программных решений

**Компьютерная платфо́рма** — в общем смысле, это любая существующая [среда выполнения](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F), в которой должен выполняться вновь разрабатываемый фрагмент [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) или [объектный модуль](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C) с учётом накладываемых этой средой ограничений и предоставляемых возможностей. Термин платформа может применяться к разным уровням абстракции, включая определенную [аппаратную архитектуру](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0), [операционную систему](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) или [библиотеку времени выполнения](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F).

Нижний слой многоуровневой организации вычислительной системы (аппаратура, операционная система, прикладное программное обеспечение), на который опираются ОС и прикладное ПО. Аппаратные платформы отличаются друг от друга [архитектурой центрального процессора](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0) и используемыми [шинами](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D0%BD%D0%B0_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)) связи функциональных блоков.

Каждой аппаратной платформе соответствуют [совместимые](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) с ней [операционные системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [прикладные программы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), которые могут на ней запускаться.

Программная платформа представляет собой общую организацию исполнения прикладных программ, задавая, например, порядок запуска программы, схему использования ею адресного пространства, зафиксированные в [архитектуре](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) плюс [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) на уровне операционной системы.

При рассмотрении [совместимости](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), или сходства, на уровне операционных систем, например, [системных вызовов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2), [файловых систем](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [пользовательской среды](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), при сравнении родственных операционных систем (например, [UNIX](https://ru.wikipedia.org/wiki/UNIX)) или семейства (например, [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows)), речь идет о совместимости на уровне API операционной системы, например, в рамках семейства ОС, а не абстрактного понятия «платформы»

**Примеры платформ ОС**

* [Win32](https://ru.wikipedia.org/wiki/Win32) — Win32 API,
* API [POSIX](https://ru.wikipedia.org/wiki/POSIX) для ОС UNIX/Linux.

Кроссплатформенность программного обеспечения — возможность исполнять его, без перекомпилирования программы, как на различных аппаратных платформах, так и под управлением разных операционных систем (иначе говоря, возможность запуска исполняемого файла на платформах различных ОС).

Примерами программного обеспечения, выполняющегося на разных аппаратных платформах и под управлением разных операционных систем, являются разнообразные программы, написанные на [языках программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) для [виртуальных машин](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0), таких, как, например, PHP, Perl, Python, Java, и многие другие, а также — кроссплатформенные [среды разработки приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8).

**Примеры**

* [Qt](https://ru.wikipedia.org/wiki/Qt)
* [GTK](https://ru.wikipedia.org/wiki/GTK)
* [Boost](https://ru.wikipedia.org/wiki/Boost)
* [Java Virtual Machine](https://ru.wikipedia.org/wiki/Java_Virtual_Machine)
* [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework)
* [Adobe AIR](https://ru.wikipedia.org/wiki/Adobe_AIR)

В рамках компетенции «Программные решения для бизнеса» для разработки используются платформы [Java или](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) .NET. В рамках .NET чаще всего используют интерфейсы программирования Windows Forms или WPF.

Windows Forms — [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) (API), отвечающий за [графический интерфейс пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) и являющийся частью [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). Данный интерфейс упрощает доступ к элементам интерфейса [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) за счет создания обёртки для существующего [Win32 API](https://ru.wikipedia.org/wiki/Windows_API)в [управляемом коде](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Причём управляемый код — классы, реализующие [API](https://ru.wikipedia.org/wiki/API) для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке [MFC](https://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes), изначально написанной на языке [C++](https://ru.wikipedia.org/wiki/C%2B%2B). С другой стороны, WF не предлагает парадигму, сравнимую с [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller). Для исправления этой ситуации и реализации данной функциональности в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является [User Interface Process Application Block](http://www.microsoft.com/downloads/details.aspx?familyid=98C6CC9D-88E1-4490-8BD6-78092A0F084E&displaylang=en), выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Внутри .NET Framework, Windows Forms реализуется в рамках [пространства имён](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE_%D0%B8%D0%BC%D1%91%D0%BD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) System.Windows.Forms.

Приложение Windows Forms представляет собой [событийно-ориентированное приложение](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), поддерживаемое Microsoft [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). В отличие от [пакетных программ](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0), большая часть времени тратится на ожидание от пользователя каких-либо действий, как, например, ввод текста в [текстовое поле](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5) или клика мышкой по [кнопке](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D0%BE%D0%BF%D0%BA%D0%B0_(%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)).

Windows Presentation Foundation (WPF) ― это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С помощью WPF можно создавать широкий спектр как автономных, так и размещенных в браузере приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений, в том числе Extensible Application Markup Language (XAML), элементами управления, привязкой данных, макетом, 2-D- и 3-D-графикой, анимацией, стилями, шаблонами, документами, мультимедиа, текстом и оформлением. WPF входит в состав Microsoft .NET Framework и позволяет создавать приложения, включающие другие элементы библиотеки классов .NET Framework.

Windows Presentation Foundation (WPF) ― это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С помощью WPF можно создавать широкий спектр как автономных, так и размещенных в браузере приложений. На следующем рисунке показан пример одного из таких приложений [Contoso Healthcare Sample Application](http://go.microsoft.com/fwlink/?LinkId=91196).

Программирование с использованием WPF

WPF существует в качестве подмножества типов .NET Framework, которые занимают большую часть в пространстве имен[System.Windows](https://msdn.microsoft.com/ru-ru/library/system.windows(v=vs.100).aspx). Пользователи, которые ранее создавали приложения с помощью .NET Framework, используя такие управляемые технологии, как ASP.NET и Windows Forms, должны быть знакомы с основами программирования WPF; создание экземпляров классов, задание свойств, вызов методов и обработка событий осуществляется с помощью одного из хорошо знакомых языков программирования .NET Framework, таких как C# или Visual Basic.

Для поддержки некоторых более мощных возможностей WPF и для упрощения процесса программирования WPF включает дополнительные программные конструкции, которые расширяют свойства и события: свойства зависимостей и перенаправленные события. Дополнительные сведения о свойствах зависимостей см. в разделе [Общие сведения о свойствах зависимости](https://msdn.microsoft.com/ru-ru/library/ms752914(v=vs.100).aspx). Дополнительные сведения о перенаправленных событиях см. в разделе [Общие сведения о перенаправленных событиях](https://msdn.microsoft.com/ru-ru/library/ms742806(v=vs.100).aspx).

Разметка и код программной части

В WPF дополнительно совершенствуется процесс программирования для разработки клиентских приложений Windows. Одним очевидным усовершенствованием является возможность разрабатывать приложения с помощью разметки и кода программной части, с которыми разработчики ASP.NET должны быть уже знакомы. Разметка Extensible Application Markup Language (XAML) обычно используется для реализации внешнего вида приложения при реализации его поведения с помощью управляемых языков программирования (кода программной части). Это разделение внешнего вида и поведения имеет следующие преимущества:

Затраты на разработку и обслуживание снижаются, так как разметка определенного внешнего вида тесно не связана с кодом определенного поведения.

Разработка более эффективна, так как разработчики, реализующие внешний вид приложения, могут это делать одновременно с разработчиками, реализующими поведение приложения.

Для реализации и совместного использования разметки XAML применяется множество средств конструирования, чтобы удовлетворить требованиям участников разработки приложений. [Microsoft Expression Blend](http://go.microsoft.com/fwlink/?LinkID=88924) предназначается для конструкторов, в то время как Visual Studio 2005 ориентируется на разработчиков.

Ниже приводится краткое описание разметки и кода программной части WPF. Дополнительные сведения об этой модели программирования см. в [Общие сведения о языке XAML (WPF)](https://msdn.microsoft.com/ru-ru/library/ms752059(v=vs.100).aspx) и в [Код программной части и XAML в WPF](https://msdn.microsoft.com/ru-ru/library/aa970568(v=vs.100).aspx).

Разметка

XAML ― это основанный на XML язык разметки, который используется для декларативной реализации внешнего вида приложения. Обычно он используется для создания окон, диалоговых окон, страниц и пользовательских элементов управления, а также для их заполнения элементами управления, фигурами и графикой.

Поскольку XAML основан на XML, UI, который формируется с его помощью, организуется в виде иерархии вложенных элементов, называемой деревом элементов. Дерево элементов предоставляет логический и интуитивно понятный способ для создания и управления UIs.

Код программной части

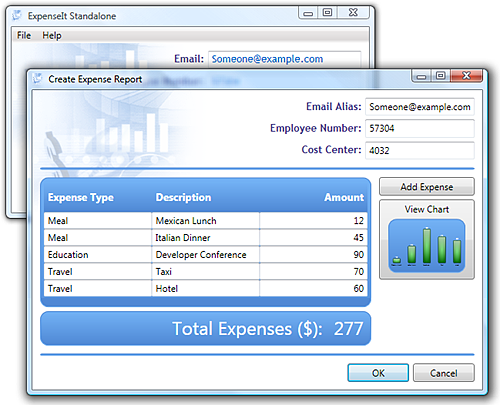
Приложение в основном предназначено для реализации функциональных возможностей, которые отвечают на взаимодействия с пользователем, включая обработку событий (например, нажатие меню, панели инструментов или кнопки) и вызов бизнес-логики и логики доступа к данным в ответ на события. В WPF такое поведение обычно реализуется в коде, который связан с разметкой. Этот тип кода называется кодом программной части.

Приложения

.NET Framework, [System.Windows](https://msdn.microsoft.com/ru-ru/library/system.windows(v=vs.100).aspx), разметка и выделенный код составляют основу разработки приложений WPF. Кроме того, WPF предоставляет полный набор средств для создания удобных и многофункциональных элементов пользовательского интерфейса. Чтобы упаковать разработанные элементы и предоставить их пользователю в виде приложений, WPF предоставляет типы и службы, вместе называемые моделью приложения. Модель приложения поддерживает разработку как автономных, так и размещенных в браузере приложений.

Автономные приложения

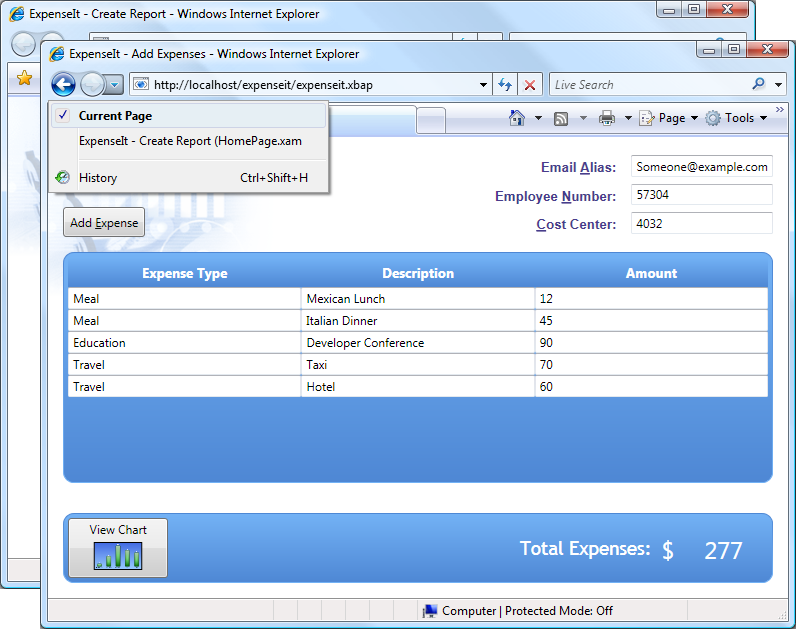
Для автономных приложений можно создавать доступные из меню и панелей инструментов окна и диалоговые окна с помощью класса [Window](https://msdn.microsoft.com/ru-ru/library/system.windows.window(v=vs.100).aspx). На следующем рисунке показано автономное приложение с главным и диалоговым окном.



Кроме того, можно использовать следующие диалоговые окна WPF: [MessageBox](https://msdn.microsoft.com/ru-ru/library/system.windows.messagebox(v=vs.100).aspx), [OpenFileDialog](https://msdn.microsoft.com/ru-ru/library/microsoft.win32.openfiledialog(v=vs.100).aspx), [SaveFileDialog](https://msdn.microsoft.com/ru-ru/library/microsoft.win32.savefiledialog(v=vs.100).aspx) и [PrintDialog](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.printdialog(v=vs.100).aspx).

Приложения, размещенные в браузере

Для приложений, размещаемых в браузере, также называемых XAML browser applications (XBAPs), можно создавать страницы ([Page](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.page(v=vs.100).aspx)) и страничные функции ([PageFunction<T>](https://msdn.microsoft.com/ru-ru/library/ms615540(v=vs.100).aspx)), по которым можно переходить с помощью гиперссылок (классы [Hyperlink](https://msdn.microsoft.com/ru-ru/library/system.windows.documents.hyperlink(v=vs.100).aspx)). На следующем рисунке показана страница в XBAP, размещенная в Internet Explorer 7.



Приложения WPF могут размещаться как в Microsoft Internet Explorer 6, так и в Internet Explorer 7. WPF предлагает два следующих параметра для альтернативных узлов переходов:

* [Frame](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.frame(v=vs.100).aspx) , чтобы размещать блоки содержимого для навигации в окнах или на страницах.
* [NavigationWindow](https://msdn.microsoft.com/ru-ru/library/system.windows.navigation.navigationwindow(v=vs.100).aspx) , чтобы размещать содержимое для навигации во всем окне.

Элементы управления

Взаимодействия с пользователем, предоставляемые моделью приложения, являются сконструированными элементами управления. В WPF "элемент управления" ― это основное понятие, относящееся к категории классов WPF, которые расположены в окне или на странице, имеют user interface (UI) и реализовывают некоторое поведение.

Элементы управления WPF по функциям

Далее перечислены встроенные элементы управления WPF.

