|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| «Национальный исследовательский ядерный университет «МИФИ» |
| **Технологический институт –** |
| филиал федерального государственного автономного образовательного учреждения высшего |
| образования «Национальный исследовательский ядерный университет «МИФИ» |

**(ТИ НИЯУ МИФИ)**

**Кафедра информационных технологий и прикладной математики**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по дисциплине «Разработка программных модулей»

**Разработка клиент-серверного приложения для автоматизации работы поставки товаров**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| **Руководитель**  ст. преподаватель |  |  |  | А.А.Шляпкин |
| *(должность)* |  | *(подпись)* |  | *(И.О. Фамилия)* |
|  |  |  |  |  |
| **Студент**  ИСП-311 |  |  |  | А.С. Шумихин |
| *(группа)* |  | *(подпись)* |  | *(И.О. Фамилия)* |

Курсовая работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_20\_\_ г.

Лесной 2023

**Содержание**

[**Введение** 3](#_Toc166882136)

[**1 Анализ предметной области** 4](#_Toc166882137)

[**2 Анализ инструментальных средств разработки** 6](#_Toc166882138)

[**3 Разработка логической модели базы данных** 8](#_Toc166882139)

[**4 Создание базы данных и разработка клиентского приложения** 9](#_Toc166882140)

[4.1 Создание базы данных 9](#_Toc166882141)

[4.2 Взаимосвязь базы данных и приложения 10](#_Toc166882142)

[4.3 Программирование клиентского приложения 10](#_Toc166882143)

[**5 Проверка работоспособности приложения** 18](#_Toc166882144)

[**Заключение** 29](#_Toc166882145)

[**Список использованных источников** 30](#_Toc166882146)

[**ПРИЛОЖЕНИЕ А. Структура таблиц БД "Доставка"** 31](#_Toc166882147)

[**ПРИЛОЖЕНИЕ Б (обязательное). Программный код на языке C#** 34](#_Toc166882148)

[9.1 Листинг формы авторизации 34](#_Toc166882149)

[9.2 Листинг главной формы 35](#_Toc166882150)

[9.3 Листинг формы добавления 49](#_Toc166882151)

[9.4 Листинг формы редактирования 64](#_Toc166882152)

[9.5 Листинг формы диалогового окна добавления заказа 84](#_Toc166882153)

[9.6 Листинг формы просмотра заказов пользователя 87](#_Toc166882154)

[**Приложение В (обязательное). Программный код создания БД** 90](#_Toc166882155)

# **Введение**

Современная динамичная бизнес-среда требует эффективных инструментов управления и оптимизации бизнес-процессов. Она характеризуется быстрыми темпами изменений, конкурентной борьбой, постоянным развитием технологий и рынков. В таких условиях компании вынуждены постоянно адаптироваться и реагировать на изменения внешней среды, чтобы оставаться конкурентоспособными. Одним из ключевых направлений в этой области является автоматизация операций по управлению запасами и поставкам товаров.

Поставки товаров – сложный процесс. Это огромный массив информации, в который входит: кто оформил заказ, когда оформил его, что входит в заказ, и прочее. Этот массив надо обрабатывать, то есть хранить информацию и ей пользоваться, например для подсчёта выручки, расчёта затрат, пересчета техники и для других целей. Именно благодаря автоматизации, этот процесс становится максимально лёгким и быстрым по времени.

Целью данной курсовой работы является создание программного модуля, способного упростить и ускорить процесс поставки товаров, снизить вероятность ошибок и повысить эффективность управления складскими запасами.

Для успешной реализации цели необходимо решить следующие задачи:

* Выявить и сформировать требования к приложению;
* Составить логическую модель базы;
* Создать базу данных;
* Разработать приложение на языке C#;
* Протестировать приложение, проверив его на работоспособность.

## **1 Анализ предметной области**

В рамках курсовой работы необходимо разработать систему, которая бы осуществляла:

* Ведение перечня товаров;
* Ведение перечня оформленных заказов;
* Ведение перечня доставок;
* Хранение информации о пользователе, работнике, для последующего взаимодействия с ними;
* Возможность для пользователей оформить заказ, выбрав товар и его количество;
* Ведение учёта действующих сотрудников;

Необходимо реализовать ролевую модель доступа к приложению, учитывающую возможность работы с ним зарегистрированных пользователей и администратора.

Представим, что есть фирма, у нее есть процессы, им требуется автоматизация. Фирма занимается перевозкой товаров от завода-изготовителя до условного заказчика.

У фирмы есть доставки, оформляемые по заказу. Эти доставки возят водители на определённый адрес, на определённом виде транспорта, загружая товар в различные контейнеры.

У фирмы есть разные виды транспорта – это связано с типом доставки. Фирма может поехать или полететь за товаром.

Чтобы оформить доставку, надо сделать заказ. Заказ оформляет менеджер.

Менеджеры – это сотрудники, которые работают условно в одном офисе, занимающиеся оформлением заказа, и редактированием информации о товарах. Помимо менеджеров, в этом же офисе есть руководитель, и другие работники, например: уборщицы, механики.

Фирма поставляет различные товары, то есть это могут быть и холодильники, и сварочные аппараты, и куча другой техники. Чтобы заказчик не искал инструменты среди холодильников, каждый товар принадлежит к какой-то категории: бытовая техника, инструменты.

Также, товары разделены по фирмам, например: холодильник от Bosch и Samsung. Каждый производитель может производить разные товары, например холодильник SamsungRB5000A, холодильник SamsungRB7200T, и ещё какую-нибудь духовку. Каждая модель товара имеет свой размер, и храниться в определенном количестве в определённом складе.

Как было сказано ранее, есть водители. Их отличие заключается в том, что они не находятся постоянно в этом офисе, этого не требует их работа, поэтому они вынесены отдельно.

Заказчик может делать несколько заказов, поэтому он связан один ко многим с заказами. Точно также и менеджер, он может оформлять заказ и других заказчиков, а водитель может возить несколько заказов. Фирмы и типы товаров тоже повторяются в товарах, потому что одна фирма может производить и стиральные машины, и холодильники, и это всё будет бытовая техника, поэтому и фирмы, и типы товаров связаны с товарами связью один ко многим. Но при этом заказ – уникален, поэтому он не должен повторяться. Это сделано с целью безопасности – представим, что с одним заказом возникла проблема – привезли не тот товар. Чтобы узнать, что пошло не так, мы должны найти тот заказ, а как его найти, если этих заказов несколько? Поэтому заказы связаны с доставкой связью один к одному, где один заказ – одна доставка

## **2 Анализ инструментальных средств разработки**

В данной работе для хранения базы данных была использована база данных MSSQL. Она представляет собой мощный и надежный инструмент для анализа данных, который обеспечивает высокую производительность, гибкость и функциональность, необходимые для успешного проведения анализа информации в различных областях исследования.

