МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Проектирование базы данных «Интернет-магазин»

Выполнил студент Мущук Артур Николаевич

(Ф.И.О.)

Руководитель проекта асс. Потапенко К.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты асс. Патапенко К.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер асс. Патапенко К.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2017

Оглавление

[**Введение** 2](#_Toc500983015)

[**1.** **Разработка модели базы данных** 3](#_Toc500983016)

[**2.** **Разработка необходимых объектов** 5](#_Toc500983017)

[**2.1. Таблицы** 5](#_Toc500983018)

[**2.2. Пользователи** 5](#_Toc500983019)

[**2.3 Процедуры** 5](#_Toc500983020)

[**3. Описание процедур импорта и экспорта данных** 7](#_Toc500983021)

[**3.1. Процедура импорта данных из XML-файла** 7](#_Toc500983022)

[**3.2. Процедура экспорта данных в XML-файл** 7](#_Toc500983023)

[**4.** **Технология SQL Server Reporting Services (SSRS)** 8](#_Toc500983024)

[**5. Резервное копирование и восстановление** 11](#_Toc500983025)

[**6.** **Тестирование** 13](#_Toc500983026)

[**6.1 Тестирование производительности базы данных** 13](#_Toc500983027)

[**7.** **Руководство пользователя** 16](#_Toc500983028)

[**7.1. Руководство пользователя администратора** 16](#_Toc500983029)

[**Заключение** 21](#_Toc500983030)

[**Список использованной литературы** 22](#_Toc500983031)

[**Приложение А** 23](#_Toc500983032)

[**Приложение Б** 25](#_Toc500983033)

[**Приложение В** 26](#_Toc500983034)

[**Приложение Г** 29](#_Toc500983035)

[**Приложение Д** 30](#_Toc500983036)

# **Введение**

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Основные функции СУБД:

* определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки;
* предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация);
* обеспечение логической и физической независимости данных;
* защита логической целостности базы данных;
* защита физической целостности;
* управление полномочиями пользователей на доступ к базе данных;
* синхронизация работы нескольких пользователей;
* управление ресурсами среды хранения;
* поддержка деятельности системного персонала.
* Обычно современная СУБД содержит следующие компоненты:
* ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
* процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;
* подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД.

СУБД существует огромное множество: Oracle, MS SQL Server, Microsoft Access, MySql и так далее. В данной работе будет использовано решение от компании MS SQL Server.

Так же для разработки приложения использовалась технология WPF и язык программирования C#. Для взаимодействия приложения с базой данных использовалась технология ADO.NET.

1. **Разработка модели базы данных**

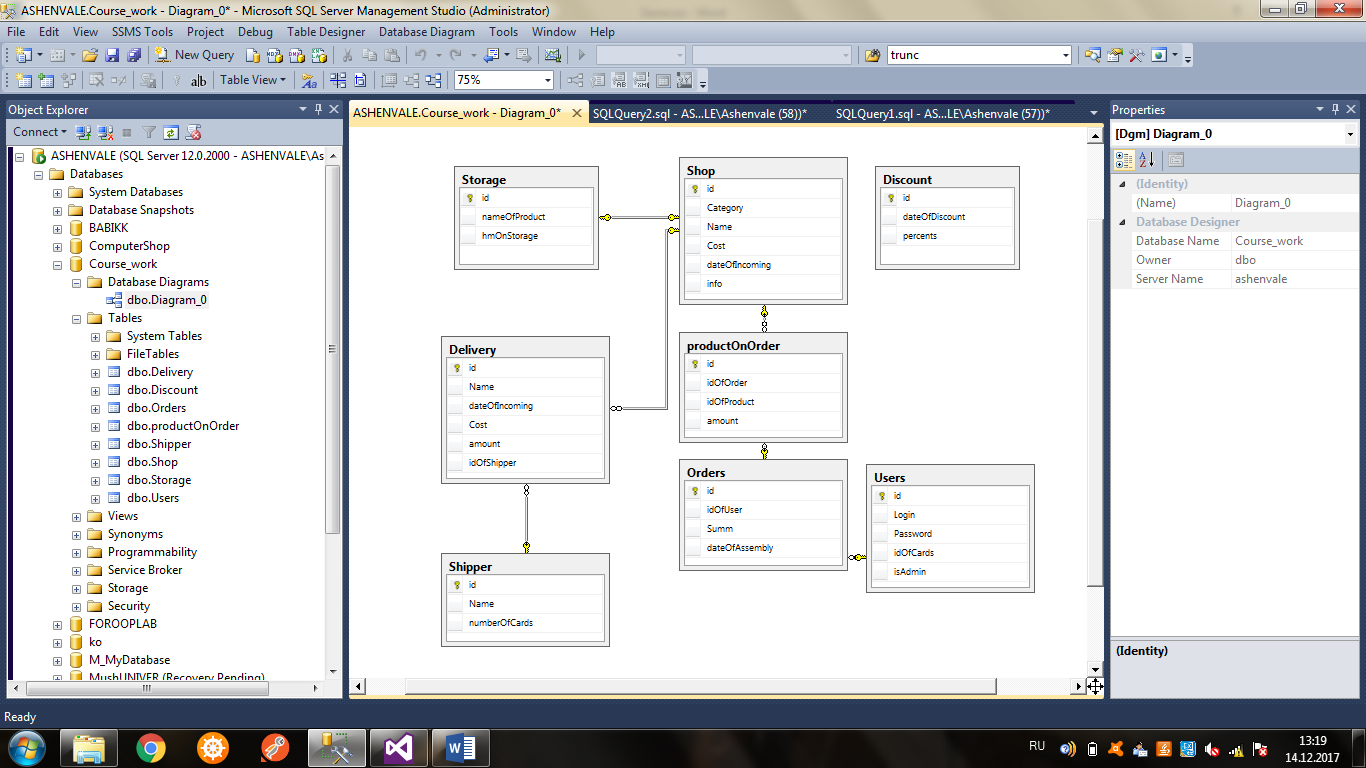


Рисунок 1.1 – Диаграмма базы данных

Для базы данных бронирования апартаментов были разработаны 7 таблиц. Диаграмма связей таблиц для необходимой базы данных представлена н рисунке 1.1.

Таблица Users, содержащая пользователей бд:

* ID - уникальный идентификатор;
* Login­ – имя пользователя;
* Password - пароль;
* idOfCards - номер карты;
* isAdmin - значение, по которому проверяется, является ли пользователь администратором.

Таблица Orders, предназначенная для хранения заказов пользователей:

* ID – уникальный идентификатор;
* idOfUser – идентификатор пользователя;
* Summ – стоимость заказа;
* dateOfAssembly – дата доставки.

Таблица productOnOrder, предназначенная для хранения товаров в заказе:

* ID – уникальный идентификатор;
* idOfOrder – идентификатор заказа;
* idOfProduct – идентификатор товара;
* amount – количество товаров.

Таблица Shop, предназначенная для хранения продаваемых товаров:

* ID – уникальный идентификатор;
* Category – категория товара;
* Name – наименование товара;
* Cost – стоимость товара в магазине;
* dateOfIncoming – дата доставки товара;
* info – информация о товаре.

Таблица Storage:

* ID – уникальный идентификатор;
* nameOfProduct – наименование товара;
* hmOnStorage – количество товаров на складе.

Таблица Delivery, предназначенная для хранения поставок товара:

* Id – уникальный идентификатор;
* Name – наименование товара;
* dateOfIncoming – дата доставки товара;
* Cost – стоимость доставки товара;
* Amount – количество товара;
* idOfShipper – идентификатор поставщика.

Таблица Shipper, предназначенная для хранения данных о поставщиках:

* ID – уникальный идентификатор;
* Name – наименование поставщика;
* numberOfCards – номер карты поставщика.

Таблица Discounts, предназначена для хранения данных о скидках:

* id – уникальный идентификатор;
* dateOfDiscount– дата скидки;
* percents – коэффицент скидки.

Скрипты для создания всех таблиц базы данных представлены в Приложении А.

1. Разработка необходимых объектов

## **2.1. Таблицы**

Таблицы являются основой любой базы данных, именно в них хранится вся информация. При проектировании базы данных было создано 8 таблиц, которые подробно описаны ранее в разделе 1, а SQL-скрипты для их создания находятся в Приложении А.

## **2.2. Пользователи**

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. На каждом этапе развития БД (проектирование, реализация, эксплуатация, модернизация и развитие, полная реорганизация) с ней связаны разные категории пользователей. При проектировании базы данных понадобилось 2 пользователя (клиент, делает заказы и администратор магазина).

Администратор магазина был наделён только привилегией на выполнение всех хранимых процедур, разработанных для данной базы данных.

Клиенту разрешено выполнять только 11 процедур, связанных с получением информации о товарах для покупки и 5 процедур, необходимых для оформления заказа.

Скрипты для создания логинов и пользователей базы данных, а также выделение привилегий на осуществление определённых операций с базой данных представлены в приложении Б.

## **Процедуры**

Использование хранимых процедур позволяет ограничить либо вообще исключить непосредственный доступ пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным. Листинги некоторых хранимых процедур представлены в приложении В.

Всего было разработано 15 процедур:

* UserSelectByLogin –процедура для проверки является ли пользователь администратором;
* SelectUsersOrders – процедура для получения всех заказов пользователя;
* SelectCurrentOrders – процедура для получения актуальных заказов;
* SelectproductByOrdersId – процедура для получения перечня товаров в заказе;
* getSummOfOrder – процедура для получения стоимости заказа;
* OrdersDateUpdate – процедура для обновления даты сборки заказа;
* getProductByCost – процедура для сортировки товаров по стоимости;
* UsersIsAdminUpdate – процедура для назначения и снятия пользователя с роли администратора;
* DeliveryInsert – процедура для создания поставки с обновлением даты поставки товара;
* productOnOrderInsert– процедура для добавления товара в заказ с изъятием товара со склада;
* ShopInsert –процедура для добавления товара в магазин, с добавлением записи о товаре на склад;
* ShopFromXml – процедура для импорта товаров из файла \*.xml;
* ShopToXml – процедура для экспорта товаров в файл \*.xml;
* UserFromXml – процедура для импорта пользователей из файла \*.xml;
* UserToXml – процедура для экспорта пользователей в файл \*.xml.

Помимо них было разработано 33 CRUD-процедуры.

Все скрипты хранимых процедур приложены в отдельных файлах в корне директории прилагаемого диска.

# **Описание процедур импорта и экспорта данных**

## **3.1. Процедура импорта данных из XML-файла**

Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся. Пример данного скрипта представлена в Приложении Г.

## **3.2. Процедура экспорта данных в XML-файл**

Для работы процедуры экспорта данных в XML-файл необходимо изменить параметры конфигурации dbo при помощи скрипта, представленного на рисунке 3.1.



Рисунок 3.2.1. – Скрипт разрешения выполнения команд в cmd

Пример скрипта процедуры экспорта в XML-файл представлен в Приложении Д.

# **Технология SQL Server Reporting Services (SSRS)**

SQL Server Reporting Services (SSRS) – это службы для разработки, построения, доставки и просмотра отчетов. С помощью этих служб можно создавать табличные, интерактивные, графические и другие более сложные отчеты с использованием диаграмм и других отчетных элементов.

SSRS реализован как веб-служба и ее администрирование, а также управление отчетами производится через веб-интерфейс. Стандартный доступ к отчетам предоставляется пользователям также через-веб интерфейс.

Отчеты составлялись с помощью мастера отчетов. Созданные отчеты представлены на рисунках 4.1-4.4.

