**Введение**

Задача курсовой работы – разработка приложения «Склад строительных материалов» в среде Visual Studio Express 2017 на языке C# используя WPF и язык XAMEL с подключением к базе данных, разработанной в среде Microsoft SQL Server Management Studio 17.

«Склад готовой продукции» – это автоматизированное рабочее место (далее АРМ) работника склада, ответственного за учет, выдачу и прием продукции на территории склада. К приложению также могут иметь доступ другие работники предприятия из управляющего состава.

Приложение должно позволять оперативно изменять информацию в системе о количестве имеющегося на складе товара.

**Обзор аналогов**

Аналогом этого приложения будет являться программа

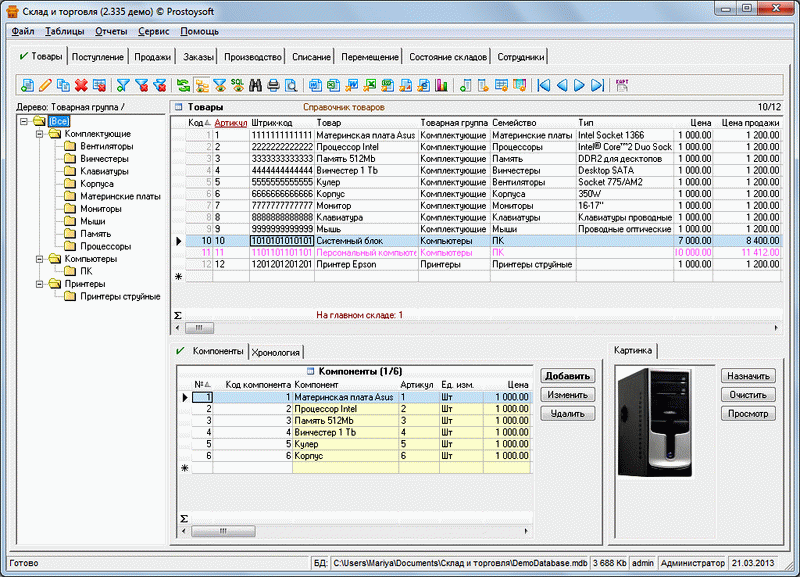
«Склад и торговля»

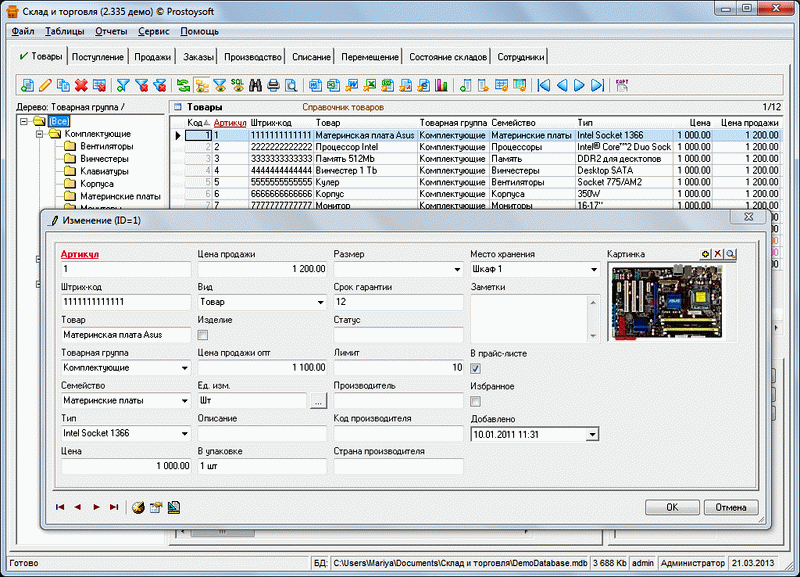
**"Склад и торговля"** - простая, надежная система для автоматизации торговли и складского учета. Программа поможет Вам легко и создавать первичные документы (счета, счета-фактуры, накладные, договоры и др.), контролировать остатки на складе, делать резервирование товаров, вести учет задолженности клиентов и поставщикам, учитывать закупочные цены и калькулировать себестоимость продуктов при производстве, подсчитывать полученную прибыль и многое другое. Широкие возможности настройки интерфейса - конструктор форм, персонализация, возможность составления шаблонов печатных документов позволяют создавать произвольные документы и отчеты. Использование кода на языке Visual Basic Script во внешних модулях позволяет неограниченно расширять функциональные возможности программы.

Основные функции программы

* Учет основных товарных операций (приход, расход, возврат, резервирование, инвентаризация)
* Учет продаж и расход товаров
* Учет заказов от покупателей и заказов поставщикам
* Внутренние перемещение, списание товаров
* Отслеживание остатков на складах
* Поддержка нескольких складов
* Учет расходов денежных средств и МБП (малоценных и быстроизнашивающихся предметов)
* Работа с прайс-листами
* Отслеживание оплат
* Начисление и учет зарплат сотрудникам
* Печать ценников, визиток
* Обработка коммерческих и логистических операций с товарами
* Движение товаров внутри склада
* Учет клиентской активности и создание заказов поставщикам на его основе
* Работа по нескольким складам (создание нескольких несвязанных учетных групп внутри одной программы складского учета)
* Контроль наличной и безналичной оплаты
* Создание документов по существующим стандартам и произвольно
* Экспорт, импорт и синхронизация любых данных, обрабатываемых программой складского учета и торговли
* Гибкая структура базы данных с настройкой под любые задачи
* Сетевой и многопользовательский режимы, гибкая настройка прав доступа с ограничением по полям и по записям
* Кастомизация - настройка меню, панелей инструментов и других элементов персонально под каждого пользователя

Скриншоты программы





1***. Описание предметной области и определение требований к системе с точки зрения предметной области.***

1.1 Предметная область

Исследуемая предметная область представляет собой приложение для небольшого склада готовой продукции, осуществляющий продажу мобильных телефонов и аксессуаров к ним.

Склад обязан вести учет продукции, находящейся на временном хранении, составление карточек, кладовых книг, описей, приходных и расходных накладных, ордеров по учету прихода, расхода, наличия, остатков продукции на складе, учет выполнения заказов по отгрузке и разгрузке готовой продукции, составление отчетов о загрузке складских площадей.

Рассмотрим типичные бизнес-процессы складского учета на не автоматизированном гипотетическом складе. Такое рассмотрение проводится с целью выявить недостатки существующей системы складского учета, а также показать необходимость автоматизации склада.

Процедура принятия продукции на склад:

* Продукция приходит на склад в сопровождении экспедитора и приходной накладной;
* Контролер на складе, проверяет приходную накладную, и регистрирует ее в книге учета входящих документов (накладных);
* Осматривает входящую продукцию, и если с ней все нормально принимает ее на склад, передавая экспедитору товара выписку (документ) о том, что товар принят на хранение;
* Грузчики отвозят товар в свободное место хранения, и контролер делает запись в книге учета о том, где хранится вновь поступившая продукция.

Склад готовой продукции не занимается никакой коммерческой деятельностью, а только осуществляет процедуру хранения продукции для сторонних лиц заинтересованных в этом.

В рамках курсовой работы было решено автоматизировать процессы выдачи и учета материалов, а также предоставить возможность руководящему составу и производителю работ получать данные о состоянии склада.

Проанализировав ситуацию на складе и выявив все минусы, постараемся создать такую систему, которая бы автоматизировала следующие операции на складе:

* Регистрация документов осуществляется с помощью ЭВМ;
* Поиск товаров для отгрузки будет проводиться путем поиска соответствующего товара в БД и просмотра информации о месте его хранении (номер склада).
* Формирование заказов, будет производиться системой автоматически.
* Формирование заявок, добавление товара или удаления товара, будет производиться системой автоматически.
* Поиск заявок и распечатка
* Распределение товара по горячим и холодным зонам хранения в зависимости от частоты попадания в заявки.

При помощи ЭВМ на складе автоматизирован учет поступления и отгрузки товаров, учет входящих и исходящих документов, количественный учет. В общем объеме учетных работ эти задачи имеют значительный удельный вес. Их автоматизация позволяет сократить ручные операции, ускорить обработку информации, повысить точность учета. В памяти ЭВМ хранится и может быть выдана на печать детальная информация о количестве поступления и отгрузки конкретного товара по каждому документу в случае несовпадения величины запаса с данными машинного учета.

Главное назначение автоматизированной системы в данном случае – повысить эффективность выполнения основных функций работников склада.

# 2. *Постановка задачи и обзор методов её решения.*

Основное преимущество автоматизации - это сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления избыточных копий и устранение возможности возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте, увеличение степени достоверности информации и увеличение скорости обработки информации; излишнее количество внутренних промежуточных документов, различных журналов, папок, заявок и т.д., повторное внесение одной и той же информации в различные промежуточные документы. Также значительно сокращает время автоматический поиск информации, который производится из специальных экранных форм, в которых указываются параметры поиска объекта.

Задача курсовой работы – разработка приложения «Склад» в среде Visual Studio Professional 2017 на языке C# с подключением к базе данных, разработанной в среде MS SQL, включающей все необходимые инструментальные средства для создания локальной базы данных. В ее файле могут храниться не только данные, но и объекты интерфейса: отчеты, формы, запросы.

Техническое задание составлялось на основании требований Заказчика, изложенных в устной форме: АРМ «Склад» должно обеспечивать оператору возможность быстрого получения данных о состоянии склада. Приложение должно предоставлять доступ к просмотру и редактированию данных в зависимости от роли пользователя в системе. Роль – совокупность связанных функций, выполняемых пользователем приложения. Доступ к приложению обеспечивается на основании введенных учетных данных. Права доступа назначаются администратором системы.

**Требования к системе в целом:**

* Система должна предоставлять полноценный оконный интерфейс Windows-приложения;
* Система должна обеспечивать безопасность на уровне разграничения прав доступа пользователей по ролям и в соответствии с организационной структурой предприятия;

**Функциональные требования**

Система должна позволять вести общую информацию по следующим справочникам:

* Пользователи;
* Заказы;
* Заказано Товаров;
* Склад
* Каталог

обрабатывая и по возможности исключая ошибки пользовательского ввода.