Для данной курсовой работы использовался язык программирования C#. Выбор языка программирования обусловлен его многочисленными достоинствами:

1. Простота изучения и использования. C# разработан с учетом простоты и понятности синтаксиса, что делает его доступным даже для начинающих программистов. Кроме того, C# поддерживает объектно-ориентированный подход к программированию, что упрощает организацию кода и повышает его читаемость.
2. Широкая поддержка. C# является одним из основных языков программирования в экосистеме Microsoft, что обеспечивает широкую поддержку и ресурсы для разработчиков. Он активно поддерживается сообществом разработчиков, что обеспечивает доступ к обучающим материалам, форумам поддержки и библиотекам кода.
3. Платформенная независимость. Хотя C# разрабатывался для платформы .NET Framework, с появлением .NET Core и .NET 5+ язык стал платформенно независимым, что позволяет использовать его для создания приложений для различных операционных систем, включая Windows, Linux и macOS.
4. Богатые возможности. C# обладает богатым набором возможностей, включая поддержку современных парадигм программирования, таких как асинхронное и параллельное программирование, LINQ для работы с данными, обработку исключений, а также множество встроенных библиотек для различных целей, таких как работа с сетью, графикой, базами данных и многое другое.
5. Интеграция с другими технологиями. C# хорошо интегрируется с другими технологиями и инструментами разработки, такими как Visual Studio, одна из самых популярных сред разработки, а также среды разработки от JetBrains, такие как Rider. Это обеспечивает разработчикам удобную среду для работы и интеграцию с другими инструментами.

В качестве среды программирования была выбрана Visual Studio 2022, позволяющая разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms.

## **3 Разработка логической модели базы данных**

На основе проведенного анализа предметной области можно выделить следующие сущности: Users, Warehouses, Container, Delivery, FirmsGoods, Goods, Orders, Roles, Transport, TypeGoods, Typeofdelivery.

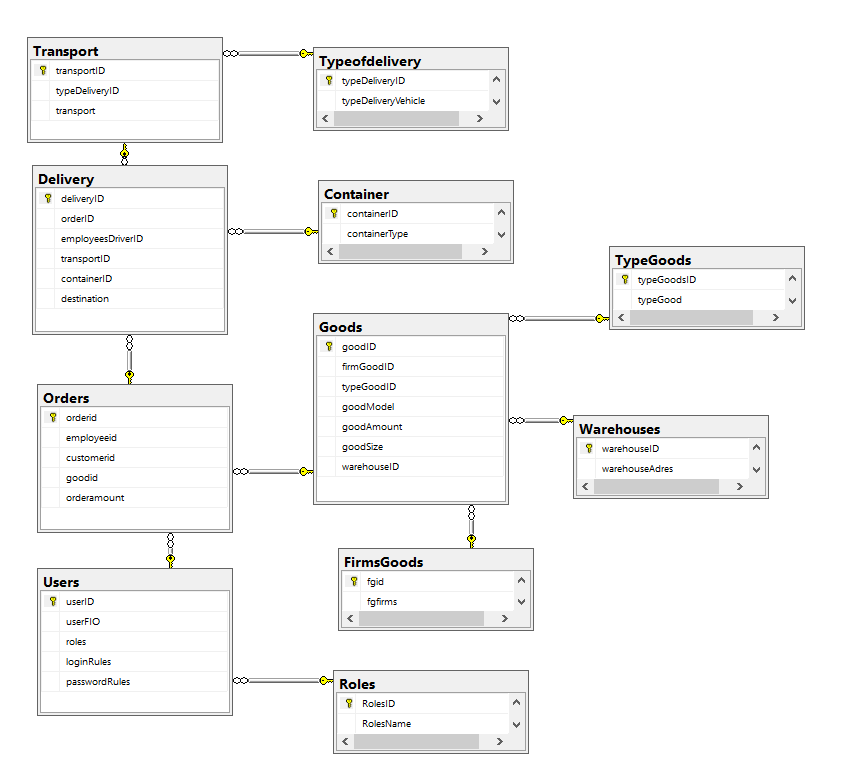
Логическая модель, разрабатываемая для целевой базы данных, достигла третьей нормальной формы. Она соответствует следующим условиям: уникальность записей, уникальность названий колонок (атрибутов) сущностей, отсутствие составных колонок, атомарность значений, функциональная зависимость не ключевых полей только от первичного ключа. Полученная логическая модель представлена на Рисунке 1.

Рисунок 1 - Логическая модель БД

## **4 Создание базы данных и разработка клиентского приложения**

### 4.1 Создание базы данных

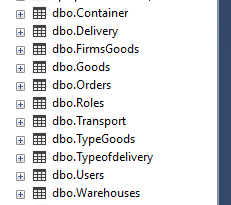
После создания логической модели БД в программе SQL Server Management Studio 19 на локальном сервере была создана база данных, состоящая из 12 таблиц: Users, Warehouses, Container, Delivery, FirmsGoods, Goods, Orders, Roles, Transport, TypeGoods, Typeofdelivery. Эти таблицы можно увидеть на Рисунке 2.

Рисунок 2 – Таблицы БД «Доставки товаров»

Таблица Container содержит информацию о видах контейнера, таблица Delivery – информацию о доставке заказов, из таблицы Orders, FirmsGoods – перечень фирм, TypeGoods – список типа товаров, Warehouses – адреса складов, Goods – таблица товаров, которая содержит информацию о товаре, в ней объединены таблицы FirmsGoods, TypeGoods, Warehouses. Transport – справочник транспорта, Typeofdelivery – виды передвижения транспорта, в таблице Users находятся все пользователи, а в таблице Roles – информация о ролях пользователей.

### 4.2 Взаимосвязь базы данных и приложения

Взаимосвязь базы и приложения осуществляется за счёт базовых компонентов библиотеки SqlClient: SqlConnection, SqlCommand, SqlDataReader, SqlAdapter. Общение с базой обеспечивается за счёт использования встроенных в компоненты функций и свойств, а также динамического изменения данных свойств.

### 4.3 Программирование клиентского приложения

После создания БД, необходимо разработать клиентское приложение. Процесс состоит из создания графического интерфейса и программирование функции для работы с БД.

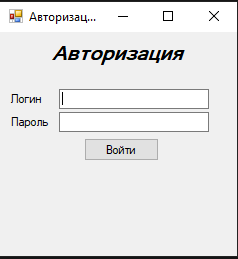
 Внешний вид окна авторизации и главной формы представлены на рисунке 3, 4 соответственно.

