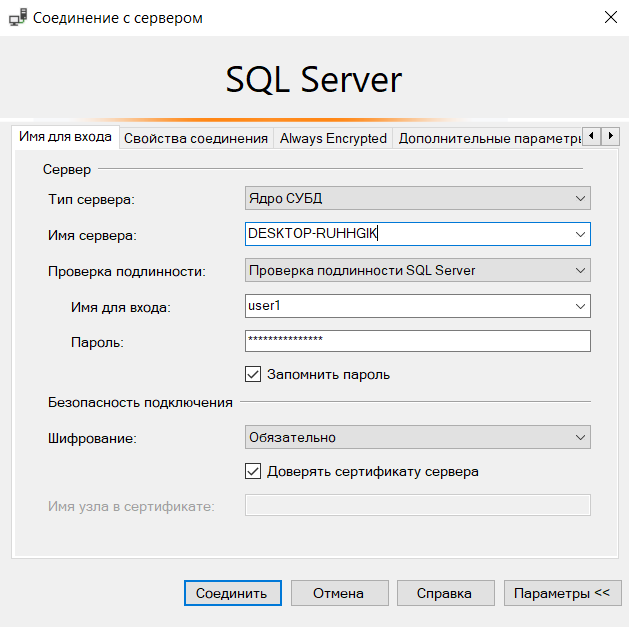
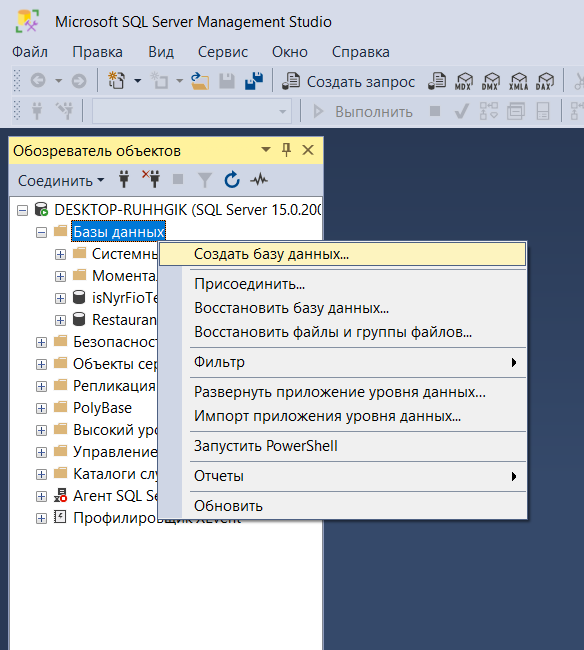
# **ч 2: Соадминистрирование баз данных и серверов**

Создание БД в sql server management studio

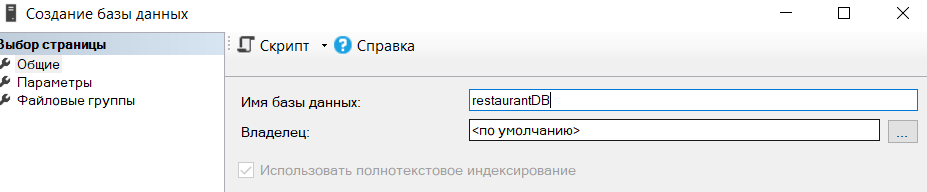
1. Вводим имя для входа и пароль. Устанавливаем соединение с сервером



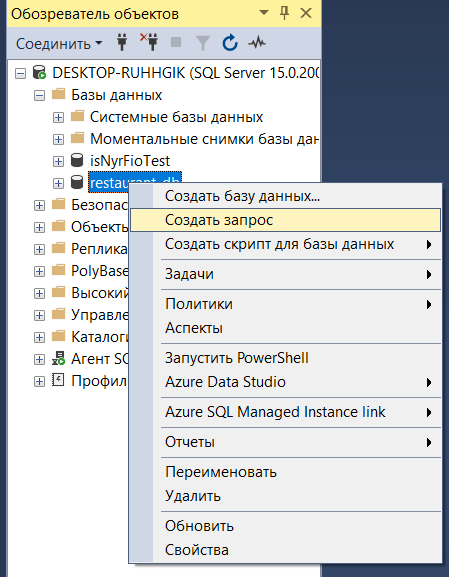
1. В обозревателе объектов выбираем «Базы данных», нажимаем правой кнопкой мыши, выбираем «Создать базу данных»



1. Заполняем поле «Имя БД», нажимаем «ОК»



1. Заполняем БД сущностями и атрибутами. Выбираем созданную БД, нажимаем правой кнопкой, выбираем «Создать запрос»



1. В открывшимся окне заполняем таблицу сущностями и атрибутами. Нажимаем на кнопку «Выполнить»

-- Таблица ролей

CREATE TABLE role (

role\_id INT PRIMARY KEY IDENTITY,

role\_nazvanie VARCHAR(250) NOT NULL

);

-- Таблица пользователей

CREATE TABLE polzovateli (

polzovatel\_id INT PRIMARY KEY IDENTITY,

imya\_polzovatelya VARCHAR(250) NOT NULL,

telefon VARCHAR(250) NOT NULL,

login VARCHAR(250),

parol VARCHAR(250) NOT NULL,

);

--Таблица сотрудники

CREATE TABLE sotrudniki (

sotrudnik\_id INT PRIMARY KEY IDENTITY,

imya\_sotrudnika VARCHAR(250) NOT NULL,

login VARCHAR(250),

parol VARCHAR(250) NOT NULL,

role\_polzovatel INT FOREIGN KEY REFERENCES role(role\_id)

);

-- Таблица меню

CREATE TABLE menu (

menu\_id INT PRIMARY KEY IDENTITY,

kategoriya VARCHAR(250) NOT NULL,

nazvanie\_bluda VARCHAR(250) NOT NULL,

ingredient VARCHAR(250),

cena float NOT NULL

);

-- Таблица запасов

CREATE TABLE zapas (

produkt\_id INT PRIMARY KEY IDENTITY,

imya\_produkta VARCHAR(250) NOT NULL,

kolichestvo FLOAT NOT NULL,

minimal\_ostatok float NOT NULL

);

-- Таблица поставщиков

CREATE TABLE postavchik (

postavchik\_id INT PRIMARY KEY IDENTITY,

nazvanie\_postavchika VARCHAR(250) NOT NULL,

);

-- Вставка данных в роли

INSERT INTO role (role\_nazvanie) VALUES

('Администратор'), ('Сотрудник');

-- Вставка данных в меню

INSERT INTO menu (kategoriya, nazvanie\_bluda, ingredient, cena ) VALUES

('Закуски', 'Цезарь с курицей', 'романо, курица, пармезан, гренки, соус Цезарь', 350.00),

('Закуски', 'Брускета с томатами', 'хлеб, помидоры, базилик, оливковое масло', 250.00),

('Закуски', 'Креветки в чесночном соусе', 'креветки, чеснок, оливковое масло, лимон', 450.00),

('Основные блюда', 'Паста Карбонара', 'паста, бекон, яйца, пармезан, черный перец', 400.00),

('Основные блюда', 'Стейк Рибай', 'говядина, соль, перец, оливковое масло', 1200.00),

('Основные блюда', 'Курица в сливочном соусе', 'куриное филе, сливки, грибы, специи', 500.00),

('Десерты', 'Торт Наполеон', 'слоеное тесто, крем, ваниль', 300.00),

('Десерты', 'Мороженое ассорти', 'молоко, сахар, ваниль, фрукты', 200.00),

('Напитки', 'Красное вино', '750 мл', 600.00),

('Напитки','Мохито', 'ром, мята, лайм, сода, сахар', 350.00);

-- Вставка данных в поставщиков

INSERT INTO postavchik(nazvanie\_postavchika) VALUES

('Птицефабрика "Здоровье"'),

('Мясокомбинат "Вкусно"'),

('Овощная база "Солнечная"'),

('Молочный завод "Молоко"'),

('Сахарный завод "Сладость"'),

('Птицефабрика "Яйцо"'),

('Винодельня "Солнечная"'),

('Итальянские продукты');

-- Вставка данных в запасы

INSERT INTO zapas (imya\_produkta, kolichestvo, minimal\_ostatok) VALUES

('Куриное филе', 150.00, 50.00),

('Говядина', 100.00, 30.00),

('Паста', 200.00, 50.00),

('Помидоры', 80.00, 20.00),

('Огурцы', 60.00, 15.00),

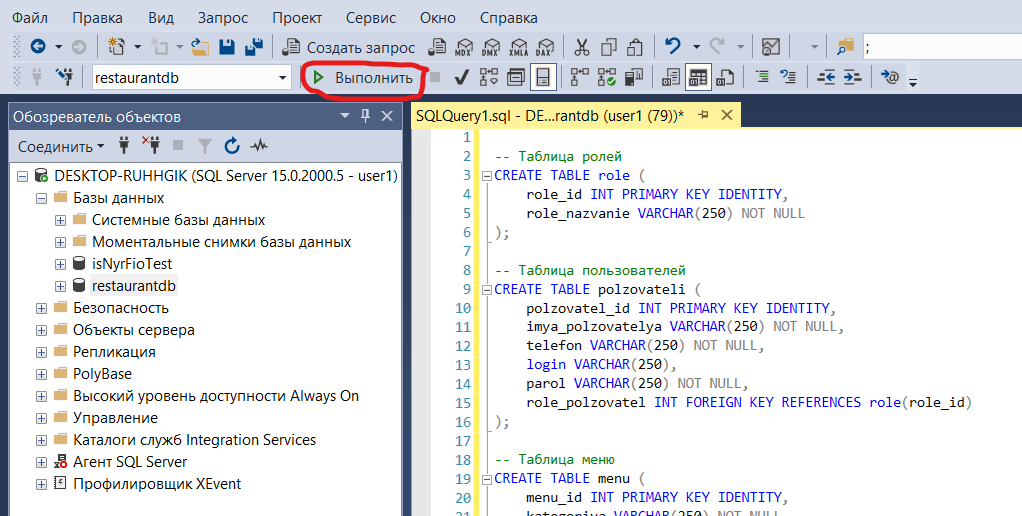
('Сливки', 50.00, 10.00),

('Яйца', 200.00, 50.00),

('Сахар', 100.00, 20.00),

('Вино красное', 50.00, 10.00),

('Молоко', 100.00, 20.00);



1. Переносим данные в таблицу «Пользователи» из файла excel.