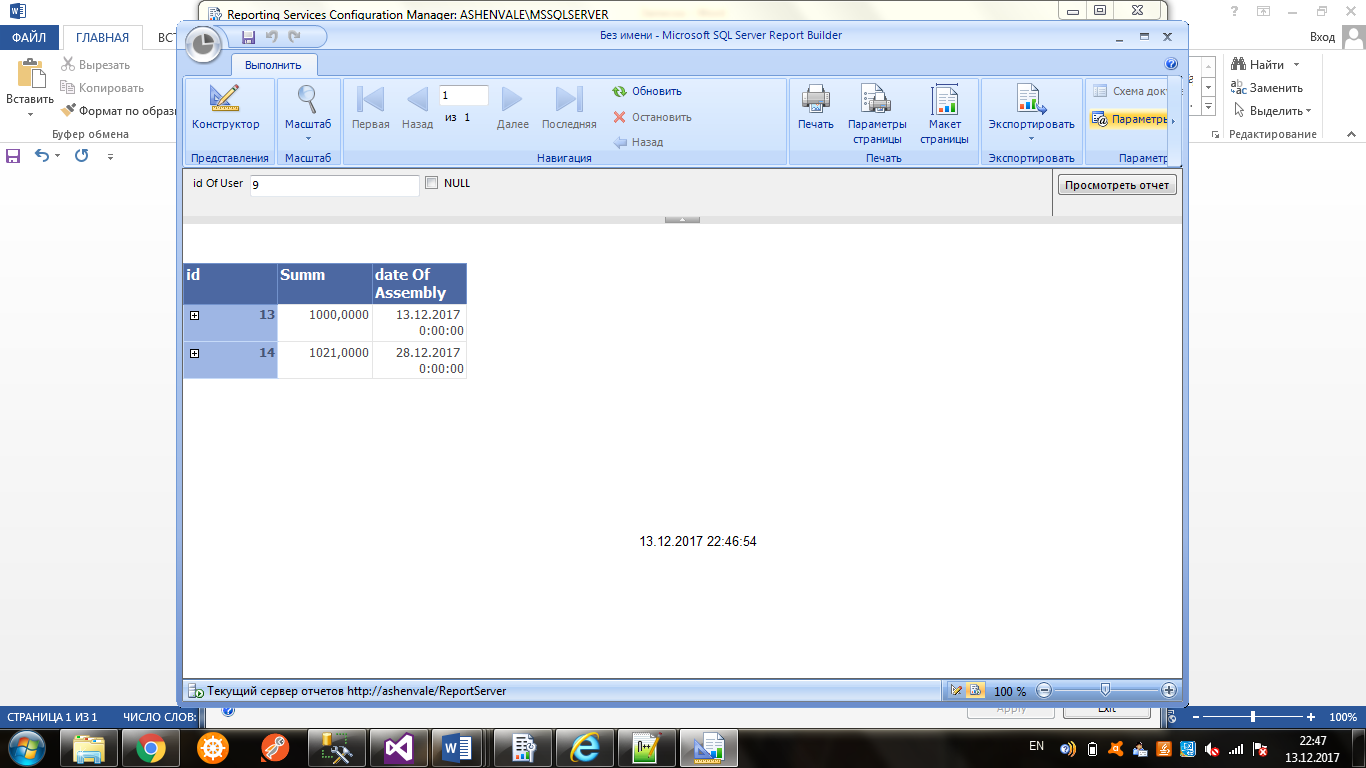


Рисунок 4.1 – Заказы пользователя.

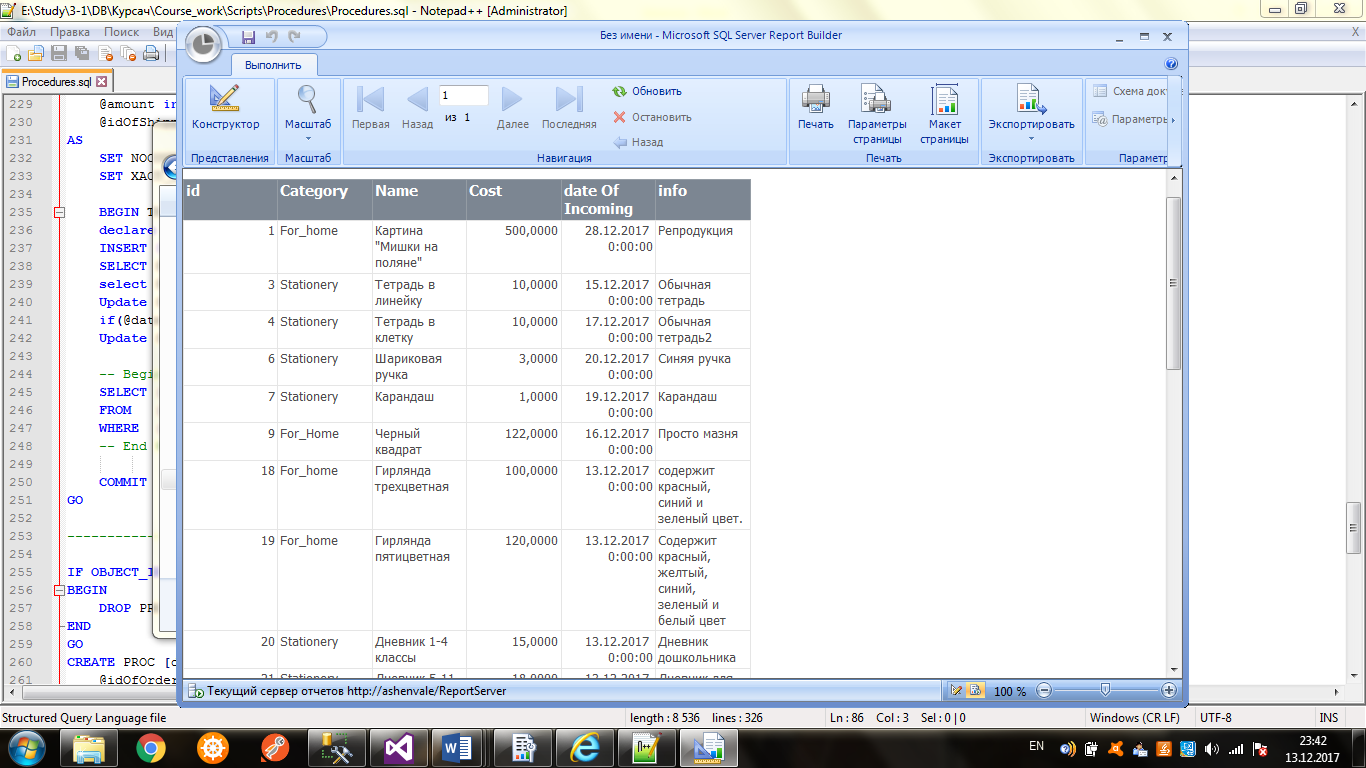


Рисунок 4.2 – Select-запрос к таблице товаров.

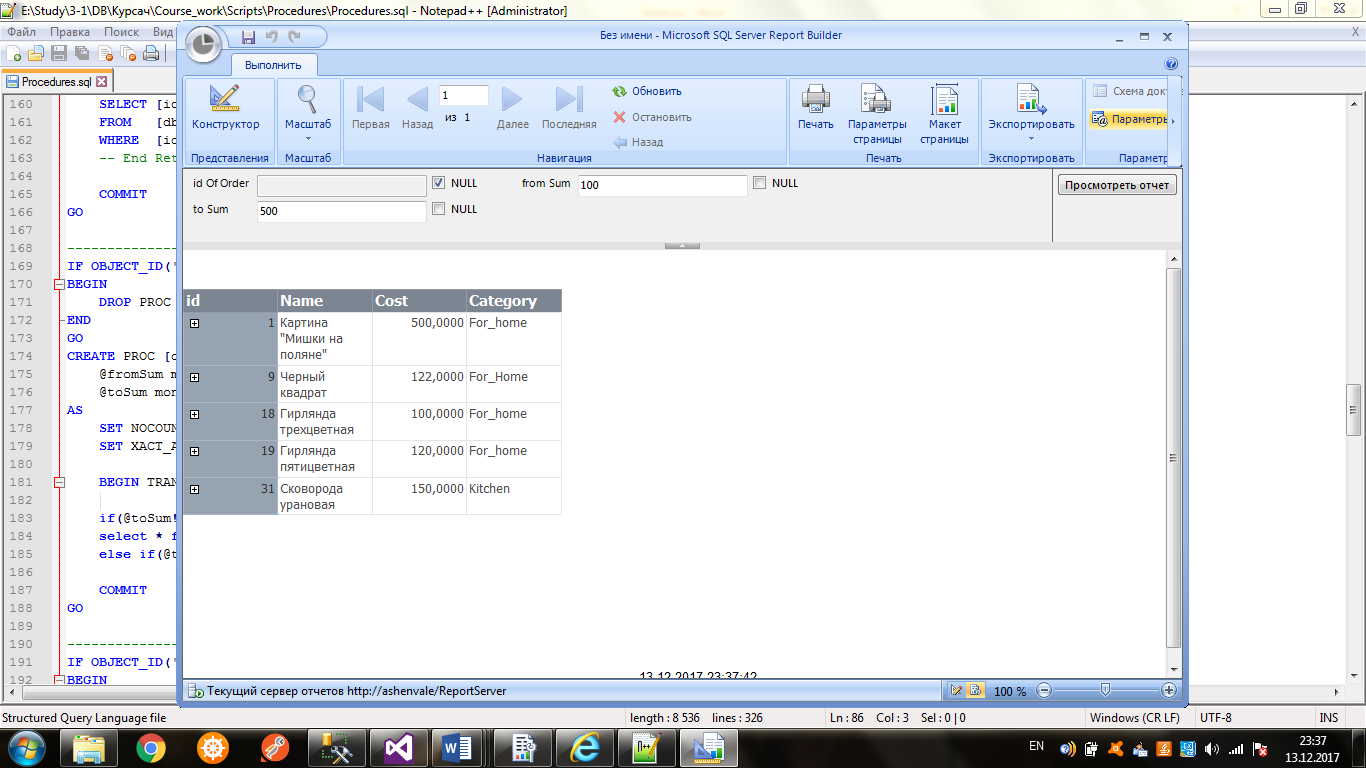


Рисунок 4.3 – Товары отсортированные по стоимости.

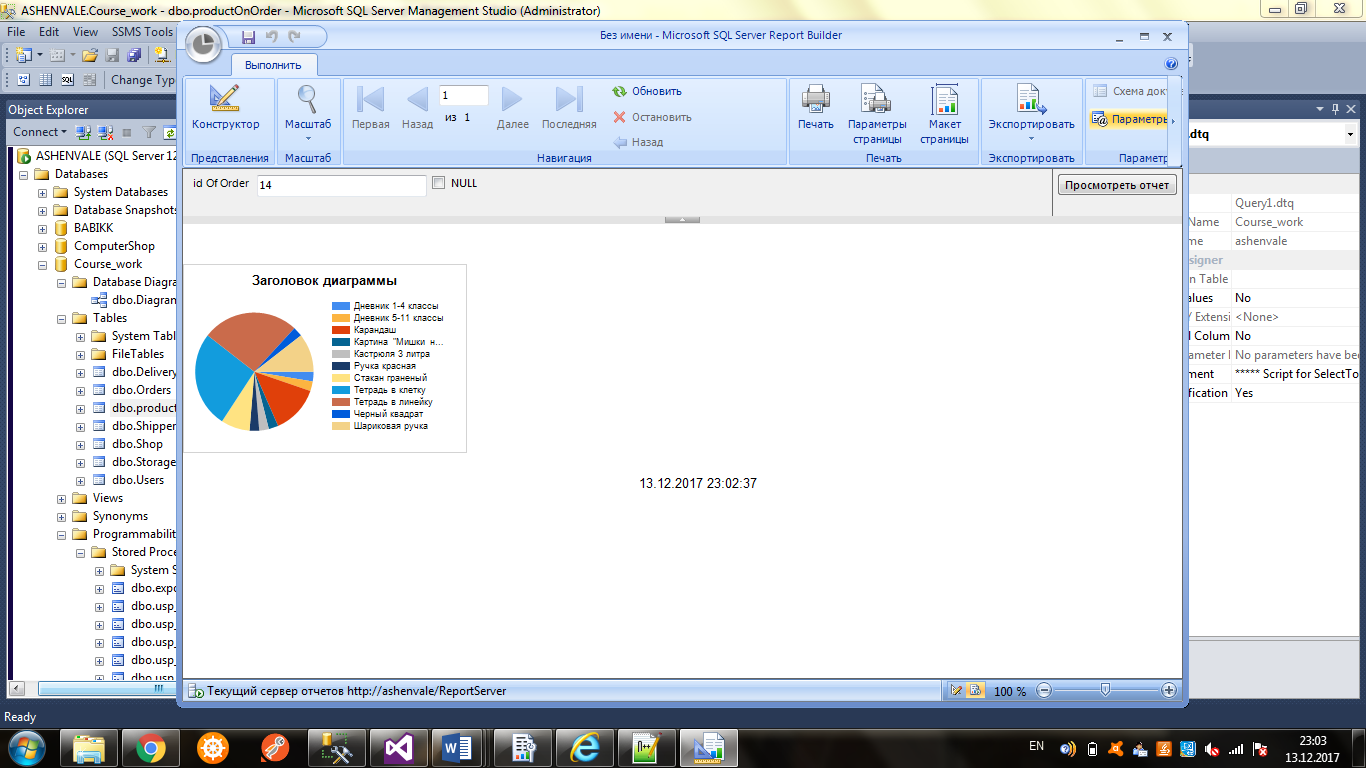


Рисунок 4.4 – Диаграмма количества товаров в заказе.

# **Резервное копирование и восстановление**

Резервное копирование базы данных будет осуществляться в папку BackupDB на диске E:\ каждый день и имя файла будет состоять из даты создания бэкапа.