Рисунок 3 – Внешний вид окна авторизации

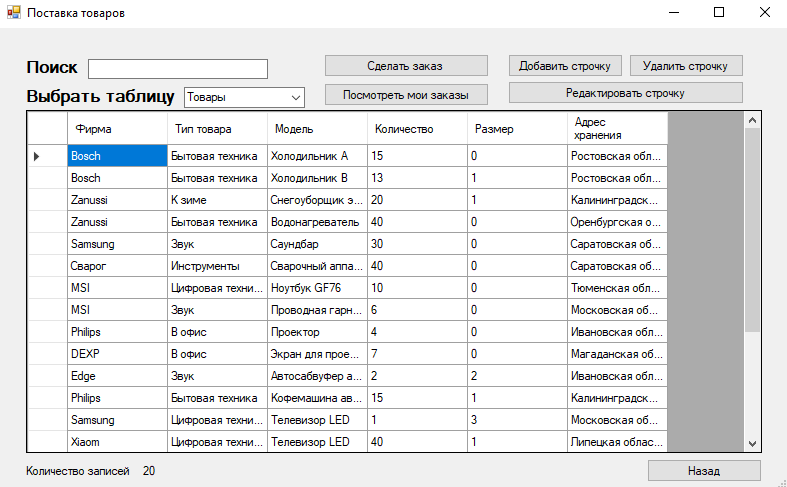


Рисунок 4 – Внешний вид главной формы в роли руководителя

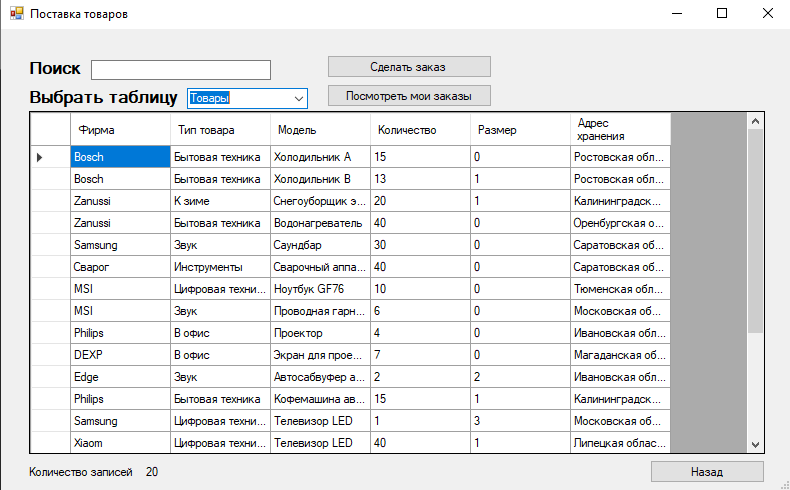
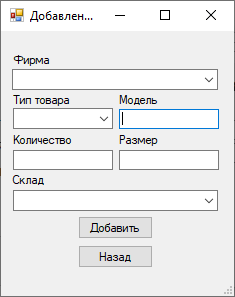
Внешний вид главной формы отличается для пользователя, так как ему нельзя вносить правки в таблицы, она представлена на рисунке 5.

Рисунок 5 – Внешний вид главной формы в роли пользователя

Для добавления данных в соответствующие таблицы в БД используются вспомогательные формы, внешний вид которых приведен на Рисунках 6-17.

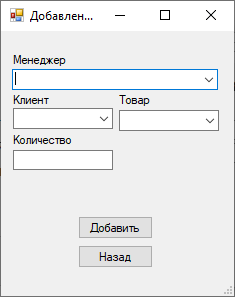
Рисунок 6 - Внешний вид формы добавления и редактирования данных о новом товаре

Рисунок 7 - Внешний вид формы добавления и редактирования данных о новом заказе

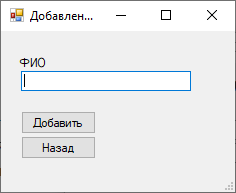


Рисунок 8 - Внешний вид формы добавления и редактирования данных о новом водителе

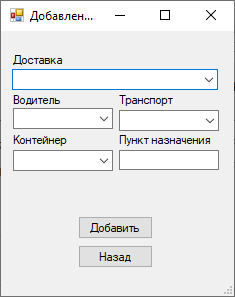
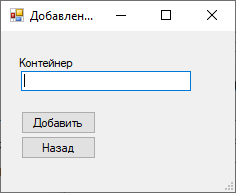


Рисунок 9 - Внешний вид формы добавления и редактирования данных о новой доставке



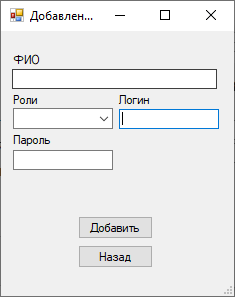
Рисунок 10 - Внешний вид формы добавления и редактирования данных о новом контейнере

Рисунок 11 - Внешний вид формы добавления и редактирования данных о новом пользователе

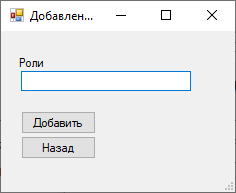


Рисунок 12 - Внешний вид формы добавления и редактирования данных о новой роли

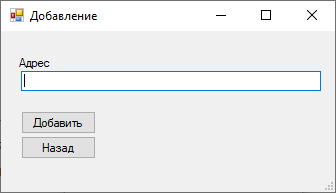


Рисунок 13 - Внешний вид формы добавления и редактирования данных о новом адресе

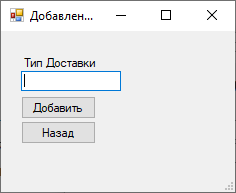


Рисунок 14 - Внешний вид формы добавления и редактирования данных о новом типе доставки

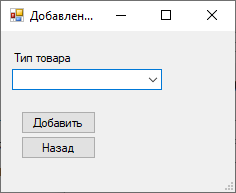
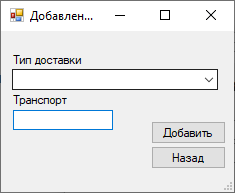


Рисунок 15 - Внешний вид формы добавления и редактирования данных о новом типе товара



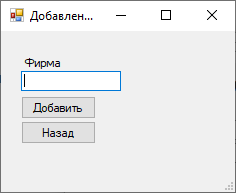
Рисунок 16 - Внешний вид формы добавления и редактирования данных о новом транспорте

Рисунок 17 - Внешний вид формы добавления и редактирования данных о новой фирме

В данной курсовой работе создано 6 форм для понятного функционала

приложения:

* FrmAuthorization – форма авторизации
* MainForm – главная форма
* AddForm – Форма добавления
* EditForm – Форма редактирования
* CheckOrder – форма отображения заказов пользователя
* InputDialog – форма-диалог для добавления заказа

Связи между формами представлены на Рисунке 18

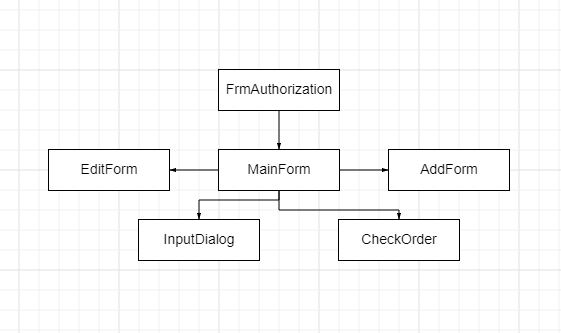


Рисунок 18 – Связи форм в приложении

Программный код для всех форм представлен в Приложении Б

## **5 Проверка работоспособности приложения**

При запуске приложения, пользователь увидит форму авторизации, представленную ранее на рисунке 3.

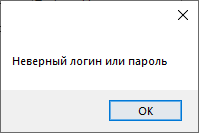
Пользователю необходимо заполнить данные логина и пароля, в случае ошибочного ввода, появится сообщение об ошибке, внешний вид которого представлен на Рисунке 19

Рисунок 19 – Сообщение о неверно веденных данных

При успешной авторизации пользователь попадает на главную форму приложения, которая была предоставлена на рисунке 5

В данной БД существует три вида ролей: менеджер, пользователь и руководитель. У руководителя есть доступ ко всем таблицам и все возможности их редактирования. У менеджера есть возможность редактировать данные о товарах, доставках, заказов, все что с ними связано, за исключением таблиц с ролями и пользователями, и возможности удаления строк. Пользователям же в свою очередь эти опции не доступны (кнопки редактирования скрыты). По умолчанию при входе отображается список товара.