* Кнопки: [Button](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.button(v=vs.100).aspx) и [RepeatButton](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.repeatbutton(v=vs.100).aspx).
* Отображение данных: [DataGrid](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.datagrid(v=vs.100).aspx), [ListView](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listview(v=vs.100).aspx) и [TreeView](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.treeview(v=vs.100).aspx).
* Выбор и отображение дат: [Calendar](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.calendar(v=vs.100).aspx) и [DatePicker](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.datepicker(v=vs.100).aspx).
* Диалоговые окна: [OpenFileDialog](https://msdn.microsoft.com/ru-ru/library/microsoft.win32.openfiledialog(v=vs.100).aspx), [PrintDialog](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.printdialog(v=vs.100).aspx) и [SaveFileDialog](https://msdn.microsoft.com/ru-ru/library/microsoft.win32.savefiledialog(v=vs.100).aspx).
* Рукописный фрагмент: [InkCanvas](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.inkcanvas(v=vs.100).aspx) и [InkPresenter](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.inkpresenter(v=vs.100).aspx).
* Документы: [DocumentViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.documentviewer(v=vs.100).aspx), [FlowDocumentPageViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.flowdocumentpageviewer(v=vs.100).aspx), [FlowDocumentReader](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.flowdocumentreader(v=vs.100).aspx), [FlowDocumentScrollViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.flowdocumentscrollviewer(v=vs.100).aspx) и [StickyNoteControl](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.stickynotecontrol(v=vs.100).aspx).
* Ввод: [TextBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.textbox(v=vs.100).aspx), [RichTextBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.richtextbox(v=vs.100).aspx) и [PasswordBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.passwordbox(v=vs.100).aspx).
* Структура: [Border](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.border(v=vs.100).aspx), [BulletDecorator](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.bulletdecorator(v=vs.100).aspx), [Canvas](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.canvas(v=vs.100).aspx), [DockPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.dockpanel(v=vs.100).aspx), [Expander](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.expander(v=vs.100).aspx), [Grid](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.grid(v=vs.100).aspx), [GridView](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.gridview(v=vs.100).aspx), [GridSplitter](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.gridsplitter(v=vs.100).aspx), [GroupBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.groupbox(v=vs.100).aspx), [Panel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.panel(v=vs.100).aspx), [ResizeGrip](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.resizegrip(v=vs.100).aspx), [Separator](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.separator(v=vs.100).aspx), [ScrollBar](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.scrollbar(v=vs.100).aspx), [ScrollViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.scrollviewer(v=vs.100).aspx), [StackPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.stackpanel(v=vs.100).aspx), [Thumb](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.thumb(v=vs.100).aspx), [Viewbox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.viewbox(v=vs.100).aspx), [VirtualizingStackPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.virtualizingstackpanel(v=vs.100).aspx), [Window](https://msdn.microsoft.com/ru-ru/library/system.windows.window(v=vs.100).aspx) и [WrapPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.wrappanel(v=vs.100).aspx).
* Мультимедиа: [Image](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.image(v=vs.100).aspx), [MediaElement](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.mediaelement(v=vs.100).aspx) и [SoundPlayerAction](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.soundplayeraction(v=vs.100).aspx).
* Меню: [ContextMenu](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.contextmenu(v=vs.100).aspx), [Menu](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.menu(v=vs.100).aspx) и [ToolBar](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.toolbar(v=vs.100).aspx).
* Переходы: [Frame](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.frame(v=vs.100).aspx), [Hyperlink](https://msdn.microsoft.com/ru-ru/library/system.windows.documents.hyperlink(v=vs.100).aspx), [Page](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.page(v=vs.100).aspx), [NavigationWindow](https://msdn.microsoft.com/ru-ru/library/system.windows.navigation.navigationwindow(v=vs.100).aspx) и [TabControl](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.tabcontrol(v=vs.100).aspx).
* Выбор: [CheckBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.checkbox(v=vs.100).aspx), [ComboBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.combobox(v=vs.100).aspx), [ListBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listbox(v=vs.100).aspx), [RadioButton](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.radiobutton(v=vs.100).aspx) и [Slider](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.slider(v=vs.100).aspx).
* Информация пользователя: [AccessText](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.accesstext(v=vs.100).aspx), [Label](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.label(v=vs.100).aspx), [Popup](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.popup(v=vs.100).aspx), [ProgressBar](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.progressbar(v=vs.100).aspx), [StatusBar](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.statusbar(v=vs.100).aspx), [TextBlock](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.textblock(v=vs.100).aspx) и [ToolTip](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.tooltip(v=vs.100).aspx).

Ввод и команды

Элементы управления наиболее часто обнаруживают входные данные пользователя и отвечают на них. Система ввода WPF использует прямые и перенаправленные события для поддержки ввода текста, управления фокусом и позиционирования мыши. Дополнительные сведения см. в разделе [Общие сведения о входных данных](https://msdn.microsoft.com/ru-ru/library/ms754010(v=vs.100).aspx).

Приложения часто предъявляют сложные требования к вводу данных. WPF предоставляет систему команд, в которой действия пользователя по вводу данных отделяются от кода, который отвечает на эти действия. Дополнительные сведения см. в разделе [Общие сведения о системе команд](https://msdn.microsoft.com/ru-ru/library/ms752308(v=vs.100).aspx).

Макет

При создании UI происходит упорядочивание элементов управления по расположению и размеру для формирования структуры. Основным требованием любой структуры является адаптируемость к изменениям размера окна и параметрам отображения. Чтобы не пришлось создавать код для адаптации структуры в таких обстоятельствах, WPF предоставляет первоклассную расширяемую систему структуры.

Основой системы структуры является относительное позиционирование, что увеличивает способность адаптации к изменяемому окну и условиям отображения. Кроме того, система структуры управляет согласованием между элементами управления для определения структуры.Такое согласование состоит из двух этапов: сначала элемент управления сообщает родительскому элементу, какое расположение и размер требуется; затем родительский элемент сообщает элементу управления, какое пространство он может занять.

Дочерние элементы управления получают доступ к системе макета через базовые классы WPF. Для распространенных макетов, таких как сетки, вложение и закрепление, WPF включает несколько элементов управления макетом.

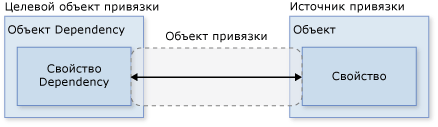
* [Canvas](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.canvas(v=vs.100).aspx) : дочерние элементы управления предоставляют свои собственные макеты.
* [DockPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.dockpanel(v=vs.100).aspx) : дочерние элементы управления выравниваются по краям панели.
* [Grid](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.grid(v=vs.100).aspx) : дочерние элементы управления располагаются по строкам и столбцам.
* [StackPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.stackpanel(v=vs.100).aspx) : дочерние элементы управления располагаются либо горизонтально, либо вертикально.
* [VirtualizingStackPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.virtualizingstackpanel(v=vs.100).aspx) : дочерние элементы управления являются виртуальными и располагаются в одной горизонтальной или вертикальной строке.
* [WrapPanel](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.wrappanel(v=vs.100).aspx) : дочерние элементы управления располагаются в порядке слева-направо и переносятся на следующую строку, когда в текущей строке не хватает места.

Привязка данных

Большинство приложений создаются для предоставления пользователям средств просмотра и редактирования данных. В приложениях WPF работа по хранению и доступу к данным уже обеспечена такими технологиями, как Microsoft SQL Server и ADO.NET. После обращения к данным и загрузки данных в управляемые объекты приложения начинается основная тяжелая работа для приложений WPF. По существу, она включает в себя две вещи:

* копирование данных из управляемых объектов в элементы управления, где данные могут отображаться и редактироваться;
* проверка того, что изменения, внесенные в данные с помощью элементов управления, скопированы обратно в управляемые объекты.

Чтобы упростить разработку приложений, WPF предоставляет механизм привязки данных для автоматического выполнения этих этапов. Основной единицей механизма привязки данных является класс [Binding](https://msdn.microsoft.com/ru-ru/library/system.windows.data.binding(v=vs.100).aspx), назначение которого привязать элемент управления (цель привязки) к объекту данных (источник привязки). Это отношение показано на следующем рисунке.



Механизм привязки данных WPF предоставляет дополнительную поддержку, включающую проверку, сортировку, фильтрацию и группировку. Кроме того, привязка данных поддерживает использование шаблонов данных для создания настраиваемого UI, чтобы связывать данные, когда UI отображается несоответствующими стандартными элементами управления WPF.

Графика

WPF представляет обширный, масштабируемый и гибкий набор графических возможностей, которые имеют следующие преимущества:

Графика, не зависящая от разрешения и устройства. Основной единицей измерения в графической системе WPF является аппаратно-независимая точка, которая составляет 1/96 часть дюйма независимо от фактического разрешения экрана и предоставляет основу для создания изображения, независимого от разрешения и устройства. Каждый аппаратно-независимый пиксель автоматически масштабируется в соответствии с числом точек на дюйм в системе, в которой он отображается.

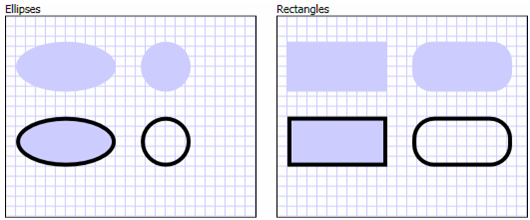
Повышенная точность. В системе координат WPF используются числа с плавающей запятой двойной точности, вместо одиночной точности. Значения преобразований и прозрачности также выражаются с помощью чисел двойной точности. Кроме того, WPF поддерживает широкую цветовую палитру (scRGB) и предоставляет встроенную поддержку для управления входными данными из различных цветовых пространств.

Дополнительная поддержка графики и анимации. WPF упрощает программирование графики за счет автоматического управления анимацией. Разработчик не должен заниматься обработкой сцен анимации, циклами визуализации и билинейной интерполяцией. Кроме того, WPF предоставляет поддержку проверки нажатия и полную поддержку альфа-компоновки.

Аппаратное ускорение. Графическая система WPF использует преимущества графического оборудования, чтобы уменьшить использование ЦП.

Двухмерные формы

WPF предоставляет библиотеку общих 2-D фигур, нарисованных с помощью векторов, таких, как прямоугольники и эллипсы, показанные на следующем рисунке.



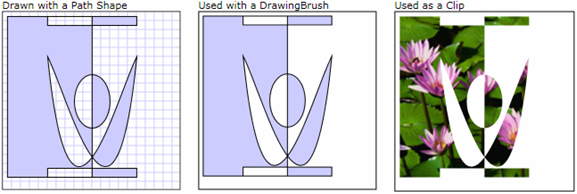
Интересная особенность фигур в том, что они могут не только отображаться; фигуры реализовывают многие возможности, ожидаемые от элементов управления, включая ввод с клавиатуры и ввод с помощью мыши.

Двухмерная геометрия

WPF предоставляет стандартный набор двухмерных (2-D) фигур. Однако, возможно, потребуется создать пользовательские фигуры для облегчения разработки настраиваемого UI. В этих целях WPF предоставляет геометрические объекты. На следующем рисунке показано использование геометрий для создания пользовательской фигуры, которая может быть нарисована непосредственно, использоваться в качестве кисти, или использоваться для отсечения других фигур и элементов управления.

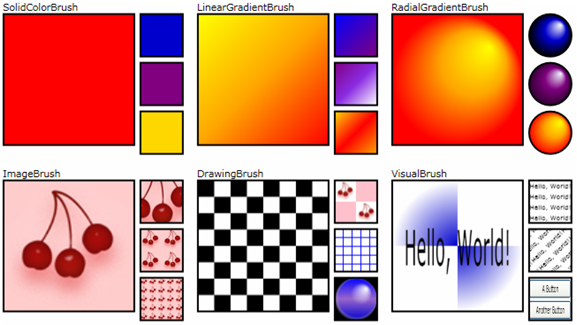
Объекты [Path](https://msdn.microsoft.com/ru-ru/library/system.windows.shapes.path(v=vs.100).aspx) могут быть использованы для рисования замкнутых, открытых, составных фигур и даже кривых поверхностей.

Объекты [Geometry](https://msdn.microsoft.com/ru-ru/library/system.windows.media.geometry(v=vs.100).aspx) могут использоваться для отсечения, проверки нажатия и отрисовки данных двухмерной графики.



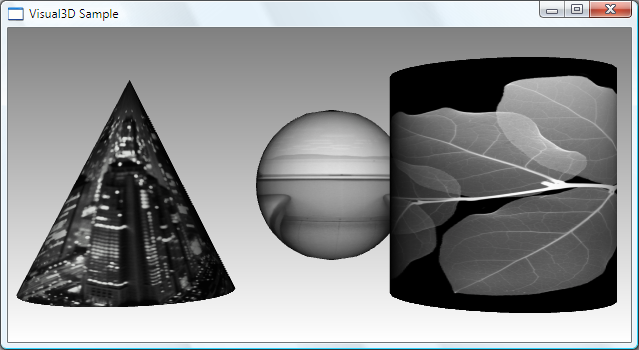
Двухмерные эффекты

Подмножество средств 2-D WPF включает визуальные эффекты, такие как градиенты, точечные рисунки, чертежи, рисунки с видео, поворот, масштабирование и наклон. Все это достигается с помощью кистей; на следующем рисунке показано несколько примеров.



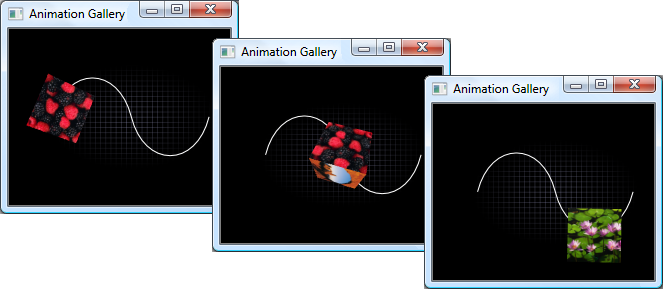
Трехмерная визуализация

WPF также включает возможности трехмерной (3-D) визуализации, интегрированные с двухмерной (2-D) графикой, что позволяет создавать более яркий и интересный UIs. Например, следующий рисунок показывает изображения 2-D, отображаемые в фигурах 3-D.



Анимация

Поддержка анимации WPF позволяет осуществлять рост, вибрацию, вращение и исчезновение элементов управления для создания интересных страничных переходов и других эффектов. Можно анимировать большинство классов WPF, даже настраиваемые классы. На следующем рисунке показана простая анимация в действии.



Мультимедиа

Одним из способов передачи богатого содержимого является использование аудиовизуальной среды. WPF предоставляет специальную поддержку для изображений, видео и аудио.

Изображения

Изображения присутствуют в большинстве приложений, и WPF предоставляет несколько способов их использования. На следующем рисунке показан UI со списком, в котором содержатся эскизные изображения. При выделении эскиза изображение показывается в полном размере.



Видео и аудио

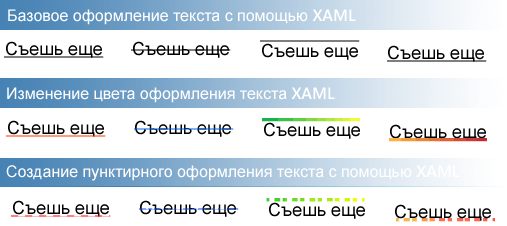
Элемент управления [MediaElement](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.mediaelement(v=vs.100).aspx) способен воспроизводить видео и аудио, и является достаточно гибким, чтобы служить основой для пользовательского проигрывателя.

Текст и типография

Для облегчения отрисовки текста высокого качества WPF предоставляет следующие возможности:

* Поддержка шрифта OpenType.
* Улучшения ClearType.
* Высокая производительность, которая использует преимущества аппаратного ускорения.
* Интеграция текста с мультимедиа, графикой и анимацией.
* Механизмы резервирования и поддержки международного шрифта.

Для демонстрации интеграции текста с графикой на следующем рисунке показано применение художественного оформления текста.



Документы

WPF предоставляет встроенную поддержку работы с тремя типами документов: документами нефиксированного формата, документами фиксированного формата и документами XML Paper Specification (XPS). WPF также предоставляет службы для создания и просмотра документов, управления документами, добавления заметок, упаковки и печати документов.

Документы нефиксированного формата

Документы нефиксированного формата разработаны для оптимизации просмотра и читаемости посредством динамической настройки и обновления содержимого при изменении размера окна и параметров дисплея.

Документы фиксированного формата

Документы фиксированного формата предназначены для приложений, в которых требуется точное представление вида "что видишь, то и получишь" (режим полного соответствия изображения на экране и распечатки WYSIWYG), особенно по отношению к печати. Документы фиксированного формата обычно используются при подготовке публикаций с помощью настольных издательских средств, обработке текста и разметке формы, где строгое соблюдение исходного дизайна страницы является обязательным.

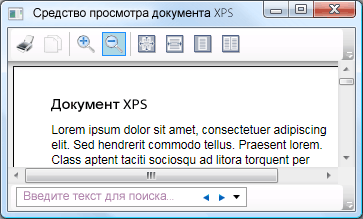
В документах фиксированного формата поддерживается точное размещение содержимого независимо от устройства. Например, документ фиксированного формата отображается на мониторе с разрешением 96 точек на дюйм точно так же, как при печати на лазерном принтере с разрешением 600 точек на дюйм или на фотонаборной машине с разрешением 4800 точек на дюйм. Макет документа остается одинаковым во всех случаях, хотя качество документа варьируется в зависимости от возможностей каждого устройства.