Листинг исходного кода резервного копирования:

DECLARE @pathName NVARCHAR(512)

SET @pathName = 'E:\BackupDB\db\_backup\_' + Convert(varchar(8), GETDATE(), 112) + '.bak'

BACKUP DATABASE [Course\_work] TO DISK = @pathName WITH NOFORMAT, NOINIT, NAME = N'db\_backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

Далее создаем файл расширения .bat, который и будет запускать скрипт резервного копирования.

Листинг исходного батника:

sqlcmd -S Ashenvale -E -i backup.sql

Создание задачи будем осуществлять с помощью планировщика заданий (запуск команды taskschd.msc).

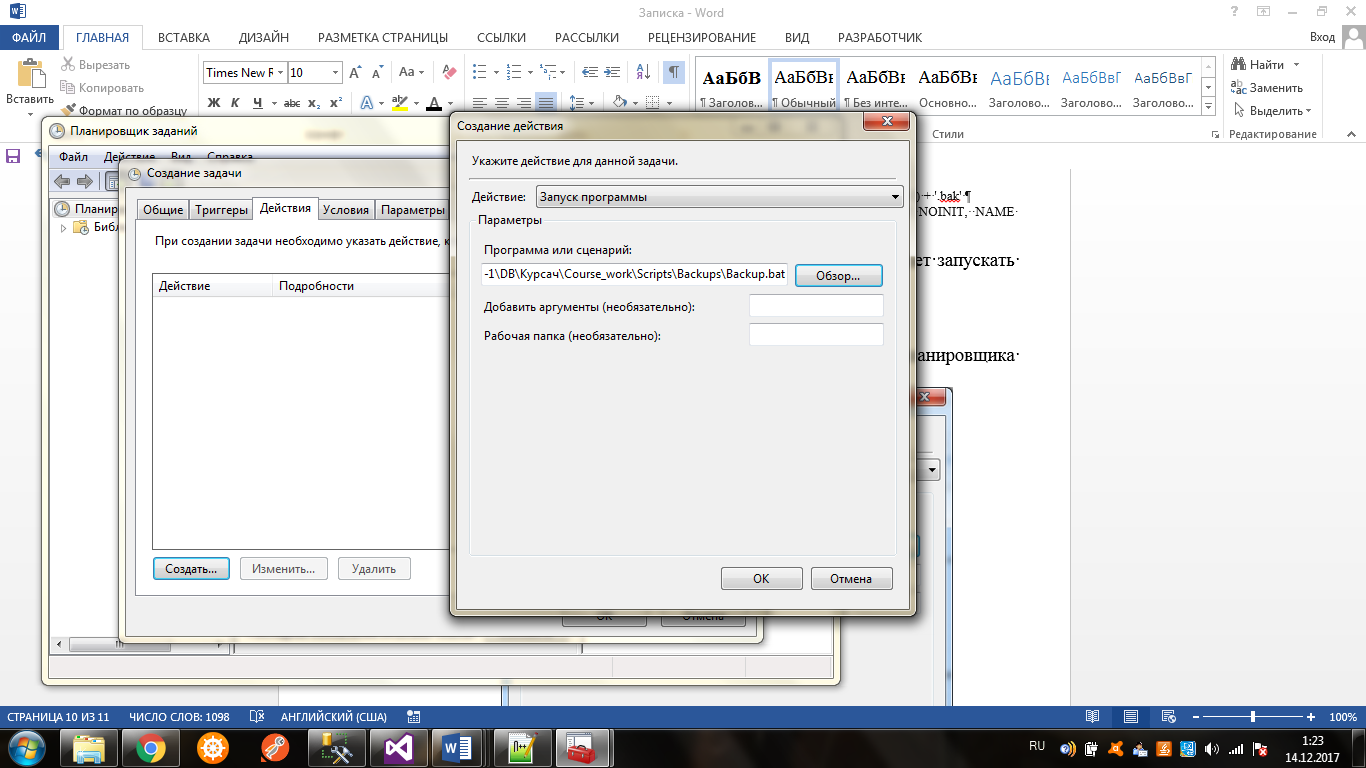


Рисунок 5.1 – Создание действия

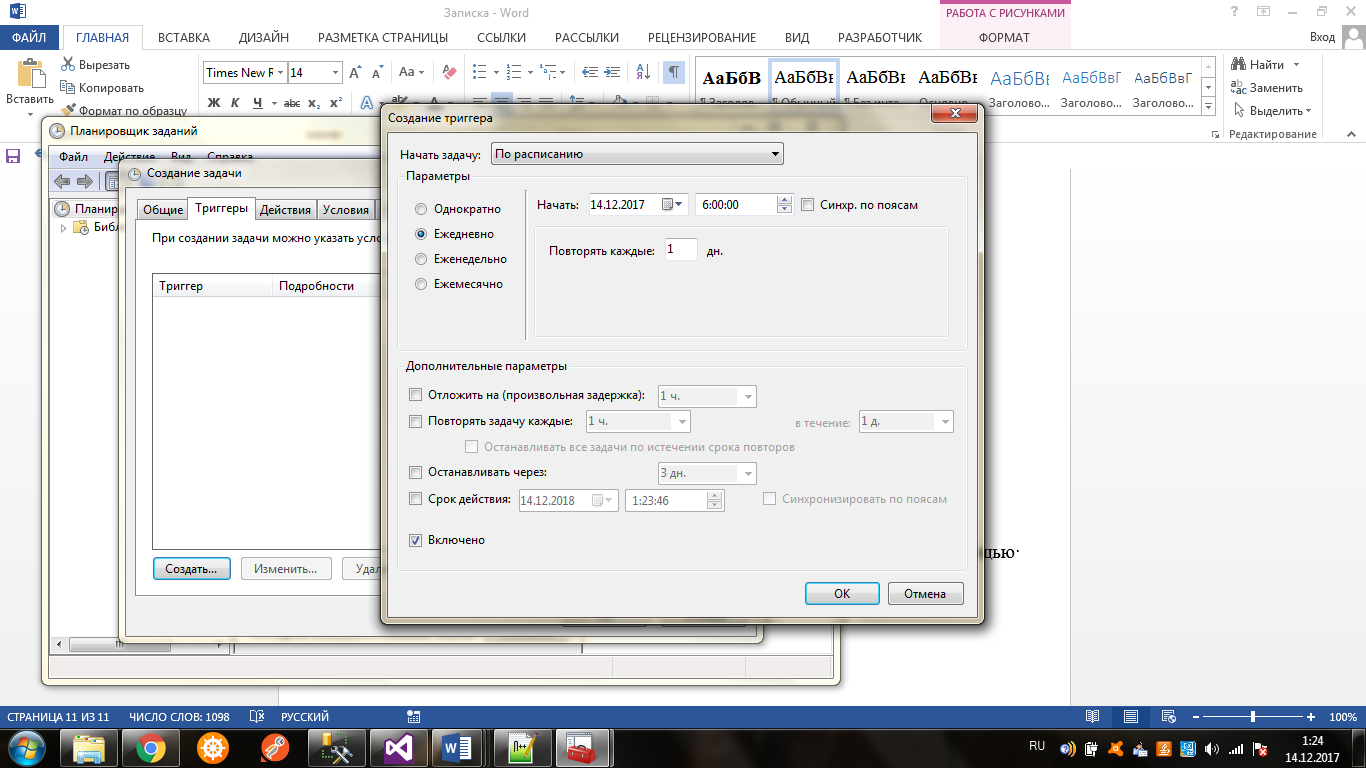


Рисунок 5.2 – Создание расписания

Восстановление базы данных можно осуществить с помощью выполнения задачи восстановления.

# **Тестирование**

## **6.1 Тестирование производительности базы данных**

Для тестирования производительности была взята за основу таблица Shop, так как она заполнена гораздо больше всех остальных таблиц.

Изначально таблица Shop была заполнена на 100000 строк. После этого был применён SELECT-запрос к данной таблице и при помощи стандартных средств IDE MS SQL Server Management Studio оценена цена выборки к таблице. Результат данной оценки запроса приведён на рисунке 7.1.

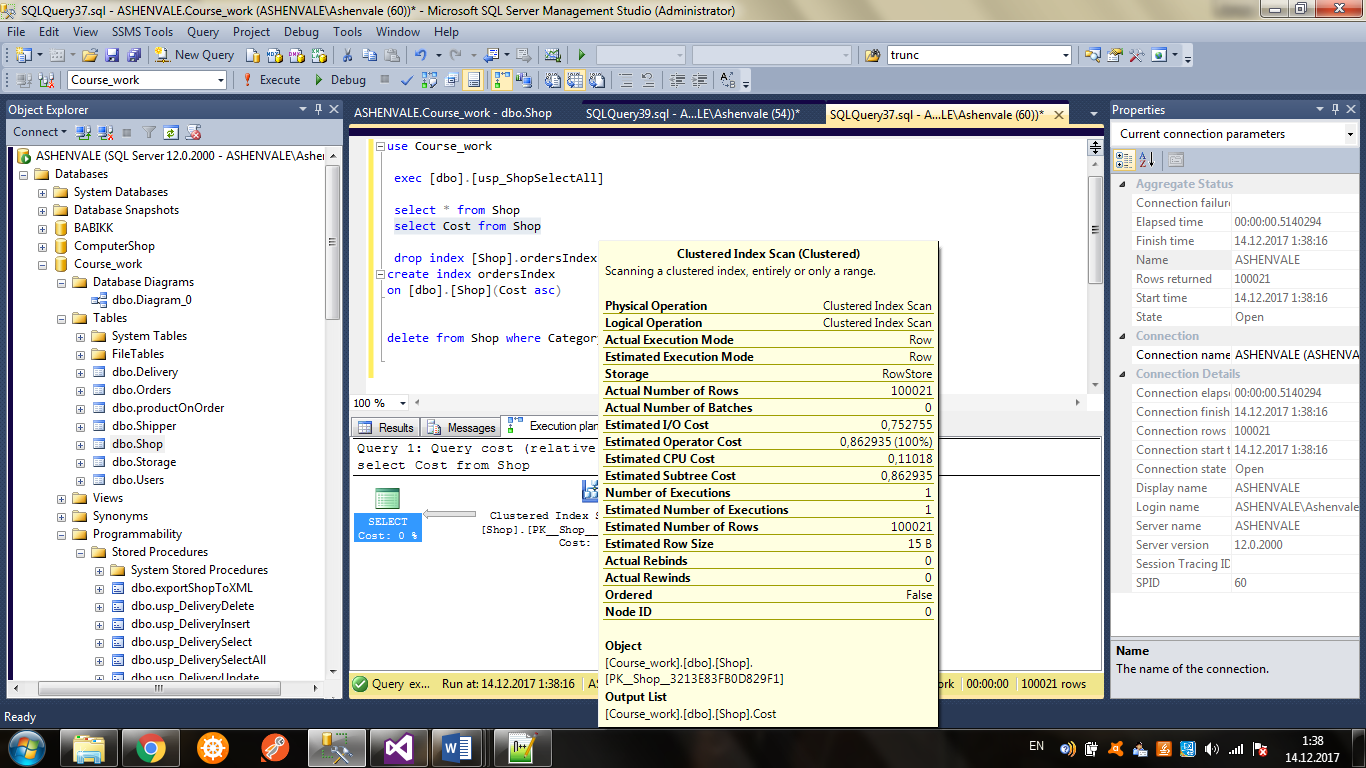


Рисунок 7.1 – Оценка запроса к таблице Shop без некластеризованного индекса

После проведения первоначальной оценки был построен некластеризованный индекс к таблице Shop по столбцу id и проведена оценка такого же SELECT-запроса к таблице Shop. Результаты, полученные во время оценки, представлены на рисунке 7.2.