Пользователь имеет возможность отфильтровать товары по алфавитному порядку, а также найти нужное через поисковую строку, что представлено на Рисунке 20.

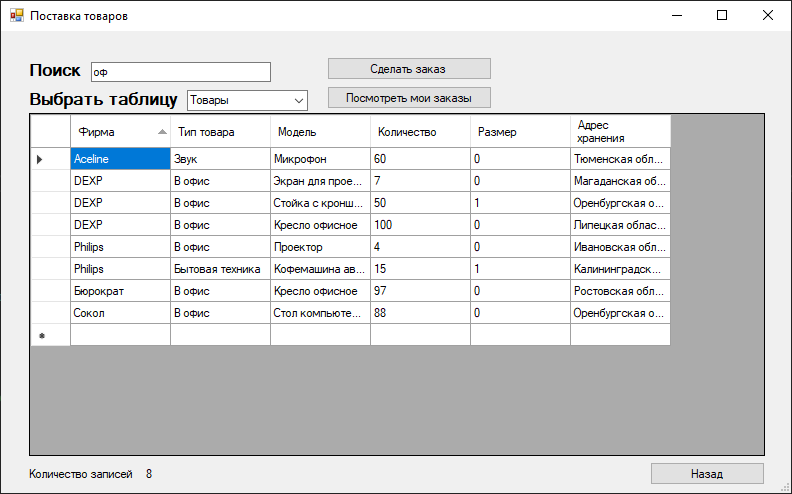


Рисунок 20 – Поисковая строка и отфильтрованный по алфавиту столбец «Фирма»

При нажатии на кнопку «Посмотреть мои заказы», открывается форма просмотра заказов, которая видна на рисунке 21

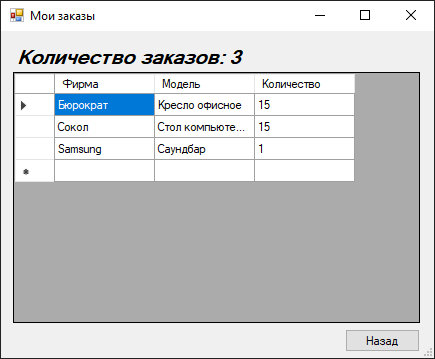


Рисунок 21 – Форма заказанных пользователем заказов

Нажав кнопку «Назад», пользователь вернется на главную форму.

Пользователь может посмотреть фирмы, которые продают товар, для этого необходимо выбрать таблицу Фирмы, которая представлена на рисунке 22.

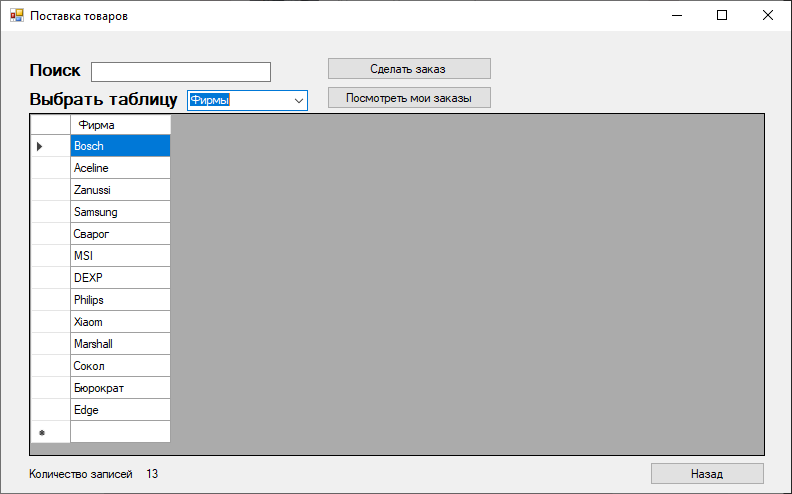


Рисунок 22 – Внешний вид формы просмотра Фирм

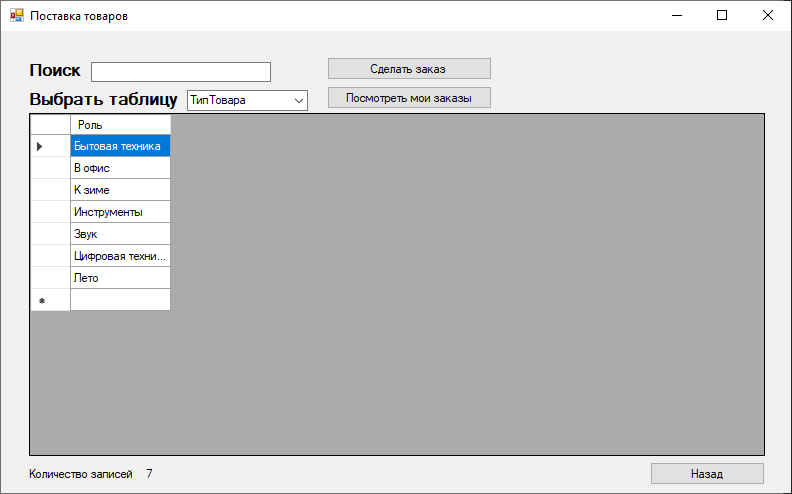
Аналогичным образом пользователь может посмотреть категории товаров, которые представлены на рисунке 23.

Рисунок 23 – Внешний вид формы просмотра категории товара

Доступа к остальным таблицам и возможности редактировать их у пользователя нет. Как было сказано ранее, остальные таблицы доступны для менеджера, кроме таблиц «Роли» и «Пользователи», а также у них отсутствует возможность удаления.

Если менеджер нажмет на таблицу «Водители», откроется таблиц, которая содержит ФИО всех водителей фирмы. Форма менеджера и таблицы представлена на рисунке 24.

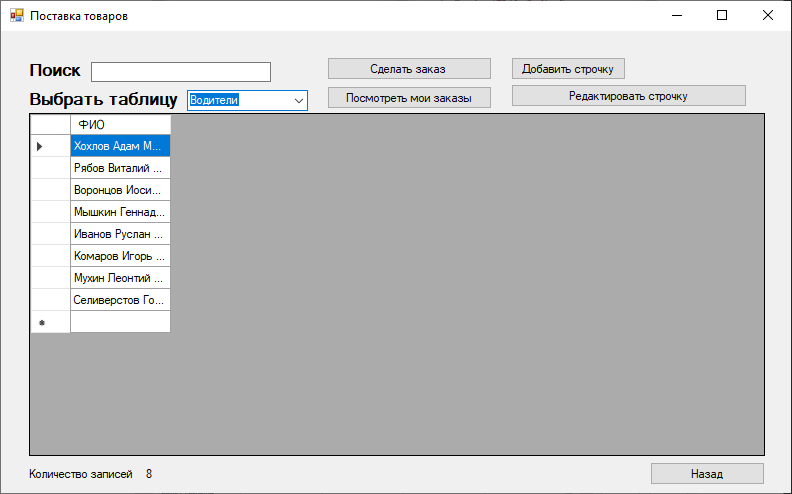


Рисунок 24 – Внешний вид формы менеджера и просмотра водителей

Последующим образом менеджер может посмотреть и остальные таблицы, а именно «Доставки», «Заказы», «Контейнер», «Склады», «ТипДоставки», «Транспорт», и описанные ранее таблицы для пользователя. Внешний вид форм представлен на рисунках 25-30.