# Проектирование системы

Процесс проектирования приложения был разбит на три этапа: Проектирование структуры классов приложения, проектирование базы данных и проектирование интерфейса.

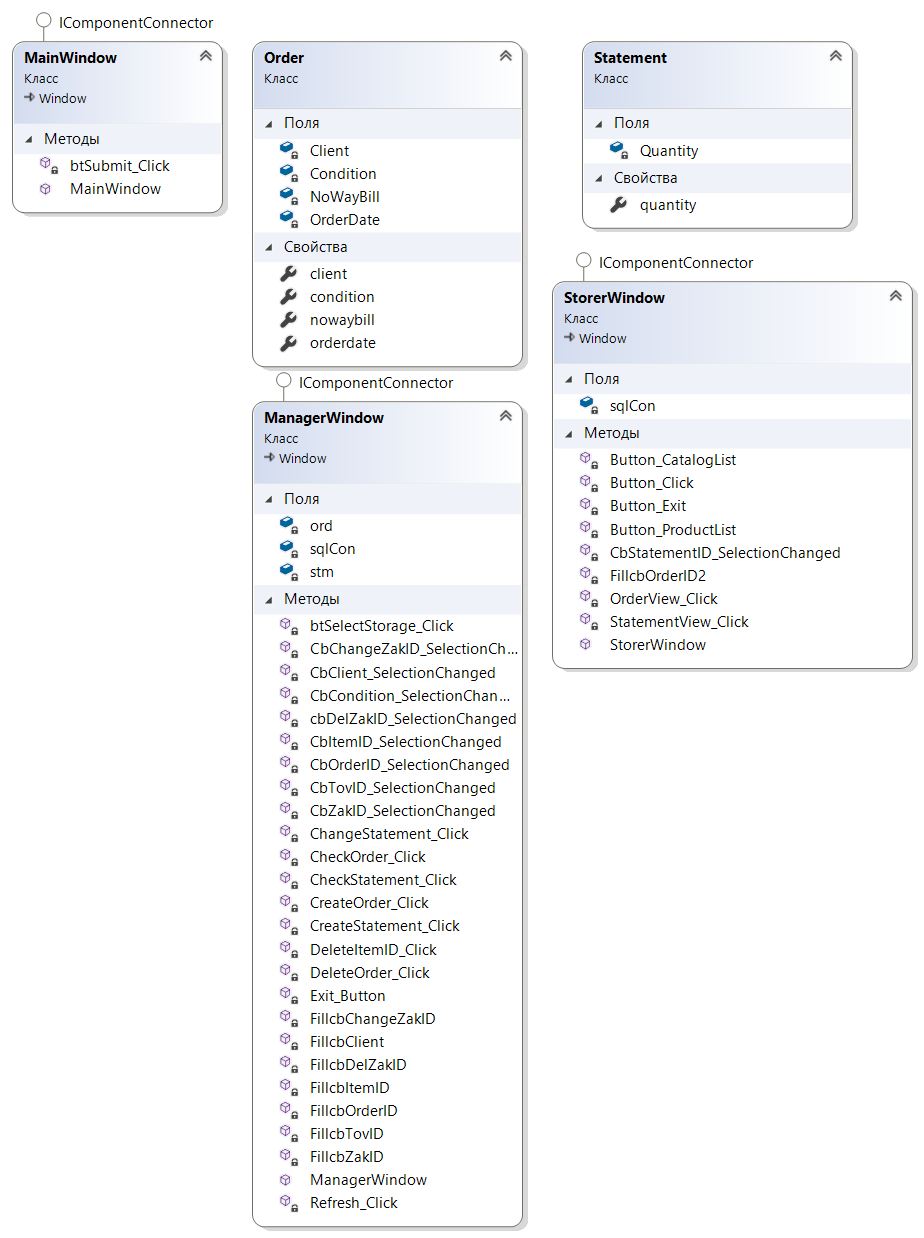
***3. Модели представления системы и их описание.***

Для разностороннего изучения разрабатываемой системы представим различные модели, разработанные в Rational Rose Enterprise Edition. Основные функции, выделенные для системы, представлены на диаграмме вариантов использования:



Диаграмма классов, характеризующее разрабатываемое приложение была сгенерирована MS Visual Studio. На них представлены классы разработанные при написании приложения.

**Диаграмма классов**



Класс MainWindow является окном ввода Логина и Пароля, после ввода которого в зависимости от выбранного Логина открывается соответствующее окно. Реализует это метод btSubmit\_Click()

Данный метод извлекает данные из таблицы Users базы данных MyStorage, который содержит логины и пароли. Затем открывает соответствующее окно.

Класс ManagerWindows является рабочим окном для Менеджера, в котором он:

* Оформляет заказы
* Оформляет Заявки
* Получает данные о товаре и остатках

Оформление заказа происходит в два этапа:

Ввод данных через текстовые поля «Клиент», «Состояние», «Номер накладной», «Дата Заказа», «Дата Отгрузки»

За создание Заказа отвечает метод CreatOrder\_Click()

Данный метод передаю хранимой процедуре CreatOrder четыре параметра:

@Client

@NoWayBill

@OrderDate

@Condition

И вносит данные в таблицу «Orders» базы данных MyStorage

Также Менеджер может просматривать, изменять или удалять заказы.

Реализуют все это методы:

DeleteOrder\_Click();

ChangeStatement\_Click();

CheckOrder\_Click();

Метод DeleteOrder\_Click() осуществляет возврат имеющихся заказов, которые находятся на этапе «Оформление» и удаляет выбранный заказ. Если Заказ уже отгружен заказ не удаляется. За это отвечает хранимая процедура DeleteOrder

Метод ChangeStatement\_Click() возвращает заказы которые находятся на этапе «Оформления» и вносит изменения в таблицу Orders при помощи хранимой процедуры ChangeStatement

Метод CheckOrder\_Click() возвращает все заказы из таблицы Orders

Используя хранимую процедуру DataOrders

Второй этап – это оформление Заявки(добавление товара в заказ).

Создание Заявки и добавление товара осуществляет метод

CreateStatement\_Click(). Этот метод передает три параметра:

@OrderID – номер заказа;

@ItemID – номер товара;

@Quantity – количество заказанного товара;

Номера Заказов и Товаров уже хранятся в выпадающих списках(ComboBox).

Просмотр Заявок осуществляет метод CheckStatement\_Click() и хранимая процедура DataStatement

Если Менеджеру понадобиться удалить товар из заявки, то это осуществит метод DeleteItemID\_Click(), который принимает в себя два параметра:

@OrderID – номера имеющихся заказов, которые находятся на этапе «Оформление»

@ItemID – номера товаров, которые имеются в Заявках

И внесет изменения в таблицу Statement, при помощи хранимой процедуры DelItemID

Данные о наименовании, количестве, цене и адресе хранения находятся в таблице Storage.

Метод btSelectStorage\_Click() осуществляет вывод информации о товаре используя хранимую процедуру DataStorage.

Методы Refresh\_Click() и Exit\_Button() отвечают за обновление окна и выход из приложения.

Класс StorerWindow является рабочим окном для Кладовщика, в котором он:

* Просмотр списка продукции
* Просмотр заказов
* Печать заказов
* Распределение товара по горячим и холодным зонам хранения

За просмотр списка продукции отвечает известный нам метод btSelectStorage\_Click();

Для того, что бы кладовщику просмотреть необходимо воспользоваться методом OrderView\_Click(), который выводит список заказов находящихся на этапе «Оформление».

После, сотрудник склада выбирает необходимый заказ и может просмотреть наличие товара в нем при помощи метода StatementView(). Воспользовавшись хранимой процедурой Statement\_Storage метод передает параметр @OrderID и возвращает данные из таблицы Statement и Storage.

Главная проблема всех складов является не рациональное размещение товара на складе.

Размещение товара на складе может быть случайным и оптимизированным. Главный принцип оптимизации размещения товаров на складе заключается в минимизации передвижений посредством разделения всего ассортимента на группы, требующие большого количества перемещений, и группы, к которым обращаются достаточно редко.

К первой группе относятся товары, имеющие значительную интенсивность потребления в течение определенного периода времени, ко второй группе – товары, которые отличаются относительно небольшой интенсивности потребления.

Для разделения всего ассортимента товаров, поступающих на склад, на эти две группы используют правило Парето(20\80), которое применительно к логистике складирования заключается в следующем.

Часто отпускаемые товары составляют лишь небольшую часть номенклатуры – около 20% наименований. Располагать их необходимо в удобных, максимально приближенных к зонам отпуска местах, вдоль так называемых «горячих» линий. Товары, которые требуются реже, составляют 80% наименований номенклатуры. Их отодвигают на второй план и размещают вдоль «холодных зон». Вдоль горячих зон должны располагаться крупногабаритные товары, а также те, которые хранятся без тары.

За распределения товара по горячим зонам хранения будет отвечать метод Button\_Click(). С помощью хранимой процедуры Raspredelenie,

таблицы Storage, Orders, Statement и NewStorage

он будет отыскивать товар, который очень часто попадает в заявки и распределять его по «горячим и холодным зонам» в порядке уменьшения.

Эта функция поможет работникам склада рационально распределять товар и сэкономит время отбора заявок.

***4. Информационная модель системы и её описание (с доказательством приведения ее к 3-ей нормальной форме).***

# База данных

# База данных для приложения разрабатывалась с учётом дальнейшего её расширения.

# Наша база данных состоит из 4 основных таблиц, касающихся нашей предметной области

# Orders – таблица заказов

# Storage – таблица учета товаров на складе

# Statement – таблица заказанных товаров

# Cataloque – таблица наименований товаров

# и двух вспомогательных таблиц необходимых для решения наших задач

# NewStorage – вспомогательная таблица хранящая адреса складских ячеек

# Users – таблица содержащая логины и пароли

## Средства проектирования

База данных спроектирована в среде Microsoft SQL Server Management Studio 17. Для работы приложения на компьютере пользователя должны быть запущены службы SQL Server.

Перенос структуры БД на другую платформу не составит труда, однако для удобства демонстрации работы приложения было решено использовать локальную БД.

Проектирование базы данных MyStorage производилось в Case-системе ErWin 4.0

ERwin - средство разработки структуры реляционных базы данных и ее описания на языке целевой СУБД (ORACLE, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server, Progress и др.).