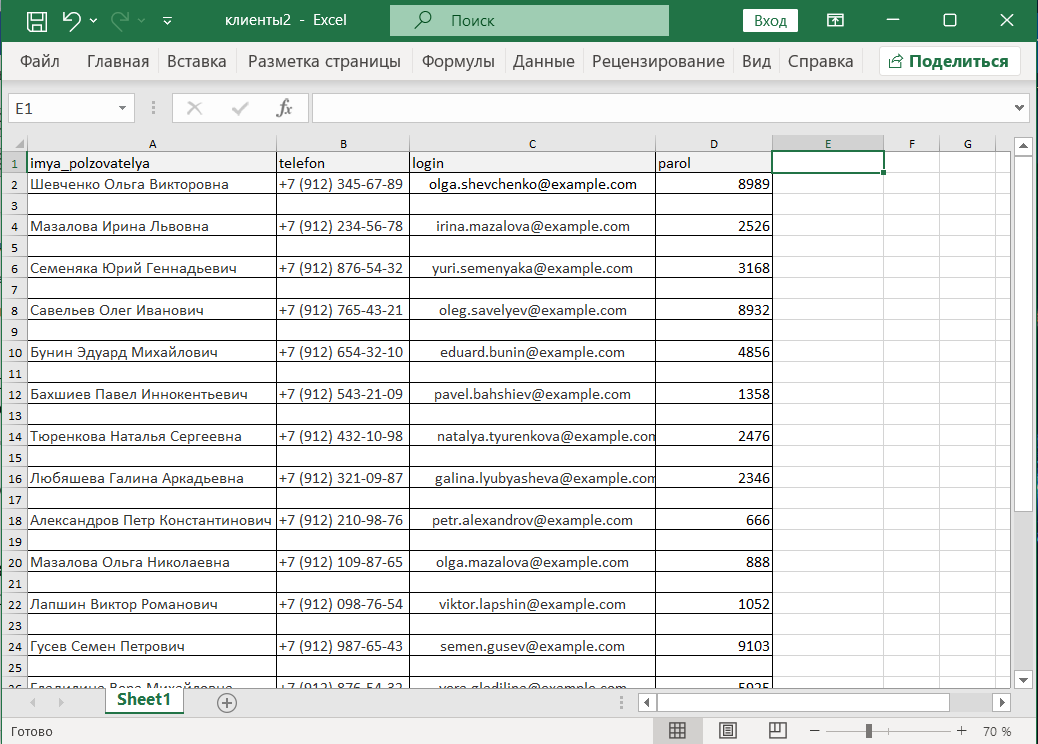
Есть несколько вариантов.

Первый - делаем запрос и заполняем таблицу (если таблица уже была создана и ее необходимо только заполнить данными)

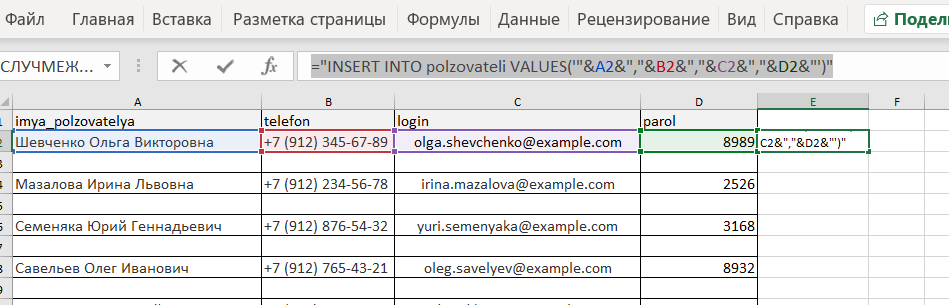
Второй – создаем таблицу через загрузку файла (если таблица не была создана)

Первый вариант

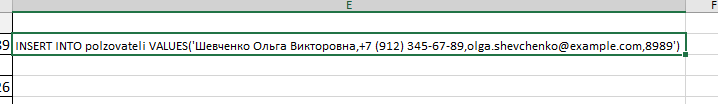
* Открываем файл excel



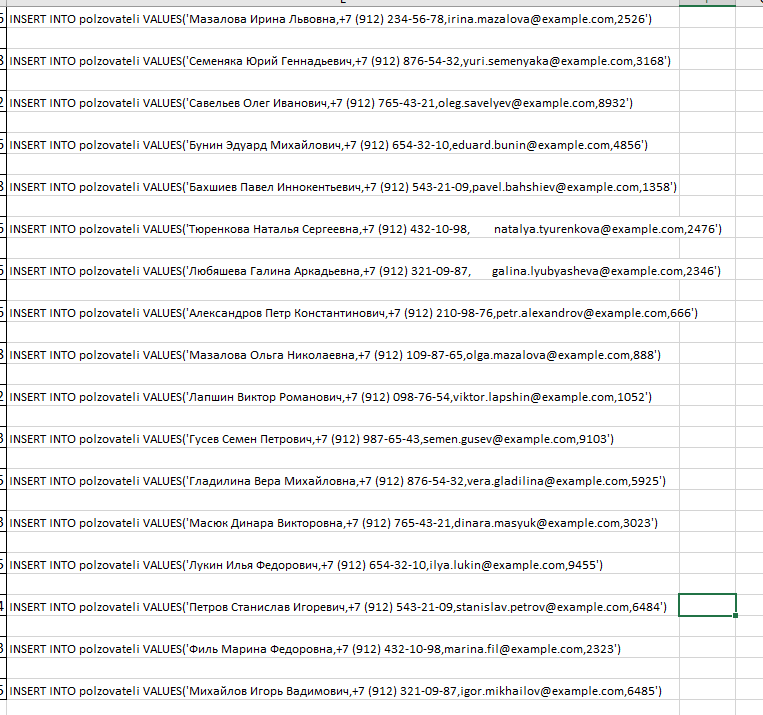
* Название столбцов excel должно соответствовать названию столбцов в таблице.
* В колонке «Е» пишем запрос ="INSERT INTO название таблицы VALUES('"&(выбираем колонку)&"')"



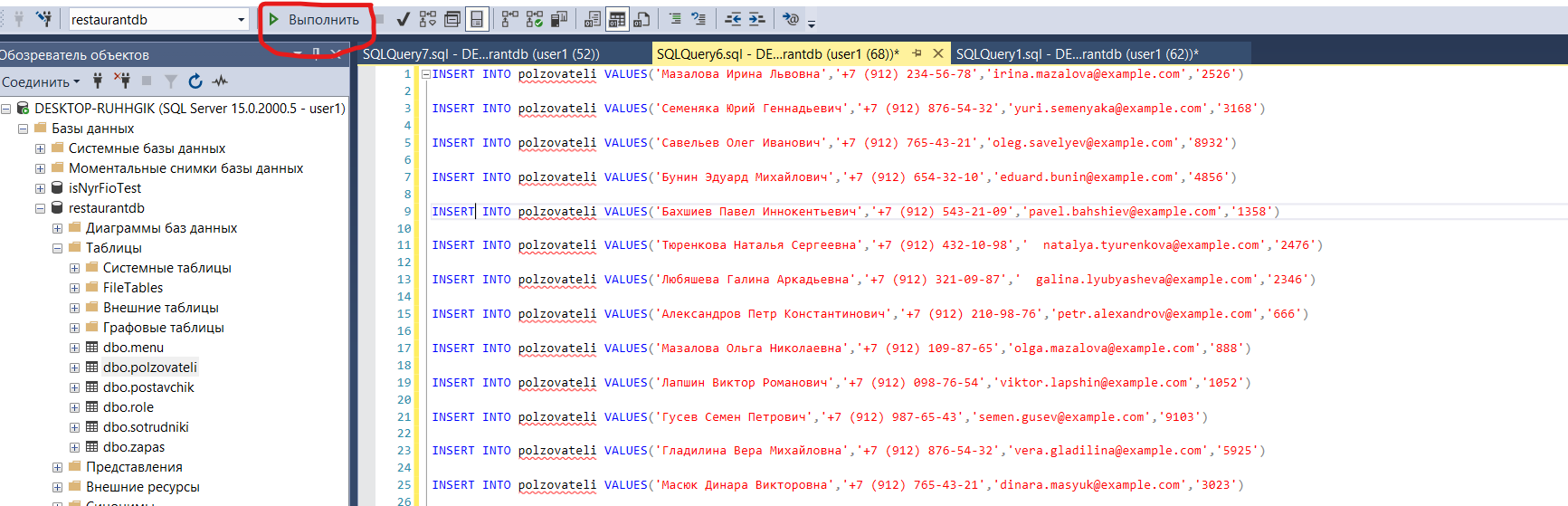
="INSERT INTO polzovateli VALUES('"&A2&"','"&B2&"','"&C2&"','"&D2&"')"