Документы XML Paper Specification (XPS) построены на основе документов фиксированного формата WPF. Документы XPS описываются схемой на основе XML, которая фактически представляет разбитый на страницы электронный документ. Открытый кросс-платформенный формат документов XPS предназначен для упрощения создания, печати и архивирования разбитых на страницы документов, а также организации совместного доступа. Технология XPS включает следующие важные возможности:

Упаковка документов XPS в файлы [ZipPackage](https://msdn.microsoft.com/ru-ru/library/system.io.packaging.zippackage(v=vs.100).aspx), соответствующие стандарту [Open Packaging Conventions](http://go.microsoft.com/fwlink/?LinkID=71255) (OPC).

* Размещение в автономных и в размещенных в браузере приложениях.
* Создание документов XPS и управление ими из приложений WPF вручную.
* Высокоточная отрисовка путем выбора устройства вывода максимального качества.
* Очередь печати принтера Windows Vista.
* Прямая отправка документов на XPS-совместимые принтеры.
* Интеграция UI с [DocumentViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.documentviewer(v=vs.100).aspx).

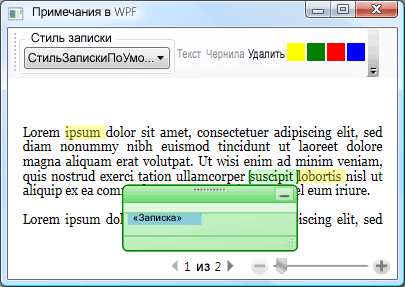
На следующем рисунке показан документ XPS, который отображается с помощью [DocumentViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.documentviewer(v=vs.100).aspx).



[DocumentViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.documentviewer(v=vs.100).aspx) также дает возможность пользователям изменять просмотр, поиск и печать документов XPS.

Заметки

Заметки ― это примечания или комментарии, которые добавляются к документу, чтобы отметить информацию или выделить интересующие элементы для дальнейшего использования. В напечатанных документах делать заметки просто, но в электронных документах возможность создания заметок часто ограничена или отсутствует. Однако в WPF для поддержки возможности создания комментариев-наклеек и выделений предоставляется система заметок. Кроме того, эти заметки можно применять к документам, размещенным в элементе управления [DocumentViewer](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.documentviewer(v=vs.100).aspx), как показано на следующем рисунке.



Упаковка

WPF [System.IO.Packaging](https://msdn.microsoft.com/ru-ru/library/system.io.packaging(v=vs.100).aspx) APIs позволяет приложениям организовывать данные, содержимое и ресурсы в единые, переносимые, удобные для распространения и для доступа упакованные документы. Для проверки подлинности элементов, содержащихся в пакете, можно включать цифровые подписи, которые гарантируют, что подписанный элемент не был подделан или изменен. Кроме того, можно ограничить доступ к защищенной информации, зашифровав пакеты с помощью системы управления правами.

Печать

.NET Framework включает подсистему печати, которую WPF дополняет поддержкой для расширенного управления системой печати.Улучшения печати включают следующее:

* Установка удаленных серверов и очередей печати в режиме реального времени.
* Динамическое обнаружение возможностей принтера.
* Динамическая установка параметров принтера.
* Перенаправление и изменение приоритета заданий на печать.

В документах XPS также имеется ключевое улучшение производительности. Существующий путь печати Microsoft Windows Graphics Device Interface (GDI) обычно подразумевает два преобразования:

первое ― преобразование документа в формат процессора печати, например в Enhanced Metafile (EMF);

второе ― преобразование в язык описания страниц принтера, например в Printer Control Language (PCL) или PostScript.

Однако документы XPS обходятся без этих преобразований, поскольку один компонент формата файла XPS является как языком обработчика заданий печати, так и языком описания страницы. Эта поддержка позволяет уменьшить как размер файла очереди, так и загрузки сетевых принтеров.

Главной задачей большей части элементов управления WPF является отображение содержимого. В WPF тип и количество элементов, которые могут составлять содержимое элемента управления, называется моделью содержимого элемента управления. Некоторые элементы управления могут содержать один элемент и один тип содержимого. Например, содержимое [TextBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.textbox(v=vs.100).aspx) — строковое значение, которое присваивается свойству [Text](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.textbox.text(v=vs.100).aspx).

Триггеры

Хотя главной задачей разметки XAML и является реализация внешнего вида приложения, XAML также можно использовать для реализации некоторых действий, выполняемых приложением. Например, с помощью триггеров изменять внешний вид приложения в зависимости от действий пользователя.

Шаблоны элементов управления

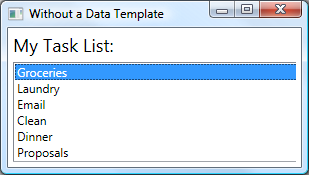
UIs по умолчанию для элементов управления WPF обычно создается из других элементов управления и фигур. Например, [Button](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.button(v=vs.100).aspx) состоит из элементов управления [ButtonChrome](https://msdn.microsoft.com/ru-ru/library/microsoft.windows.themes.buttonchrome(v=vs.100).aspx) и [ContentPresenter](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.contentpresenter(v=vs.100).aspx).

[ButtonChrome](https://msdn.microsoft.com/ru-ru/library/microsoft.windows.themes.buttonchrome(v=vs.100).aspx) обеспечивает стандартный внешний вид кнопки, в то время как[ContentPresenter](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.contentpresenter(v=vs.100).aspx) отображает содержимое кнопки, заданное свойством [Content](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.contentcontrol.content(v=vs.100).aspx).

Иногда внешний вид элемента управления по умолчанию может не сочетаться с общим внешним видом приложения. В этом случае можно с помощью [ControlTemplate](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.controltemplate(v=vs.100).aspx) изменить внешний вид UI элемента управления без изменения его содержимого и поведения.

Шаблоны данных

В то время как шаблон элемента управления позволяет задавать внешний вид элемента управления, шаблон данных позволяет задавать внешний вид содержимого элемента управления. Шаблоны данных часто используются для улучшения способа отображения данных, связанных с элементом управления. На следующем рисунке показан внешний вид по умолчанию элемента управления [ListBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listbox(v=vs.100).aspx), который связан с коллекцией объектов Task, где каждый объект имеет имя, описание и приоритет.



Внешний вид по умолчанию ― это вид, который ожидается от [ListBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listbox(v=vs.100).aspx). Однако внешний вид по умолчанию каждого объекта коллекции содержит только имя. Чтобы отобразить имя, описание и приоритет объекта коллекции, внешний вид по умолчанию элементов связанного списка элемента управления [ListBox](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listbox(v=vs.100).aspx)должен быть изменен с помощью [DataTemplate](https://msdn.microsoft.com/ru-ru/library/system.windows.datatemplate(v=vs.100).aspx).

Ресурсы

Элементы управления в приложении должны использовать один и тот же внешний вид, который может включать любые шрифты и цвета фона для шаблонов элементов управления, шаблонов данных и стилей. С помощью поддержки WPF для ресурсов user interface (UI) можно инкапсулировать эти ресурсы в одном расположении для повторного использования.

Темы и обложки

С точки зрения визуального восприятия тема определяет глобальный внешний вид системы Windows и приложений, которые в ней запускаются. Windows поставляется с несколькими темами. Например, Microsoft Windows XP поставляется с темами Windows XP и Windows Classic, а Windows Vista поставляется с темами Windows Vista и Windows Classic. Внешний вид, определяемый темой, задает внешний вид по умолчанию для приложения WPF. Однако WPF не поддерживает прямую интеграцию с темами Windows. Поскольку внешний вид WPF определяется шаблонами, WPF включает по одному шаблону для каждой известной темы Windows, в том числе Aero (Windows Vista), Classic (Microsoft Windows 2000), Luna (Microsoft Windows XP) и Royale (Microsoft Windows XP Media Center Edition 2005). Эти темы упакованы в словари ресурсов, которые применяются, если ресурсы не найдены в приложении. Внешний вид многих приложений задается с помощью этих тем; сохраняющаяся согласованность с внешним видом Windows помогает пользователям быстрее освоиться с большинством приложений.

С другой стороны, опыт работы пользователя с некоторыми приложениями не обязательно связан с стандартными темами. Например, Microsoft Windows Media Player работает с аудио- и видеоданными, и здесь преимущество имеют пользователи с опытом работы в другом стиле. Такие UIs чаще предоставляют настраиваемые, специфичные для приложения темы. Такие темы называются "обложки", и приложения, которые их используют, часто предоставляют средства настройки различных аспектов обложек. Microsoft Windows Media Player имеет множество собственных обложек и обложек от сторонних производителей.

Пользовательские элементы управления

Хотя WPF и обеспечивает поддержку настройки, могут возникнуть ситуации, в которых существующие элементы управления WPF не удовлетворяют требованиям приложения или его пользователей. Это возможно в следующих ситуациях:

Нужный UI не может быть создан путем настройки внешнего вида и поведения существующих реализаций WPF.

Нужное поведение не поддерживается (или поддерживается частично) существующими реализациями WPF.

Тем не менее, в этом случае можно воспользоваться одной из трех моделей WPF для создания нового элемента управления. Каждая модель предназначена для определенного скрипта и требует, чтобы пользовательский элемент управления был производным от конкретного базового класса WPF. Далее приводится описание этих трех моделей.

Модель пользовательского элемента управления. Пользовательский элемент управления производится из [UserControl](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.usercontrol(v=vs.100).aspx) и состоит из одного или нескольких других элементов управления.

Модель элемента управления. Пользовательский элемент управления производится из [Control](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.control(v=vs.100).aspx) и используется для построения реализаций, в которых внешний вид и поведение разделены с помощью шаблонов, подобно большей части элементов управления WPF. Создание элемента управления, производного от [Control](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.control(v=vs.100).aspx), предоставляет по сравнению с пользовательскими элементами управления большую свободу для создания нестандартного UI, но может потребовать дополнительных усилий.

Модель элемента .NET Framework. Пользовательский элемент управления производится от [FrameworkElement](https://msdn.microsoft.com/ru-ru/library/system.windows.frameworkelement(v=vs.100).aspx), когда его внешний вид определяется пользовательской логикой визуализации (не шаблонами).

Советы и рекомендации по WPF

Как любая платформа разработки, WPF может использоваться множеством способов для достижения нужного результата. Для гарантий, что приложения WPF предоставляют требуемое взаимодействие с пользователем и удовлетворяют требованиям аудитории в целом, в данном разделе предлагаются советы и рекомендации по специальным возможностям, глобализации и локализации, а также производительности.

Рассмотрим примеры работы с платформой .NET на примере выполнения задания для демонстрационного экзамена 2016 года(WPF).

Ознакомиться с полной версией задания и использованными ресурсами можно на ресурсе Академии Ворлдскиллс Россия по ссылке: <https://drive.google.com/drive/u/0/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-lkn>

Сессия 1 данного Конкурсного задания состоит из следующей документации / файлов:

1. WSR2018\_TP09\_14+\_С1.pdf (Инструкция к первой сессии)

2. marathon-skills-2018-database-mysql.sql (Сценарий SQL для создания таблиц с данными для MySQL)

3. marathon-skills-2018-database-mssql.sql (Сценарий SQL для создания таблиц с данными для Microsoft SQL Server)

4. marathon-skills-2018-staff-import.xlsx (Данные для импорта)

5. marathon-skills-2018-testing-data-s1.pdf (Тестирование системы)

ВВЕДЕНИЕ

В этой сессии вы начнете разработку приложения и базы данных для MarathonSkills 2016. Дизайнер предоставил вам набор системной документации, так что вы можете построить систему в соответствии с потребностями клиента. Найдите время для знакомства с предоставленными материалами.

Создайте базу данных, а затем импортируйте туда необходимые данные. Затем создайте приложение: часть окон, которые будут доступны пользователям системы.

Файл: marathon-skills-2018-testing-data-s1.pdf предоставлен вам для того, чтобы вы смогли протестировать систему

ИНСТРУКЦИЯ УЧАСТНИКУ

К концу этой сессии, у вас должны быть достигнуты следующие результаты, необходимые для того, чтобы заказчик был спокоен, что система будут завершена вовремя. Убедитесь, что вы следуете предоставленному руководству по стилю во всех частях системы. Убедитесь, что вы предоставляете соответствующие проверки и сообщения об ошибках во всех частях системы. Убедитесь, что все соответствующие кнопки / ссылки работают в конце сессии. Убедитесь, что вы используете соответствующие соглашения об именах для всех частей системы по мере необходимости.

ПРАКТИЧЕСКИЕ РЕЗУЛЬТАТЫ

1.1 СОЗДАНИЕ БАЗЫ ДАННЫХ Создайте базу данных, используя знакомую вам платформу (MySQL / MSSQLServer) на сервере баз данных, который вам предоставлен.

1.2 ЗАГРУЗКА ДАННЫХ Сценарий SQLпредоставлен для вас, чтобы создать большинство таблиц и вставки данных в них. Все, что вам нужно сделать, это импортировать сценарий SQL в вашу базу данных. Выберите сценарий SQL, который подходит для вашей платформы: MySQL: marathon-skills-2016-database-mysql.sql СерверSQL: marathon-skills-2016-database-mssql.sql Таблица Сотрудники (персонал, должности) не включены в этот сценарий SQL. См результаты 1.3 и 1.4.

1.3 СОЗДАТЬ ТАБЛИЦЫ ДЛЯ ПЕРСОНАЛА СОГЛАСНО СПЕЦИФИКАЦИИ Обратитесь к диаграмме базы данных (ERD) и словарю данных. Создайте таблицы сотрудников (Position, Staff, Timesheet) согласно спецификации.

1.4 ИМПОРТ ДАННЫХ ПЕРСОНАЛА Все данные сотрудников были представлены в marathon-skills-2018-staff-import.xlsx. Эти данные не отформатированы для импортирования непосредственно в базу данных, вам необходимо отформатировать данные и загрузить их в таблицы, которые вы только что создали. Поле Summary Information не требуется. В поле "FullName" в формате "Имя Фамилия" используются разные символы разделителя. Убедитесь, что адреса электронной почты в правильном формате.

1.5 СОЗДАТЬ ПРИЛОЖЕНИЕ Создайте приложение, используя выбранную вами платформу .NET (или Java).

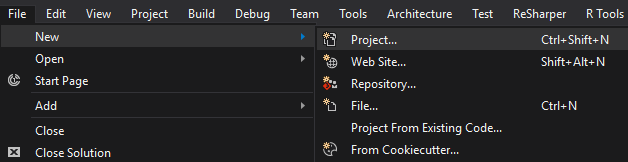
1.6 СОЗДАНИЕ "1. ГЛАВНЫЙ ЭКРАН СИСТЕМЫ" Создайте главное меню системы, как указано в "1. Главный экран системы"в презентации. Каждое окно / страница приложения, который имеет "? дней? часов и? минут до начала гонки в "в нижней части экрана должно автоматически обновляться в режиме реального времени. Рассчитать количество времени, оставшегося до начала первого MarathonSkills 2018 начинается (2018-11-24 6:00).

1.7 СОЗДАНИЕ ОКНА "7. ПОДРОБНАЯ ИНФОРМАЦИЯ” Создание окна подробная информация как указано в “7. Подробная информация” в презентации.