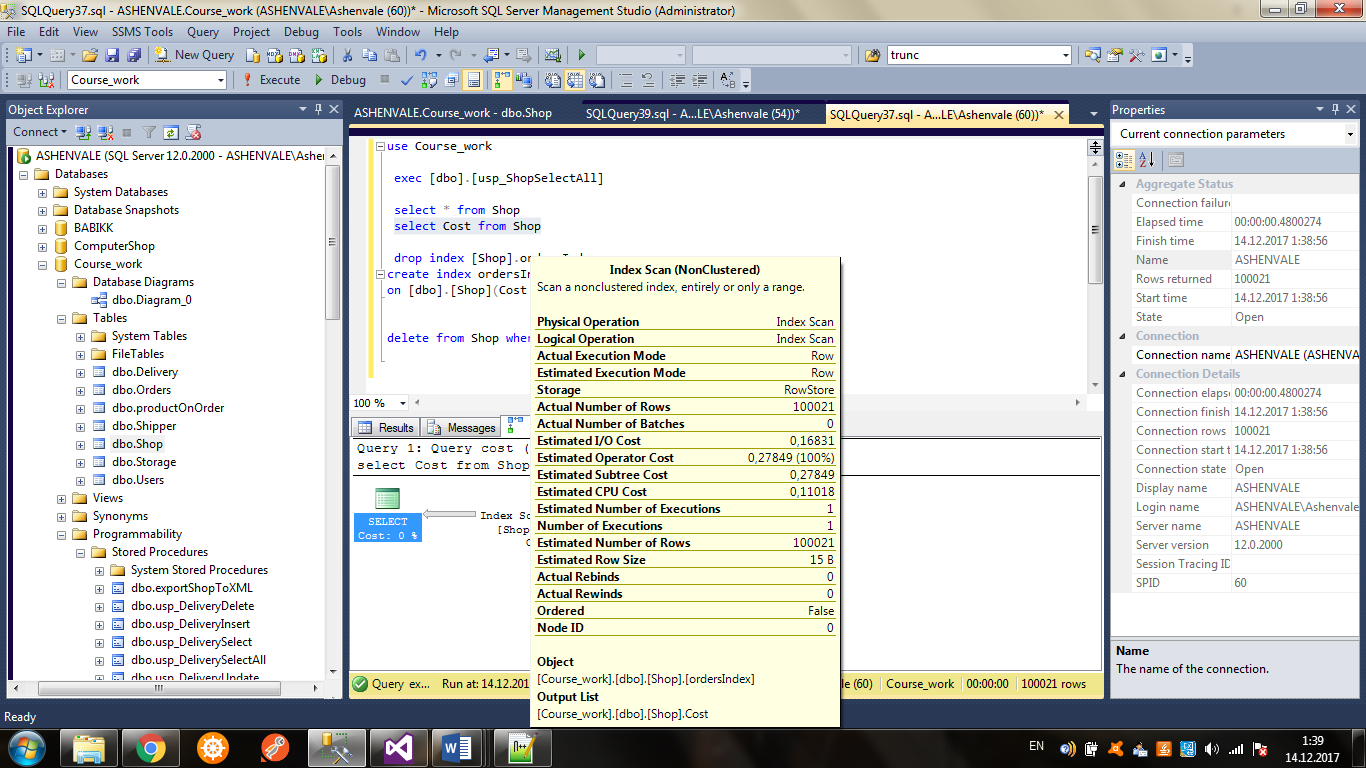


Рисунок 7.2 – Оценка запроса к таблице с построенным некластеризованным индексом

По результатам проведённых оценок до и после построения некластеризованного индекса, можно сделать вывод, что после создания индекса получили:

* стоимость по параметру “Estimated Operator Cost” в 3 раза меньше;
* стоимость по параметру “Estimated I/O Cost” в 5 раз;
* стоимость по параметру “Estimated Subtree Cost” в 3 раза.

Таким образом, постройка индекса к таблице была более чем оправдана, так как мы получили прирост производительности в 3 и более раза, в зависимости от параметра, характеризующего запрос.

* 1. **Тестирование приложения.**

При разработке приложения были учтены возможные исключительные ситуации.

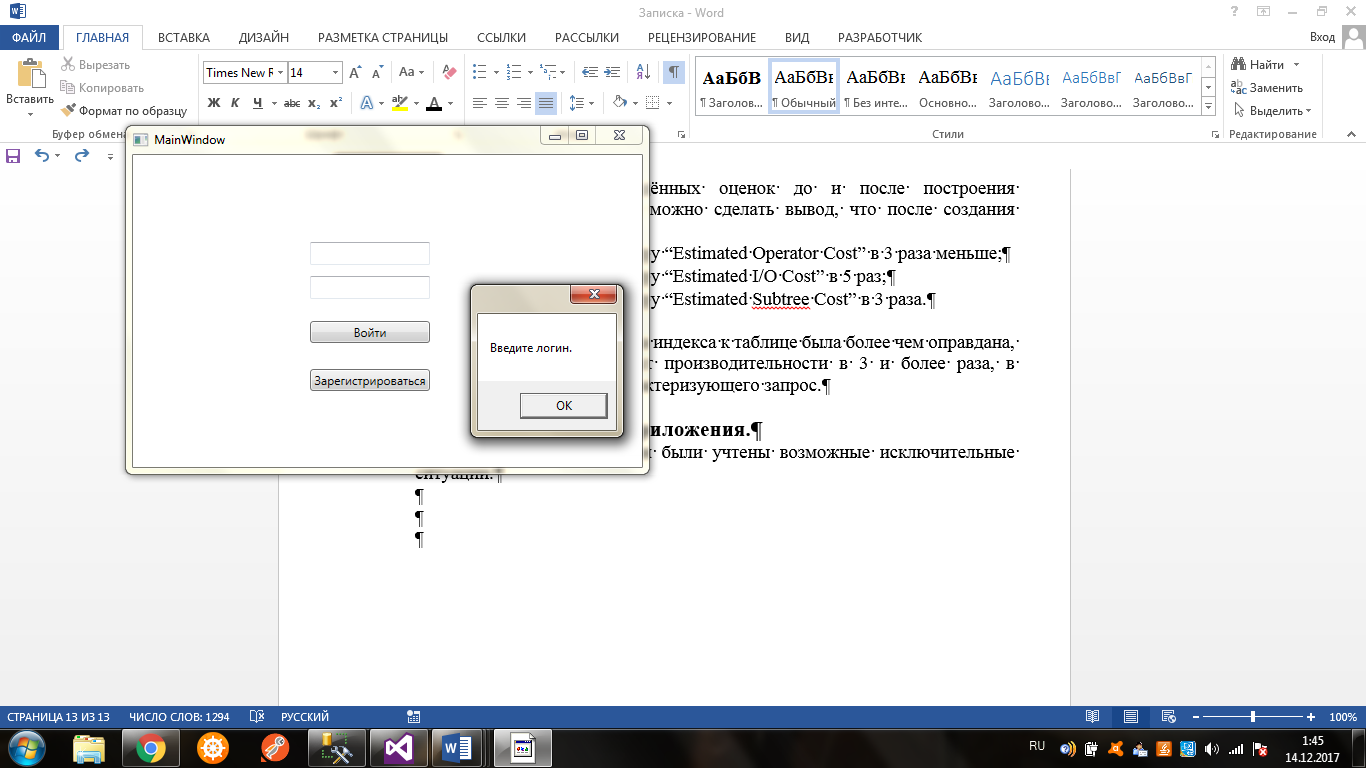


Рисунок 6.2.1. Пустое поле Логина.

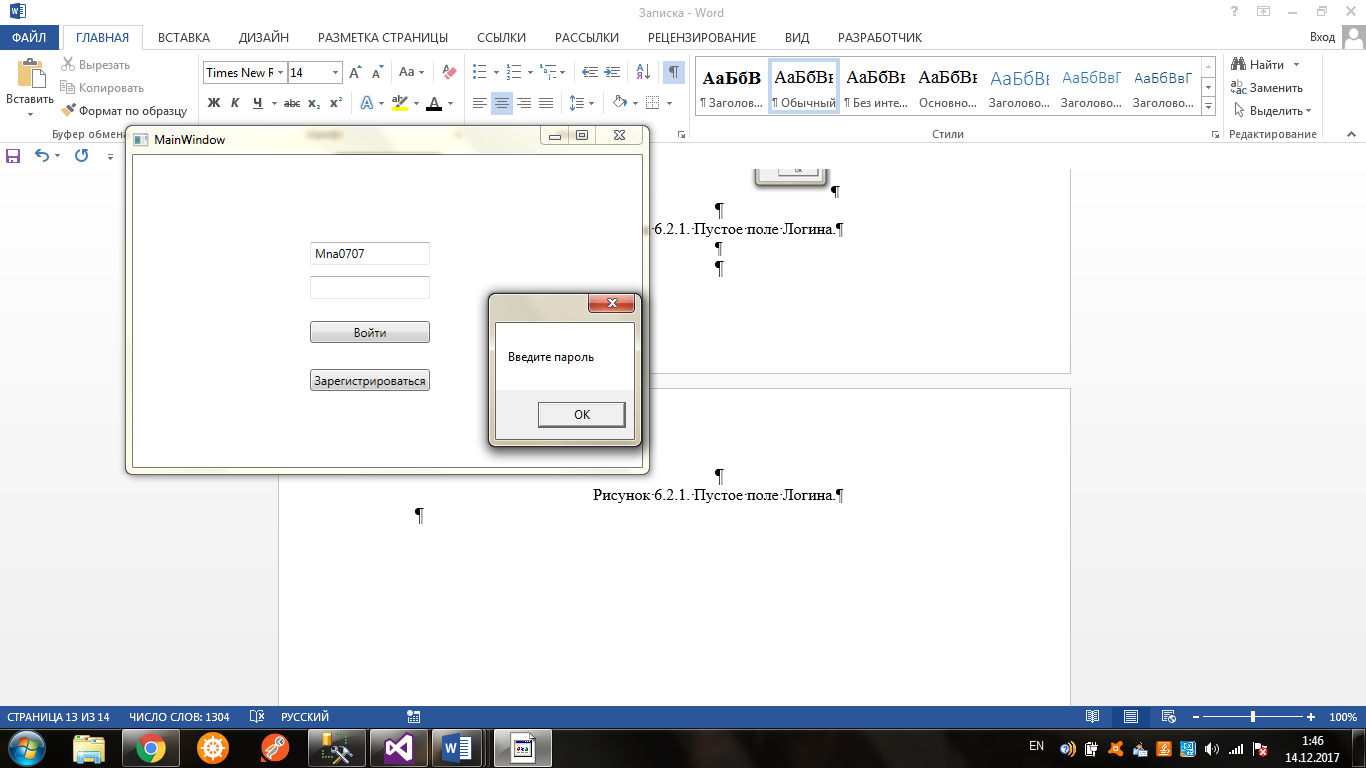


Рисунок 6.2.2. Пустое поле пароля.

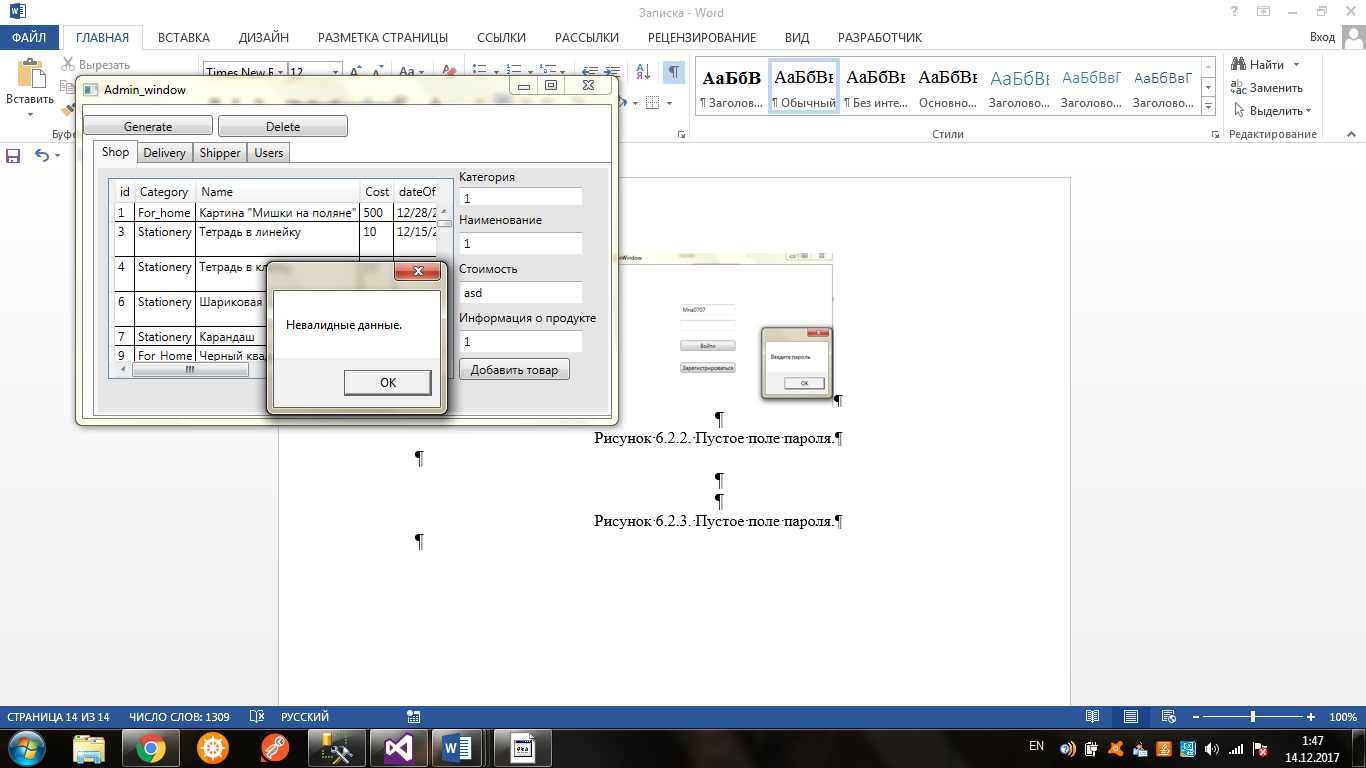


Рисунок 6.2.3. Ввод различного рода невалидных данных.