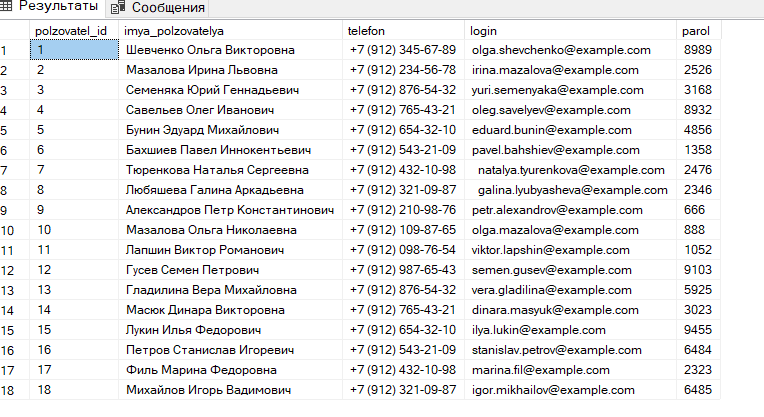


* Протягиваем формулу по странице



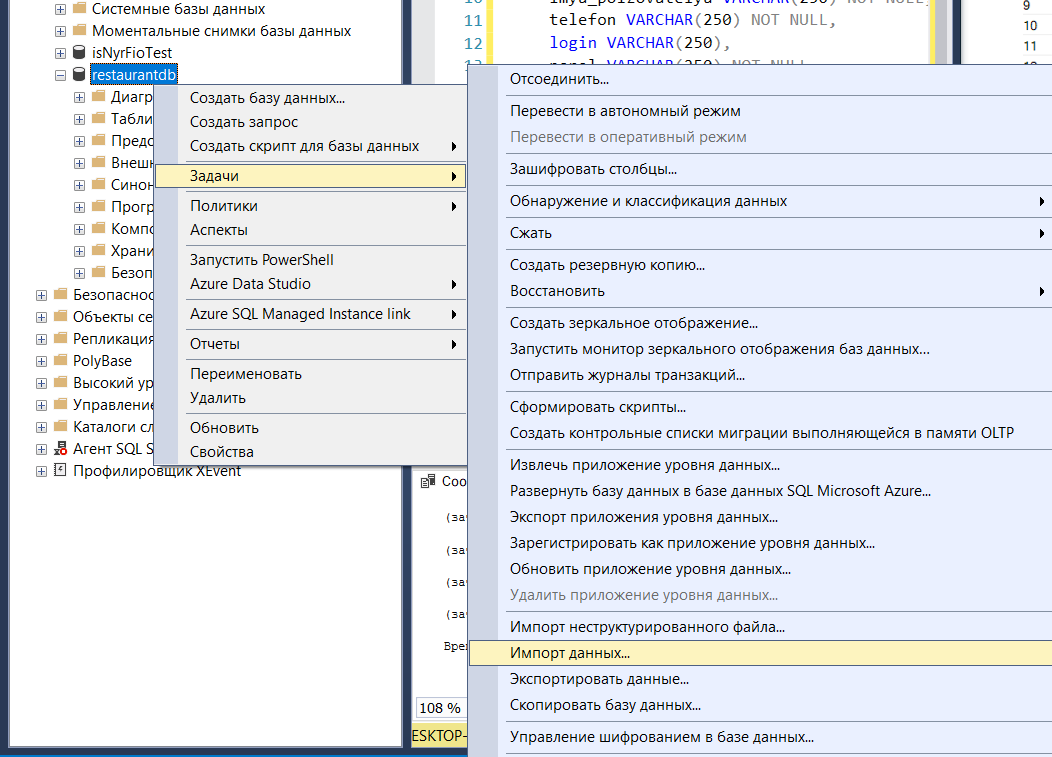
* Копируем массив данных. Делаем запрос в БД



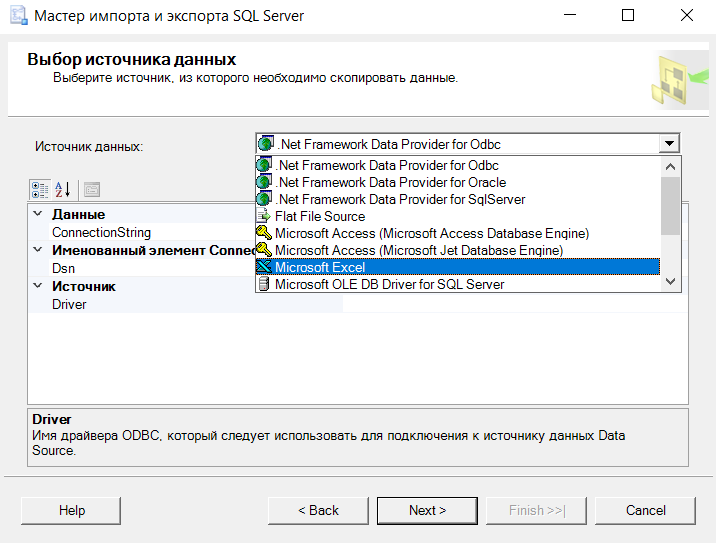


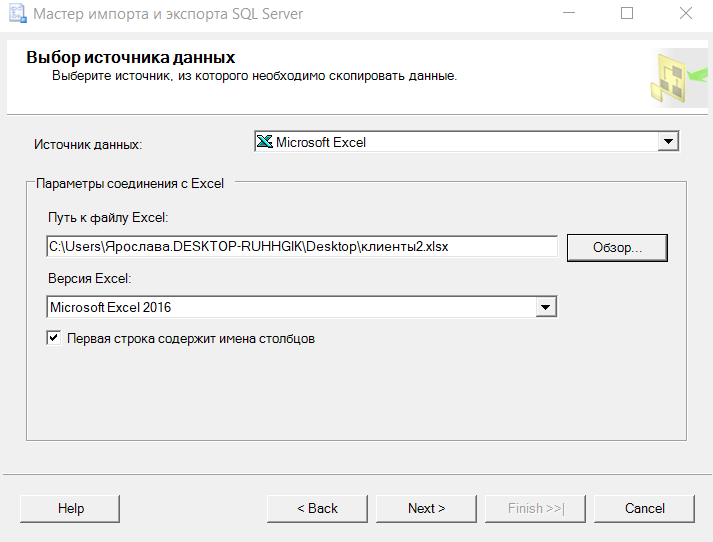
Второй вариант

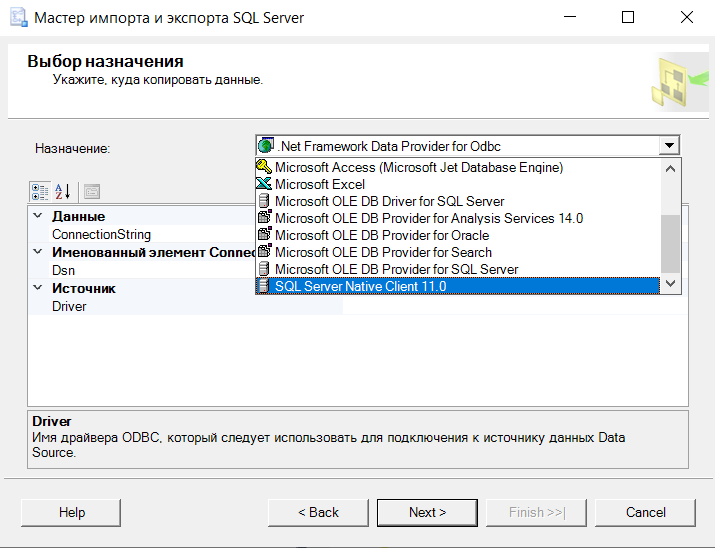
* Правой кнопкой мыши нажимаем на БД, выбираем «Задачи» - «Импорт данных»



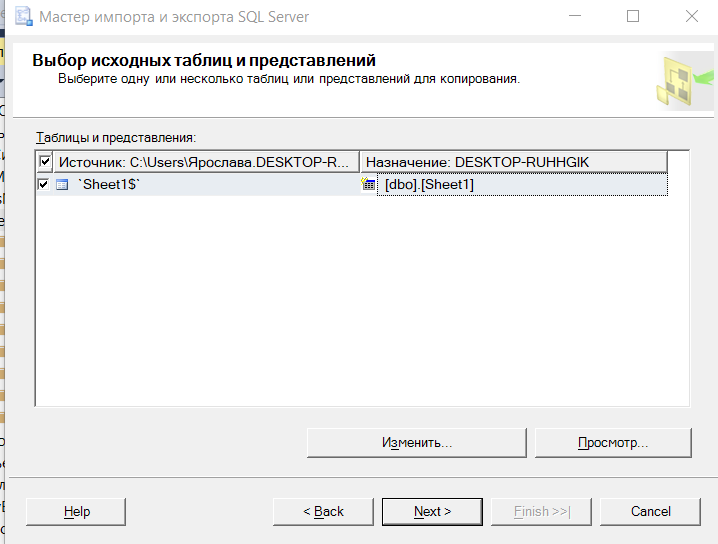
В открывшимся окне в источнике данных выбираем «Microsoft Excel» и указываем версию. Указываем путь к файлу

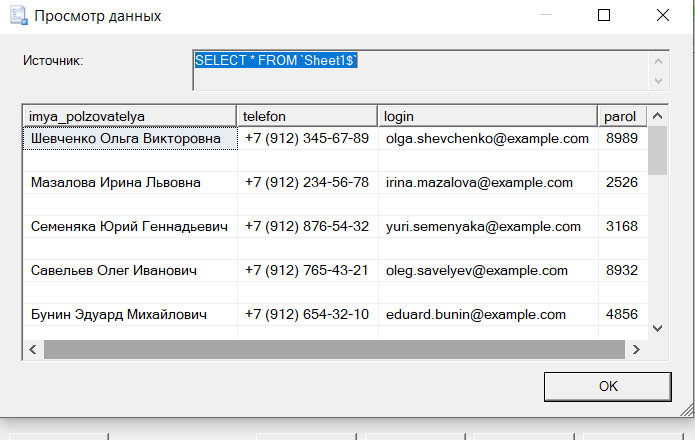




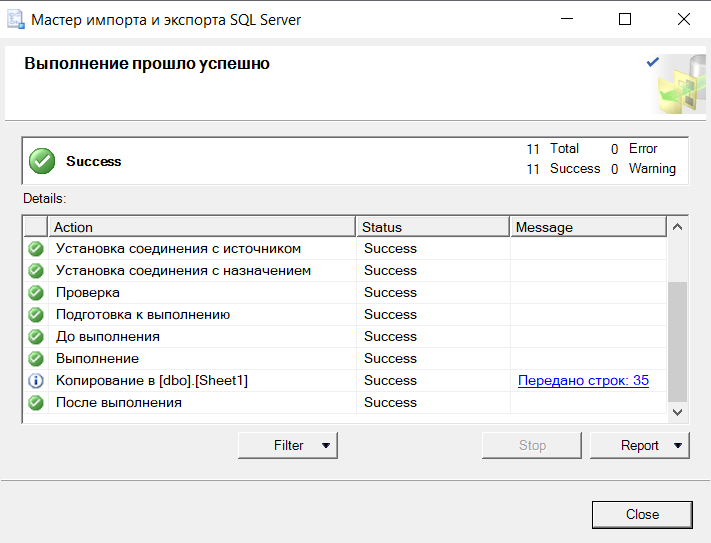


В открывшимся окне есть возможность просмотр таблицы. Можно поменять тип данных, если нажать на кнопку «изменить»