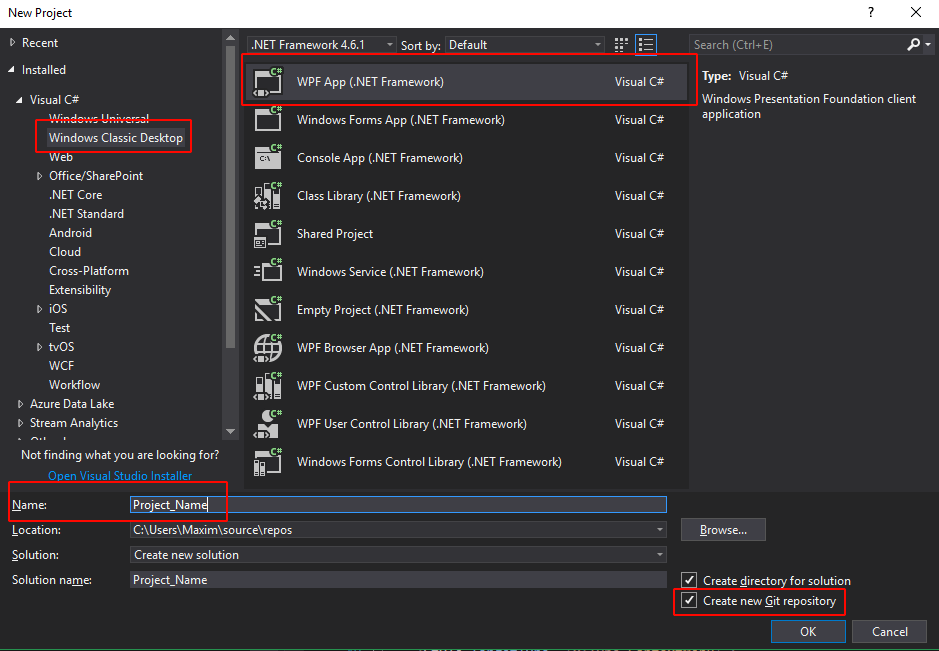
1.8 СОЗДАНИЕ ОКНА "10. СПИСОК БЛАГОТВОРИТЕЛЬНЫХ ОРГАНИЗАЦИЙ" Создать страницу, как описано в "10. Список благотворительных организаций»в презентации. Эта страница отображает все благотворительные организации, перечисленные в базе данных вместе с их логотипами (при условии, что они есть в общих ресурсах), чтобы показать бегунам благотворительные организации, которые их могут поддержать.

СОЗДАНИЕ ПРОЕКТА

Запустите Vi sual Studio 2017, перейдите File – Project.



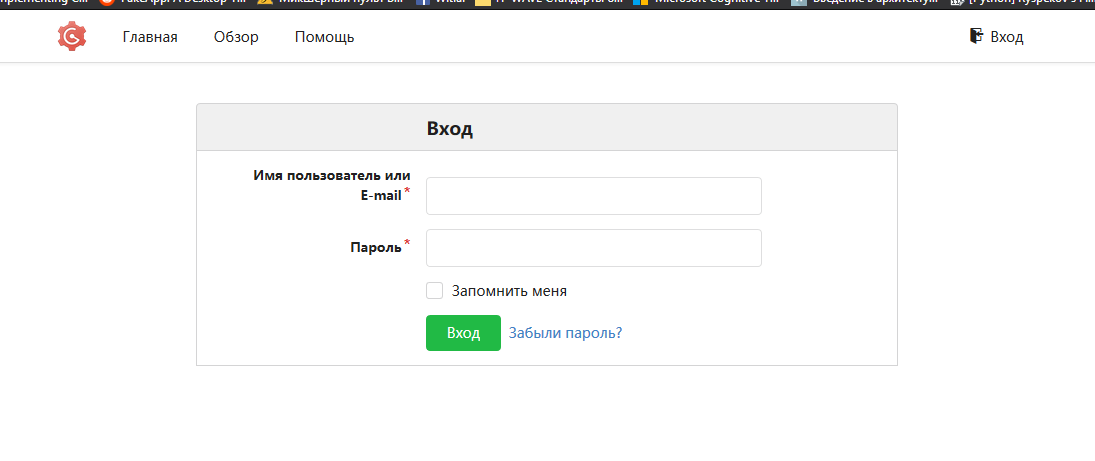
Выберите Visual C# - Windows Classic Desktop – WPF App, укажите имя проекта, а также поставьте галочку «Create new git repositoty» чтобы включить проект в систему контроля версий.



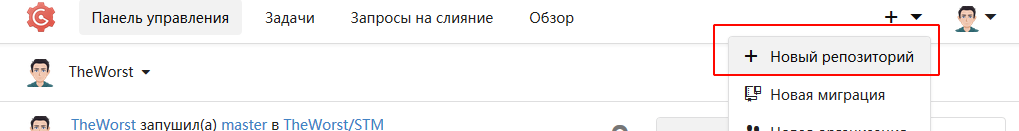
После нажатия кнопки «OK» создастся проект с пустой формой.

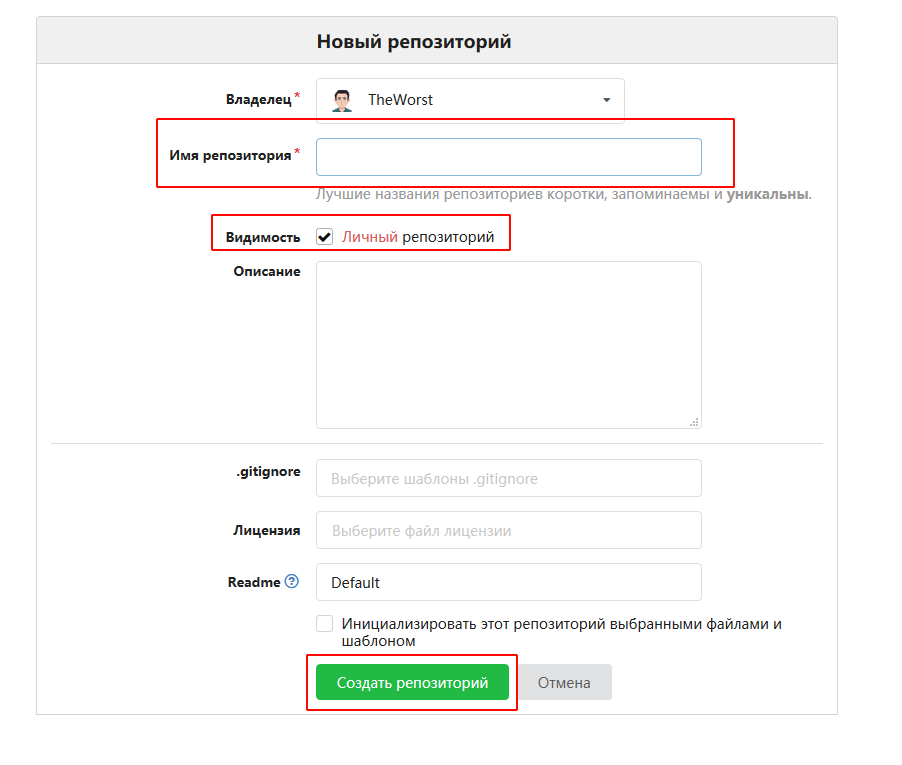
СОЗДАНИЕ РЕПОЗИТОРИЯ

Откройте в браузере GOGS. Авторизуйтесь под своей учетной записью.



На верхней панели нажмите «+» и выберите «Новый репозиторий».

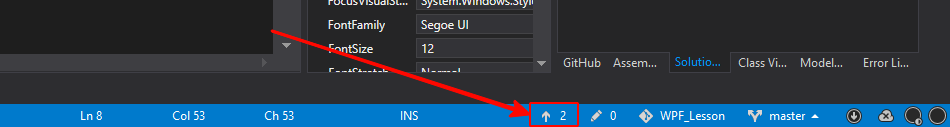


После чего укажите имя репозитория, поставьте галочку что репозиторий будет являться личным. И создайте его путем нажатия на «Создать репозиторий».

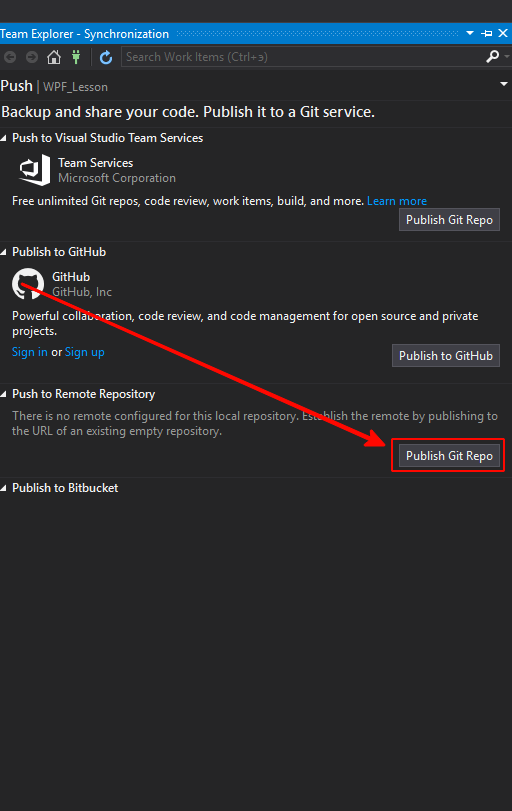
После создания репозитория появится следующее окно.

Создадим в проекте «Resource Dictionary».

Для того чтобы подключить проект, откройте проект в Visual Studio, затем внизу нажмите на стрелку направленную вверх.



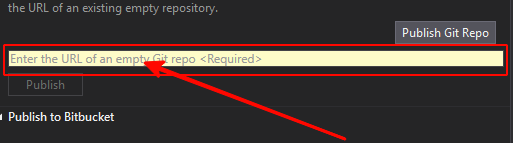
После нажатия у вас откроется окно «Team Explorer», в котором нажмите «Publish Git Repo»



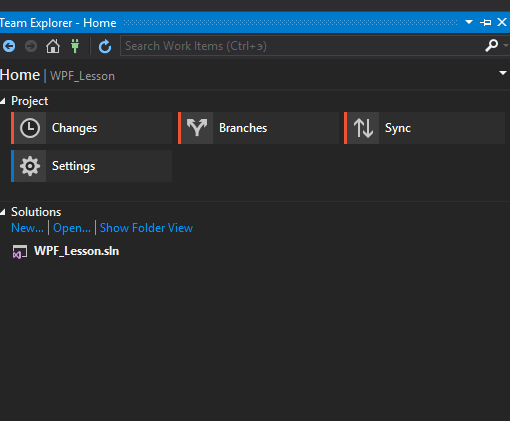
После чего у вас откроется окно в которое необходимо вставить ссылку на репозиторий. Для получения ссылки, откройте GOGS, авторизуйтесь, зайдите в созданный репозиторий (урок Создание репозитория) и там скопируйте на него ссылку.



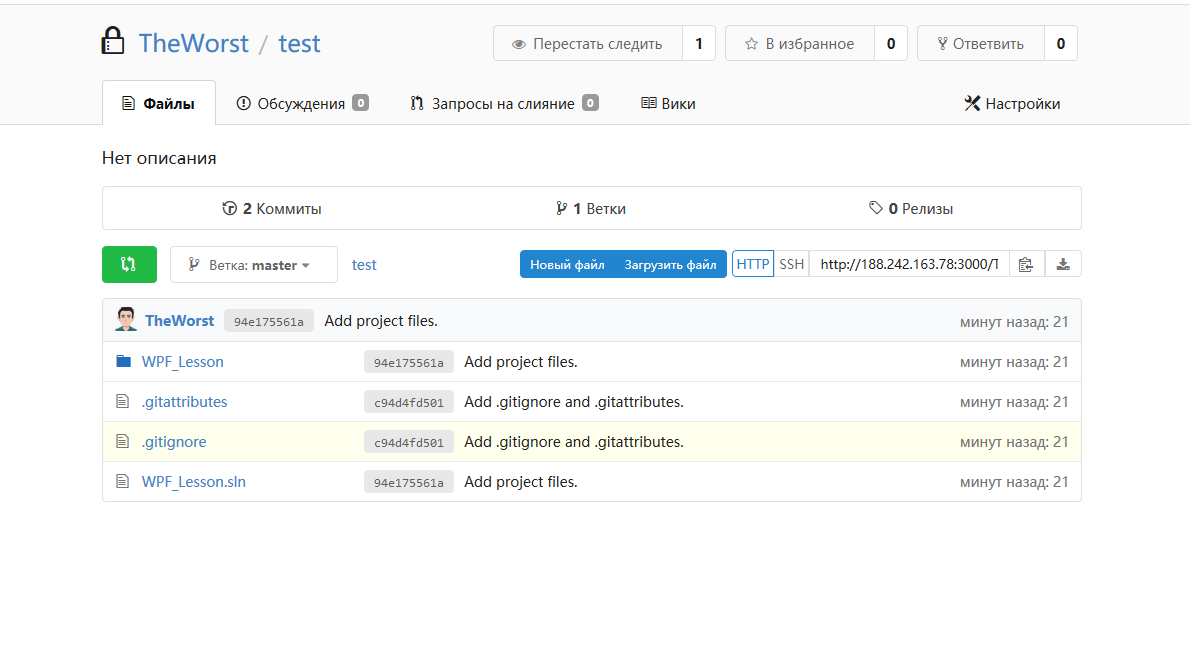
После того как скопируете ссылку вставьте ее в поле подключения удаленного репозитория в «Team Explorer» и нажмите кнопку «Publish»



После нажатия кнопки, проект будет сохранен на удаленном репозитории, а также создастся связь между локальным и удаленным репозиторием.



Проверить результат сохранения проекта можно на GOGS в репозитории.

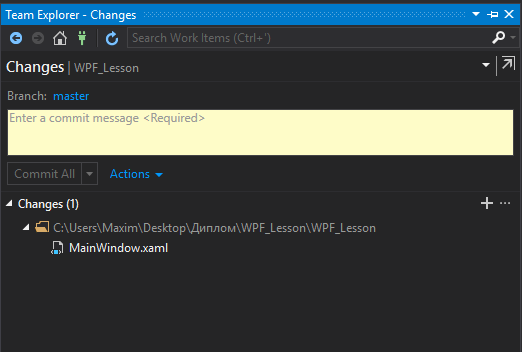


СОХРАНЕНИЕ ИЗМЕНЕНИЙ В РЕПОЗИТОРИИ

По мере того как будут вноситься правки в проект, счетчик изменений будет расти (указывается внизу Visual Studio)

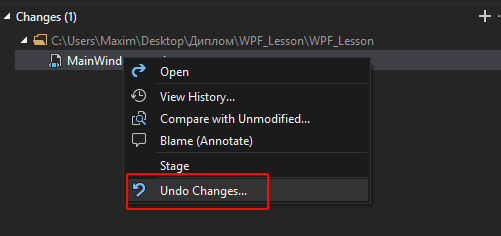


Чтобы сохранить проект в репозитории, необходимо нажать на счетчик (карандаш). После этого появится окно Team Explorer.

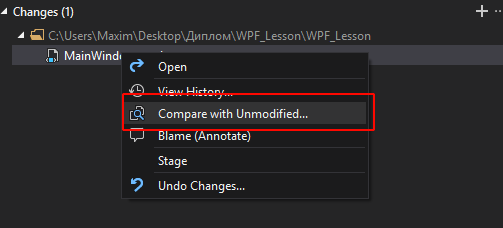


В данном окне показывается ветка (branch) в которую будет сохранятся изменения и файлы (Changes) в которых произошли изменения.