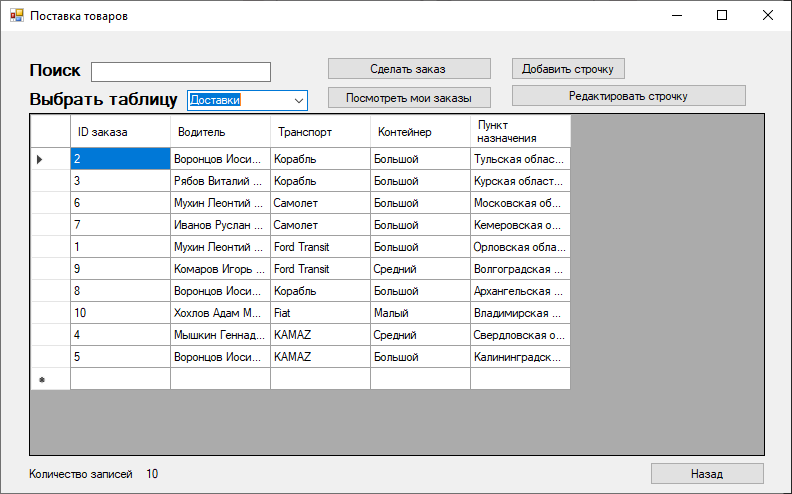


Рисунок 25 – Внешний вид формы «Доставки»

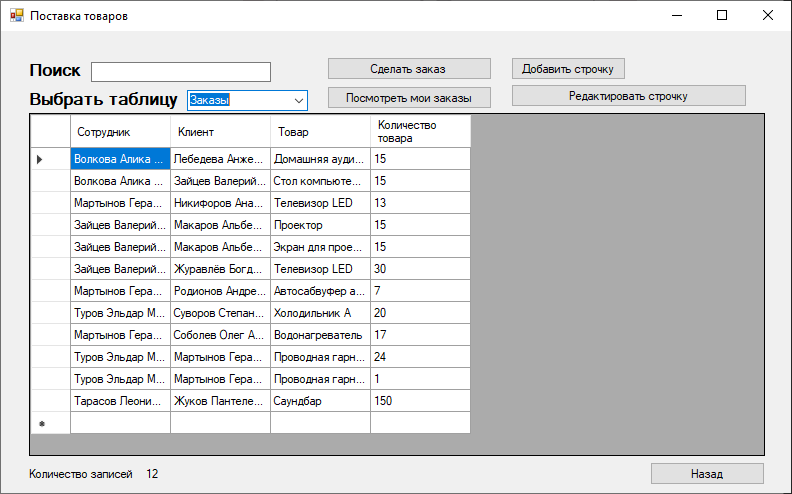


Рисунок 26 – Внешний вид формы «Заказы»

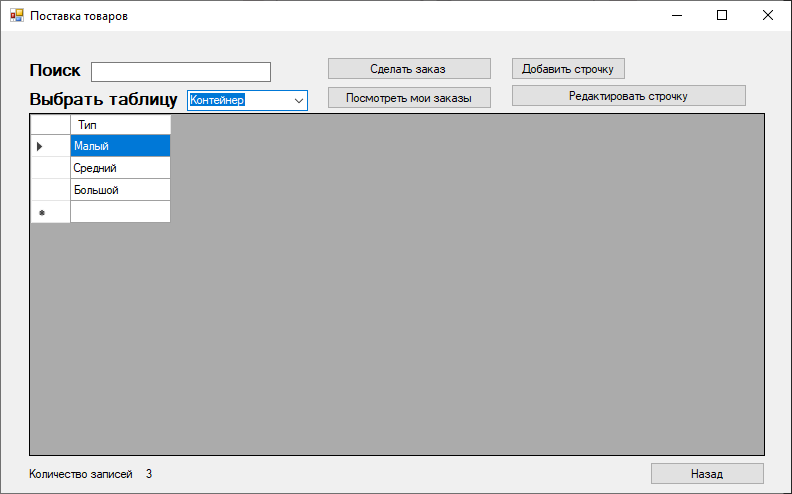


Рисунок 27 – Внешний вид формы «Контейнер»

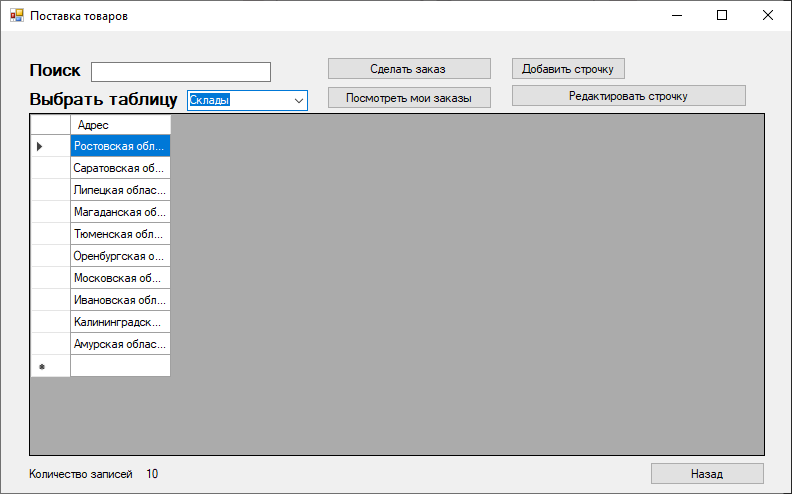


Рисунок 28 – Внешний вид формы «Склады»

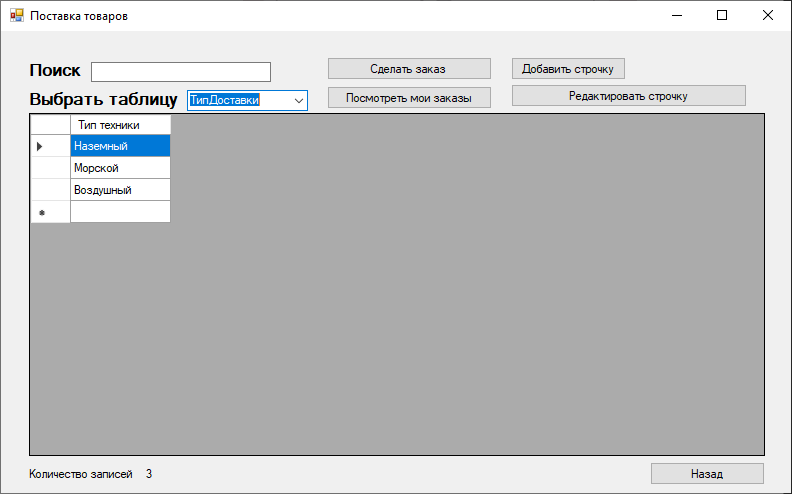


Рисунок 29 – Внешний вид формы «ТипДоставки»