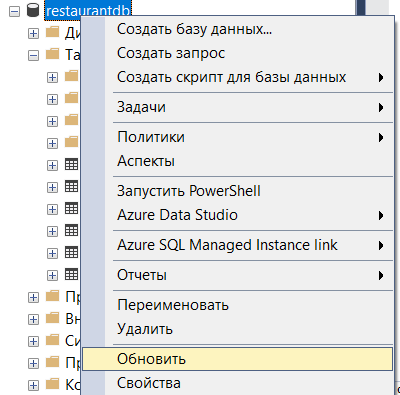


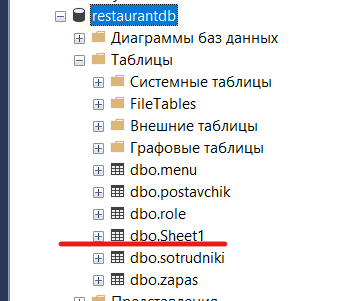


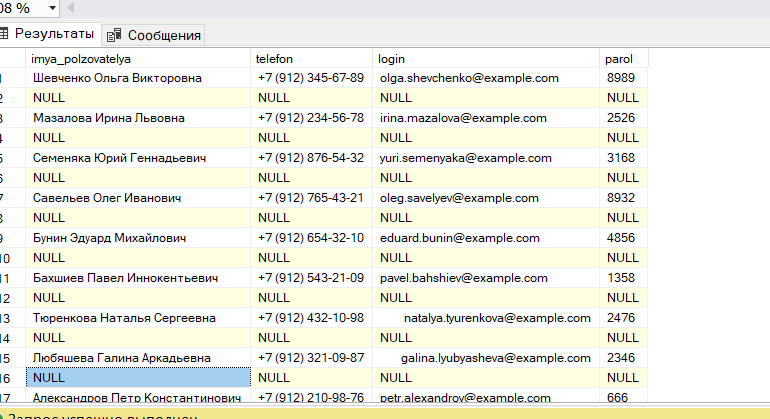
Итог:



Необходимо обновить БД, чтобы отображалась таблица. Для этого нажимаем на БД правой кнопкой мыши и выбираем «Обновить»



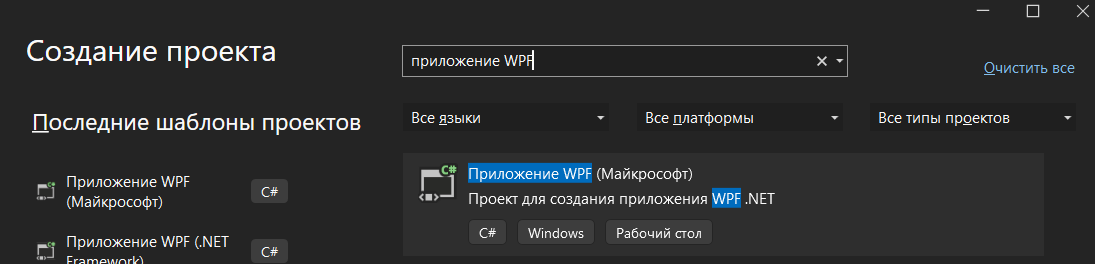


Данные в Excel были записаны через строку, поэтому, в таблице результата, через строчку есть значения «NULL». В таких случаях удобнее использовать Первый способ.

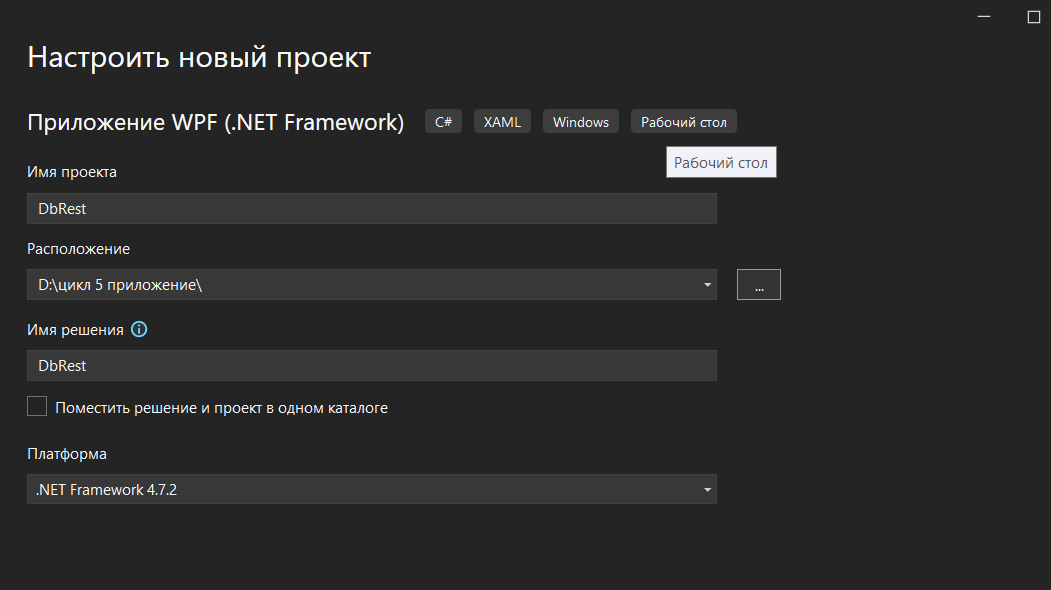
Ч 3: Проектирование и разработка информационной системы.

Разработайте форму для авторизации зарегистрированных пользователей с ролями "Администратор" и "Пользователь".

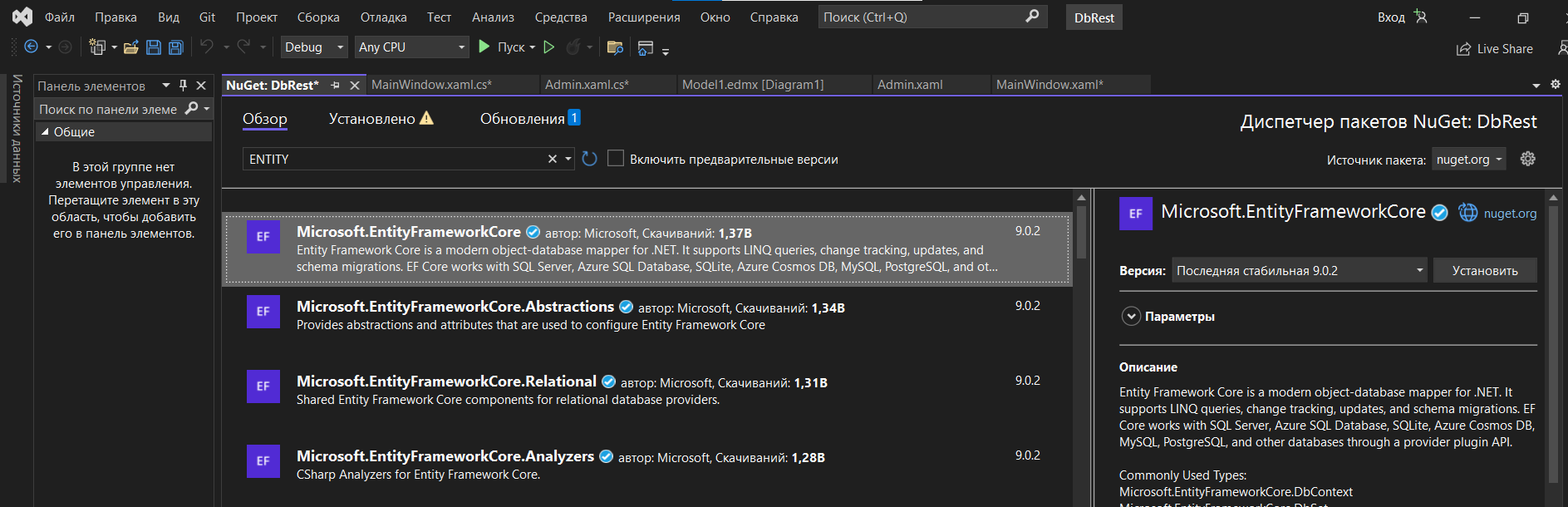
1. Открываем Visual Studio. Выбираем шаблон WPF(.NET Framework) C#



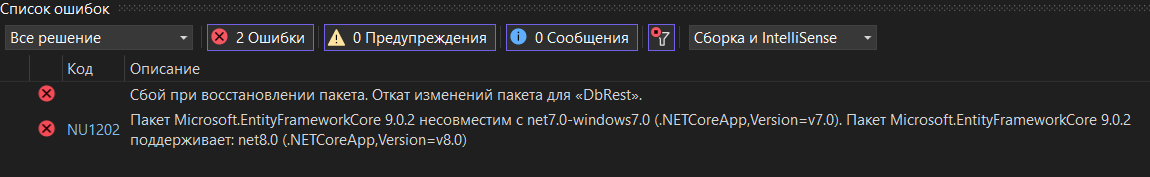
1. Называем проект, указываем путь (не называйте также как таблицы или базы данных)



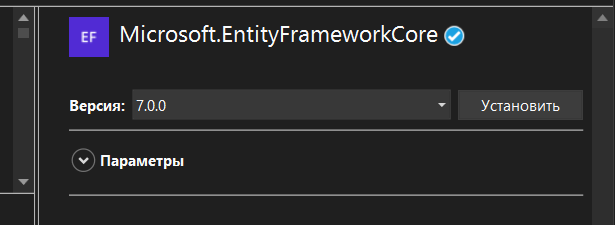
1. Устанавливаем фрейм ворк. В поиске визуал стодио пишем NUGET управление пакетами. Заходим в обзор, там пишем ENTITY скачиваем файл Microsowt.EntityFrameworkCore



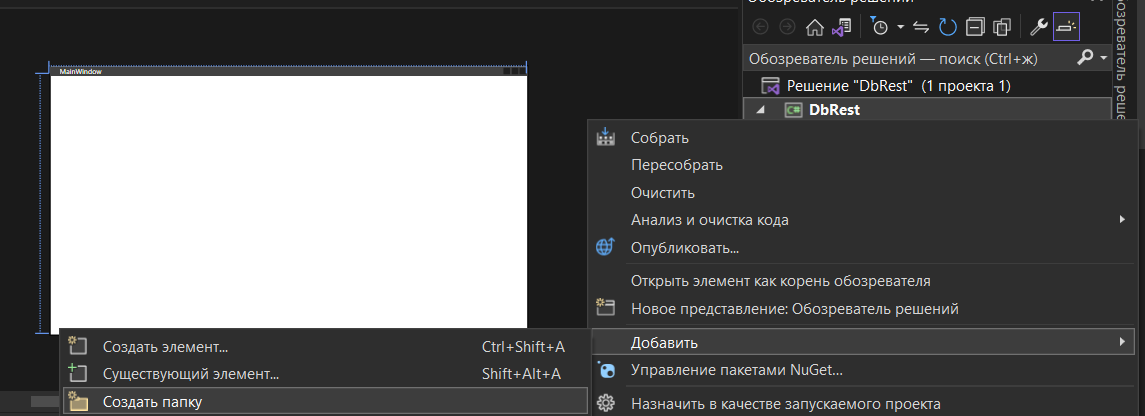
При загрузки может возникнуть ошибка при установлении пакета из-за неподдерживаемой версии языка или среды разработки. Для этого надо выбрать версию пакета, которая будет подходить под вашу версию.