**УРОВНИ МОДЕЛЕЙ ERWIN**

Возможны две точки зрения на информационную модель данных и, соответственно, два уровня модели.

***Первый - логический уровень*** (точка зрения пользователя) означает прямое отображение фактов из реальной жизни. Термин "логический уровень" в ERwin соответствует концептуальной (или инфологической) модели.

***Второй – физический, на котором рассматривается использование конкретной СУБД (даталогический уровень)***, определяются типы данных (например, целое или вещественное число), индексы для таблиц и др.

ERwin предоставляет возможности создавать и управлять этими двумя различными уровнями представления одной диаграммы (модели), равно как и иметь много вариантов отображения на каждом уровне.

Этапы построения информационной модели данных:

1. Построение инфологической модели данных

* *определение сущностей;*
* *определение зависимостей между сущностями;*
* *определение атрибутов сущностей.*

1. Построение даталогической модели данных

* *переход к физическому описанию модели: назначение соответствий имя сущности - имя таблицы, атрибут сущности - атрибут таблицы;*
* *задание первичных и вторичных ключей;*
* *приведение модели к требуемому уровню нормальной формы.*

1. Создание схемы базы данных

* *задание типов данных колонок таблиц в соответствии с типами целевой СУБД;*
* *задание ограничений триггеров, процедур;*
* *генерация базы данных.*

Ознакомившись с результатами бизнес-анализа предметной области, были выявлены информационные сущности, представленные в следующей таблице.

|  |  |
| --- | --- |
| Заказы | Описывает заказы, сделанные клиентами по покупке товаров . Атрибутами её являются номер заказа (внутренний), дата заказа, дата отгрузки, номер накладной (заполняется при отгрузке товара после его оплаты), состояние заказа (оформлен, отгружен). |
| Склад товаров | Описывает товары, находящиеся на складе и предлагаемые на продажу. Атрибутами ее являются наименование и спецификация товара, количество и цена, а также остаток на складе. |

По результатам бизнес - анализа выявлены отношения между сущностями, представленные в следующей таблице.

|  |  |
| --- | --- |
| Заказы-Склад Товаров | Каждый заказ состоит из многих товаров, поставленных на склад для продажи, каждый товар со склада может входить в различные заказы. |

Описание инфологической модели данных.

У нас есть две сущности Orders(Заказы) и Storage(Склад)

# Между этими двумя сущностями существует идентифицирующая связь (сущности зависимы).

# 

# Каждый заказ содержит товар/Товар может быть включен в Заказ

Опишем атрибуты сущностей.



Сущность Orders(Заказы) включает атрибуты

* OrderID – ID нашего заказа
* NoWayBill – номер накладной
* OrderDate – Дата заказа
* DispatchDate – дата отправки
* Condition – состояние заказа

Сущность Storage(Склад) включает

* ItemName – наименование товара
* Specification - спецификация
* Balance – остаток на складе
* Adress – адрес хранения
* ItemCost – отпускная цена

Описание даталогической модели данных.

Зададим первичные и вторичные ключи.

Так в таблице Storage(Склад) введем новое поле

ItemID которое будет нашем первичным ключом.

В таблице Orders(Заказы) у нас уже есть первичный ключ OrderID, соответственно там будет находится вторичный ключ ItemID тем самым создавая связь между двумя таблицами.

Заменим связи «многие ко многим» создав вспомогательную таблицу Statement включающая в себя в качестве вторичных ключей, первичные ключи из двух связываемых таблиц.

Так же создадим вспомогательную таблицу Cataloque, включающую в себя поля ItemName( Наименование) и ItemCode(Код Товара), где ItemCode будет первичным ключом в данной таблице и вторичным ключом в таблице Storage. Связи между этими таблицами будут не идентифицированы, так как сущности независимы.



К каждой таблице зададим все параметры полей в соответствии с типами целевой СУБД



Хранимые процедуры являются еще одной формой выполнения запросов к базе данных. Но по сравнению с ранее рассмотренными запросами, которые посылаются из приложения базе данных, хранимые процедуры определяются на сервере и предоставляют большую производительность и являются более безопасными.

Объект SqlCommand имеет встроенную поддержку хранимых процедур. В частности у него определено свойство CommandType, которое в качестве значения принимает значение из перечисления System.Data.CommandType. И значение System.Data.CommandType.StoredProcedure как раз указывает, что будет использоваться хранимая процедура.

Но чтобы использовать хранимые процедуры, нам надо их вначале создать.  
В нашем приложении мы использовали 17 хранимых процедур.

Диаграмма информационной модели представлена на рисунке ниже



***5.Описание алгоритмов реализующих бизнес логику***

Описание алгоритма работы менеджера сделанная в Rational Rose

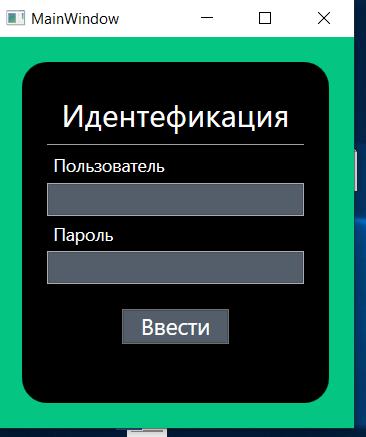


И описание алгоритма работы кладовщика сделанная в Rational Rose



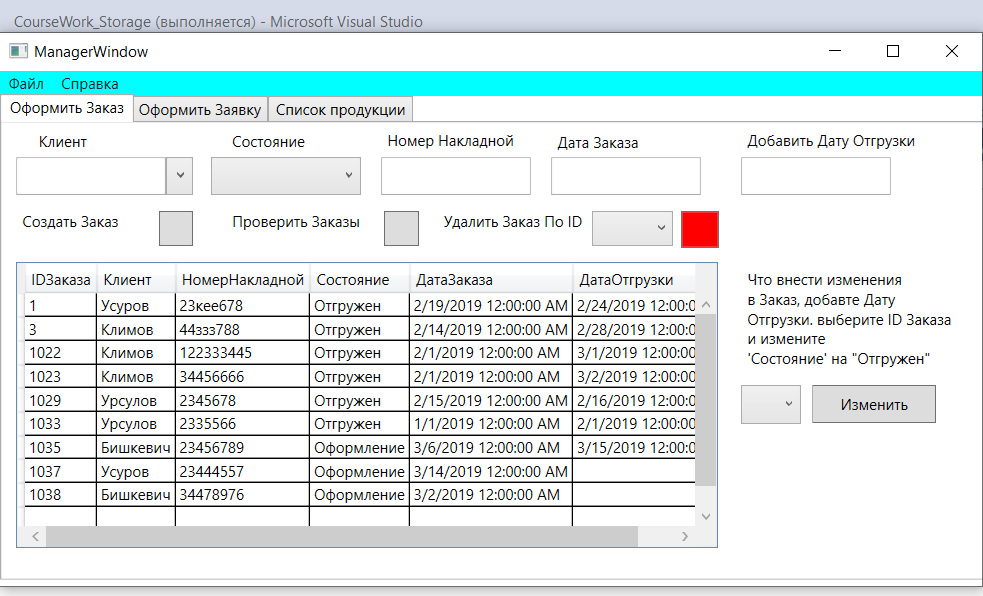
***6.Руководство пользователя***

После того как пользователь запустит наше приложение появляется главное окно, требующее ввести логин и пароль.



Введя Логин и пароль открывается соответствующее окно.

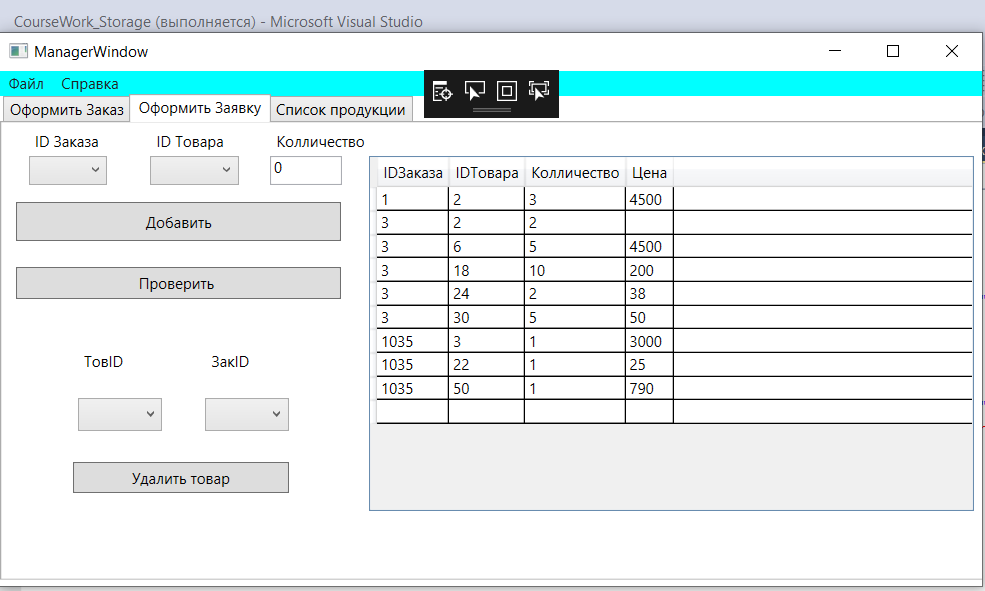
Окно менеджера выглядит так:



Для создания заказа необходимо ввести Имя клиента либо выбрать его в выпадающем списке, выбрать состояние «Оформление» ввести номер накладной и Дату заказа. Затем нажать на кнопку «Создать заказ». Что бы проверить правильность созданного заказа необходимо нажать на кнопку «Проверить Заказы». В отгруженные заказы необходимо вносить некоторые изменения. Нужно выбрать номер ID Заказа изменить Состояние заказа на «Отгружен» и ввести дату отгрузки. Затем нажать на кнопку «Изменить». Данные будут занесены в таблицу Orders в БД.