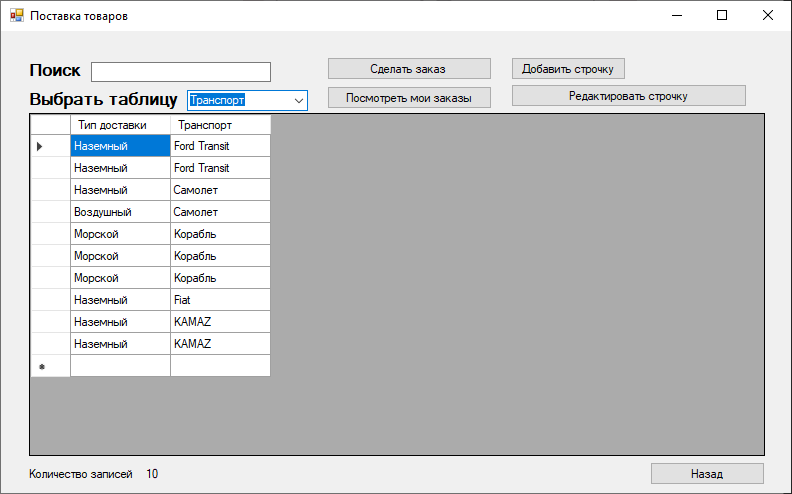


Рисунок 30 – Внешний вид формы «Транспорт»

Менеджер может добавить строку в таблицу, для этого в желаемой таблице надо нажать на кнопку «Добавить строчку», после чего откроется одна из форм, представленных на рисунках 6-17

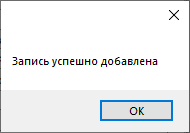
 После заполнения полей корректными данными, при нажатии кнопку «Добавить строчку», появится сообщение о успешной операции, представленное на рисунке 31.

Рисунок 31 – Окно успешного добавления записи в таблицу

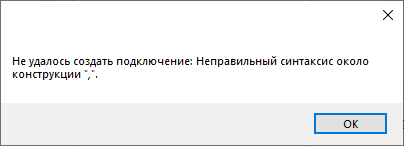
 В случае ввода некорректных данных, программа выдаст ошибку о некорректных данных, которая представлена на рисунке 32.

Рисунок 32 – Ошибка при добавлении записи

Если менеджер нажмет два раза на нужную строчку, или выбрав необходимую строчку и нажмёт «Редактировать строчку», откроется окно редактирования, которое идентично схоже на форму добавления, которая была описана ранее, только данные редактируемой строки автоматически передаются в поля. Внеся необходимые изменения, и нажав кнопку «Редактировать», появится сообщение об успешном редактировании, представленное на рисунке 33.

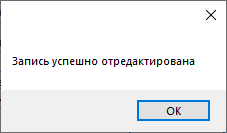


Рисунок 33 – Окно успешного редактирования записи в таблице

Если редактирование пройдёт не успешно, появится аналогичное окно с ошибкой, как при добавлении, представленное на рисунке 32.

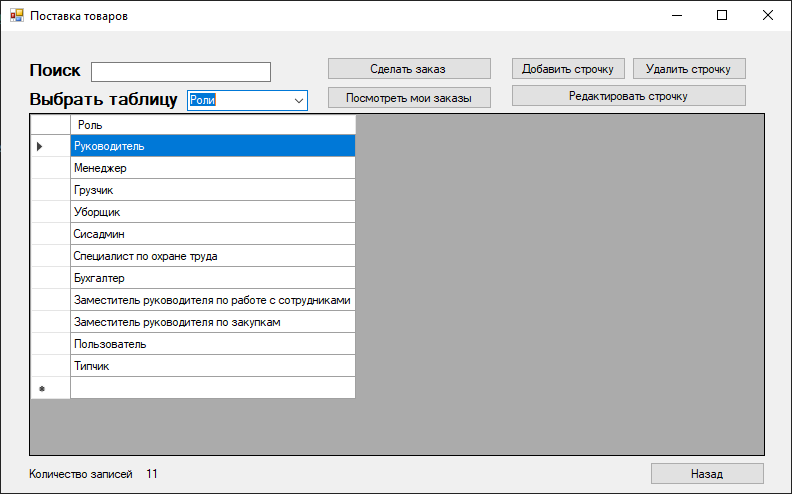
На этот моменте функционал менеджера заканчивается. Функционал руководителя абсолютно идентичен, как было сказано ранее, но у него есть доступ к таблицам «Роли» и «Пользователи», представленные на рисунке 34-35.

Рисунок 34 – Внешний вид формы ролей

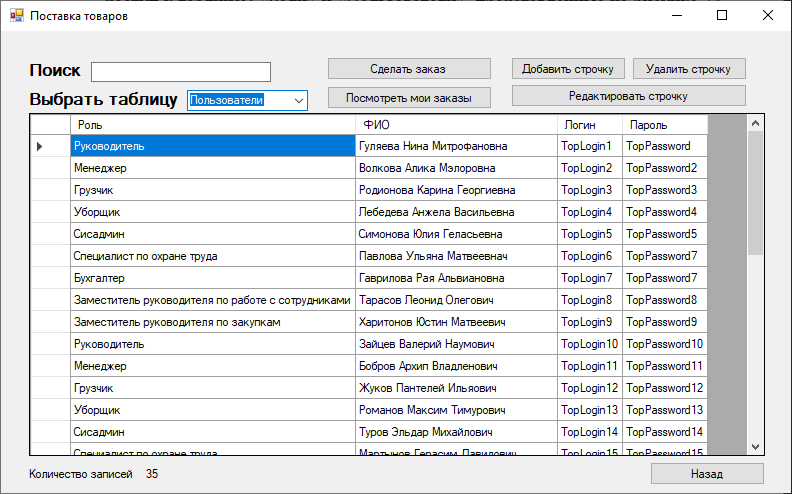


Рисунок 35 – Внешний вид формы пользователей

В этой главе был протестирован основной функционал приложения: управляющие элементы выполняют возложенные на них функции; каждый запрос пользователя проверяется на корректность и, в случае успеха проверки, он выполняется.

# **Заключение**

В результате проделанной работы была создана логическая модель базы данных, разработано клиентское приложение на языке C#. Все поставленные задачи были реализованы, цель достигнута.

Приложение соответствует выдвинутым требованиям, успешно их выполняя.

Проверка на работоспособность прошла хорошо, явных ошибок выявлено не было, следовательно приложение можно использовать.

В дальнейшем планируется расширить созданный функционал, добавив возможность автоматизации регистрации пользователя, систему расчёта стоимости товара и объёма посылки. Созданная структура базы позволяет изменять и дополнять приложение, существенно не меняя его и базу.

# **Список использованных источников**

1. Sufyan bin Uzayr. Mastering SQL – A Beginner's Guide (Mastering Computer Science). CRC Press, 2023. 244 p.
2. Е. А. Горшкова, Н. Г. Графеева, Б. А. Новиков; под ред. Е. В. Рогова. – Основы технологий баз данных: учеб. пособие /— 2-е изд. — М.: ДМК Пресс, 2020. — 582 с.
3. П. В. Евдокимов. – С# НА ПРИМЕРАХ / СПб.: Наука и Техника, 2016. – 304 с.