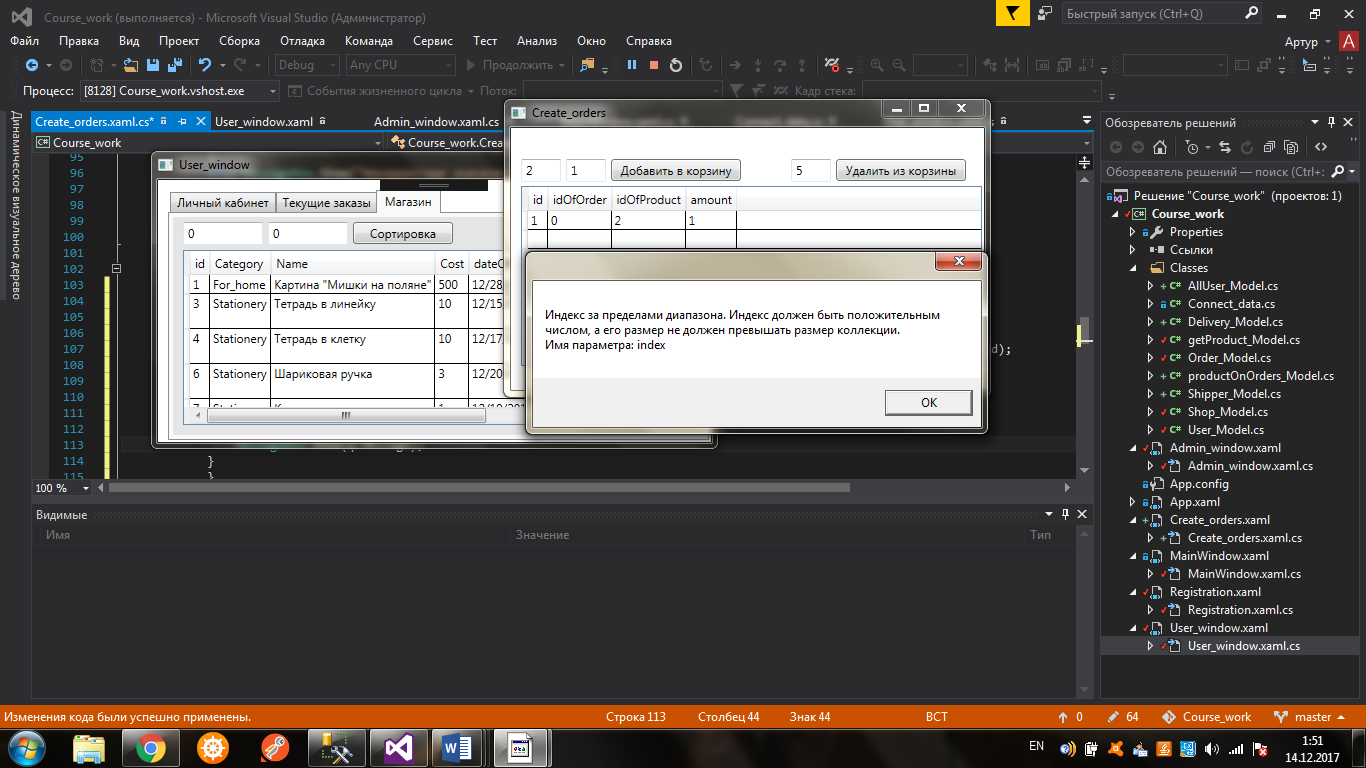


Рисунок 6.2.4. Несуществующий индекс.

# **Руководство пользователя**

## **7.1. Руководство пользователя администратора**

По умолчанию в базе данных создан один администратор, который впоследствии может назначать данную привилегию другим пользователям.

Чтобы начать управление необходимо авторизоваться. На рисунке 7.1 показано общее для всех окно авторизации

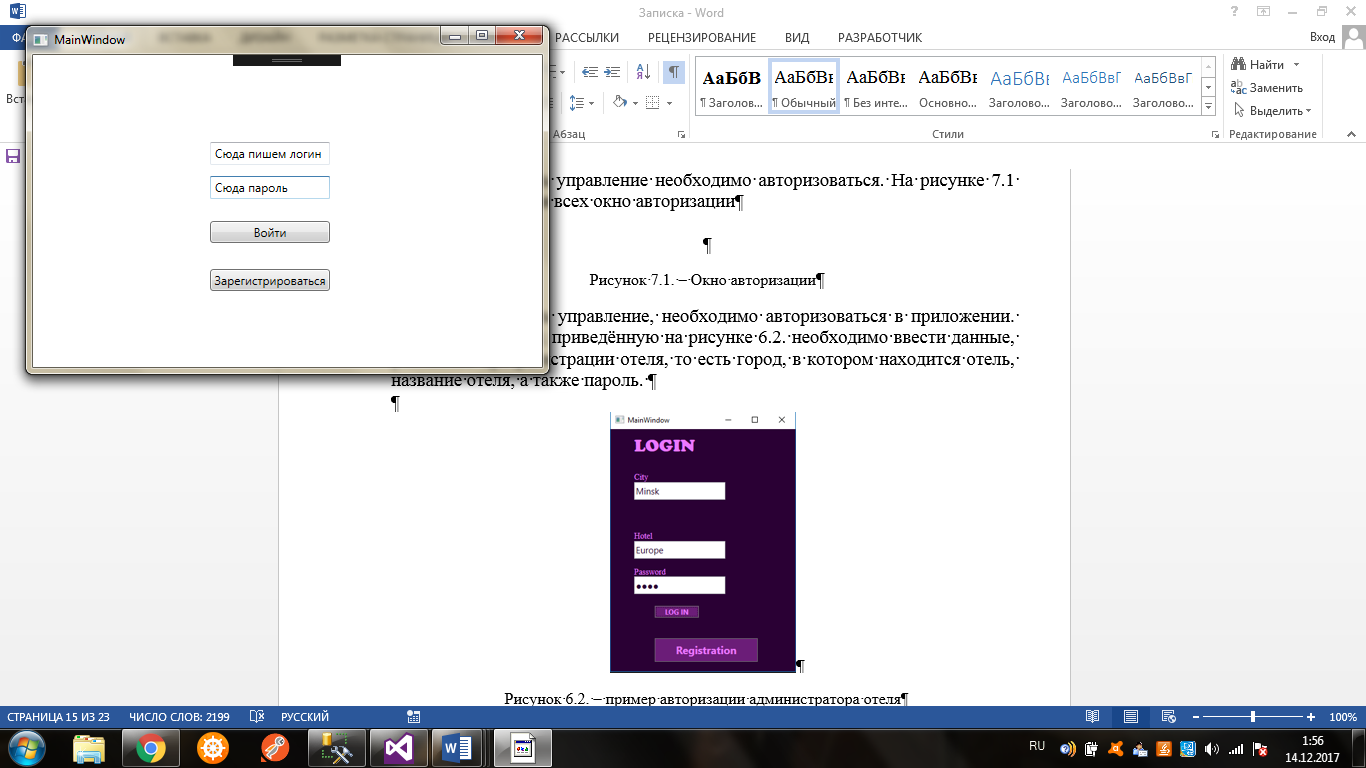


Рисунок 7.1.1. Окно авторизации

В окне администратора есть 4 вкладки: Shop, Delivery, Shippers, Users, где в левой части окна будет выведен список записей в одноименной таблице, а в правой- элементы для управления данными в таблицах.

Вкладка Shop предназначена для добавления товаров в магазин:

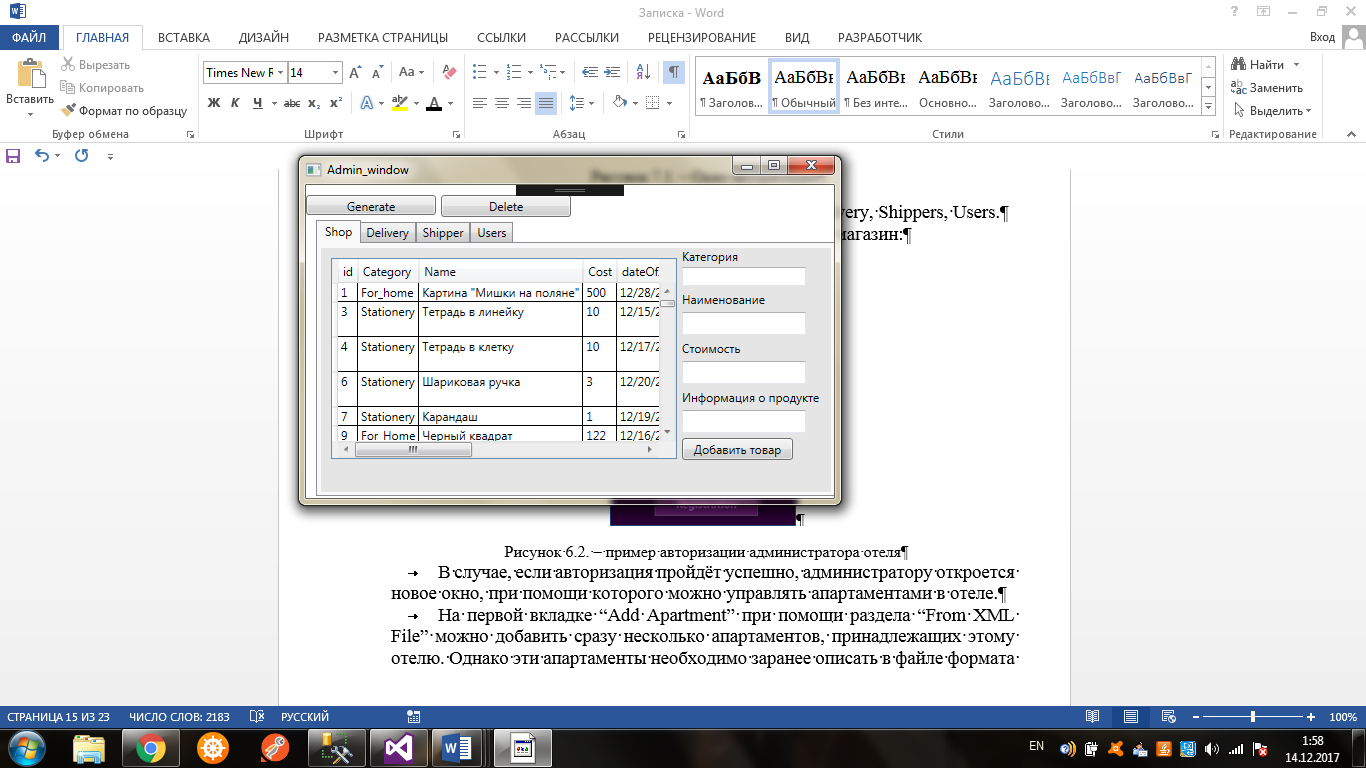


Рисунок 7.1.2. Вкладка Shop

Вкладка Delivery предназначена для добавления поставок товара

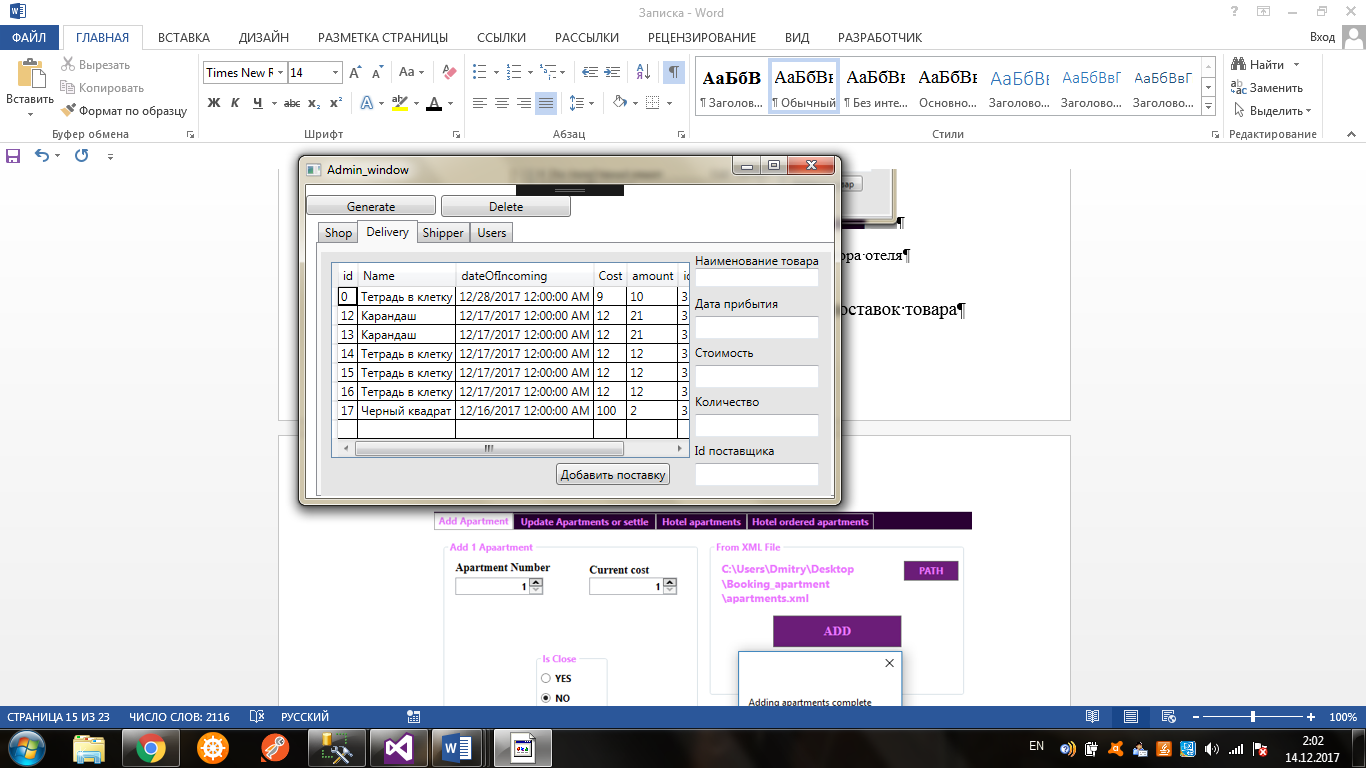


Рисунок 7.1.3. Вкладка Delivery.

Вкладка Shipper предназначена для управления записями о поставщиках.

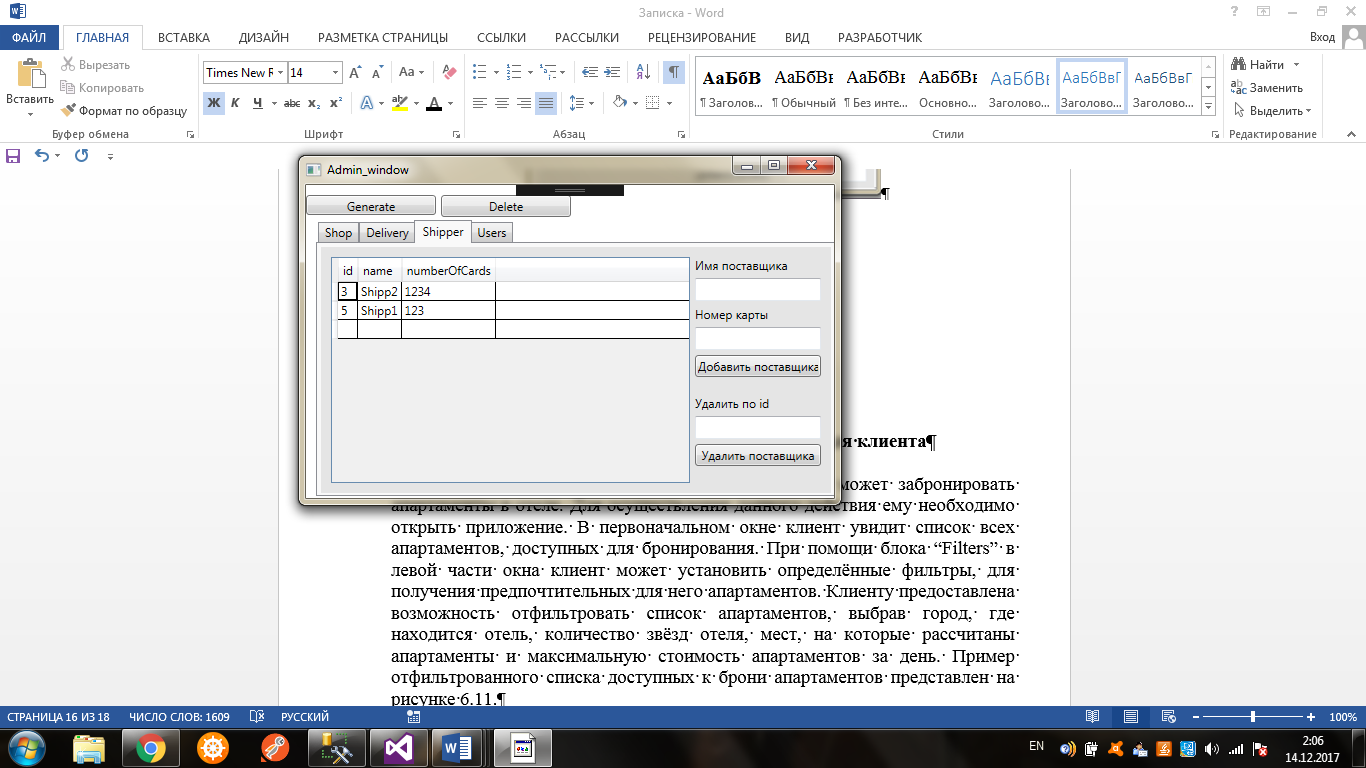


Рисунок 7.1.4. Вкладка Shipper.

Вкладка Users предназначена для выдачи и изъятия прав администратора у пользователей.

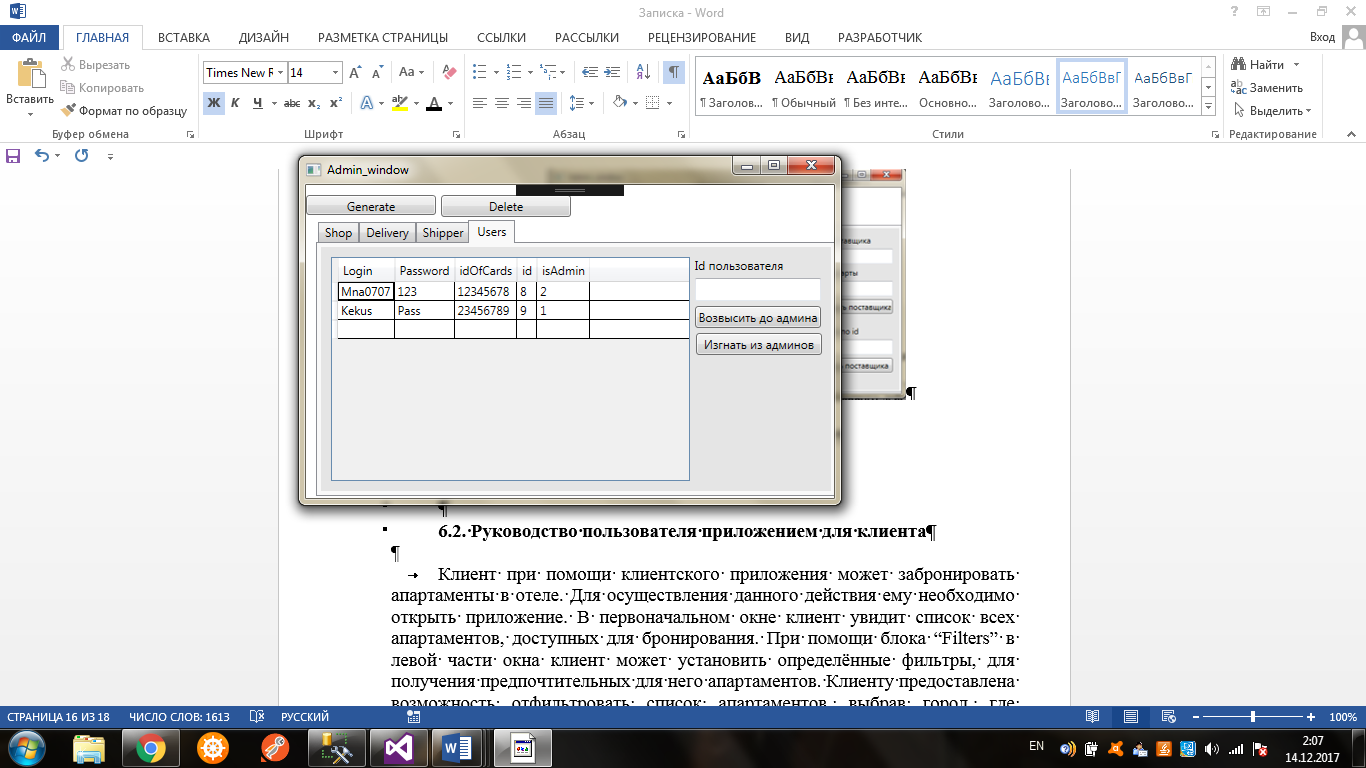


Рисунок 7.1.5. Вкладка Users.

* 1. **Руководство пользователя приложением для клиента**

Окно регистрации предназначено для регистрации новых пользователей.

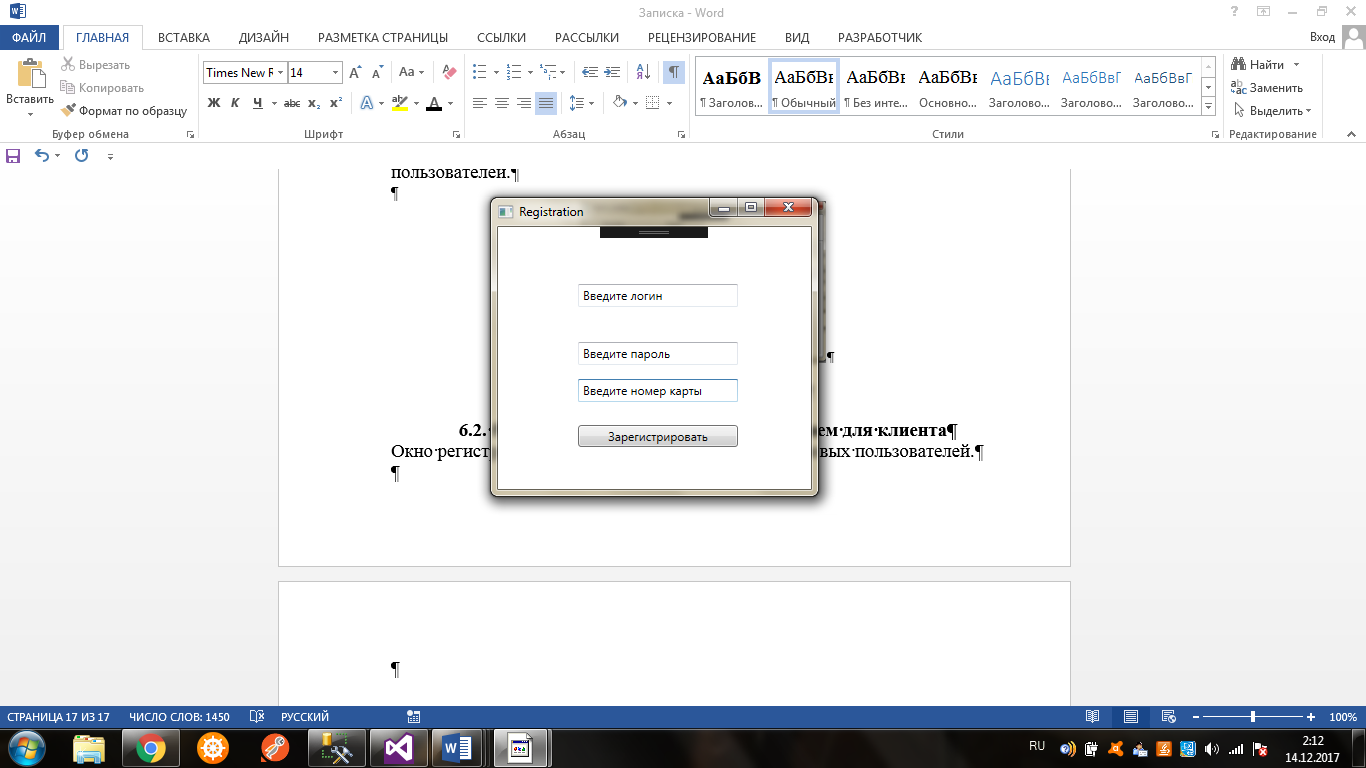


Рисунок 7.2.1. Окно регистрации.