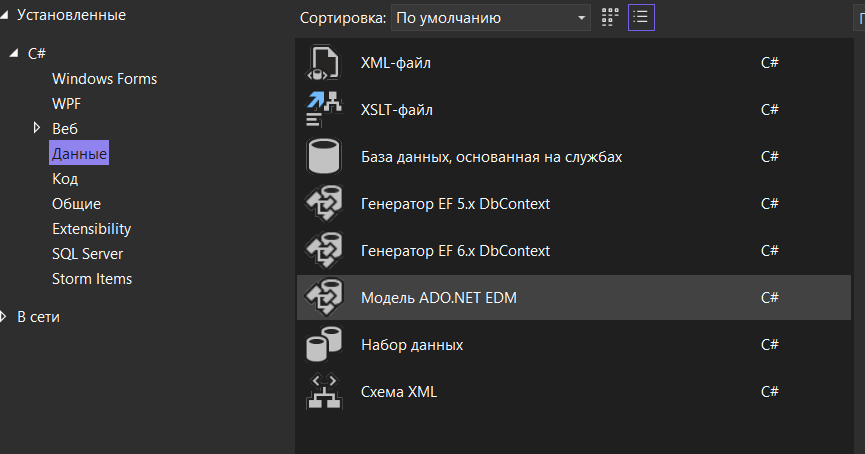
Наша версия разработки 7.0. Значит выбираем такой же версии пакет



1. Создаем отдельную папку в проекте, куда переместим все файлы, связанные с БД

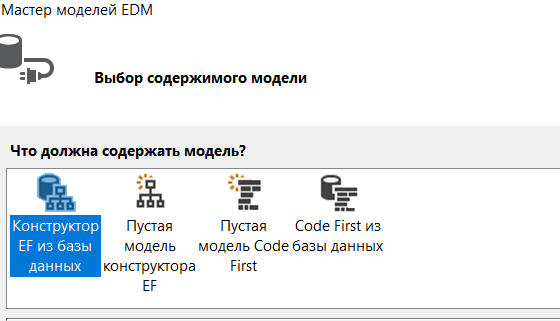


1. В папке создаем «новый элемент». Выбираем строку «Данные», из левого списка и строчку «Модель ADO.NET EDM» Также вводим название для создаваемой модели. Нажимаем «Добавить»

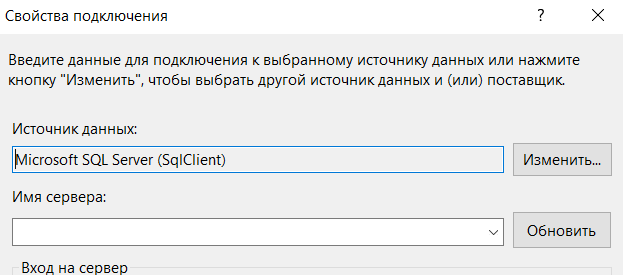


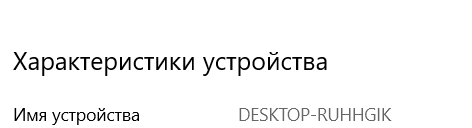


1. Выбираем «Конструктор EF из базы данных» и нажимаем «Далее».

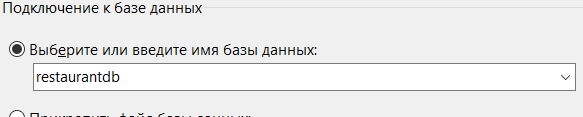


1. Нажимаем «Создать соединение». Указываем сервер (название вашего устройства). Проверяем источник данных – Microsoft SQL Server



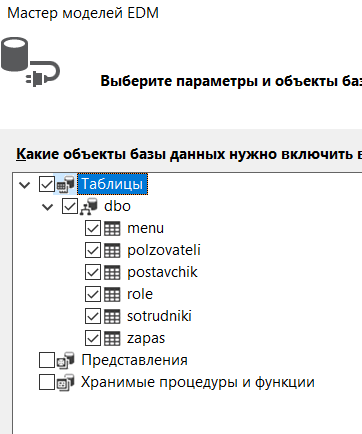


Дальше выбираем подключение к нашей БД



Нажимаем «ОК»

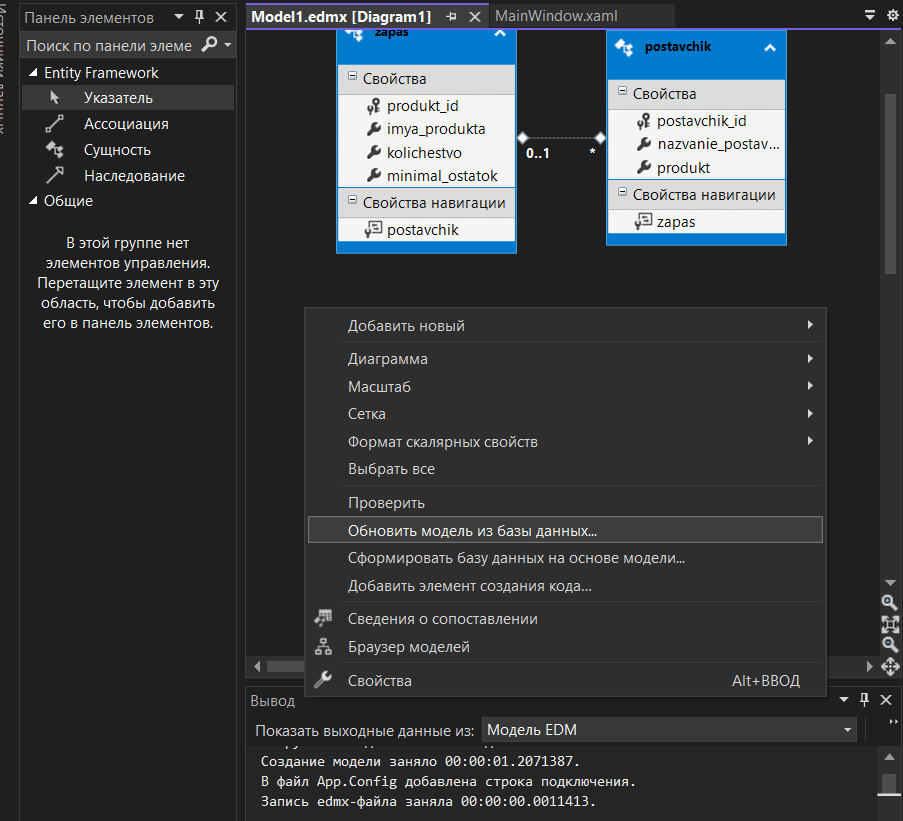
1. Нажимаем «Далее», «Далее». Ставим галочку перед «Таблицы». Нажимаем «Готово»



В случае если вы обновили свою базу данных на сервере

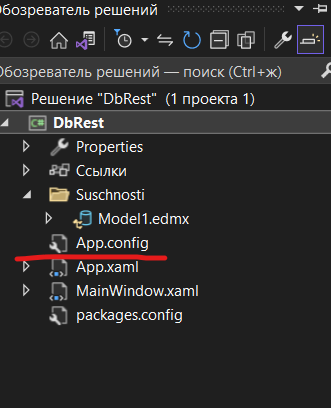
(добавили атрибуты, таблицы и т.д.), то просто кликните по пустой области ПКМ и

выберите «Обновить модель из базы данных»

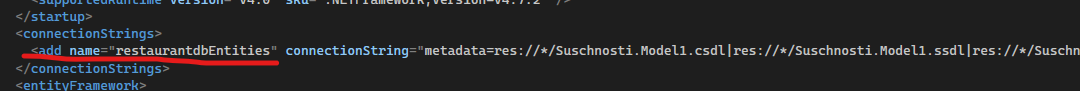


В App.config находится строка соединения с базой данных, в которой name это имя нашей БД. Также в этой строке есть указание сервера, с

который производится соединение. При наличии идентичной базы на разных серверах (дома, в колледже), этот сервер можно изменить.