# **ПРИЛОЖЕНИЕ А. Структура таблиц БД "Доставка"**

Таблица A.1 – Поля таблицы Container

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| containerID | int, PK | ID контейнера |
| containerType | varchar(50) | Тип контейнера |

Таблица A.2 – Поля таблицы Delivery

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| deliveryID | int, PK | ID доставки |
| orderID | int | ID заказа |
| employeesDriverID | int | ID водителя |
| transportID | int | ID транспорта |
| containerID | int | ID контейнера |
| destination | varchar(150) | Пункт назначения |

Таблица A.3 – Поля таблицы EmployeeDriver

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| employeesDriverID | int, PK | ID водителя |
| employeeDriverFIO | varchar(50) | ФИО водителя |

Таблица A.4 – Поля таблицы FirmsGoods

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| gid | int, PK | ID товара фирмы |
| fgfirms | varchar(25) | Название фирмы |

Таблица A.5 – Поля таблицы Goods

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| goodID | int, PK | ID товара |
| firmGoodID | int | ID товара фирмы |
| typeGoodID | int | ID типа товара |
| goodModel | varchar(50) | Модель товара |
| goodAmount | int | Количество товара |
| goodSize | float | Размер товара |
| warehouseID | int | ID склада |

Таблица A.6 – Поля таблицы Orders

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| Orderid | int, PK | ID заказа |
| employeeid | int | ID сотрудника |
| customerid | int | ID клиента |
| goodid | int | ID товара |
| orderamount | int | Количество заказа |

Таблица A.7 – Поля таблицы Roles

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| RolesID | int, PK | ID роли |
| RolesName | varchar(50) | Название роли |

Таблица A.8 – Поля таблицы Transport

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| transportID | int, PK | ID транспорта |
| typeDeliveryID | int | ID типа доставки |
| transport | varchar(50) | Транспорт |

Таблица A.9 – Поля таблицы TypeGoods

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| typeGoodsID | int, PK | ID типа товара |
| typeGood | varchar(100) | Тип товара |

Таблица A.10 – Поля таблицы Typeofdelivery

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| typeDeliveryID | int, PK | ID типа доставки |
| typeDeliveryVehicle | varchar(100 | Вид транспортного средства |

Таблица A.11 – Поля таблицы Users

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| userID | int, PK | ID пользователя |
| userFIO | varchar(50) | ФИО пользователя |
| roles | int | Роль пользователя |
| loginRules | varchar(50) | Логин |
| passwordRules | varchar(50) | Пароль |

Таблица A.12 – Поля таблицы Warehouses

| **Название поля** | **Тип поля** | **Комментарий** |
| --- | --- | --- |
| warehouseID | int, PK | ID склада |
| warehouseAdres | varchar(250) | Адрес склада |

# **ПРИЛОЖЕНИЕ Б (обязательное). Программный код на языке C#**

## 9.1 Листинг формы авторизации

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace test\_subd

{

public partial class frmAuthorization : Form

{

static SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);

public frmAuthorization()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void btnConnect\_Click(object sender, EventArgs e) // Войти

{

try

{

if (connect.State == ConnectionState.Closed)

connect.Open();

int userID;

// если процедуры нет

SqlCommand command = new SqlCommand($"Select \* from [Users] where loginRules = '{tbLogin.Text}' and passwordRules = '{tbPassword.Text}' ", connect);

// подготавливаем команду для чтения

SqlDataReader dataReader = command.ExecuteReader();

dataReader.Read();

if (dataReader.HasRows) // если есть хоть одна строка

{

userID = dataReader.GetInt32(0);

MainForm fm = new MainForm(connect, dataReader.GetInt32(2), userID);

fm.ShowDialog();

this.Hide();

}

else

MessageBox.Show("Неверный логин или пароль");

SqlCommand logRequst = new SqlCommand();

connect.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось создать подключение: {ex.Message}");

return;

}

}

}

}

## 9.2 Листинг главной формы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Runtime.Remoting.Contexts;

using Microsoft.SqlServer.Server;

using static System.Windows.Forms.DataFormats;

using System.Data.Common;

using System.Diagnostics.Eventing.Reader;

using System.Reflection.Emit;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Security.Cryptography;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Menu;

namespace test\_subd

{

public partial class MainForm : Form

{

SqlConnection connect;

int roleForm;

int userID;

public MainForm(SqlConnection cnct, int rl, int uID)

{

userID = uID;

connect = cnct;

roleForm = rl;

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

dataGridView1.CellDoubleClick += DataGridViewCellDoubleClick;

if (roleForm == 2 || roleForm == 3 || roleForm == 4 || roleForm == 6 || roleForm == 7 || roleForm == 10)

{

addString.Visible = false;

delString.Visible = false;

EditStr.Visible = false;

cmbTables.Items.RemoveAt(10);

cmbTables.Items.RemoveAt(7);

cmbTables.Items.RemoveAt(6);

cmbTables.Items.RemoveAt(5);

cmbTables.Items.RemoveAt(4);

cmbTables.Items.RemoveAt(3);

cmbTables.Items.RemoveAt(2);

cmbTables.Items.RemoveAt(1);

cmbTables.Items.RemoveAt(0);

}

else if (roleForm == 5 || roleForm == 8 || roleForm == 9)

{

delString.Visible = false;

cmbTables.Items.RemoveAt(5);

cmbTables.Items.RemoveAt(4);

}

SqlConnection sqlConnect = new SqlConnection(Properties.Settings.Default.connectionString); // тут надо вставить переменную

sqlConnect.Open();

SqlCommand logRequst = new SqlCommand();

logRequst.CommandText = $"SELECT \* FROM \_Товары";

logRequst.Connection = sqlConnect;

// SqlAdapter - прослойка между источником данных и базой данных

SqlDataAdapter adapter = new SqlDataAdapter(logRequst);

DataSet dataSet = new DataSet();

// заполняем источник данных полученными из адаптера записями

adapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

lblCount.Text = Convert.ToString(dataSet.Tables[0].Rows.Count); //считаем строчки

sqlConnect.Close();

dataGridView1.Columns[0].Visible = false;

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void txtbSearch\_TextChanged(object sender, EventArgs e)

{

var columnNames = dataGridView1.Columns.Cast<DataGridViewColumn>()

.Select(x => "[" + x.HeaderText + "]")

.ToList();

// Создаем строку фильтрации

string filterExpression = string.Empty;

foreach (var columnName in columnNames)

{

if (!string.IsNullOrEmpty(filterExpression))

filterExpression += " OR ";

filterExpression += $"CONVERT({columnName}, 'System.String') LIKE '%{txtbSearch.Text}%'";

}

// Применяем фильтр к строкам DGV1

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = filterExpression;

// Подсчитываем количество отфильтрованных строк

int rowCount = (dataGridView1.DataSource as DataTable).DefaultView.Count;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

// Выводим количество найденных строк в label1

lblCount.Text = rowCount.ToString();

}

public void RefreshData()

{

// Создаем подключение к базе данных

using (SqlConnection sqlConnect = new SqlConnection(Properties.Settings.Default.connectionString))

{

// Открываем подключение

sqlConnect.Open();

// Создаем команду SQL для выполнения запроса

SqlCommand logRequst = new SqlCommand();

logRequst.Connection = sqlConnect;

logRequst.CommandText = $"SELECT \* FROM \_{cmbTables.SelectedItem.ToString()}";