Окно пользователя позволяет получить список заказов пользователя, товары в заказе, оформить заказ просмотреть товары и посмотреть личную информацию.

Вкладка личного кабинета выполняет чисто декоративную роль: показывает вам ваш логин и номер карты.

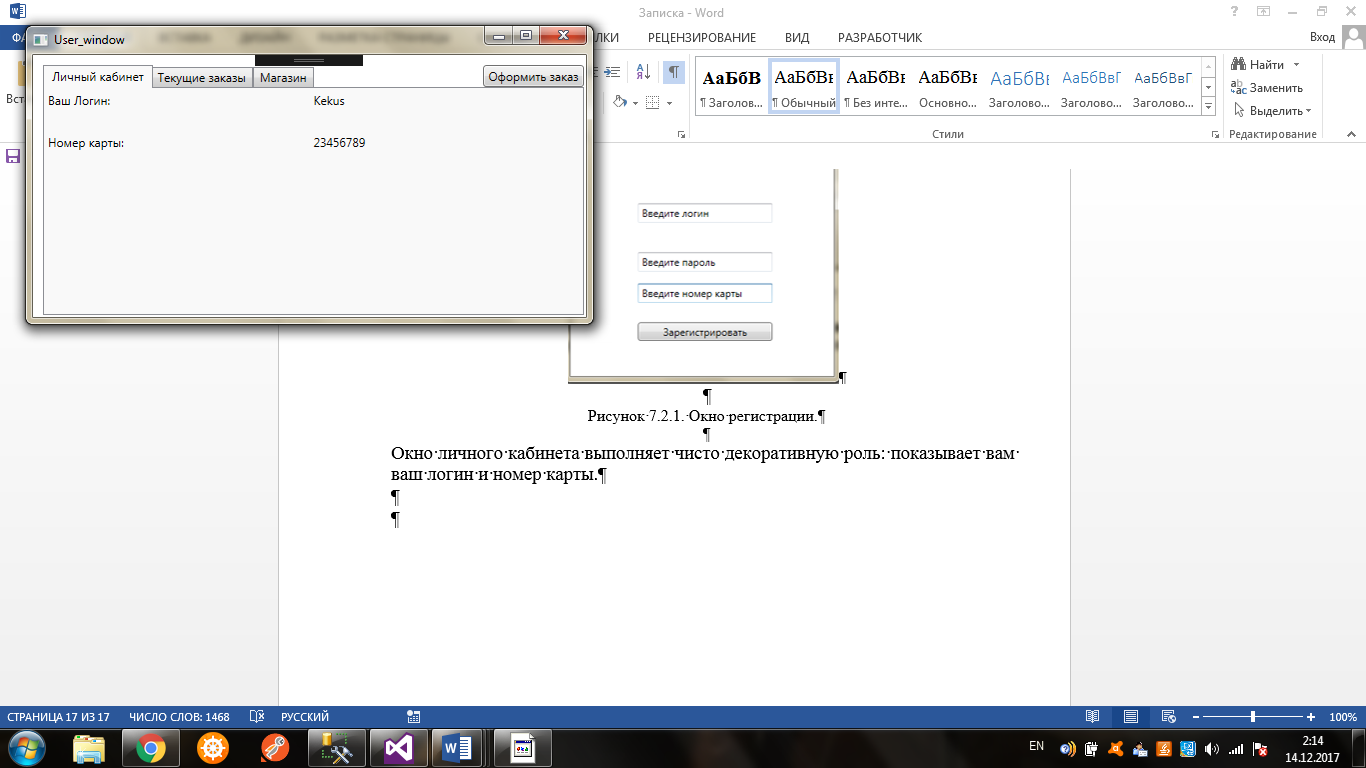


Рисунок 7.2.2. Вкладка личного кабинета.

Вкладка Текущие заказы позволяет вам получить список актуальных заказов пользователя, полный список заказов пользователя, список товаров в заказе.

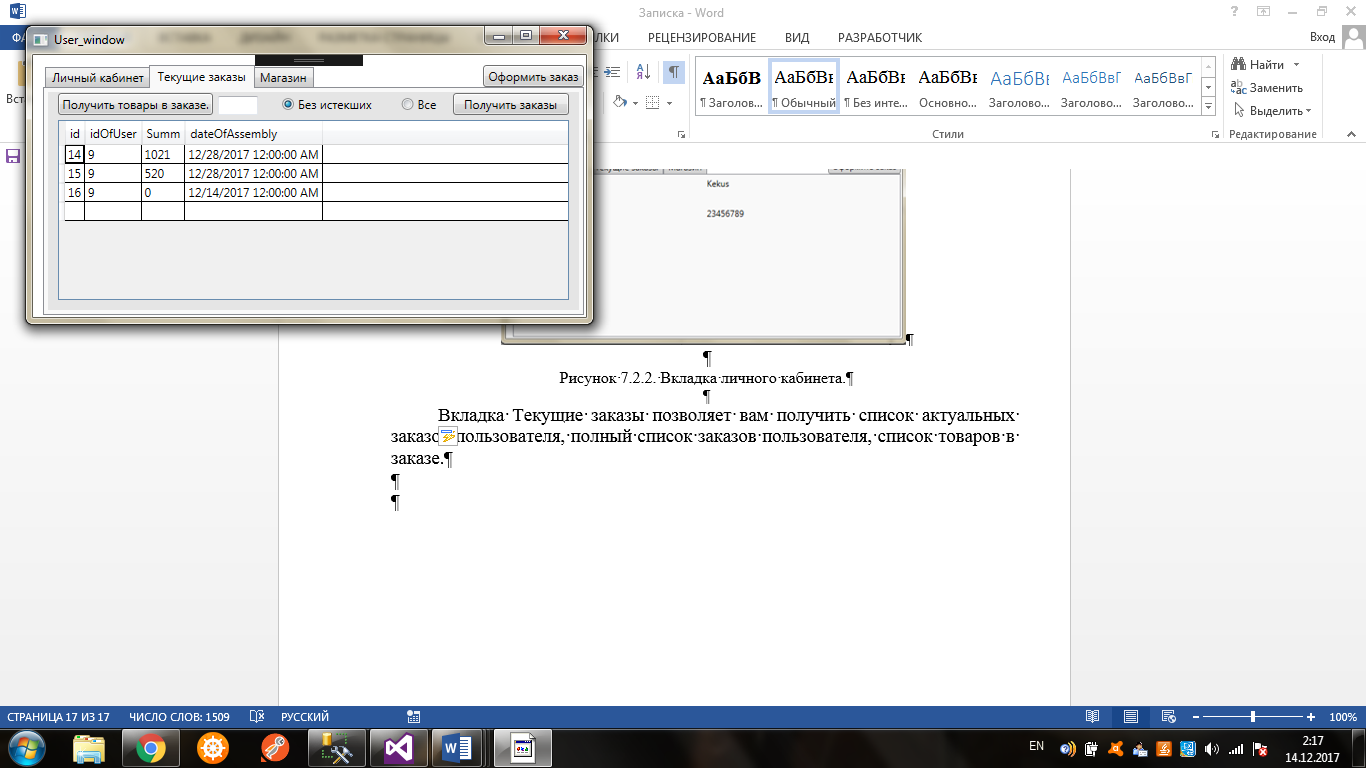


Рисунок 7.2.3. Вкладка заказов.

Вкладка Магазин позволяет вам просмотреть список товаров в магазине, отсортировать их по цене. При установке в обоих полях нулей, вы получите весь список товаров.

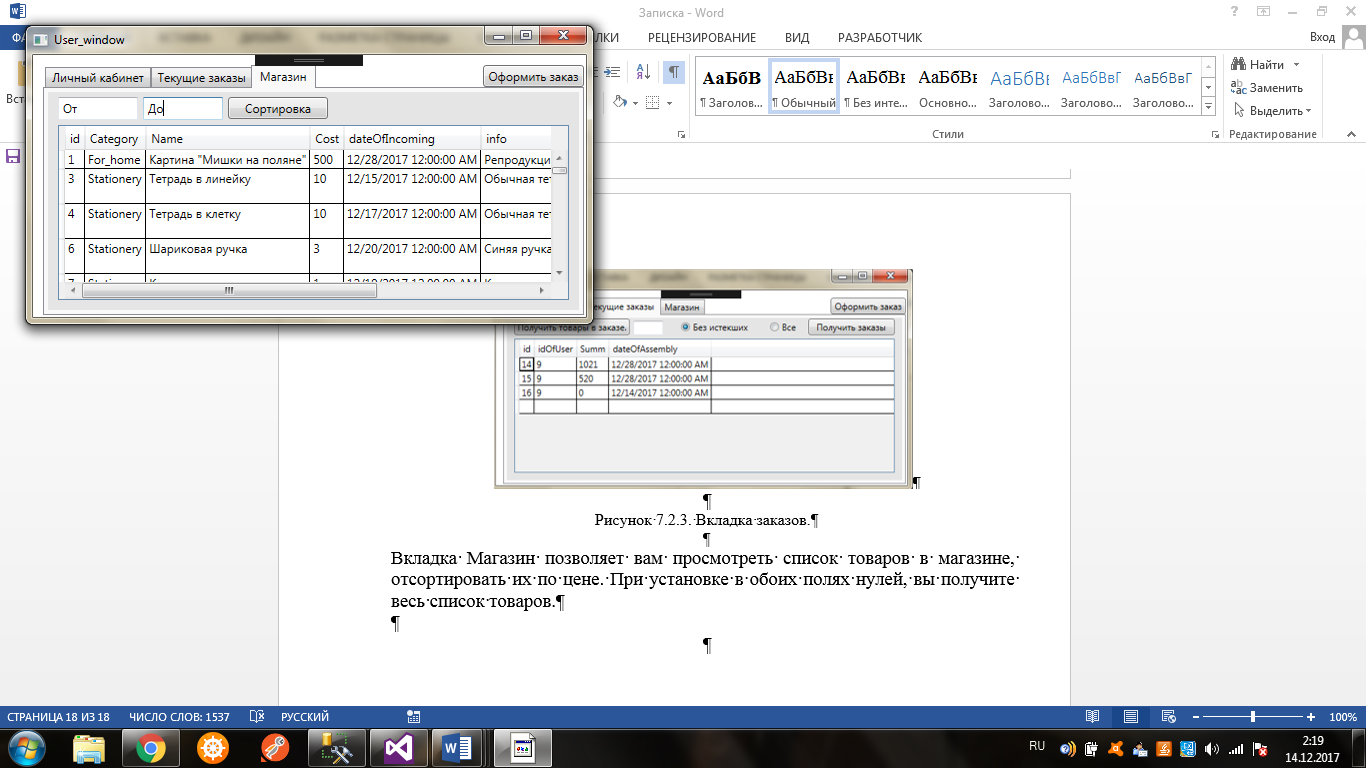


Рисунок 7.2.4. Вкладка магазина.

Окно оформления заказа позволяет вам добавить товары в корзину(вписываете в поле id и количество, нажимаете кнопку добавить). Убрать товары из корзины. Когда вы добавите все товары, нажмите кнопку оформить заказ.