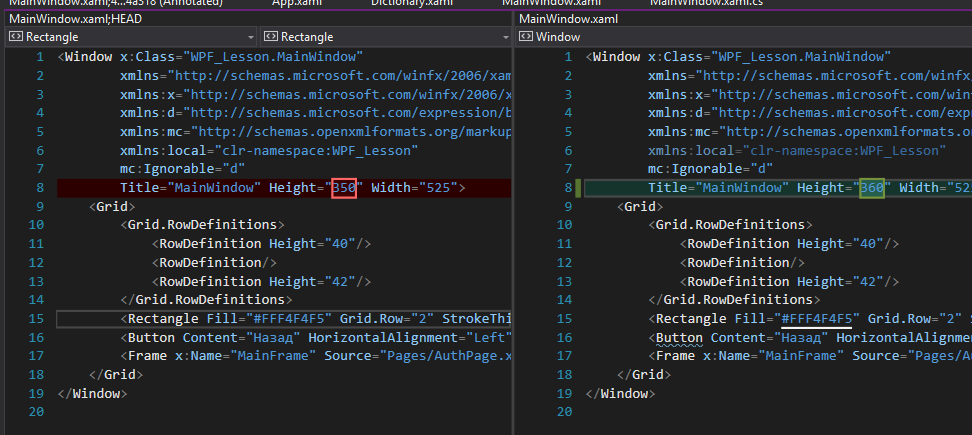
Если изменения были произведены ошибочно, то их можно откатить на предыдущею версию, путем нажатия на необходимом файле правой кнопкой мыши и выбрав пункт «Undo Changes»



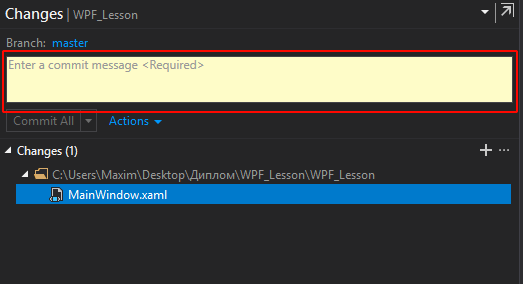
Если необходимо посмотреть изменения в файле (сравнить с предыдущей версией), то необходимо нажать правую кнопку мыши на нужном файле и выбрать пункт меню «Compare with Unmodified»



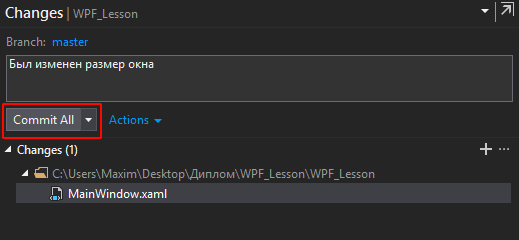
После чего откроется окно с изменениями, в одном окне предыдущая версия, в другом актуальная.



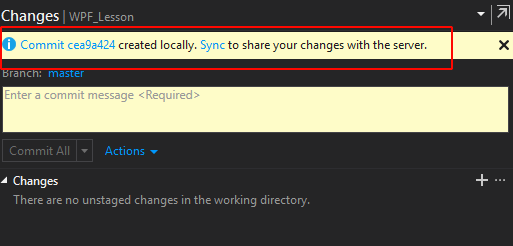
Для того чтобы сохранить изменения необходимо написать текст об изменениях в данной версии.



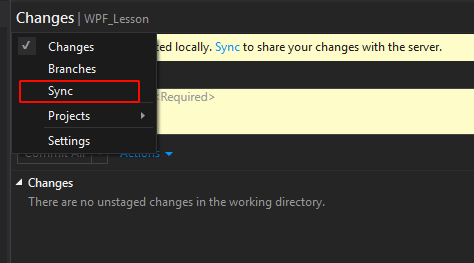
После ввода изменений, необходимо нажать кнопку «Commit All» для сохранения изменений в локальный репозиторий.



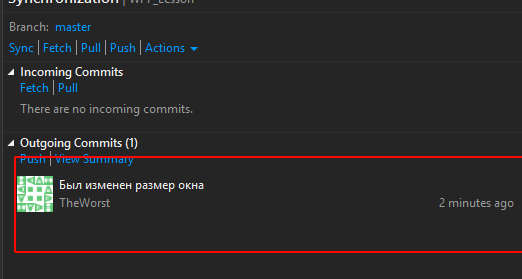
После чего будет выведено сообщение о успешном сохранении проекта. Так же будет предложено синхронизировать локальный репозиторий с удаленным.



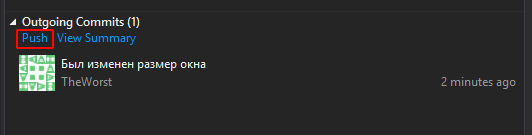
Теперь необходимо отправить изменения локального репозитория на удаленный репозиторий. Для этого необходимо выбрать пункт Sync из выпадающего меню.



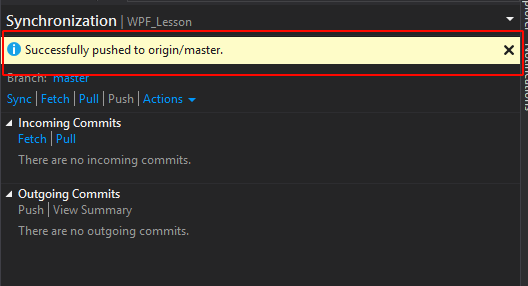
В данном пункте меню, внизу будет отображаться ваш коммит (изменения).



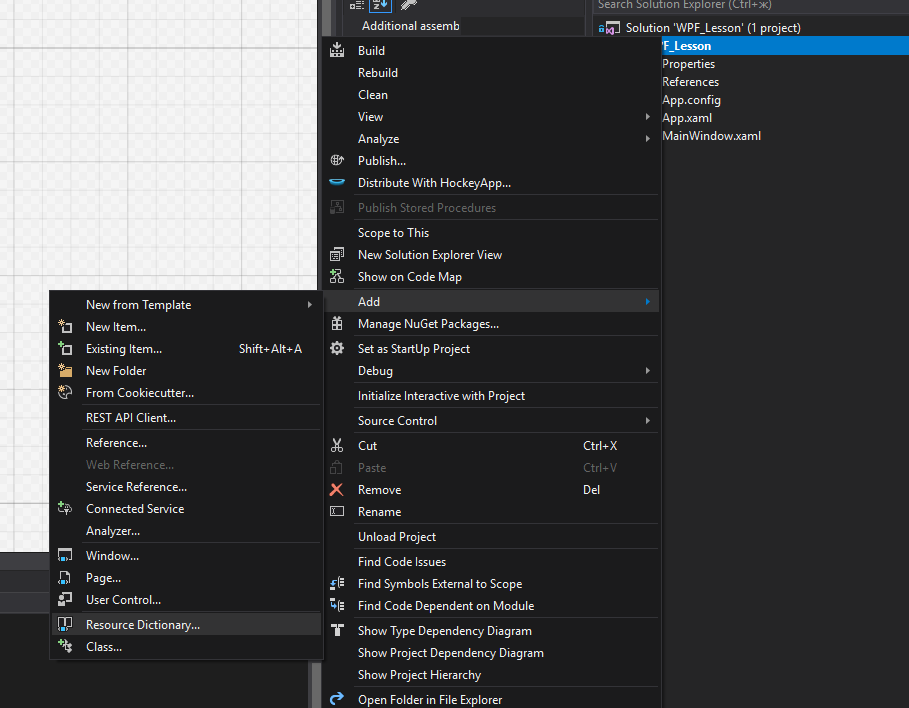
Для того чтобы отправить изменения необходимо нажать на кнопку «Push»



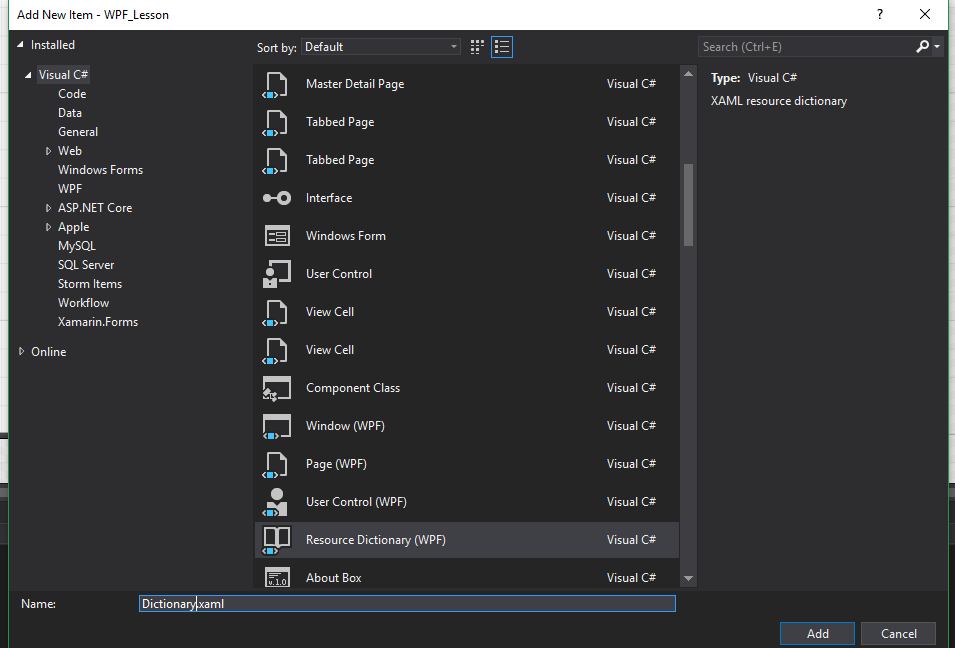
После нажатия кнопку локальный репозиторий будет сохранен на удаленном. И после успешного завершения будет выведено соответствующее сообщение.



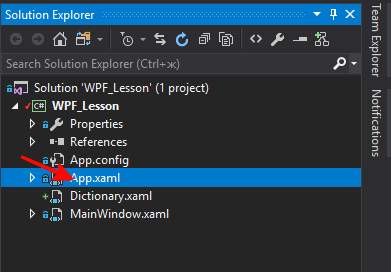
СОЗДАНИЕ СЛОВАРЯ СТИЛЕЙ



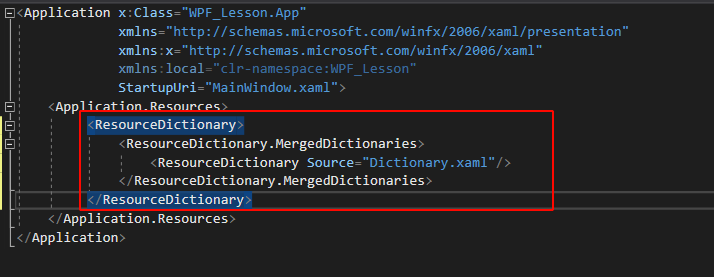
Укажем имя «Dictionary.xaml» и создадим его.



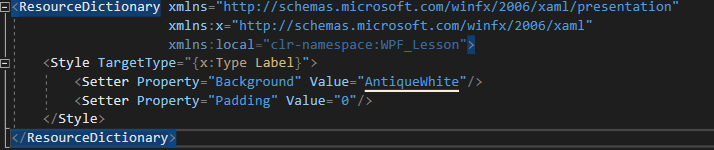
Теперь словарь необходимо подключить к проекту (чтобы компоненты могли обращаться к стилям) для этого откроем «App.xaml»

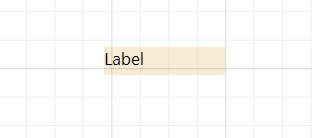


И впишем следующий код для подключения словаря.



После данных операций можно начинать писать стили и константы. Напишем первый стиль, который будет менять задний фон и внутренний отступ, и он применяться ко всем компонентам типа «Label»

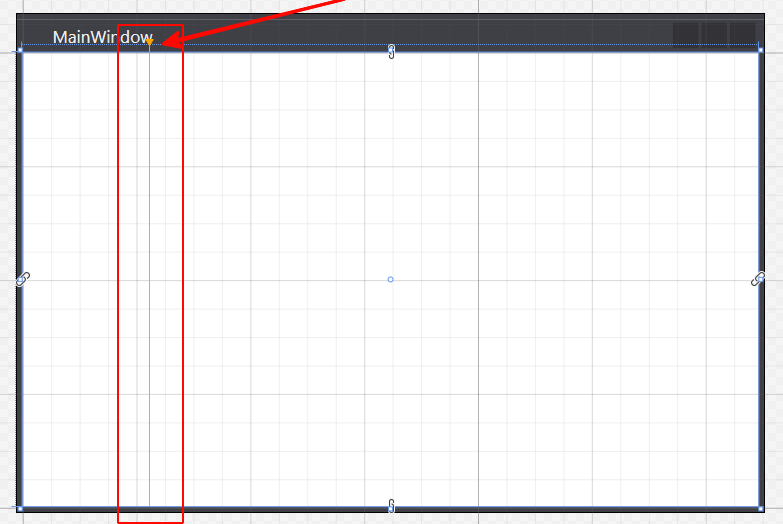




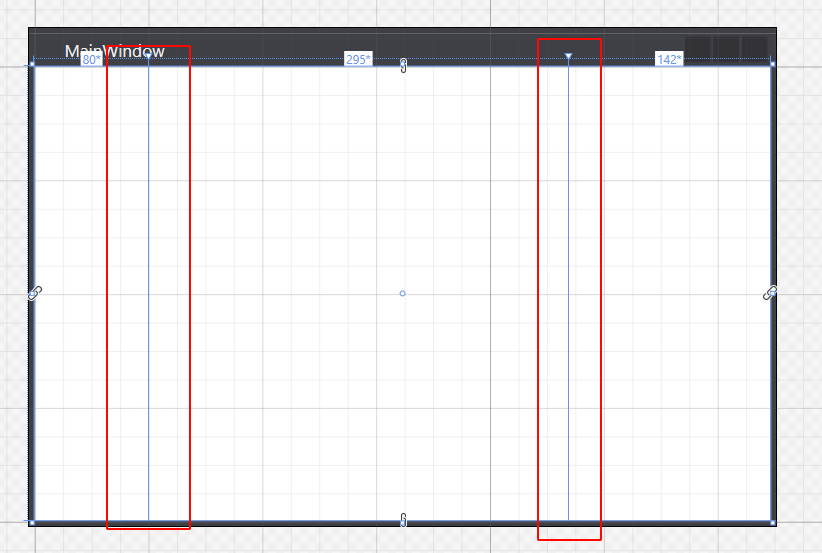
РАЗМЕЩЕНИЕ КОНТЕНТА ПО ЦЕНТРУ ФОРМЫ

Откройте форму или страницу. Выберите компонент Grid, после чего по краям (сверху и слева) появится возможность добавлять строки и столбцы.

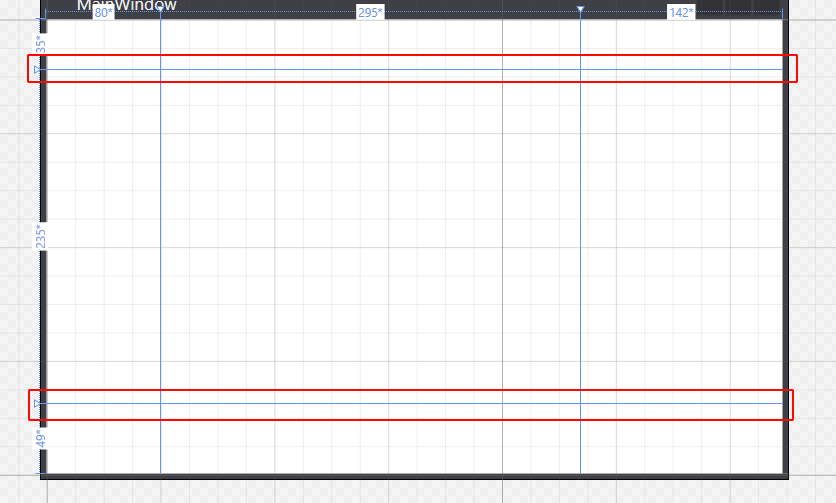
Добавим первый столбец.