Если необходимо удалить заказ, нужно выбрать его в выпадающем списке, который будет выбрасывать только те заказы, которые находятся в статусе «Оформление» и затем нажать на кнопку «Удалить заказ по ID»

Добавление товара в заявку происходит по следующим этапам:

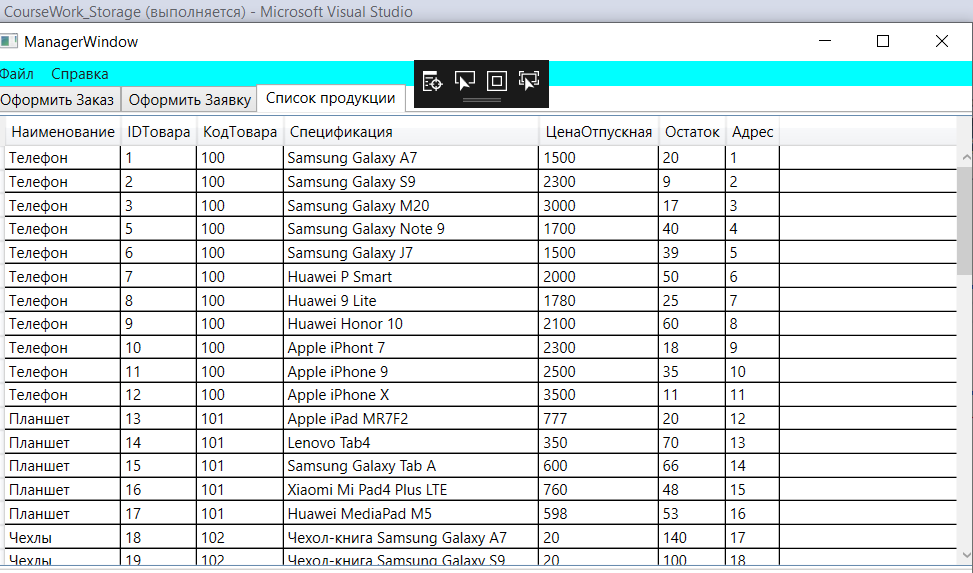


Выбор ID Заказа, ID Товара, Количества товара и затем нажимаем кнопку «Добавить».

Товар мы выбираем из списка продукции выбрав закладку Список продукции, нажав в панеле меню Файл ->Вывод продукции.

Затем мы можем проверить товар, который находится в заявках нажав на кнопку «Проверить»

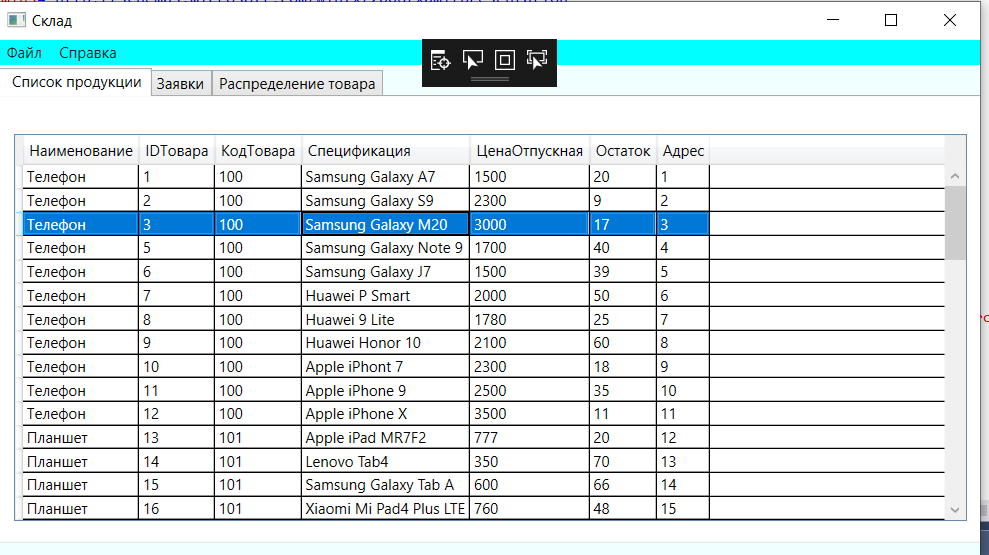
Если нам нужно удалить товар из заявки, выбираем ID товара и ID заказа в выпадающих списках и нажимаем кнопку «Удалить товар»



Каталог продукции

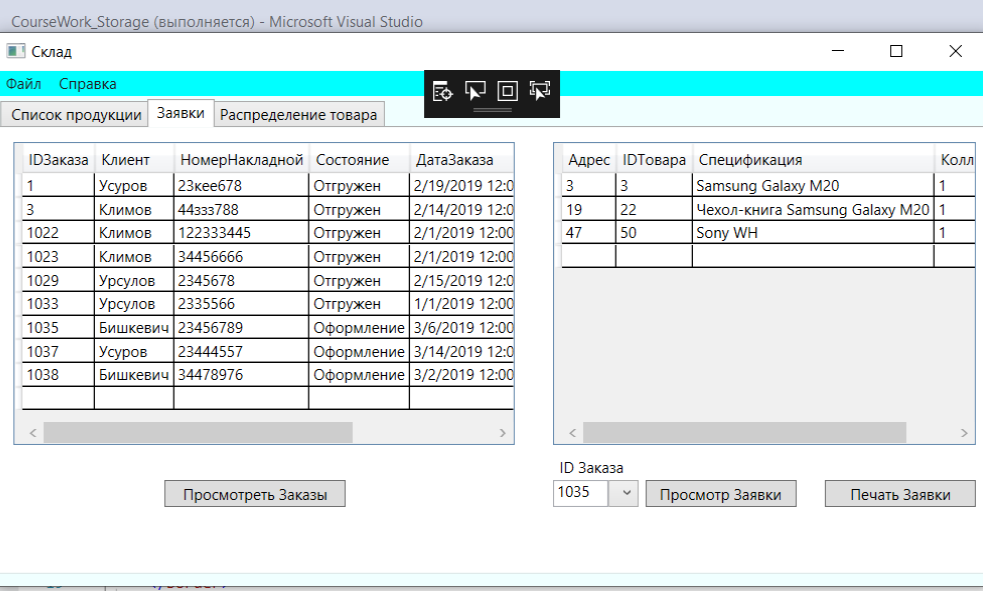
Обновление выпадающих и списков и выход из программы осуществляется через Файл->Обновить и Файл->Выход соответственно.

Окно кладовщика имеет такой вид:



Список продукции выбирается через Файл->Каталог продукции.

Далее для того, что бы распечатать заявки, необходимо выбрать графу Заявки и нажать на кнопку «Просмотреть Заказы». Затем выбрать ID Заказа и нажать кнопку «Просмотр Заявки». Перед нами появится список их ID заказанных товаров их адресов хранения и количества в заказе. Печать осуществляется через кнопку «Печать Заявки».

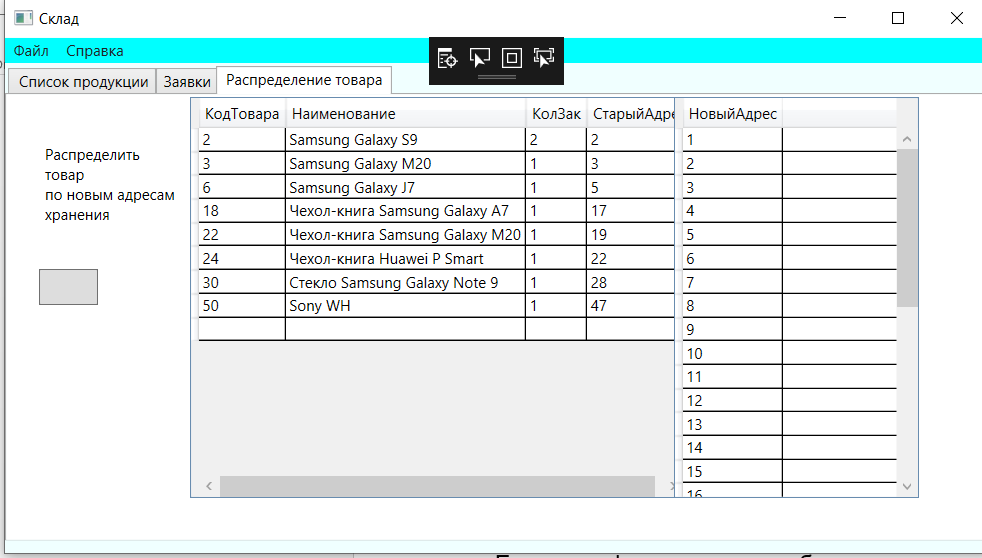


Еще одна функция которая будет полезна кладовщику, это распределение товара по горячим и холодным зонам хранения, в зависимости от частоты попадания в заказы.

Для этого необходимо выбрать графу «Распределение товара»

И нажать кнопку «Распределить товар по новым адресам хранения»

Там мы увидем старые адреса таблицу с товарами со старыми адресами хранения, и новую таблицу с новыми адресами хранения товара.