Название БД



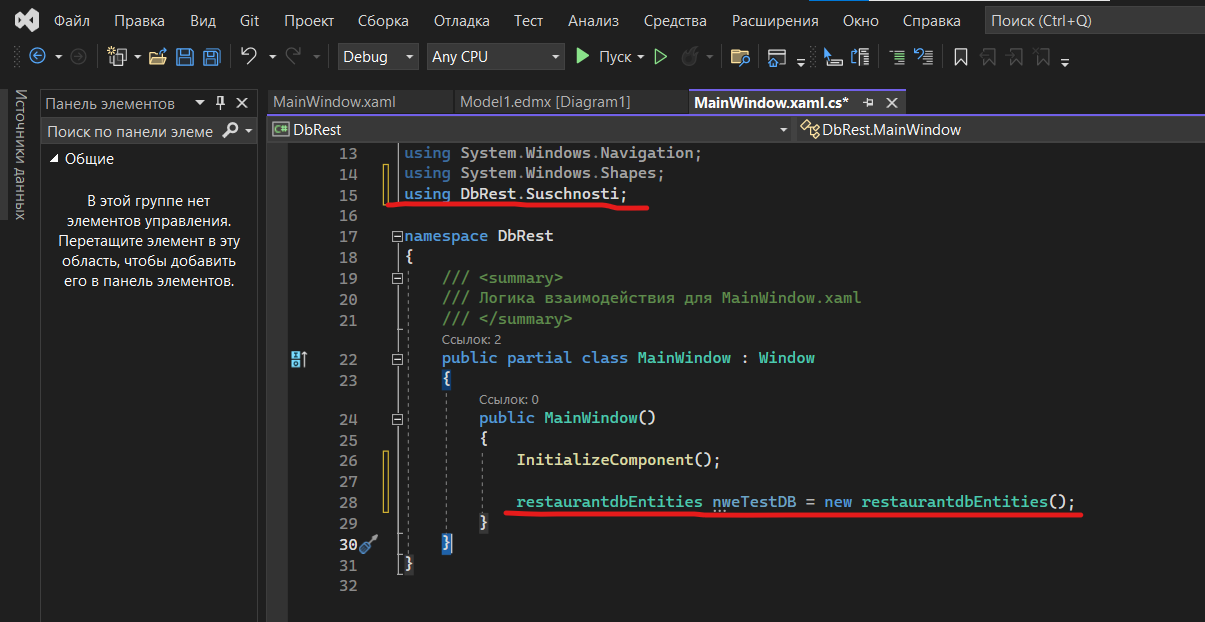
Название сервера



Дальше открываем файл MainWindow.xamal.cs

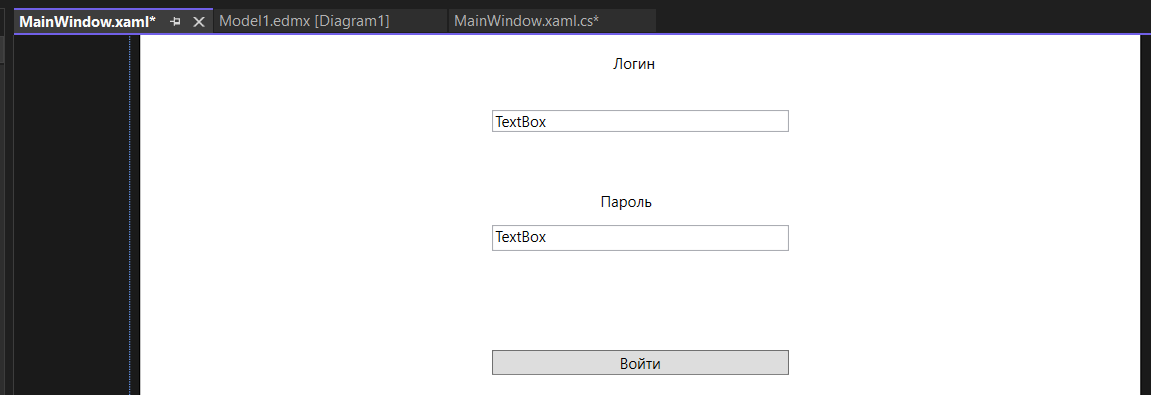
И пишем using DbRest.Suschnosti; - Название проекта + папка в которой находится БД – Это необходимо прописать в каждом окне

restaurantdbEntities nweTestDB = new restaurantdbEntities(); - вводим переменную, через которую можно обращаться к таблице БД

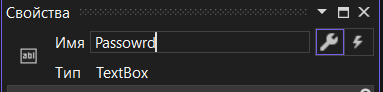


Переходим к окну MainWindow.xmal

Вставляем в форму textbox, lable, button

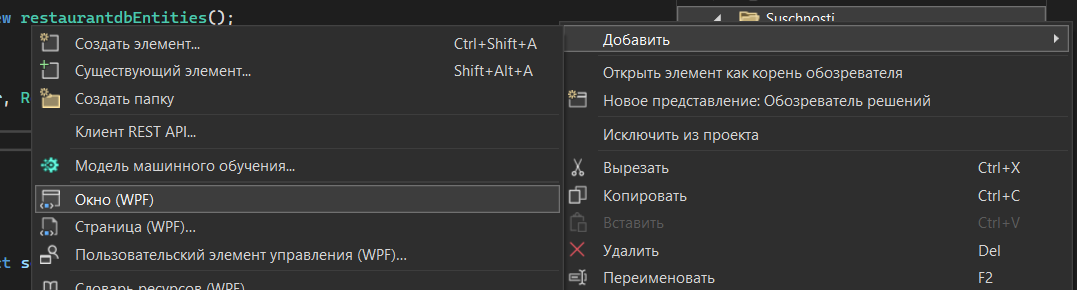


Нажимаем правой кнопкой мыши на Textbox и выбираем свойство. В свойствах задаем имя «login», «password»



Создаем окна для пользователя, администратора и сотрудника.

В обозревателе решений нажимаем на нашу папку Suschnosti правой кнопкой мыши и выбираем «Добавить» - «Окно» Называем их «AdminWindow», «SotrudnikWindow», «PolzovatelWindow»



В первом окне (где у нас находится пароль и логин для входа) прописываем условия входа (это будет цикл) и для каждого пользователя подвязываем окно.

private void Button\_Click(object sender, RoutedEventArgs e)

{

string login = Login.Text;

string passowrd = Password.Password;

if (login == "admin" && passowrd == "123")

{

MessageBox.Show("Вы вошли как админ", "Успешный вход");

AdminWindow adminWindow = new AdminWindow();

adminWindow.Show();

}

else if (login == "vikapika" && passowrd == "456")

{

MessageBox.Show("Вы вошли как сотрудник", "Успешный вход");

SotrudnikiWindow sotrudnikiWindow = new SotrudnikiWindow();

sotrudnikiWindow.Show();

}

else if (login == "gege" && passowrd == "789")

{

MessageBox.Show("Вы вошли как клиент", "Успешный вход");

PolzovateliWindow polzovateliWindow = new PolzovateliWindow();

polzovateliWindow.Show();

}

else

{

MessageBox.Show("Неправильный логин или пароль", "Ошибка");

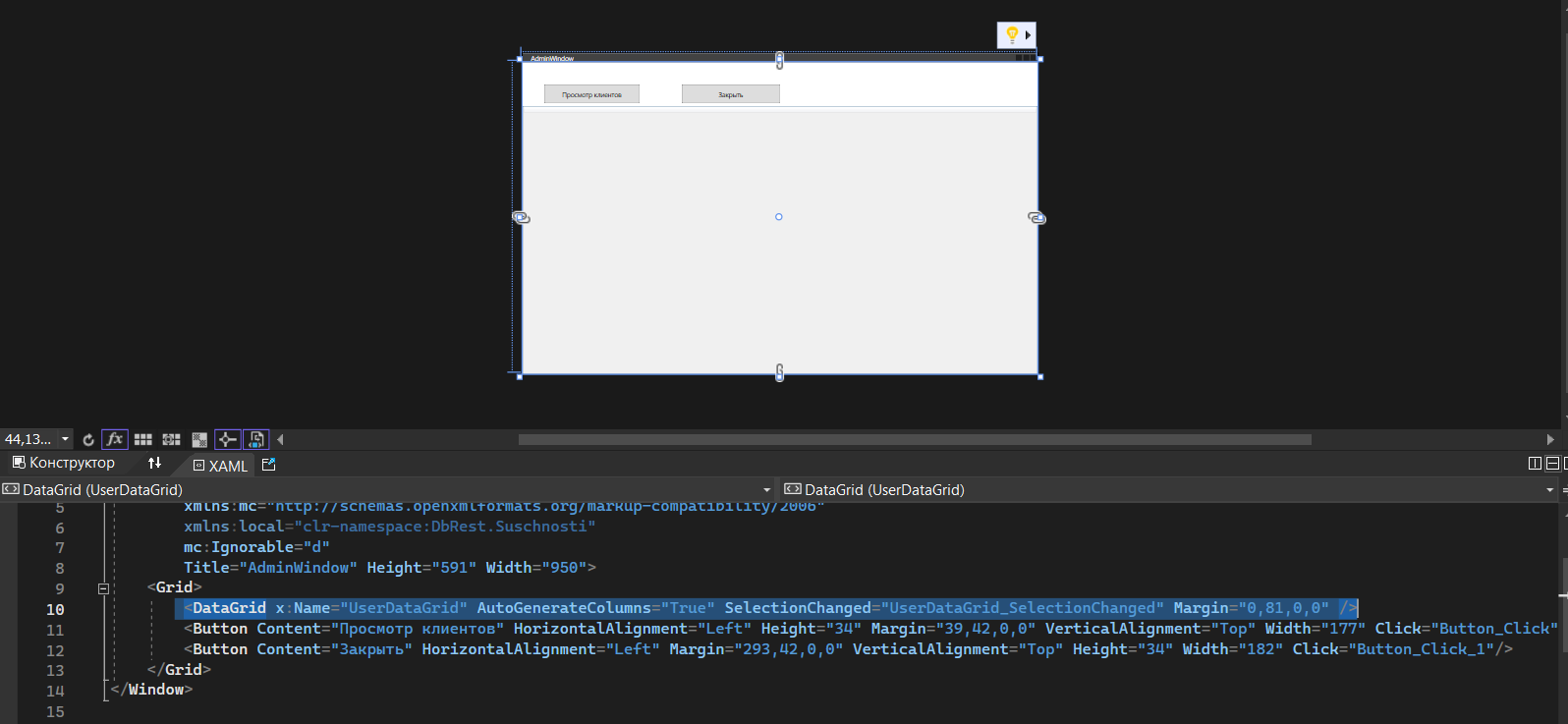
}

**Разработаем окно для администратора**

В окне администратора будут отображаться пользователи(посетители),

Меню, запас продуктов и поставщики. Так же все эти данные можно будет добавлять, удалять и изменить.

Для расположения кнопок на странице AdminWindow.xaml добавляем DataGrid



Добавляем кнопки и вызываем метод. Переходим на страницу кода.

Для кнопки «Просмотр пользователей» добавляем код

using (restaurantdbEntities db = new restaurantdbEntities())

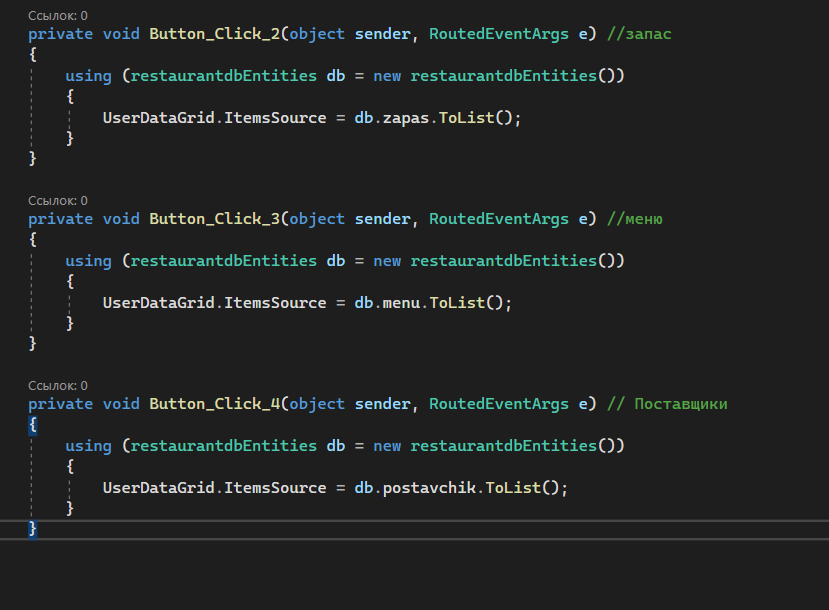
{

UserDataGrid.ItemsSource = db.polzovateli.ToList();