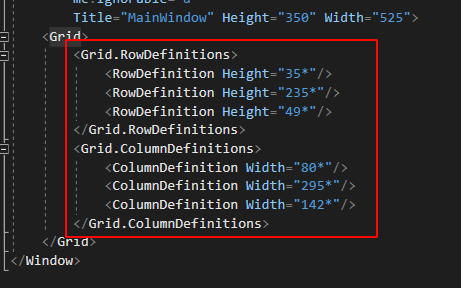
Добавим еще один столбец.



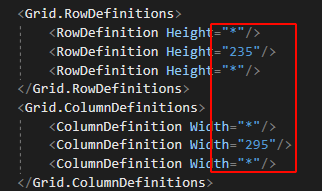
Таким же образом добавим две строки.



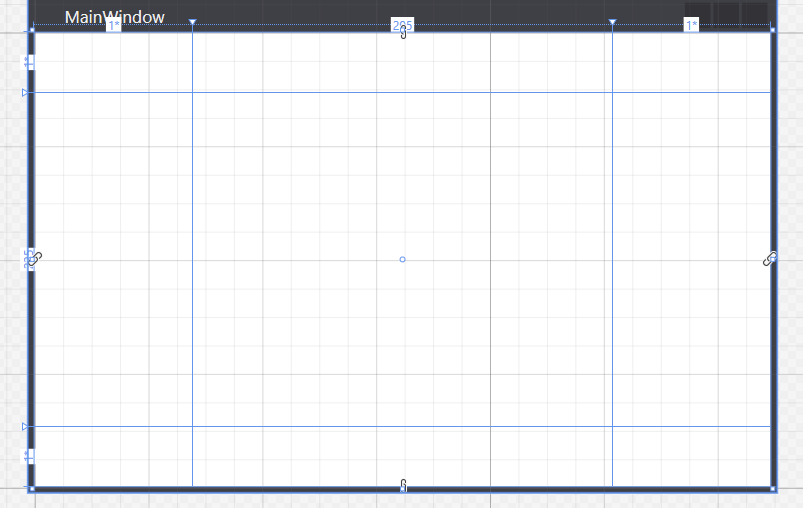
После всех добавление мышкой, мы можем увидеть следующий код XAML.



Так же столбцы и строки можно добавлять, редактируя код XAML. Здесь же можно редактировать размер строк и столбцов. Чтобы разместить элементы по центру (которые будут во второй строки и во втором столбце) необходимо задать размеры строкам и столбцам. Зададим столбцам и строкам следующие размеры.



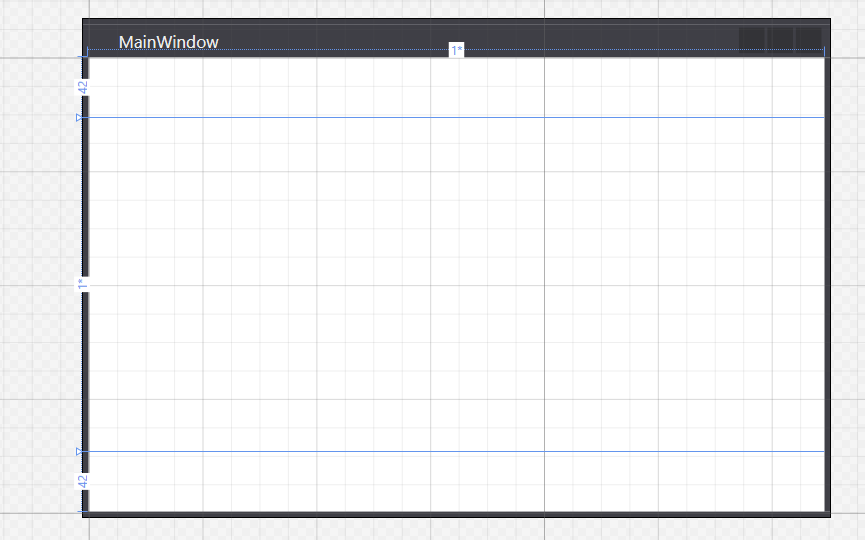
После чего на форме увидим следующее.

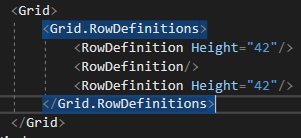


В результате средний столбец и средняя строка имеют фиксированный размер, в то время как последние и первые имеют динамический размер (меняется в зависимости от размера контейнера, окна).

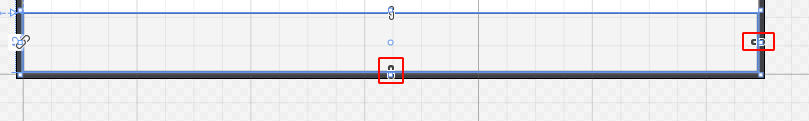
СОЗДАНИЕ БАЗОВОЙ ФОРМЫ

Создадим несколько строк с фиксированными размерами.





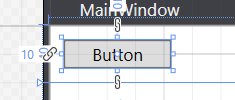
Добавим компонент «Rectangle» на форму и разместим его в 3 строке. Компонент необходимо растянуть на всю ширину и высоту строки, а затем закрепить высоту и ширину (чтобы он мог растягиваться)



Укажем у компонента «Rectangle» свойство «StrokeThickness» равное 0, это задаст размер границы прямоугольника.

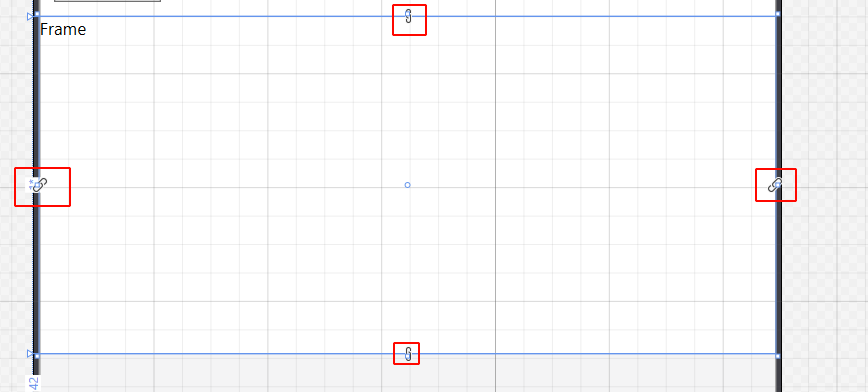


Добавим на форму кнопку и разместим в левом углу первой строки. И укажем имя кнопки «Назад», которая в последствии будет выполнять соответствующею функцию.



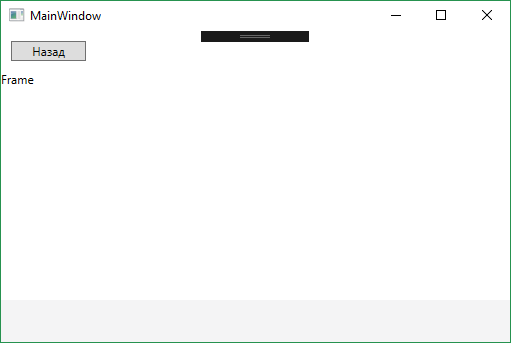


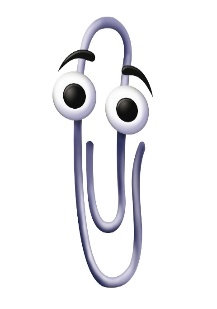
Добавим на форму компонент «Frame» во вторую строку, в него будут загружать страницы (Авторизация, Регистрация и другие). Закрепим компонент по высоте и ширине. Укажем имя ему «MainFrame»



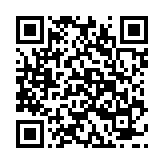


Запустим проект и увидим примерно следующее.



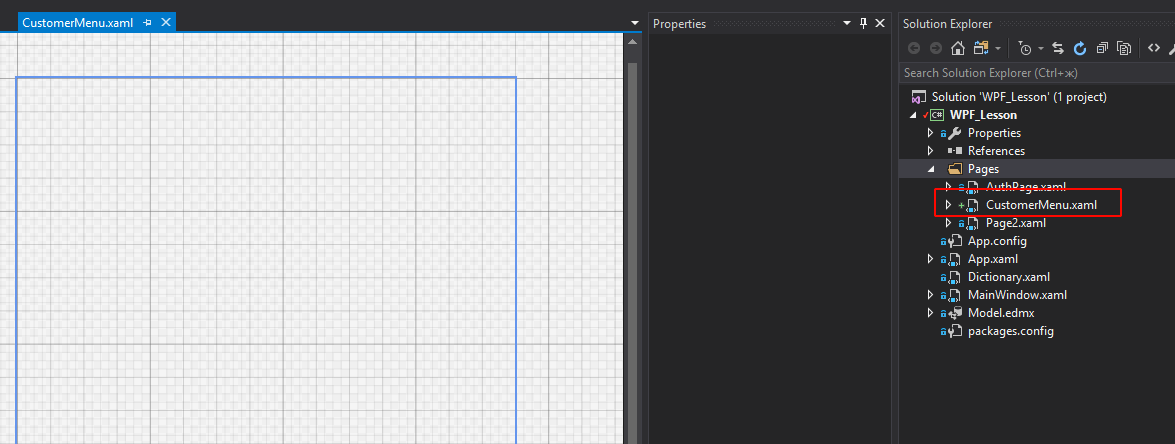


Для выполнения практического задания можно ознакомиться с обучающими видео

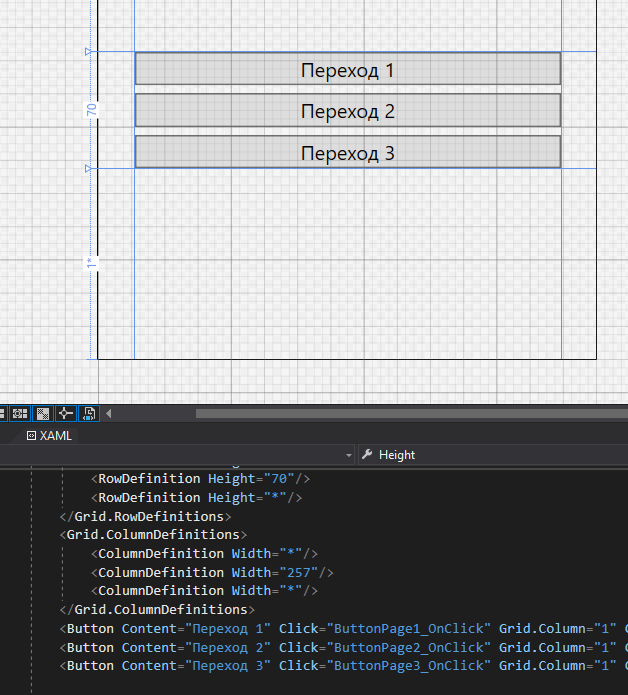


СОЗДАНИЕ МЕНЮ ПОЛЬЗОВАТЕЛЯ

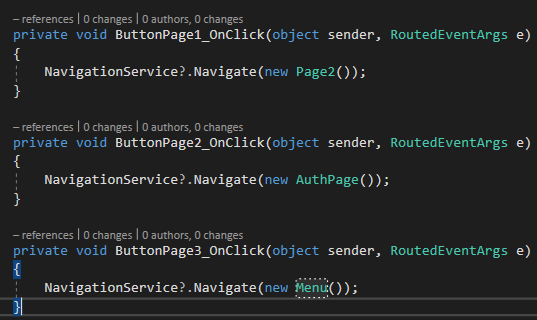
Создадим новую страницу, назовем ее CustomerMenu.

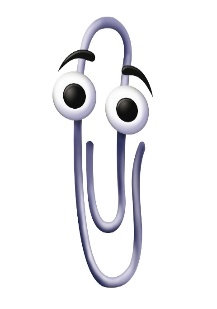


Добавим несколько кнопок, разместим их по центру и добавим им обработчик события клик.