***7. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ И ОЦЕНКА ВЫПОЛНЕНИЯ ЗАДАЧ***

Тестированию были подвергнуты основные функции приложения, которыми пользователь будет пользоваться постоянно. Результаты тестирования говорят о том, что приложение работает в соответствии с требованиями. Список тестовых случаев (тест план) и результаты их выполнения представлены ниже:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Описание теста | Ожидаемый результат | Статус |
| ТС\_001 | Проверка запуска приложения | Открывается  Главное Окно | Выполнено успешно |
| ТС\_002 | Проверка ввода логина и пароля  1.Вводим логин/пароль менеджера(Manager\123)  2.Вводим логин\пароль кладовщика(Storer\qwerty) | 1.Открывается окно менеджера  2.Открывается окно кладовщика | Выполнено успешно |
| ТС\_003 | Создание заказа:  1.Заполняем поля: «Клиенты», «Состояние»,  «Номер Накладной», «Дата Заказа»  2.Нажимаем кнопку создать заказ. | 1.Поле «Состояние» должно выдать два значения «Оформление» и «Отгружен».  Выбираем «Оформление»  В поле «Клиенты» должен выпадать список раннее веденных клиентов  2.Данные должны быть занесены в таблицу Ordes | Выполнено успешно |
| ТС\_004 | Удаление заказа который создали и находи в состоянии «Оформления»  1.Нажимаем кнопку Файл->Обновить  2.Выбираем в комбобоксе ID Заказа  3.Меняем состояние заказа «Оформление» на «Отгружен»  4.Заполняем поле «Дата Загрузки»  5.Нажимаем кнопку изменить  6.Удаление заказа, который находится в состоянии «Отгружен» | 1.В комбобоксе «Изменить» после нажатия кнопки Обновления должен обновиться список ID заказов.  2.После нажатия кнопки «Изменить» должны быть внесены изменения в таблицу Orders  3. Заказ, который находится в состоянии «Отгружен» не должен удалиться | Выполнено успешно |
| ТС\_005 | Проверка заказов  Нажимаем на кнопку «Проверить заказы» что бы увидеть старые и вновь созданные заказы. | Должен появится весь список заказов из таблицы Orders | Выполнено успешно |
| TC\_006 | 1.Вывести список продукции:  Заходим в раздел «Список продукции», нажимаем кнопку Файл->Вывод продукции | Должен вывестись список продукции с:  Наименованием,ID товара, Код товара, Спецификация, Отпускная цена, Остаток, Адрес | Выполнено успешно |
| ТС\_007 | Оформление заявки:  1.Выбираем ID заказа  2.Выбираем ID товара  3. Заполняем количество товара  4.Нажимаем кнопку «Добавить» | 1.В выпадающем списке «ID заказа»  Должны выпасть id тех заказов, которые находятся в состоянии «Оформления»  2. В выпадающем списке «ID товара» выпадают id того товара, который имеется на складе.  3.После нажатия «Добавить» данные должны сохранится в таблице Statement | Выполнено успешно |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Описание теста | Ожидаемый результат | Статус |
| ТС\_008 | 1.Проверяем добавленные товары в заказе  2.Проверяем остаток товара на складе после добавление в заказ | 1.Данные из таблицы Statement  2. Остаток товара на складе должен уменьшится на то количество которое указано в заказе | Выполнено успешно |
| ТС\_009 | Удаление товара из заказа:  1.Выбираем ID Заказа  2.Выбираем ID товара  3.Нажимаем кнопку «Удалить товар» | 1.В выпадающем списке ID Заказов выпадают id тех заказов, которые находятся в состоянии «Оформление»  2.В выпадающем списке ID Товаров выпадают только те id которые находятся в заказах  3.Данные сохраняются в таблице Statement | Выполнено успешно |
| ТС\_10 | Выход из приложения  Нажать кнопку  Файл->Выход | Приложение должно завершить работу | Выполнено успешно |
| TC\_11 | Просмотр заказов  Нажимаем на кнопку «Просмотреть Заказы» | Вывод списка заказов из таблицы Orders | Выполнено успешно |
| ТС\_12 | Просмотреть заявки:  1.Ввести ID Заказа  2.Нажать кнопку «Просмотр Заявки» | 1.В выпадающем списке ID Заказа должны выпасть те заказы, которые находятся в состоянии «Оформления»  2.После нажатия кнопки «Просмотр Заявки» должен выпасть список содержащий ID товара, адрес, спецификация и количество, того заказа, который был выбран | Выполнено успешно |
| ТС\_13 | Распределение товара по горячим и холодным зонам хранения  Нажать на кнопку «Распределить товар по новым адресам хранения» | Товар меняет свое место хранения. На экран выводятся две таблицы со старыми и новыми местами хранения | Выполнено успешно |
|  |  |  |  |

***ЗАКЛЮЧЕНИЕ***

В рамках курсового проекта был разработан программный продукт, представляющий собой приложение " Склад готовой продукции ", предназначенная для учета товара на складе, автоматизации работы менеджеров и кладовщиков.

В ходе курсового проекта было пройдено несколько этапов, на каждом из которых получены результаты:

* Была рассмотрена предметная область и выделены основные сущности.
* Были сформулированы функциональные требования в рамках диаграммы вариантов использования на языке UML. Это позволило перед началом разработки четко задать разграничение доступных действий для пользователей с различными уровнями доступа к данным.

Проведено проектирование базы данных припомощи приложения ErWin и Microsoft SQL Server Management Studio.

Проведена разработка тест-плана основных функций веб-приложения.

В результате проделанной работы, я получил опыт создания веб-приложений на основе технологии .NET (C#) Framework.

***СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ***

1. Сайт MSDN о Visual C# [Электронный ресурс] URL: <http://msdn.microsoft.com/ru-ru/vcsharp/default.aspx>

2. Сайт ASP.NET [Электронный ресурс] URL: <http://www.asp.net/mvc/mvc3>

3. Магдануров Г. И., Юнев В. А. ASP.NET MVC Framework . – СПб.: БХВ-Петербург, 2010

4. «Базы данных» Карпова, Издательство «Питер», 2003 год.

5. «Разработка программных проектов на основе RUP» Гари Поллис, Лиз Огастин, Крис Лоу, Джас Мадхар, Издательство «Бином», 2005 год.

6. Интернет сайты http://www.globalcio.ru/ и https://geekbrains.ru/.

7. Электронный конспект лекций ИИТ БГУИР по предмету СВПП, ТППОИС, ООП, ОПБД, ТПО.

***Приложение***

**ManagerWindow c#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Data.SqlClient;

using System.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Text.RegularExpressions;

using System.Collections.ObjectModel;

namespace CourseWork\_Storage

{

/// <summary>

/// Interaction logic for ManagerWindow.xaml

/// </summary>

///

public class Order

{

private string Client;

private string NoWayBill;

private string Condition;

public string condition

{

get

{

return Condition;

}

set

{

string c = value;

if (c.Length == 0)

{

throw new ArgumentException("Поле должно быть заполнено");

}

else

{

Condition = value;

}

}

}

public string nowaybill

{

get

{

return NoWayBill;

}

set

{

string w = value;

if (w.Length >= 10)

throw new ArgumentException("Номер накладной не может иметь больше 10 символов");

else

{

NoWayBill = value;

}

}

}

private string OrderDate;

public string client

{

get

{

return Client;

}

set

{

Regex r = new Regex(@"[0-9]");

string t = value;

if (r.IsMatch(t))

{

throw new ArgumentException("Фамилия не должна содержать цифры");

}

else

{

Client = value;

}

}

}

public string orderdate

{

get

{

return OrderDate;

}

set

{

string d = value;

if (d.Length == 0)

{

throw new ArgumentException("Поле должно быть заполнено");

}

else

{

OrderDate = value;

}

}

}

}

public class Statement

{

private double Quantity;

public double quantity

{

get

{

return Quantity;

}

set

{

if (value <= 0)

{

throw new ArgumentException("Значение должно быть положительным!");

}

else

{

Quantity = value;

}

}

}

}

public partial class ManagerWindow : Window

{

Order ord;

Statement stm;

public ManagerWindow()

{

InitializeComponent();

ord = new Order();

stm = new Statement();

tbClient.DataContext = ord;

tbWayBill.DataContext = ord;

tbDateOrder.DataContext = ord;

cbCondition.DataContext = ord;

cbClient.DataContext = ord;

tbQuantity.DataContext = stm;

string[] st = { "Оформление", "Отгружен" };

foreach(string a in st)

{

cbCondition.Items.Add(a);

}

FillcbClient();

FillcbOrderID();

FillcbItemID();

FillcbTovID();

FillcbZakID();

FillcbDelZakID();

FillcbChangeZakID();

}

SqlConnection sqlCon = new SqlConnection(@"Data Source=DESKTOP-H6UBJSJ; Initial Catalog=MyStorage; Integrated Security=True;");

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("CreateOrder", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

sqlcmd.Parameters.AddWithValue("@Client", SqlDbType.VarChar).Value = tbClient.Text.Trim();

sqlcmd.Parameters.AddWithValue("@NoWayBill", SqlDbType.VarChar).Value = tbWayBill.Text.Trim();

sqlcmd.Parameters.AddWithValue("@OrderDate", SqlDbType.Date).Value = tbDateOrder.Text.Trim();

sqlcmd.Parameters.AddWithValue("@Condition", SqlDbType.Date).Value = cbCondition.Text.Trim();

sqlcmd.ExecuteNonQuery();

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand cmd = new SqlCommand("DataOrders", sqlCon);

cmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(cmd);

DataTable dt = new DataTable("Orders");

dataAdp.Fill(dt);

DataGrid2.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

finally

{

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("DeleteOrder", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

sqlcmd.Parameters.AddWithValue("@OrderID", SqlDbType.Int).Value = cbDelZakID.Text.Trim();

sqlcmd.ExecuteNonQuery();

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand cmd = new SqlCommand("DataStatement", sqlCon);

cmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(cmd);

DataTable dt = new DataTable("Statement");

dataAdp.Fill(dt);

DataGrid3.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("CreateStatement", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

sqlcmd.Parameters.AddWithValue("@OrderID", SqlDbType.Int).Value = cbOrderID.Text.Trim();

sqlcmd.Parameters.AddWithValue("@ItemID", SqlDbType.Int).Value = cbItemID.Text.Trim();

sqlcmd.Parameters.AddWithValue("@Quantity", SqlDbType.Int).Value = tbQuantity.Text.Trim();

sqlcmd.ExecuteNonQuery();

sqlCon.Close();

cbTovID.Items.Add(cbItemID.Text);

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

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

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=DESKTOP-H6UBJSJ; Initial Catalog=MyStorage; Integrated Security=True;");

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("DataStorage", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(sqlcmd);

DataTable dt = new DataTable("Storage");

dataAdp.Fill(dt);

DataGrid4.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("DElItemID", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

sqlcmd.Parameters.AddWithValue("@OrderID", SqlDbType.Int).Value = cbZakID.Text.Trim();

sqlcmd.Parameters.AddWithValue("@ItemID", SqlDbType.Int).Value = cbTovID.Text.Trim();

sqlcmd.ExecuteNonQuery();

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

private void Exit\_Button(object sender, RoutedEventArgs e)

{

System.Diagnostics.Process.GetCurrentProcess().Kill();

this.Close();

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("ChangeStatement", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

sqlcmd.Parameters.AddWithValue("@OrderID", SqlDbType.Int).Value = cbChangeZakID.Text.Trim();

sqlcmd.Parameters.AddWithValue("@DispatchDate", SqlDbType.Int).Value = tbDispatchDate.Text.Trim();

sqlcmd.Parameters.AddWithValue("@Condition", SqlDbType.Int).Value = cbCondition.Text.Trim();

sqlcmd.ExecuteNonQuery();