}

UserDataGrid.ItemsSource = db.polzovateli.ToList(); - указываем какие данные будут вытягивать из БД

Делаем так для всех кнопок



Добавляем кнопку закрытия окна

if (MessageBox.Show("Вы уверены, что хотите закрыть окно?", "Подтверждение",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

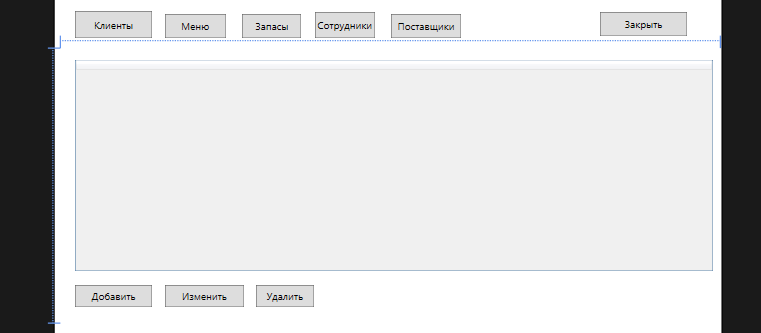
this.Close();

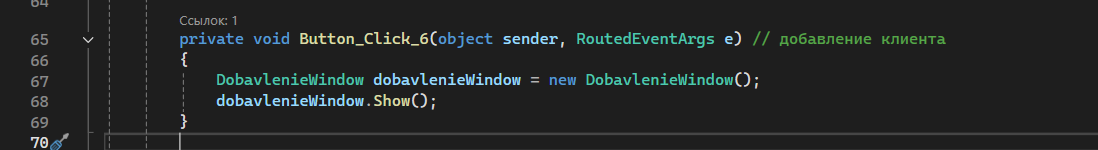
}

**Добавляем кнопки «Добавить», «Изменить», «Удалить»**

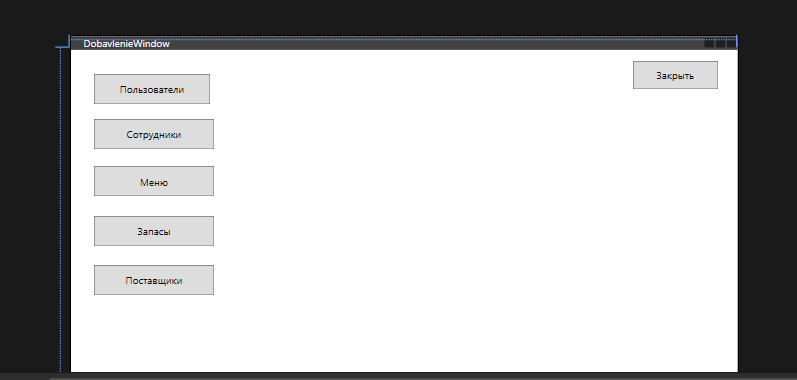
Для кнопки «Добавить» необходимо сделать отдельное окно. Мы поместим ее на панели администратора, но, при нажатии на кнопку будет выводиться отдельное окно, в нем надо будет выбрать куда и какую информацию мы будем добавлять, а дальше уже от выбранной категории будет выводиться форма заполнения данных.

1. Добавляем кнопку на панель, вызываем метод. В кнопке прописываем связь с другим окном. У нас оно будет называться «DobavlenieWindow»



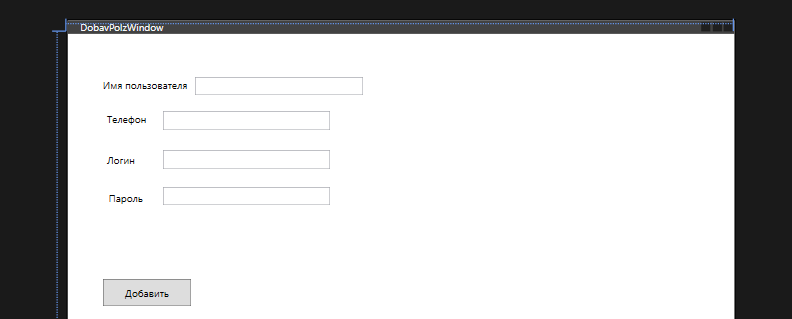
****

1. В окне «DobavlenieWindow» добавляем кнопки

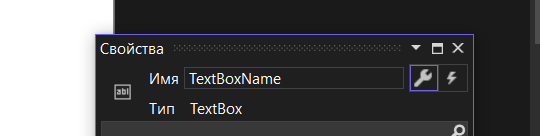


1. К каждой кнопке привязываем окно добавления

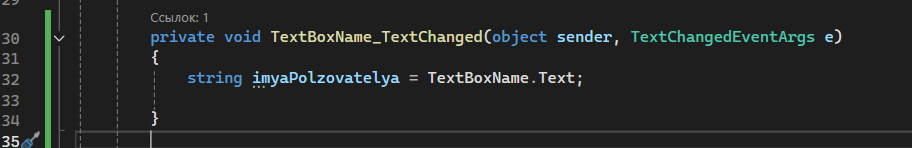
Окно добавления пользователей



TextBox для каждого поля в свойствах задаем имя



В вызванном методе TextBox присваиваем полученный текс переменной. Делаем так во всех TextBox.



Прописываем логику кнопки сохранения

private void Button\_Click(object sender, RoutedEventArgs e)

{

// Получаем данные из текстовых полей

string imyaPolzovatelya = ((TextBox)FindName("TextBoxName")).Text;

string telefon = ((TextBox)FindName("TextBoxTelefon")).Text;

string login = ((TextBox)FindName("TextBoxLogin")).Text;

string parol = ((TextBox)FindName("TextBoxPassword")).Text;

try

{

using (restaurantdbEntities db = new restaurantdbEntities())

{

// Проверяем, существует ли уже пользователь с таким логином

if (db.polzovateli.Any(u => u.login == login))

{

MessageBox.Show("Пользователь с таким логином уже существует.");

return;

}

// Создаем нового пользователя

var newUser = new polzovateli

{

imya\_polzovatelya = imyaPolzovatelya,

telefon = telefon,

login = login,

parol = parol

};

db.polzovateli.Add(newUser);

db.SaveChanges();

MessageBox.Show("Пользователь успешно добавлен.");

this.Close(); // Закрываем окно после добавления

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

}

Создаем окно для сотрудников

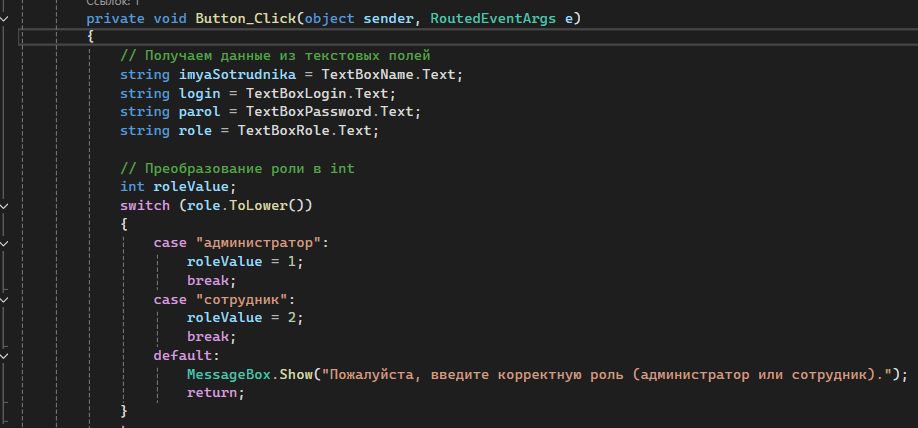


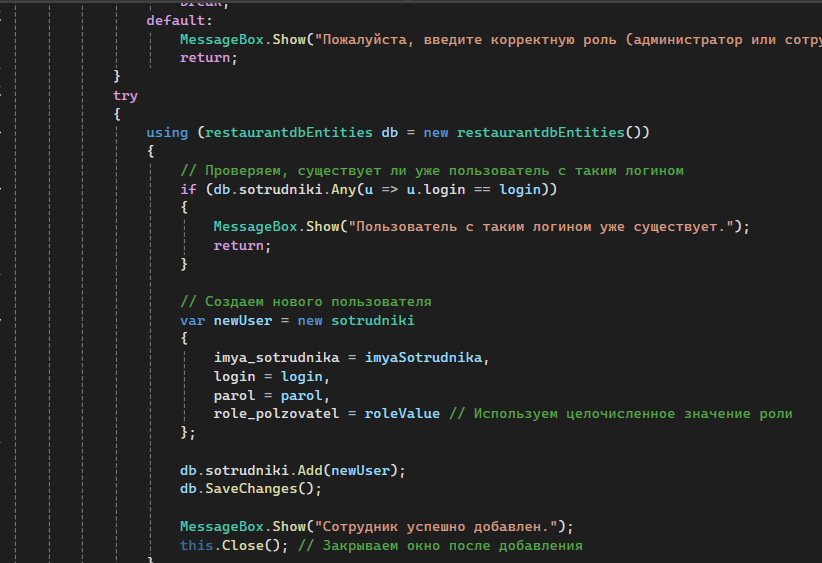
Очень важно! Поле «Роль» в базе данных у нас не varchar, а int.