И в коде каждому обработчику укажем свой переход



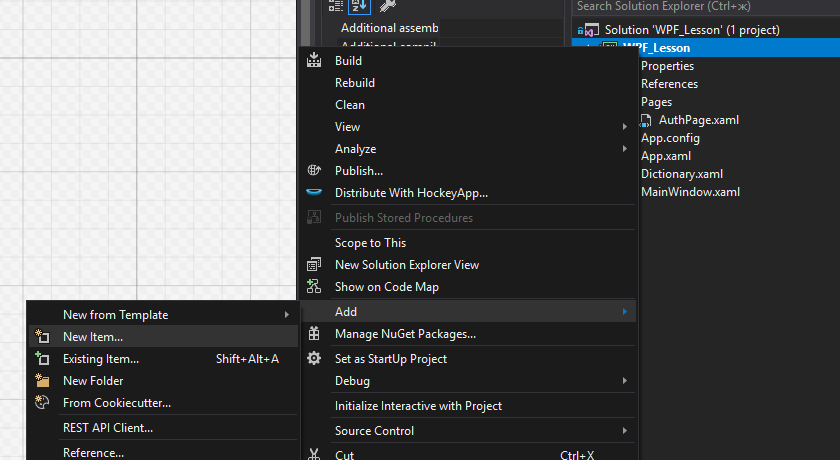


Для выполнения практического задания можно ознакомиться с обучающими видео

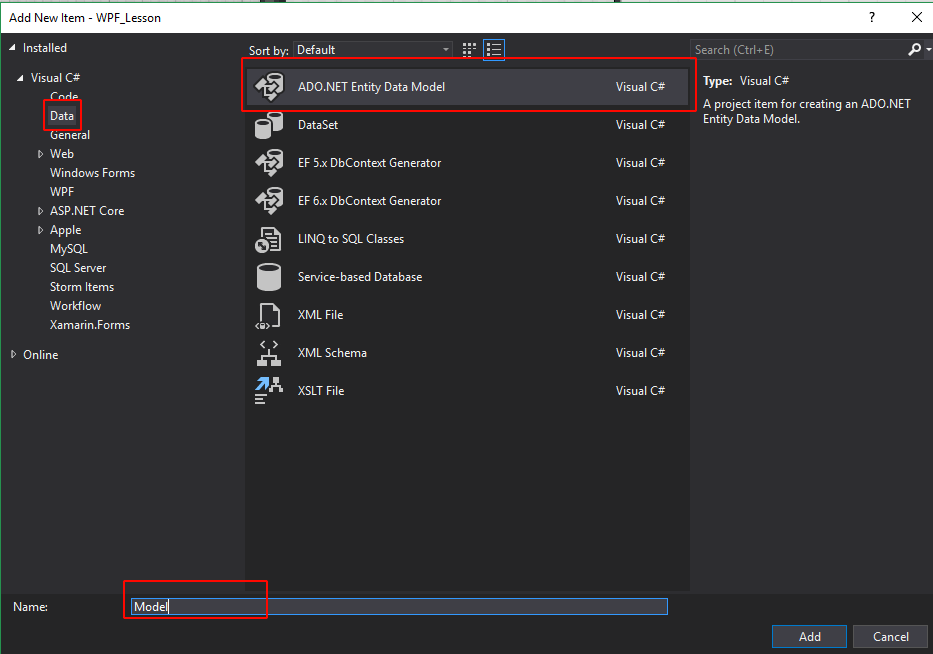


СОЗДАНИЕ ПОДКЛЮЧЕНИЯ К БАЗЕ ДАННЫХ

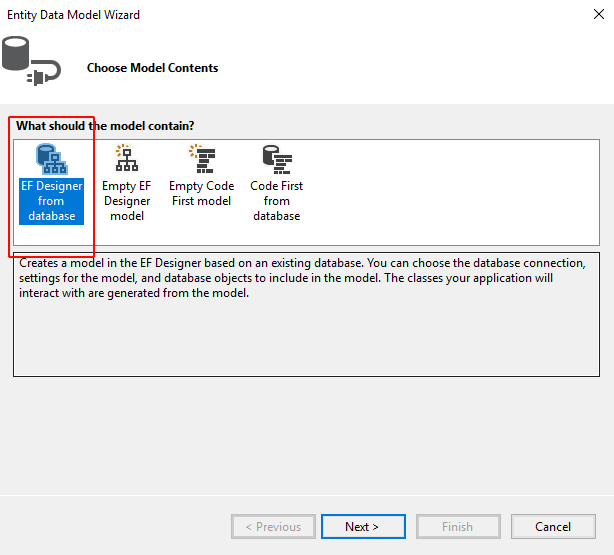
Добавим в проект новый объект



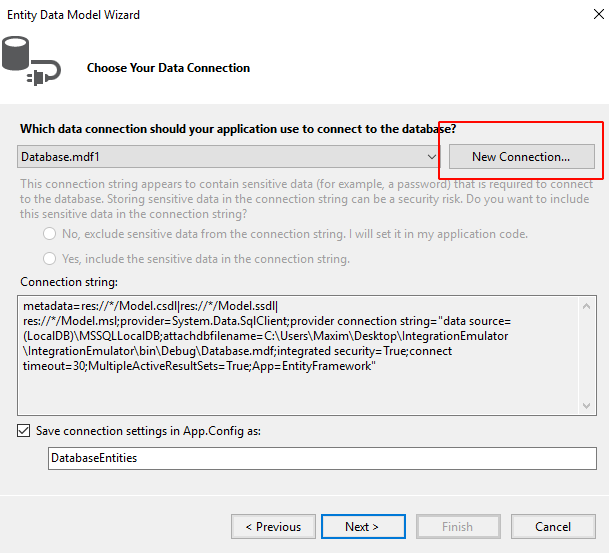
Выберем Data – ADO.NET Entity Data Model, назовем Model и добавим.



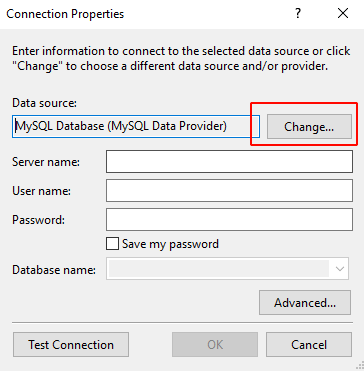
Появится диалоговое окно и выберем пункт «EF Designer from database»



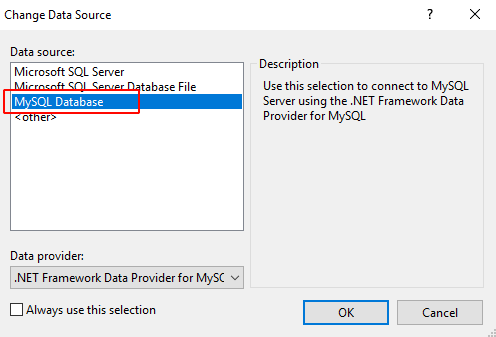
После появится следующее окно. Нажмем «New Connection»



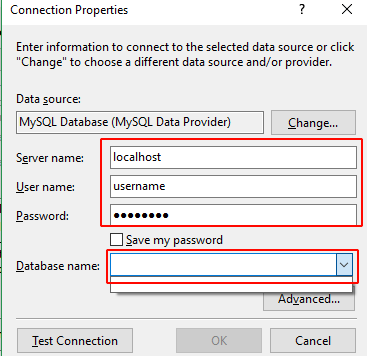
В данном окне выберем «Change».



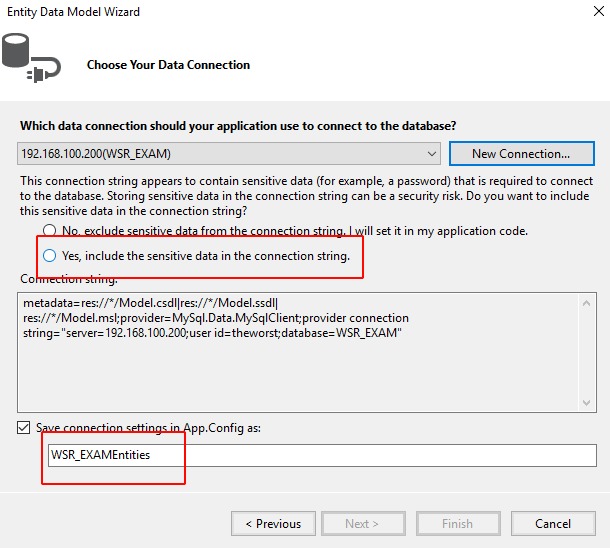
В следующем окне выберем «MySQL Database» и нажмем «ОК».



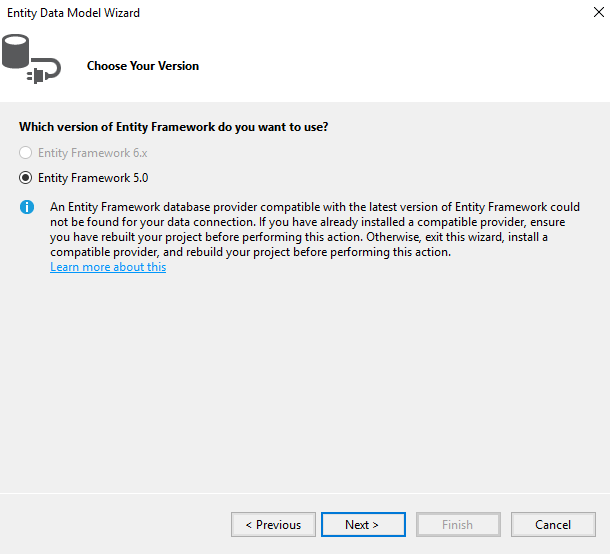
После выбора введем данные сервера: IP адрес, логин и пароль. После выберем нужную базу данных.



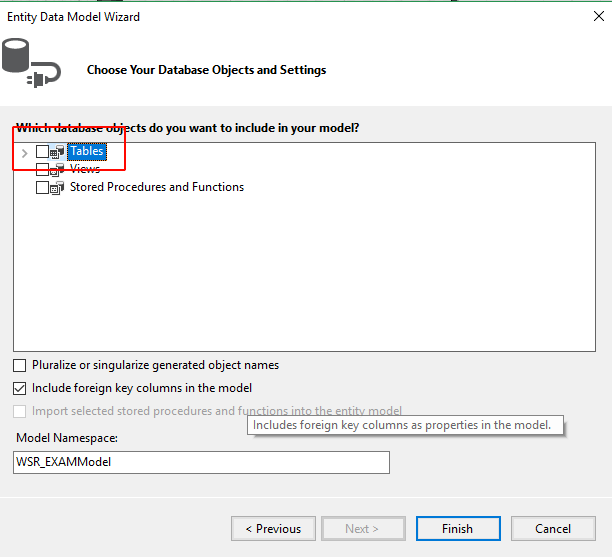
На следующем окне, выберем созданное подключение и поставим «Yes, include the sensitive data in the connection string», а также укажем имя нашего объекта базы данных «Entitites».



В следующем окне оставим по умолчанию:

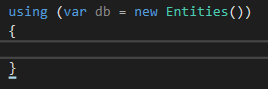


В этом окне выберем таблицы, которые хотим использовать в проекте:

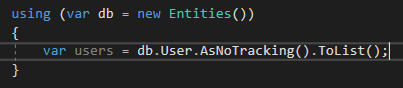


ПОЛУЧЕНИЕ ДАННЫХ ИЗ БАЗЫ ДАННЫХ

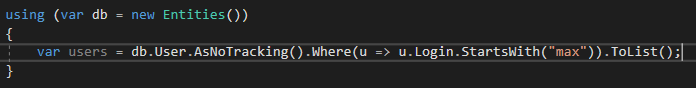
Для подключения к базе данных необходимо создать контекст.



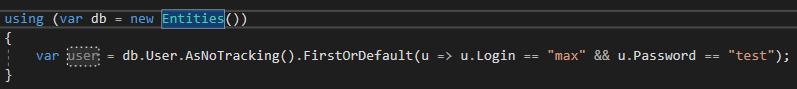
Теперь загрузим всю таблицу пользователей.

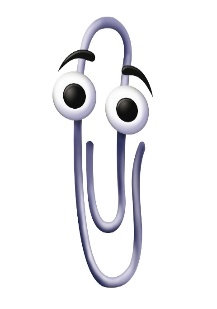


Получим пользователей по определенному критерию



Получим пользователя по определенным критериям



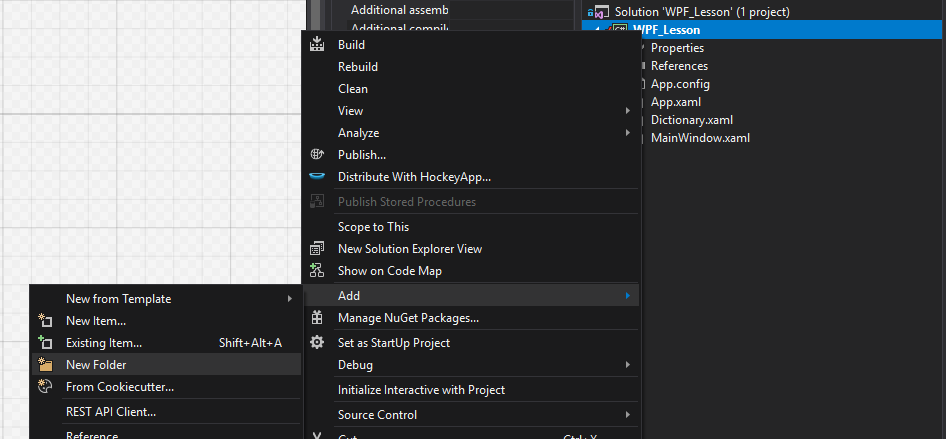


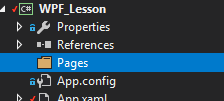
Для выполнения практического задания можно ознакомиться с обучающими видео



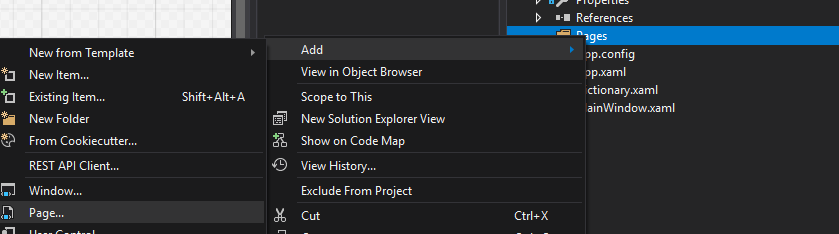
Создание формы авторизации

Добавим в проект новую папку и назовем ее «Pages», в этой папке будут находится страницы (авторизация, регистрация и другие)

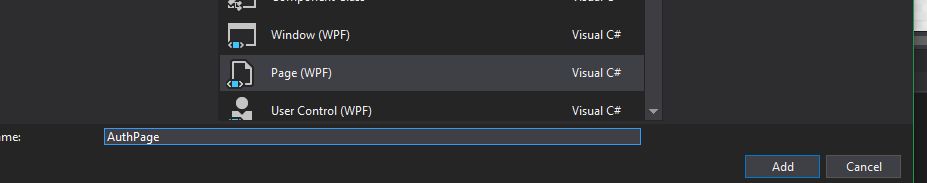


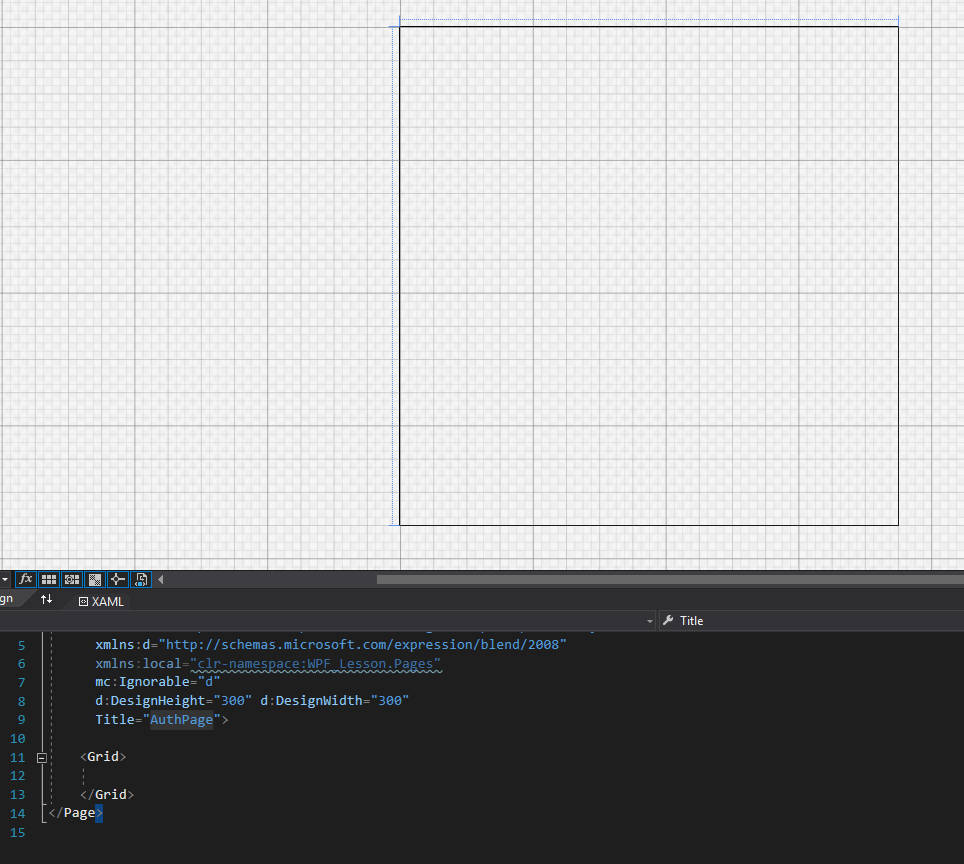


Добавим в папку новую страницу.