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

private void FillcbClient()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectClient", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while(drd.Read())

{

cbClient.Items.Add(drd["Client"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void FillcbOrderID()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectOrderID", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbOrderID.Items.Add(drd["OrderID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void FillcbItemID()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectItemID", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbItemID.Items.Add(drd["ItemID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void FillcbTovID()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectItemForDel", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbTovID.Items.Add(drd["ItemID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void FillcbZakID()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectOrderForDel", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbZakID.Items.Add(drd["OrderID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void FillcbDelZakID()

{

cbDelZakID.Items.Clear();

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectZakIDForDel", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter da = new SqlDataAdapter(sqlcmd);

da.UpdateCommand = sqlcmd;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbDelZakID.Items.Add(drd["OrderID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void FillcbChangeZakID()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("cbChangeZakID", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbChangeZakID.Items.Add(drd["OrderID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void CbCondition\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbCondition.Text = cbCondition.SelectedItem.ToString();

}

private void CbClient\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

tbClient.Text = cbClient.SelectedItem.ToString();

}

private void CbOrderID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbOrderID.Text = cbOrderID.SelectedItem.ToString();

}

private void CbItemID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbItemID.Text = cbItemID.SelectedItem.ToString();

}

private void CbTovID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbTovID.Text = cbTovID.SelectedItem.ToString();

}

private void CbZakID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbZakID.Text = cbZakID.SelectedItem.ToString();

}

private void CbChangeZakID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbChangeZakID.Text = cbChangeZakID.SelectedItem.ToString();

}

private void cbDelZakID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cbDelZakID.Text = cbDelZakID.SelectedItem.ToString();

}

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

{

cbClient.Items.Add(tbClient.Text);

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlCmd = new SqlCommand("SELECT OrderID FROM Orders ", sqlCon);

sqlCmd.CommandType = CommandType.Text;

SqlDataReader drd = sqlCmd.ExecuteReader();

while (drd.Read())

{

cbOrderID.Items.Add(drd["OrderID"].ToString());

cbDelZakID.Items.Add(drd["OrderID"].ToString());

cbZakID.Items.Add(drd["OrderID"].ToString());

cbChangeZakID.Items.Add(drd["OrderID"].ToString());

}

sqlCon.Close();

}

}

}

**StorerWindow C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Data.SqlClient;

using System.Data;

using System.IO;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace CourseWork\_Storage

{

/// <summary>

/// Interaction logic for StorerWindow.xaml

/// </summary>

public partial class StorerWindow : Window

{

public StorerWindow()

{

InitializeComponent();

FillcbOrderID2();

}

SqlConnection sqlCon = new SqlConnection(@"Data Source=DESKTOP-H6UBJSJ; Initial Catalog=MyStorage; Integrated Security=True;");

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("DataStorage", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(sqlcmd);

DataTable dt = new DataTable("Storage");

dataAdp.Fill(dt);

DataGrid.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("DataStorage", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(sqlcmd);

DataTable dt = new DataTable("Storage");

dataAdp.Fill(dt);

DataGrid.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

this.Close();

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand cmd = new SqlCommand("DataOrders", sqlCon);

cmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(cmd);

DataTable dt = new DataTable("Orders");

dataAdp.Fill(dt);

DataGrid5.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("Statement\_Storage", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

sqlcmd.Parameters.AddWithValue("@OrderID", SqlDbType.VarChar).Value = tbStatementID.Text.Trim();

SqlDataAdapter dataAdp = new SqlDataAdapter(sqlcmd);

DataTable dt = new DataTable("Storage");

dataAdp.Fill(dt);

DataGrid6.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

private void FillcbOrderID2()

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand sqlcmd = new SqlCommand("SelectOrderID", sqlCon);

sqlcmd.CommandType = CommandType.StoredProcedure;

SqlDataReader drd = sqlcmd.ExecuteReader();

while (drd.Read())

{

cbStatementID.Items.Add(drd["OrderID"].ToString());

}

sqlCon.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void CbStatementID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

tbStatementID.Text = cbStatementID.SelectedItem.ToString();

}

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

{

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand cmd = new SqlCommand("Raspredelenie", sqlCon);

cmd.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

dataAdp.Fill(dt);

DataGrid10.ItemsSource = dt.DefaultView;

dataAdp.Update(dt);

SqlCommand cmd2 = new SqlCommand("Raspredelenie2", sqlCon);

cmd2.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdp2 = new SqlDataAdapter(cmd2);

DataTable dt2 = new DataTable();

dataAdp2.Fill(dt2);

DataGrid11.ItemsSource = dt2.DefaultView;

dataAdp.Update(dt2);

sqlCon.Close();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

}

}

**MainWindow C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Data.SqlClient;

using System.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace CourseWork\_Storage

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=DESKTOP-H6UBJSJ;Initial Catalog=MyStorage; Integrated Security=True;");

try

{

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

string query = "SELECT COUNT(1) FROM Users WHERE UserName=@UserName AND UserPassword=@UserPassword";

SqlCommand sqlCmd = new SqlCommand(query, sqlCon);

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Parameters.AddWithValue("@UserName", txtUsername.Text);

sqlCmd.Parameters.AddWithValue("@UserPassword", txtPassword.Password);

int count = Convert.ToInt32(sqlCmd.ExecuteScalar());

if(count==1)

{

ManagerWindow dashbord = new ManagerWindow();

StorerWindow dashbord2 = new StorerWindow();

AdminWindow dashbord3 = new AdminWindow();

switch(txtUsername.Text)

{

case "Manager":

dashbord.Show();

break;

case "Storer":

dashbord2.Show();

break;

case "Admin":

dashbord3.Show();

break;

}

this.Close();

}

else

{

MessageBox.Show("Имя пользователя и пороль не коректны");

}

sqlCon.Close();

}

catch(Exception)

{

}

finally

{

}

}

}

}

**ManagerWindow XAML**

<Window x:Class="CourseWork\_Storage.ManagerWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CourseWork\_Storage"

mc:Ignorable="d"

Title="ManagerWindow" Height="450" Width="800">

<Window.Resources>

<Style TargetType="{x:Type TextBox}">

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="true">

<Setter Property="ToolTip" Value="{Binding RelativeSource={x:Static RelativeSource.Self}, Path=(Validation.Errors)[0].ErrorContent }">

</Setter>

<Setter Property="Background" Value="Pink"></Setter>

</Trigger>

</Style.Triggers>

</Style>

<ControlTemplate x:Key="ErrorTemp">

<StackPanel Orientation="Horizontal">

<Border BorderThickness="2" BorderBrush="Red">

<AdornedElementPlaceholder></AdornedElementPlaceholder>

</Border>

<TextBlock Foreground="Red" FontSize="30">!</TextBlock>

</StackPanel>

</ControlTemplate>

</Window.Resources>

<Grid>

<Menu Height="20" VerticalAlignment="Top" Margin="0,1,0,0" Background="Aqua">

<MenuItem Header="Файл">

<StackPanel>

<Button x:Name="ProductList" Height="20" Width="115" Content="Вывод продукции" Margin="-30,0,0,0" Click="btSelectStorage\_Click"/>

</StackPanel>

<StackPanel>

<Button x:Name="Refresh" Height="20" Width="115" Content="Обновить" Margin="-30,0,0,0" Click="Refresh\_Click" />

</StackPanel>

<StackPanel>

<Button x:Name="Exit" Content="Выход" Height="20" Width="115" Margin="-30,0,0,0" Click="Exit\_Button"/>

</StackPanel>

</MenuItem>

<MenuItem Header="Справка"/>

</Menu>

<TabControl HorizontalAlignment="Left" Height="389" VerticalAlignment="Bottom" Width="792" Margin="0,0,0,5">

<TabControl.Background>

<ImageBrush></ImageBrush>

</TabControl.Background>

<TabItem Header="Оформить Заказ">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="132\*"/>

<ColumnDefinition Width="7\*"/>

<ColumnDefinition Width="29\*"/>

<ColumnDefinition Width="22\*"/>

<ColumnDefinition Width="597\*"/>

</Grid.ColumnDefinitions>

<Label Content="Клиент" HorizontalAlignment="Left" VerticalAlignment="Top" RenderTransformOrigin="1.211,0.577" Margin="23,0,0,0" Width="100" />

<TextBox x:Name="tbClient" TextWrapping="Wrap" Text="{Binding Path=client, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" Margin="10,26,2,304.8" Height="30" Width="120" />

<Label Content="Номер Накладной" HorizontalAlignment="Left" Margin="112.2,-1,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.105,-0.385" Height="25" Width="120" Grid.Column="4"/>

<TextBox x:Name="tbWayBill" TextWrapping="Wrap" Text="{Binding Path=nowaybill, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" Height="30" Width="120" Margin="112.2,26,364.4,304.8" Grid.Column="4"/>

<Label Content="Дата Заказа" Width="100" Margin="248.2,1,248.4,0" RenderTransformOrigin="0.68,0.6" Height="25" VerticalAlignment="Top" Grid.Column="4"/>

<TextBox x:Name="tbDateOrder" Height="30" TextWrapping="Wrap" Text="{Binding Path=orderdate, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" Width="120" Margin="248.2,26,228.4,304.8" Grid.Column="4"/>

<TextBox x:Name="tbDispatchDate" Height="30" TextWrapping="Wrap" Text="" Width="120" Margin="400.2,26,76.4,304.8" Grid.Column="4"/>

<DataGrid x:Name="DataGrid2" HorizontalAlignment="Left" Height="229" Margin="10,110,0,0" VerticalAlignment="Top" Width="561" Grid.ColumnSpan="5"/>

<Button x:Name="btCreateOrder" Content="" HorizontalAlignment="Left" Margin="124,69,0,0" VerticalAlignment="Top" Width="27" Height="28" FontWeight="Bold" Foreground="#FF1E43E8" FontSize="14" Click="CreateOrder\_Click" RenderTransformOrigin="-0.77,0.5" Grid.ColumnSpan="3"/>