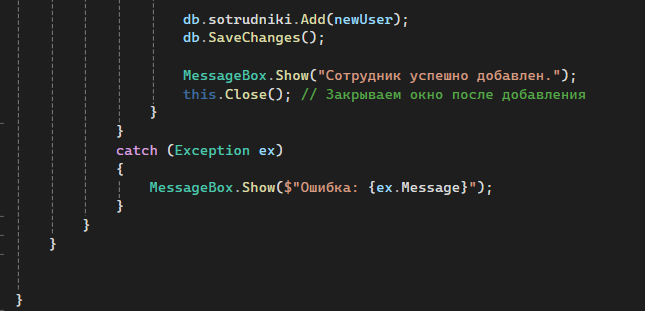
1. Преобразуем текст в число

Для начала в методе TextBox пишем string role = TextBoxRole.Text;

1. В кнопке прописываем







private void Button\_Click(object sender, RoutedEventArgs e)

{

// Получаем данные из текстовых полей

string imyaSotrudnika = TextBoxName.Text;

string login = TextBoxLogin.Text;

string parol = TextBoxPassword.Text;

string role = TextBoxRole.Text;

// Преобразование роли в int

int roleValue;

switch (role.ToLower())

{

case "администратор":

roleValue = 1;

break;

case "сотрудник":

roleValue = 2;

break;

default:

MessageBox.Show("Пожалуйста, введите корректную роль (администратор или сотрудник).");

return;

}

try

{

using (restaurantdbEntities db = new restaurantdbEntities())

{

// Проверяем, существует ли уже пользователь с таким логином

if (db.sotrudniki.Any(u => u.login == login))

{

MessageBox.Show("Пользователь с таким логином уже существует.");

return;

}

// Создаем нового пользователя

var newUser = new sotrudniki

{

imya\_sotrudnika = imyaSotrudnika,

login = login,

parol = parol,

role\_polzovatel = roleValue // Используем целочисленное значение роли

};

db.sotrudniki.Add(newUser);

db.SaveChanges();

MessageBox.Show("Сотрудник успешно добавлен.");

this.Close(); // Закрываем окно после добавления

}

}

catch (Exception ex)

{

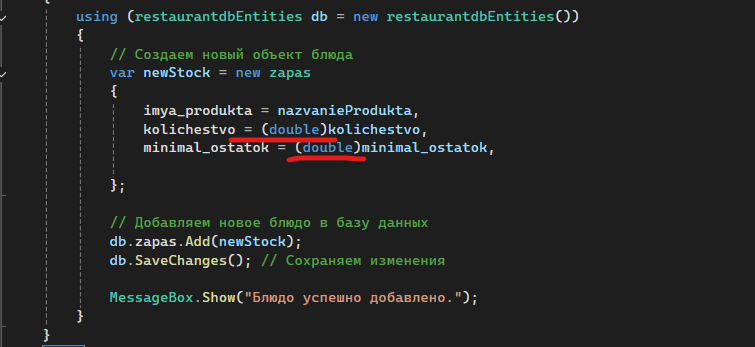
MessageBox.Show($"Ошибка: {ex.Message}");

}

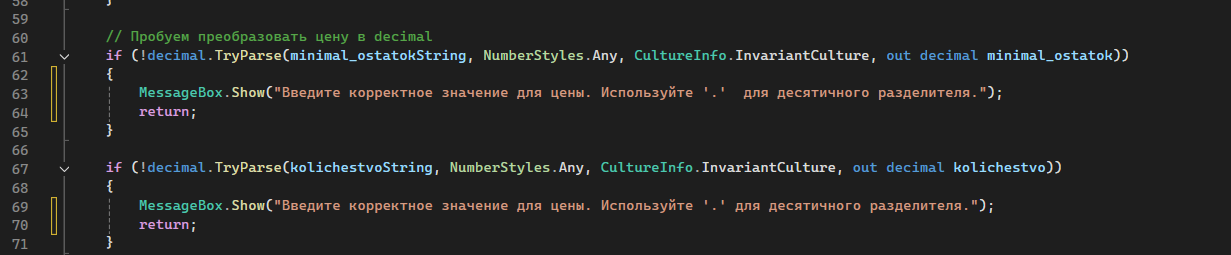
Окно «Запас»

Так как в БД у нас количество и минимальный остаток – Float, то необходимо преобразовать строковое значение в числовое. Делаем то же самое, что и в примере страницы «»Сотрудники», только немного добавим изменения.

В «Создаем новый объект блюда» перед числовой переменной добавляем (double).



И прописываем два условия для количества и минимального остатка



private void TextBoxName\_TextChanged(object sender, TextChangedEventArgs e)

{

string nazvanieProdukta = TextBoxName.Text;

}

private void TextBoxKolichestvo\_TextChanged(object sender, TextChangedEventArgs e)

{

string kolichestvo = TextBoxKolichestvo.Text;

}

private void TextBoxMinOst\_TextChanged(object sender, TextChangedEventArgs e)

{

string minimal\_ostatok = TextBoxMinOst.Text;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

string nazvanieProdukta = TextBoxName.Text;

string kolichestvoString = TextBoxKolichestvo.Text;

string minimal\_ostatokString = TextBoxMinOst.Text;

{

// Проверяем ввод цены

if (string.IsNullOrWhiteSpace(minimal\_ostatokString))

{

MessageBox.Show("Введите корректное значение для цены.");

return;

}

// Пробуем преобразовать цену в decimal

if (!decimal.TryParse(minimal\_ostatokString, NumberStyles.Any, CultureInfo.InvariantCulture, out decimal minimal\_ostatok))

{

MessageBox.Show("Введите корректное значение для цены. Используйте '.' для десятичного разделителя.");

return;

}

if (!decimal.TryParse(kolichestvoString, NumberStyles.Any, CultureInfo.InvariantCulture, out decimal kolichestvo))

{

MessageBox.Show("Введите корректное значение для цены. Используйте '.' для десятичного разделителя.");

return;

}

try

{

using (restaurantdbEntities db = new restaurantdbEntities())

{

// Создаем новый объект блюда

var newStock = new zapas

{

imya\_produkta = nazvanieProdukta,

kolichestvo = (double)kolichestvo,

minimal\_ostatok = (double)minimal\_ostatok,

};

// Добавляем новое блюдо в базу данных

db.zapas.Add(newStock);

db.SaveChanges(); // Сохраняем изменения

MessageBox.Show("Блюдо успешно добавлено.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении блюда: {ex.Message}");

}

}

}

Аналогично создаем остальные окна. Добавляем логику кнопки «Закрыть» для окна «Добавление».

private void Button\_Click\_5(object sender, RoutedEventArgs e)//закрыть

{

if (MessageBox.Show("Вы уверены, что хотите закрыть окно?", "Подтверждение",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

this.Close();

}

}

На окне «DobavlenieWindow» добавляем кнопки «Изменить», «Удалить»

Кнопка «Удаление»

private void Button\_Click\_7(object sender, RoutedEventArgs e) // удаление

{

if (UserDataGrid.SelectedItem == null)

{

MessageBox.Show("Выберите запись для удаления.");

return;

}

try

{

using (restaurantdbEntities db = new restaurantdbEntities())

{

object selectedItem = UserDataGrid.SelectedItem;

if (selectedItem is polzovateli user)

{

var entity = db.polzovateli.FirstOrDefault(u => u.polzovatel\_id == user.polzovatel\_id);

if (entity != null) db.polzovateli.Remove(entity);

}

else if (selectedItem is sotrudniki sotrudnik)

{

var entity = db.sotrudniki.FirstOrDefault(s => s.sotrudnik\_id == sotrudnik.sotrudnik\_id);

if (entity != null) db.sotrudniki.Remove(entity);

}

else if (selectedItem is menu dish)

{

var entity = db.menu.FirstOrDefault(m => m.menu\_id == dish.menu\_id);

if (entity != null) db.menu.Remove(entity);

}

else if (selectedItem is zapas stock)

{

var entity = db.zapas.FirstOrDefault(z => z.produkt\_id == stock.produkt\_id);

if (entity != null) db.zapas.Remove(entity);

}

else if (selectedItem is postavchik supplier)

{

var entity = db.postavchik.FirstOrDefault(p => p. postavchik\_id == supplier.postavchik\_id);

if (entity != null) db.postavchik.Remove(entity);

}

else

{

MessageBox.Show("Не удалось определить таблицу.");

return;

}

db.SaveChanges();

MessageBox.Show("Запись успешно удалена.");

// Обновляем DataGrid

UserDataGrid.ItemsSource = db.polzovateli.ToList();

UserDataGrid.ItemsSource = db.sotrudniki.ToList();

UserDataGrid.ItemsSource = db.menu.ToList();

UserDataGrid.ItemsSource = db.zapas.ToList();

UserDataGrid.ItemsSource = db.postavchik.ToList();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

}

}

Кнопка «Изменить»

private void Button\_Click\_8(object sender, RoutedEventArgs e) // обновление данных

{

using (restaurantdbEntities db = new restaurantdbEntities())

{

UserDataGrid.ItemsSource = db.polzovateli.ToList();

UserDataGrid.ItemsSource = db.sotrudniki.ToList();

UserDataGrid.ItemsSource = db.menu.ToList();

UserDataGrid.ItemsSource = db.zapas.ToList();

UserDataGrid.ItemsSource = db.postavchik.ToList();

}

}

Так же к ней прописываем сохранение данных

private void UserDataGrid\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e) // сохранение изменений

{

if (e.EditAction == DataGridEditAction.Commit)

{

using (restaurantdbEntities db = new restaurantdbEntities())

{

object editedItem = e.Row.Item;

if (editedItem is polzovateli user)

{

var entity = db.polzovateli.FirstOrDefault(u => u.polzovatel\_id == user.polzovatel\_id);

if (entity != null)

{

entity.imya\_polzovatelya = user.imya\_polzovatelya;

entity.telefon = user.telefon;

entity.login = user.login;

entity.parol = user.parol;

}

}

else if (editedItem is sotrudniki employee)

{

var entity = db.sotrudniki.FirstOrDefault(s => s.sotrudnik\_id == employee.sotrudnik\_id);

if (entity != null)

{

entity.imya\_sotrudnika = employee.imya\_sotrudnika;

entity.login = employee.login;

entity.parol = employee.parol;

entity.role\_polzovatel = employee.role\_polzovatel;

}

}

else if (editedItem is menu dish)

{

var entity = db.menu.FirstOrDefault(m => m.menu\_id == dish.menu\_id);

if (entity != null)

{

entity.kategoriya = dish.kategoriya;

entity.nazvanie\_bluda = dish.nazvanie\_bluda;

entity.ingredient = dish.ingredient;

entity.cena = dish.cena;

}

}

else if (editedItem is zapas stock)

{

var entity = db.zapas.FirstOrDefault(z => z.produkt\_id == stock.produkt\_id);

if (entity != null)

{

entity.imya\_produkta = stock.imya\_produkta;

entity.kolichestvo = stock.kolichestvo;

}

}

else if (editedItem is postavchik supplier)

{

var entity = db.postavchik.FirstOrDefault(p => p.postavchik\_id == supplier.postavchik\_id);

if (entity != null)

{

entity.nazvanie\_postavchika = supplier.nazvanie\_postavchika;

}

}

else

{

MessageBox.Show("Не удалось определить таблицу.");

return;

}

db.SaveChanges();

MessageBox.Show("Изменения успешно сохранены.");

}

}

}