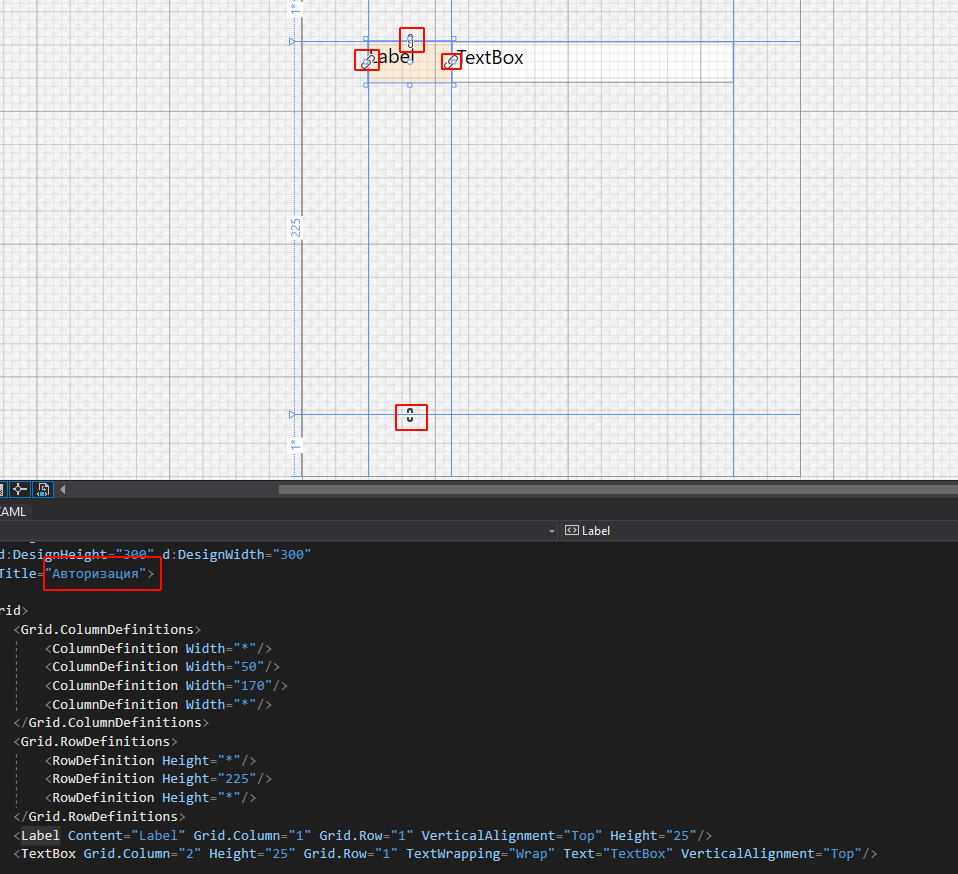


Назовем страницу AuthPage и создадим ее. После чего появится пустая страница.

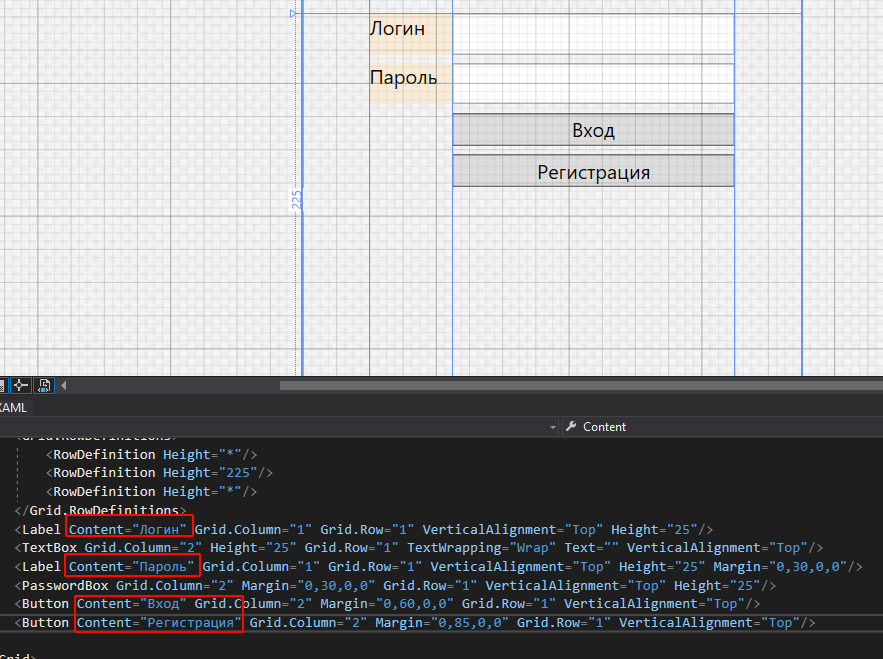




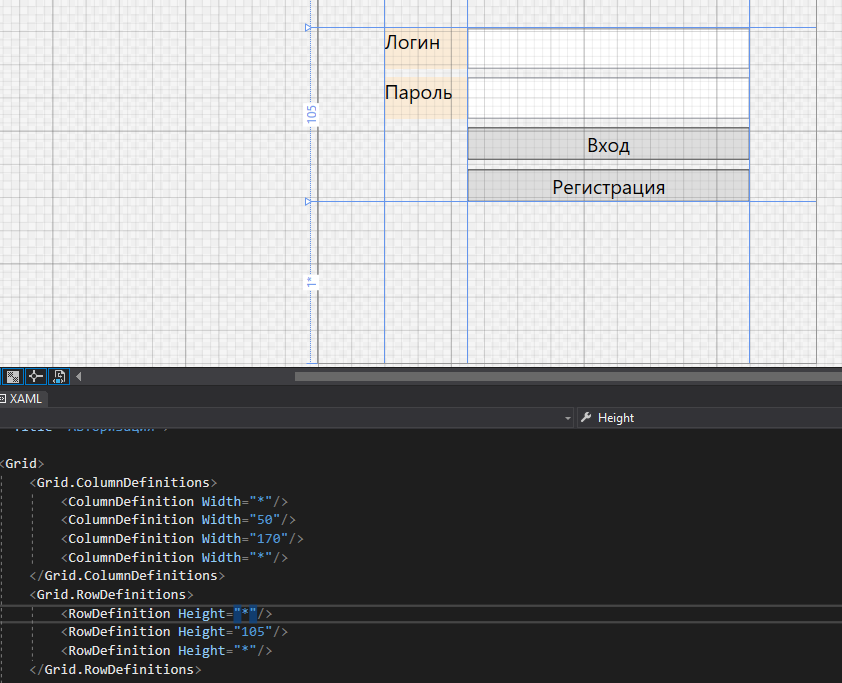
Добавим на форму компоненты «Label» и «TextBox», а затем от центруем их и добавим еще один столбец (для размещения лейблов). Также не забудем переименовать страницу, а затем закрепить лейбл и текстбокс, чтобы они растягивались по ширине.



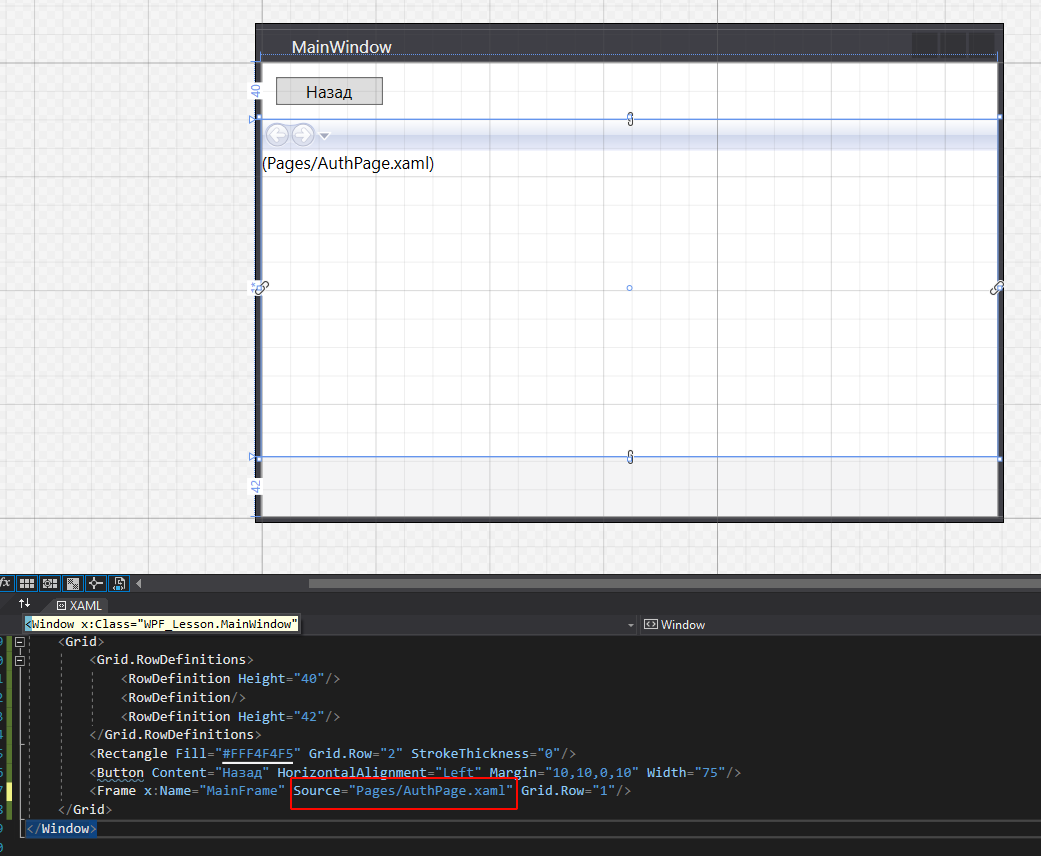
Добавим остальные компоненты по аналогии и переименуем их.



В результате получится примерно так.



Перейдем на главную форму и в компоненте «Frame» укажем в свойстве «Source» нашу страницу «AuthPage.xaml»

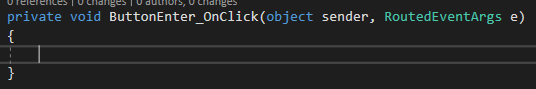


Запустим проект и увидим, что при запуске теперь отображается страница авторизации.

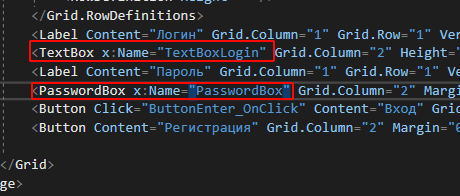


Теперь добавим функционал. Добавим обработчик события на кнопку вход.

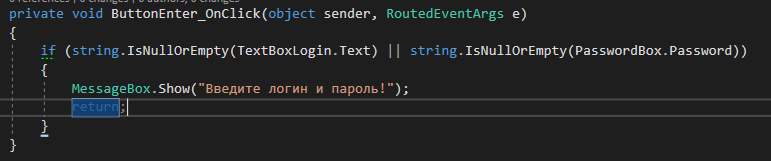




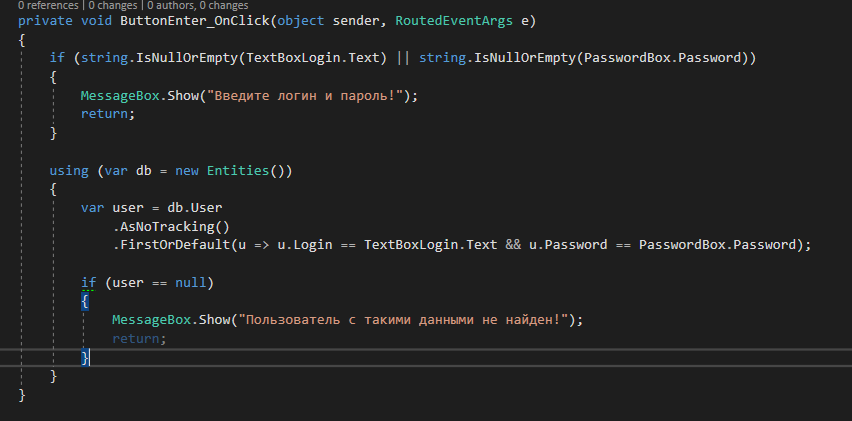
Добавим полям имена:



Добавим в код базовую проверку

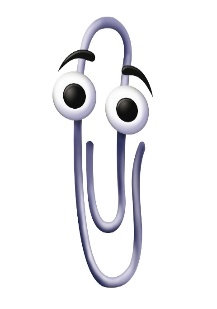


Добавим запрос к базе данных:



И теперь добавим переходы в зависимости от роли на меню пользователя (для этого необходимо создать страницы меню для каждого типа пользователя, CustomerMenu или DirectorMenu и тд)



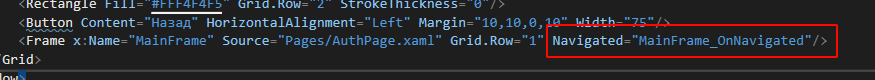


Для выполнения практического задания можно ознакомиться с обучающими видео



ПЕРЕХОД МЕЖДУ СТРАНИЦАМИ

Перейдем на базовую форму и создадим обработчик события Navigated у Frame.



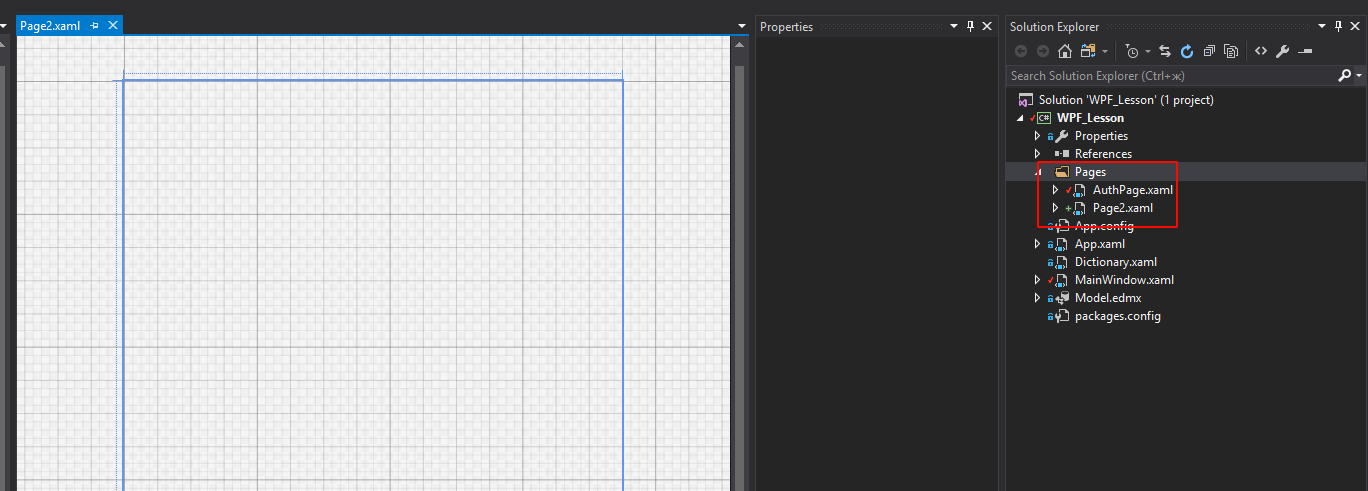
В обработчик события напишем следующий код. Сначала мы проверяем что получили ли мы страницу на вход, затем устанавливаем заголовок формы в соответствии с шаблоном, после в зависимости от страницы отображаем или скрываем кнопку «Назад».



А в обработчик события нажатия кнопки «Назад». При нажатии на кнопку будет выполнен переход назад, если такой возможен.



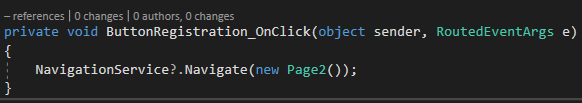
Добавим еще одну страницу.



Теперь выполним переход со страницы AuthPage на Page2. Перейдем на страницу AuthPage и добавим обрабочик события на кнопку «Регистрация»:



Напишем следующий код в обработчике:



После этого переход по кнопке будет осуществляться на страницу Page2, а по кнопке «Назад» обратно на AuthPage.