<Button x:Name="btDelOrder" Content="" Background="Red" HorizontalAlignment="Left" Margin="352.2,69,0,0" VerticalAlignment="Top" Width="30" Height="30" Foreground="#FFF30A0A" FontWeight="Bold" FontSize="14" Click="DeleteOrder\_Click" Grid.Column="4"/>

<Button x:Name="CheckOrder" Content="" HorizontalAlignment="Left" Margin="114.2,69,0,0" VerticalAlignment="Top" Width="28" Height="28" RenderTransformOrigin="0.555,0.55" Click="CheckOrder\_Click" Grid.Column="4"/>

<Label Content="Состояние" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.411,0.195" Width="73" Grid.Column="3" Grid.ColumnSpan="2"/>

<Button x:Name="btChange" Content="Изменить" HorizontalAlignment="Left" Margin="457.2,208,0,0" VerticalAlignment="Top" Width="99" Height="31" Click="ChangeStatement\_Click" Grid.Column="4" RenderTransformOrigin="1,0.45"/>

<Canvas HorizontalAlignment="Left" Height="14" Margin="19.2,72,0,0" VerticalAlignment="Top" Width="42" Grid.Column="4"/>

<ComboBox x:Name="cbCondition" HorizontalAlignment="Left" Margin="26.8,26,0,0" VerticalAlignment="Top" Width="120" Height="30" SelectionChanged="CbCondition\_SelectionChanged" Grid.ColumnSpan="3" Grid.Column="2"/>

<ComboBox x:Name="cbClient" HorizontalAlignment="Left" Margin="130,26,0,0" VerticalAlignment="Top" Width="21" Height="30" SelectionChanged="CbClient\_SelectionChanged" Grid.ColumnSpan="3"/>

<Label Content="Создать Заказ" Width="100" Margin="10,64,22,263.8"/>

<Label Content="Проверить Заказы" Grid.Column="3" HorizontalAlignment="Left" Margin="10,64,0,0" VerticalAlignment="Top" Width="121" Height="33" Grid.ColumnSpan="2"/>

<Label Content="Удалить Заказ По ID" Grid.Column="4" Margin="157.2,64,315.4,263.8" RenderTransformOrigin="0.506,0.579"/>

<Label Content="Добавить Дату Отгрузки" Grid.Column="4" HorizontalAlignment="Left" Margin="400.2,-1,0,0" VerticalAlignment="Top" Width="156" RenderTransformOrigin="-0.008,0.43"/>

<Label Content="Что внести изменения&#xD;&#xA;в Заказ, добавте Дату&#xD;&#xA;Отгрузки. выберите ID Заказа &#xD;&#xA;и измените&#xD;&#xA;'Состояние' на &quot;Отгружен&quot;" Grid.Column="4" HorizontalAlignment="Left" Margin="400.2,110,0,0" VerticalAlignment="Top" Width="187" Height="111"/>

<ComboBox x:Name="cbDelZakID" Grid.Column="4" Margin="281.2,69,250.4,263.8" SelectionChanged="cbDelZakID\_SelectionChanged" ItemsSource="{Binding}"/>

<ComboBox x:Name="cbChangeZakID" Grid.Column="4" Margin="400.2,208,148.4,121.8" SelectionChanged="CbChangeZakID\_SelectionChanged"/>

</Grid>

</TabItem>

<TabItem Header="Оформить Заявку">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="49\*"/>

<ColumnDefinition Width="198\*"/>

<ColumnDefinition Width="539\*"/>

</Grid.ColumnDefinitions>

<Label Content="ID Заказа" HorizontalAlignment="Left" Margin="20,0,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.289,0.269" Height="25" Width="120" Grid.ColumnSpan="2"/>

<Label Content="ID Товара" HorizontalAlignment="Left" Margin="68.2,0,0,0" VerticalAlignment="Top" Height="25" Width="120" Grid.Column="1"/>

<Label Content="Колличество" HorizontalAlignment="Left" Margin="164.2,0,0,0" VerticalAlignment="Top" Height="25" Width="120" RenderTransformOrigin="0.867,0.72" Grid.ColumnSpan="2" Grid.Column="1"/>

<TextBox x:Name="tbQuantity" HorizontalAlignment="Left" Height="23" Margin="164.2,25,0,0" TextWrapping="Wrap" Text="{Binding Path=quantity, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" VerticalAlignment="Top" Width="57" Grid.Column="1" Grid.ColumnSpan="2"/>

<DataGrid x:Name="DataGrid3" HorizontalAlignment="Left" Height="284" Margin="44.8,25,0,0" VerticalAlignment="Top" Width="484" Grid.Column="2"/>

<Button x:Name="btAdd" Content="Добавить" HorizontalAlignment="Left" Margin="10,62,0,0" VerticalAlignment="Top" Width="260" Height="31" Click="CreateStatement\_Click" Grid.ColumnSpan="3"/>

<Button x:Name="tbCheck" Content="Проверить " Margin="10,114,516.4,221.8" RenderTransformOrigin="0.227,2.194" Height="25" Click="CheckStatement\_Click" Grid.ColumnSpan="3"/>

<Button x:Name="btDelTov" Content="Удалить товар" HorizontalAlignment="Left" Margin="6.2,270,0,0" VerticalAlignment="Top" Width="173" Height="25" Click="DeleteItemID\_Click" Grid.Column="1"/>

<Label Content="ТовID" HorizontalAlignment="Left" Margin="10.2,176,0,0" VerticalAlignment="Top" Width="61" RenderTransformOrigin="1.108,0.273" Grid.Column="1"/>

<Label Content="ЗакID" HorizontalAlignment="Left" Margin="112.2,176,0,0" VerticalAlignment="Top" Width="48" Grid.Column="1"/>

<ComboBox x:Name="cbOrderID" Margin="20,25,164.2,312.8" SelectionChanged="CbOrderID\_SelectionChanged" RenderTransformOrigin="1.288,0.539" Grid.ColumnSpan="2"/>

<ComboBox x:Name="cbItemID" Margin="68.2,25,59.2,312.8" SelectionChanged="CbItemID\_SelectionChanged" Grid.Column="1" RenderTransformOrigin="2.065,0.27"/>

<ComboBox x:Name="cbTovID" Margin="10.2,219,120.2,115.8" SelectionChanged="CbTovID\_SelectionChanged" Grid.Column="1" RenderTransformOrigin="1.05,0.392"/>

<ComboBox x:Name="cbZakID" Grid.Column="1" Margin="112.2,219,19.2,115.8" SelectionChanged="CbZakID\_SelectionChanged"/>

</Grid>

</TabItem>

<TabItem Header="Список продукции">

<DataGrid x:Name="DataGrid4" Margin="2,0,-2,0"/>

</TabItem>

**StorerWindow XAML**

<Window x:Class="CourseWork\_Storage.StorerWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CourseWork\_Storage"

mc:Ignorable="d"

Title="Склад" Height="450" Width="800" Background="Azure">

<Grid>

<Menu Height="20" VerticalAlignment="Top" Margin="0,1,0,0" Background="Aqua">

<MenuItem Header="Файл">

<StackPanel>

<Button x:Name="ProductList" Height="20" Width="115" Content="Вывод продукции" Margin="-30,0,0,0" Click="Button\_ProductList"/>

</StackPanel>

<StackPanel>

<Button x:Name="CatalogList" Height="20" Width="115" Content="Каталог продукции" Margin="-30,0,0,0" Click="Button\_CatalogList" />

</StackPanel>

<StackPanel>

<Button x:Name="Exit" Content="Выход" Height="20" Width="115" Margin="-30,0,0,0" Click="Button\_Exit"/>

</StackPanel>

</MenuItem>

<MenuItem Header="Справка"/>

</Menu>

<TabControl HorizontalAlignment="Left" Height="380" Margin="0,0,0,10" VerticalAlignment="Bottom" Width="792">

<TabItem Header=" Список продукции">

<Grid Background="White" Margin="0,18,0,0">

<DataGrid x:Name="DataGrid" AutoGenerateColumns="True" HorizontalAlignment="Left" Height="310" Margin="10,10,0,0" VerticalAlignment="Top" Width="762"/>

</Grid>

</TabItem>

<TabItem Header="Заявки">

<Grid Background="White">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="28\*"/>

<ColumnDefinition Width="168\*"/>

<ColumnDefinition Width="6\*"/>

<ColumnDefinition Width="51\*"/>

<ColumnDefinition Width="39\*"/>

<ColumnDefinition Width="494\*"/>

</Grid.ColumnDefinitions>

<DataGrid x:Name="DataGrid5" HorizontalAlignment="Left" Height="242" VerticalAlignment="Top" Width="398" Margin="10,10,0,0" Grid.ColumnSpan="6"/>

<DataGrid x:Name="DataGrid6" HorizontalAlignment="Left" Height="242" Margin="146.8,10,0,0" VerticalAlignment="Top" Width="336" Grid.Column="5"/>

<Button Content="Просмотреть Заказы" HorizontalAlignment="Left" Margin="102,280,0,0" VerticalAlignment="Top" Width="144" Height="22" Grid.ColumnSpan="4" Click="OrderView\_Click" Grid.Column="1"/>

<TextBox x:Name="tbStatementID" Grid.Column="5" HorizontalAlignment="Left" Height="22" Margin="146.8,280,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="44" RenderTransformOrigin="0.493,0.896"/>

<Button Content="Просмотр Заявки" Grid.Column="5" HorizontalAlignment="Left" Margin="219.8,280,0,0" VerticalAlignment="Top" Width="120" Height="22" Click="StatementView\_Click"/>

<Label Content="ID Заказа" Grid.Column="5" HorizontalAlignment="Left" Margin="146.8,257,0,0" VerticalAlignment="Top" Width="65" Height="36"/>

<Button x:Name="btPrintStatement" Content="Печать Заявки" Grid.Column="5" HorizontalAlignment="Left" Margin="362.8,280,0,0" VerticalAlignment="Top" Width="120" Height="22"/>

<ComboBox x:Name="cbStatementID" Grid.Column="5" HorizontalAlignment="Left" Margin="190.8,280,0,0" VerticalAlignment="Top" Width="24" SelectionChanged="CbStatementID\_SelectionChanged"/>