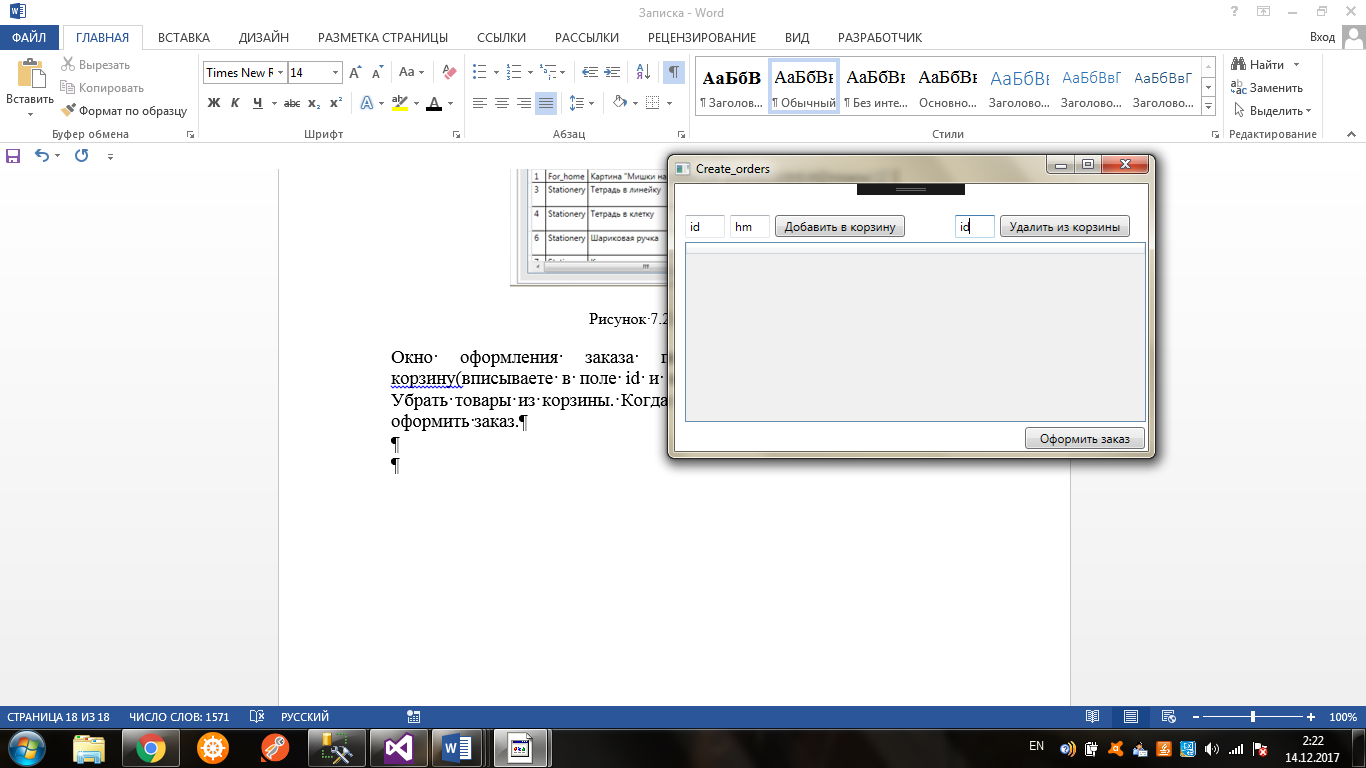


Рисунок7.2.5. Оформление заказа.

# **Заключение**

В данном курсовом проекте была разработана база данных для интернет-магазина. Также было разработано приложение для взаимодействия с базой данных. Помимо этого, было настроено резервное копирование бд на случай сбоя работы, а также разобрана технология составления отчетов.

В соответствии с полученным результатом, можно сказать, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список использованной литературы**

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. ProfessorWeb .NET & Web Programming [Электронный ресурс] – Режим доступа: [https://professorweb.ru](https://vk.com/away.php?to=https%3A%2F%2Fprofessorweb.ru&cc_key=) – Дата доступа: 30.11.2017.
4. Надёжное обслуживание баз MS SQL Server [Электронный ресурс] – Режим доступа: <https://habrahabr.ru/company/softlab/blog/266033/> - Дата доступа: 01.12.2017.

# **Приложение А**

create table Users --пользователи

(id int identity(1,1) primary key,--ид пользователя

Login nvarchar(16) unique not null, --логин

Password nvarchar(16) not null, --пароль

idOfCards int unique, --номер карты

isAdmin int default 0 --админ ли

);

create table Shop --магазин

(

id int identity(1,1) primary key, --ид товара

Category nvarchar(32) not null, --категория

Name nvarchar(32) not null unique, --наименование

Cost money not null, --стоимость

dateOfIncoming date not null, --дата поставки

info nvarchar(100) --информация

);

create table Shipper --продавец

(

id int identity(1,1) primary key,

Name nvarchar(32) not null,--наименование

numberOfCards int unique not null --номер карты

);

Create table Delivery--поставки

(

id int identity(1,1) primary key,

Name nvarchar(32) not null,--наименование товара

dateOfIncoming date not null,--дата поставки

Cost money not null,--стоимость одной единицы товара

amount int not null,--количество

idOfShipper int

);

create table productOnOrder --товар в заказе

(

id int identity(1,1) primary key,

idOfOrder int not null, --ид заказа

idOfProduct int not null, --наименование товара

amount int not null --количество

);

create table Orders--заказы

( id int identity(1,1) primary key,--ид заказа

idOfUser int not null,--ид пользователя

Summ money not null,--сумма

dateOfAssembly date not null--дата сборки

);

create table Storage--склад

(id int identity(1,1) primary key,

nameOfProduct nvarchar(32) unique, --наименование товара

hmOnStorage int default 0 --количество на складе

);

create table Discount--скидки

(id int identity(1,1) primary key,--идентификатор

dateOfDiscount date unique not null, --дата скидки

percents real not null--коэффицент

);

# **Приложение Б**

use master;

CREATE LOGIN [IS\_user] WITH PASSWORD=N'IS\_user',

DEFAULT\_DATABASE=[Course\_work], DEFAULT\_LANGUAGE=[русский],

CHECK\_EXPIRATION=OFF, CHECK\_POLICY=ON

GO

ALTER LOGIN [IS\_user] ENABLE

GO

CREATE LOGIN [IS\_admin] WITH PASSWORD=N'IS\_admin',

DEFAULT\_DATABASE=[Course\_work], DEFAULT\_LANGUAGE=[русский],

CHECK\_EXPIRATION=OFF, CHECK\_POLICY=ON

GO

ALTER LOGIN [IS\_admin] ENABLE

GO

use Course\_work;

CREATE USER [user] FOR LOGIN [IS\_user];

GO

CREATE USER [admin] FOR LOGIN [IS\_admin];

GO

CREATE ROLE [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_DeliverySelect] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_OrdersSelect] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_productOnOrderSelect] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_ShipperSelect] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_ShopSelect] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_ShopSelectAll] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_UsersSelect] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_SelectCurrentOrders] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_SelectUsersOrders] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_SelectproductByOrdersId] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_getProductByCost] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_OrdersInsert] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_productOnOrderInsert] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_getSummOfOrder] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_OrdersSumUpdate] TO [Course\_work\_user];

GRANT EXECUTE ON [dbo].[usp\_OrdersDateUpdate] TO [Course\_work\_user];

go

sp\_addrolemember 'Course\_work\_user', 'user';

GRANT EXECUTE ON SCHEMA::[dbo] TO [admin]

# **Приложение В**

-------------------------Выборка Пользователя по логину.

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROC [dbo].[usp\_UserSelectByLogin]

@login nvarchar(16),

@password nvarchar(16)

AS

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

declare @adm int;

select @adm=[isAdmin]from Users where Login=@login and Password=@password;

if (@adm!=0)

select \* from Users where Login=@login and Password=@password;

return @adm;

COMMIT

--------------------Выбор актуальных заказов конкретного пользователя

IF OBJECT\_ID('[dbo].[usp\_SelectCurrentOrders]') IS NOT NULL

BEGIN

DROP PROC [dbo].[usp\_SelectCurrentOrders]

END

GO

CREATE PROC [dbo].[usp\_SelectCurrentOrders]

@idOfUser int

AS

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

SELECT \* FROM [dbo].[Orders] where (([dateOfAssembly]>=CONVERT(date, SYSDATETIME())) and ([idOfUser]=@idOfUser))

COMMIT

GO

---------------------------Получить стоимость заказа

IF OBJECT\_ID('[dbo].[usp\_getSummOfOrder]') IS NOT NULL

BEGIN

DROP PROC [dbo].[usp\_getSummOfOrder]

END

GO

CREATE PROC [dbo].[usp\_getSummOfOrder]

@idOfOrder int

AS

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

declare @currSum money;

create table #t (summ money);

SELECT @currSum=SUM(productOnOrder.amount\*Shop.Cost) FROM [dbo].[productOnOrder], [dbo].Shop where (Shop.id=productOnOrder.idOfProduct) and (productOnOrder.idOfOrder=@idOfOrder)

insert into #t values (@currSum);

if(@currSum!=0)

select \* from #t;

COMMIT

return @currSum;

GO

------------проверка даты прибытия

IF OBJECT\_ID('[dbo].[usp\_OrdersDateUpdate] ') IS NOT NULL

BEGIN

DROP PROC [dbo].[usp\_OrdersDateUpdate]

END

GO

CREATE PROC [dbo].[usp\_OrdersDateUpdate]

@id int

AS

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

declare @maxdate date;

select @maxdate=max(dateOfIncoming) from [dbo].Shop inner join [dbo].productOnOrder on Shop.id=productOnOrder.idOfProduct

where productOnOrder.idOfOrder=@id;

UPDATE [dbo].[Orders]

SET [dateOfAssembly]=@maxdate

WHERE [id] = @id

-- Begin Return Select <- do not remove

SELECT [id], [idOfUser], [Summ], [dateOfAssembly]

FROM [dbo].[Orders]

WHERE [id] = @id

-- End Return Select <- do not remove

COMMIT

GO

---------------------Получить товары в диапазоне цен

IF OBJECT\_ID('[dbo].[usp\_getProductByCost] ') IS NOT NULL

BEGIN

DROP PROC [dbo].[usp\_getProductByCost]

END

GO

CREATE PROC [dbo].[usp\_getProductByCost]

@fromSum money,

@toSum money

AS

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

if(@toSum!=0 and @toSum>=@fromSum)

select \* from [dbo].Shop where Shop.Cost between @fromSum and @toSum;

else if(@toSum=0 and @fromSum=0) select \* from [dbo].Shop;

COMMIT

GO

----------------------Вставка данных поставки изменяющая даты в магазине.

USE [Course\_work];

GO

IF OBJECT\_ID('[dbo].[usp\_DeliveryInsert]') IS NOT NULL

BEGIN

DROP PROC [dbo].[usp\_DeliveryInsert]

END

GO

CREATE PROC [dbo].[usp\_DeliveryInsert]

@Name nvarchar(32),

@dateOfIncoming date,

@Cost money,

@amount int,

@idOfShipper int = NULL

AS

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

declare @shopDate Date;

INSERT INTO [dbo].[Delivery] ([Name], [dateOfIncoming], [Cost], [amount], [idOfShipper])

SELECT @Name, @dateOfIncoming, @Cost, @amount, @idOfShipper;

select @shopDate=Shop.dateOfIncoming from Shop where Shop.Name=@Name;

Update Storage set hmOnStorage+=@amount where @Name=Storage.NameOfProduct;

if(@dateOfIncoming>@shopDate)

Update Shop set dateOfIncoming=@dateOfIncoming where Shop.Name=@Name;

-- Begin Return Select <- do not remove

SELECT [id], [Name], [dateOfIncoming], [Cost], [amount], [idOfShipper]

FROM [dbo].[Delivery]

WHERE [id] = SCOPE\_IDENTITY()

-- End Return Select <- do not remove

COMMIT

GO

# **Приложение Г**

IF OBJECT\_ID('[dbo].[insertUsersFromXml]') IS NOT NULL

BEGIN

DROP PROC [dbo].[insertUsersFromXml]

END

GO

CREATE PROC [dbo].[insertUsersFromXml]

@path nvarchar(256)

AS

begin

SET NOCOUNT ON --отлючить вывод кол-ва обработанных строк

SET XACT\_ABORT ON --ролбэк транзакции и прекращение процедуры

declare @count1 int=0;

declare @count2 int=0;

set @count1 = (select count(\*) from Shop)

BEGIN TRAN

declare @results table (x xml) --таблица для рерзультата прочтения файла xml

declare @sql nvarchar(300)=

'SELECT

CAST(REPLACE(CAST(x AS VARCHAR(MAX)), ''encoding="utf-16"'', ''encoding="utf-8"'') AS XML)

FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)'; --чтение xml

INSERT INTO @results EXEC (@sql) -- -> результат в поле таблицы @results

declare @xml XML = (SELECT TOP 1 x from @results); -- поле с xml таблицы в переменную типа xml

INSERT INTO Users(Login,Password,idOfCards,isAdmin)

SELECT

C3.value('Login[1]', 'nvarchar(16)') AS Login,

C3.value('Password[1]', 'nvarchar(16)') AS Password,

C3.value('idOfCards[1]', 'int') AS idOfCards,

C3.value('isAdmin[1]', 'int') AS isAdmin

FROM @xml.nodes('Root/Users') AS T3(C3)

COMMIT;

set @count2 = (select count(\*) from Shop)

return @count2 - @count1;

end;

# **Приложение Д**

IF OBJECT\_ID('[dbo].[exportUsersToXML]') IS NOT NULL

BEGIN

DROP PROC [dbo].[exportUsersToXML]

END

GO

CREATE PROC [dbo].[exportUsersToXML]

@path nvarchar(256) --path to file

AS

begin

BEGIN TRAN

declare @sql nvarchar(500)=

'bcp "SELECT \* '+ --select

'FROM [dbo].[Users] '+ --select

' FOR XML PATH(''Users''), ROOT(''Root'')" queryout "'+@path+'" -S Ashenvale -d Course\_work -w -T '; --once\_item\_name/ root\_name/ path to file/ server path/ warchar + T

EXEC xp\_cmdshell @sql;

COMMIT;

end;

GO