</Grid>

</TabItem>

<TabItem Header="Распределение товара">

<Grid>

<DataGrid x:Name="DataGrid10" HorizontalAlignment="Left" Height="321" VerticalAlignment="Top" Width="396" Margin="145,0,0,0"/>

<DataGrid x:Name="DataGrid11" HorizontalAlignment="Left" Height="321" Margin="532,0,0,0" VerticalAlignment="Top" Width="196"/>

<Button x:Name="bt1" Content="" Margin="24,138,714.4,185.8" Click="Button\_Click"/>

<Label Content="Распределить&#xD;&#xA;товар&#xD;&#xA;по новым адресам&#xD;&#xA;хранения" Height="100" Margin="24,33,646.4,219.8"/>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Window>

**MainWindow XAML**

<Window x:Class="CourseWork\_Storage.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CourseWork\_Storage"

mc:Ignorable="d"

Title="MainWindow" Height="350" Width="300" FontSize="14" Background="#04c582">

<Border Background="Black" CornerRadius="20" Margin="20">

<StackPanel Margin="20">

<Label Content="Идентефикация" Foreground="White" FontSize="25" HorizontalAlignment="Center"/>

<Separator></Separator>

<Label Content="Пользователь" Foreground="White"/>

<TextBox Name="txtUsername" Background="#545d6a" Foreground="White" FontSize="18"/>

<Label Content="Пароль" Foreground="White"/>

<PasswordBox Name="txtPassword" Background="#545d6a" Foreground="White" FontSize="18"/>

<Button Name="btSubmit" Click="btSubmit\_Click" Content="Ввести" Margin="60 20" Background="#545d6a" Foreground="White" FontSize="18"/>

</StackPanel>

</Border>

</Window>

**Скрипт БД**

USE [master]

GO

/\*\*\*\*\*\* Object: Database [MyStorage] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

CREATE DATABASE [MyStorage]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'MyStorage', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\MyStorage.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'MyStorage\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\MyStorage\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [MyStorage] SET COMPATIBILITY\_LEVEL = 140

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [MyStorage].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [MyStorage] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [MyStorage] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [MyStorage] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [MyStorage] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [MyStorage] SET ARITHABORT OFF

GO

ALTER DATABASE [MyStorage] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [MyStorage] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [MyStorage] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [MyStorage] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [MyStorage] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [MyStorage] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [MyStorage] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [MyStorage] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [MyStorage] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [MyStorage] SET DISABLE\_BROKER

GO

ALTER DATABASE [MyStorage] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [MyStorage] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [MyStorage] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [MyStorage] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [MyStorage] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [MyStorage] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [MyStorage] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [MyStorage] SET RECOVERY SIMPLE

GO

ALTER DATABASE [MyStorage] SET MULTI\_USER

GO

ALTER DATABASE [MyStorage] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [MyStorage] SET DB\_CHAINING OFF

GO

ALTER DATABASE [MyStorage] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [MyStorage] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [MyStorage] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [MyStorage] SET QUERY\_STORE = OFF

GO

USE [MyStorage]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Cataloque] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Cataloque](

[ItemCode] [int] NOT NULL,

[ItemName] [varchar](40) NOT NULL,

CONSTRAINT [PK\_Cataloque] PRIMARY KEY CLUSTERED

(

[ItemCode] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[NewStorage] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[NewStorage](

[NewAdress] [int] NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Orders] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Orders](

[OrderID] [int] IDENTITY(1,1) NOT NULL,

[Client] [varchar](30) NOT NULL,

[NoWayBill] [varchar](10) NULL,

[OrderDate] [date] NOT NULL,

[DispatchDate] [date] NULL,

[Condition] [varchar](15) NULL,

CONSTRAINT [PK\_Orders] PRIMARY KEY CLUSTERED

(

[OrderID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Statement] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Statement](

[OrderID] [int] NOT NULL,

[ItemID] [int] NOT NULL,

[Quantity] [real] NOT NULL,

[Price] [real] NULL,

CONSTRAINT [PK\_Statement] PRIMARY KEY CLUSTERED

(

[OrderID] ASC,

[ItemID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Storage] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Storage](

[ItemID] [int] IDENTITY(1,1) NOT NULL,

[ItemCode] [int] NOT NULL,

[Specification] [varchar](50) NOT NULL,

[ItemCost] [real] NOT NULL,

[Balance] [int] NULL,

[Adress] [int] NULL,

CONSTRAINT [PK\_Storage] PRIMARY KEY CLUSTERED

(

[ItemID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Users](

[UserID] [int] IDENTITY(1,1) NOT NULL,

[UserName] [varchar](50) NOT NULL,

[UserPassword] [varchar](20) NOT NULL,

CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED

(

[UserID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Statement] WITH CHECK ADD CONSTRAINT [FK\_Statement\_Orders] FOREIGN KEY([OrderID])

REFERENCES [dbo].[Orders] ([OrderID])

GO

ALTER TABLE [dbo].[Statement] CHECK CONSTRAINT [FK\_Statement\_Orders]

GO

ALTER TABLE [dbo].[Statement] WITH CHECK ADD CONSTRAINT [FK\_Statement\_Storage] FOREIGN KEY([ItemID])

REFERENCES [dbo].[Storage] ([ItemID])

GO

ALTER TABLE [dbo].[Statement] CHECK CONSTRAINT [FK\_Statement\_Storage]

GO

ALTER TABLE [dbo].[Storage] WITH CHECK ADD CONSTRAINT [FK\_Storage\_Cataloque] FOREIGN KEY([ItemCode])

REFERENCES [dbo].[Cataloque] ([ItemCode])

GO

ALTER TABLE [dbo].[Storage] CHECK CONSTRAINT [FK\_Storage\_Cataloque]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[cbChangeZakID] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[cbChangeZakID]

as

select OrderID from Orders

where Condition='оформление'

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ChangeStatement] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[ChangeStatement]

@OrderID int,

@DispatchDate date,

@Condition varchar(20)

AS

Update Orders Set DispatchDate=@DispatchDate

Where OrderID=@OrderID and Condition='оформление'

Update Orders Set Condition=@Condition

where OrderID=@OrderID and Condition='оформление'

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CreateOrder] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[CreateOrder]

@Client varchar(30), @NoWayBill varchar(10), @OrderDate date, @Condition varchar(15)

As

insert into Orders

values( @Client, @NoWayBill, @OrderDate,Null, @Condition)

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CreateStatement] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[CreateStatement]

@OrderID int,

@ItemID int,

@Quantity real

As

Declare @Balance int

Declare @ItemCost real

Select @Balance=Balance, @ItemCost=ItemCost From Storage

where ItemID=@ItemID

If @Balance>=@Quantity

Begin

Insert into Statement

Values(@OrderID, @ItemID, @Quantity,null)

Update Statement Set Price=@Quantity\*@ItemCost

where ItemID=@ItemId

Update Storage SET Balance=Balance-@Quantity

where ItemID=@ItemID

Return 0

End

Else

Return 1

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[DataOrders] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[DataOrders]

As

Select OrderID as ID\_Заказа, Client AS Клиент, NoWayBill AS Номер\_Накладной, Condition as Состояние,

OrderDate as Дата\_Заказа, DispatchDate as Дата\_Отгрузки

From Orders

Return

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[DataStatement] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[DataStatement]

as

Begin

Select OrderId as ID\_Заказа, ItemID as ID\_Товара, Quantity as Колличество, Price as Цена from Statement

End

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[DataStorage] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[DataStorage]

As

begin

Select Cataloque.ItemName as Наименование, Storage.ItemID as ID\_Товара, Storage.ItemCode as Код\_Товара, Storage.Specification as Спецификация, Storage.ItemCost as Цена\_Отпускная, Storage.Balance as Остаток, Storage.Adress as Адрес

from Storage inner join Cataloque on Cataloque.ItemCode=Storage.ItemCode

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[DeleteOrder] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[DeleteOrder]

@OrderID int

As

If exists (Select\*From Orders Where OrderID=@OrderID and Condition='Оформление')

Begin

Update Storage Set Balance=Balance+Quantity from Statement

where Statement.OrderID=@OrderID and Statement.ItemID=Storage.ItemID

Delete From Statement where OrderID=@OrderID

Delete From Orders where OrderID=@OrderID

End

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[DElItemID] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[DElItemID]

@ItemID int,

@OrderID int

AS

Declare @Balance int

Declare @Quantity real

Begin

Select @Quantity=Quantity FRom Statement

Where ItemID=@ItemID And OrderID=@OrderID

Update Storage Set Balance=Balance+@Quantity

Where ItemID=@ItemID

Delete From Statement

Where ItemID=@ItemID

End;

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Raspredelenie] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[Raspredelenie]

as

select Statement.ItemID as КодТовара, Storage.Specification as Наименование, count (Orders.OrderID) as КолЗак,

Storage.Adress as Старый\_Адрес from Statement inner join Orders on Statement.OrderID=Orders.OrderID

inner join Storage on Statement.ItemID=Storage.ItemID

where Statement.ItemID between 1 and 61

group by Statement.ItemID, Storage.Specification, Storage.Adress

order by КолЗак

Desc

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Raspredelenie2] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[Raspredelenie2]

as

select NewAdress as Новый\_Адрес from NewStorage

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectClient] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[SelectClient]

As

Select Client From Orders

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectItemForDel] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[SelectItemForDel]

as

Select ItemID From Statement

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectItemID] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[SelectItemID]

as

Select ItemID from Storage

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectOrderForDel] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[SelectOrderForDel]

as

Select distinct OrderID from Statement

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectOrderID] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[SelectOrderID]

as

select OrderID from Orders

where Condition='оформление'

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectZakIDForDel] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[SelectZakIDForDel]

as

Select OrderID from Orders

where Condition='оформление'

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Statement\_Storage] Script Date: 30.03.2019 17:09:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[Statement\_Storage]

@OrderID int

AS

select Storage.Adress as Адрес, Statement.ItemID as ID\_Товара, Storage.Specification as Спецификация,

Statement.Quantity as Колличество

From Statement inner join Storage on Statement.ItemID=Storage.ItemID

Where Statement.OrderID=@OrderID

GO

USE [master]

GO

ALTER DATABASE [MyStorage] SET READ\_WRITE

GO