// Создаем адаптер данных

SqlDataAdapter adapter = new SqlDataAdapter(logRequst);

// Создаем набор данных

DataSet dataSet = new DataSet();

// Заполняем набор данных данными из адаптера

adapter.Fill(dataSet);

// Привязываем данные к DataGridView на форме 2

dataGridView1.DataSource = dataSet.Tables[0];

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

// Обновляем текст метки с количеством строк

lblCount.Text = dataSet.Tables[0].Rows.Count.ToString();

}

}

private void cmbTables\_SelectedIndexChanged(object sender, EventArgs e)

{

SqlConnection sqlConnect = new SqlConnection(Properties.Settings.Default.connectionString);

SqlCommand logRequst = new SqlCommand();

logRequst.CommandText = $"SELECT \* FROM \_{cmbTables.SelectedItem.ToString()}";

logRequst.Connection = sqlConnect;

// SqlAdapter - прослойка между источником данных и базой данных

SqlDataAdapter adapter = new SqlDataAdapter(logRequst);

DataSet dataSet = new DataSet();

// заполняем источник данных полученными из адаптера записями

adapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

lblCount.Text = Convert.ToString(dataSet.Tables[0].Rows.Count); //считаем строчки

}

private void btnExit\_Click(object sender, EventArgs e)

{

// Создаем экземпляр первой формы

Form frmAuthorization = new frmAuthorization();

// Показываем первую форму

frmAuthorization.Show();

// Закрываем текущую форму (вторую форму)

this.Close();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void addString\_Click(object sender, EventArgs e)

{

string boxItem = cmbTables.SelectedItem.ToString();

this.Hide();

//Form fm = Application.OpenForms["Form3"];

AddForm fm = new AddForm(connect, roleForm, boxItem);

fm.ShowDialog();

}

private void delString\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Вы не выбрали строку для удаления", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Получаем выбранную строку

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

// Формируем текст для подтверждения удаления

string confirmationText = $"Вы действительно хотите удалить из таблицы ";

// Получаем значения всех столбцов выбранной строки и добавляем их к тексту подтверждения

for (int i = 0; i < selectedRow.Cells.Count; i++)

{

confirmationText += $"{selectedRow.Cells[i].Value.ToString()}";

if (i < selectedRow.Cells.Count - 1)

confirmationText += " - ";

}

// Показываем окно подтверждения

DialogResult result = MessageBox.Show(confirmationText, "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

// Если пользователь подтвердил удаление, выполняем операцию

if (result == DialogResult.Yes)

{

try

{

using (SqlConnection sqlConnect = new SqlConnection(Properties.Settings.Default.connectionString))

{

sqlConnect.Open();

SqlCommand logRequest = new SqlCommand();

logRequest.Connection = sqlConnect;

string columnName = dataGridView1.Columns[0].HeaderText;

string condition = selectedRow.Cells[0].Value.ToString();

string tableName = cmbTables.SelectedItem.ToString();

if (cmbTables.SelectedItem.ToString() == "Товары")

{

logRequest.CommandText = $"DELETE FROM Goods WHERE goodID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Товары";

}

if (cmbTables.SelectedItem.ToString() == "Контейнер")

{

logRequest.CommandText = $"DELETE FROM Container WHERE containerID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Контейнер";

}

if (cmbTables.SelectedItem.ToString() == "Доставки")

{

logRequest.CommandText = $"DELETE FROM Delivery WHERE deliveryID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Доставки";

}

if (cmbTables.SelectedItem.ToString() == "Водители")

{

logRequest.CommandText = $"DELETE FROM EmployeeDriver WHERE employeeDriverID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Водители";

}

if (cmbTables.SelectedItem.ToString() == "Фирмы")

{

logRequest.CommandText = $"DELETE FROM FirmsGoods WHERE fgid = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Фирмы";

}

if (cmbTables.SelectedItem.ToString() == "Заказы")

{

logRequest.CommandText = $"DELETE FROM Orders WHERE orderid = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Заказы";

}

if (cmbTables.SelectedItem.ToString() == "Роли")

{

logRequest.CommandText = $"DELETE FROM Roles WHERE RolesID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Роли";

}

if (cmbTables.SelectedItem.ToString() == "Транспорт")

{

logRequest.CommandText = $"DELETE FROM Transport WHERE transportID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Транспорт";

}

if (cmbTables.SelectedItem.ToString() == "ТипТовара")

{

logRequest.CommandText = $"DELETE FROM TypeGoods WHERE typeGoodsID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_ТипТовара";

}

if (cmbTables.SelectedItem.ToString() == "ТипДоставки")

{

logRequest.CommandText = $"DELETE FROM Typeofdelivery WHERE typeDeliveryID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_ТипДоставки";

}

if (cmbTables.SelectedItem.ToString() == "Пользователи")

{

logRequest.CommandText = $"DELETE FROM Users WHERE userID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Пользователи";

}

if (cmbTables.SelectedItem.ToString() == "Склады")

{

logRequest.CommandText = $"DELETE FROM Warehouses WHERE warehouseID = {selectedRow.Cells[0].Value.ToString()}";

logRequest.ExecuteNonQuery();

logRequest.CommandText = "SELECT \* FROM \_Склады";

}

// SqlAdapter - прослойка между источником данных и базой данных

SqlDataAdapter adapter = new SqlDataAdapter(logRequest);

DataSet dataSet = new DataSet();

// заполняем источник данных полученными из адаптера записями

adapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

lblCount.Text = Convert.ToString(dataSet.Tables[0].Rows.Count); //считаем строчки

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка удаления: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void EditStr\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Вы не выбрали строку для редактирования", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Получаем выбранную строку

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

// Формируем текст для подтверждения удаления

string confirmationText = $"Вы действительно хотите редактировать запись: ";

// Получаем значения всех столбцов выбранной строки и добавляем их к тексту подтверждения

for (int i = 0; i < selectedRow.Cells.Count; i++)

{

confirmationText += $"{selectedRow.Cells[i].Value.ToString()}";

if (i < selectedRow.Cells.Count - 1)

confirmationText += " - ";

}

// Показываем окно подтверждения

DialogResult result = MessageBox.Show(confirmationText, "Подтверждение редактирования", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

// Если пользователь подтвердил удаление, выполняем операцию

if (result == DialogResult.Yes)

{

string boxItem = cmbTables.SelectedItem.ToString();

string condition = selectedRow.Cells[0].Value.ToString();

this.Hide();

try

{

EditForm fm = new EditForm(connect, roleForm, boxItem, condition);

fm.ShowDialog();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка редактирования: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void DataGridViewCellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (roleForm == 1 || roleForm == 5 || roleForm == 8 || roleForm == 9)

{

// Получаем выбранную строку

DataGridViewRow selectedRow = dataGridView1.CurrentRow;

// Формируем текст для подтверждения удаления

string confirmationText = $"Вы действительно хотите редактировать запись: ";

// Получаем значения всех столбцов выбранной строки и добавляем их к тексту подтверждения

for (int i = 0; i < selectedRow.Cells.Count; i++)

{

confirmationText += $"{selectedRow.Cells[i].Value.ToString()}";

if (i < selectedRow.Cells.Count - 1)

confirmationText += " - ";

}

DialogResult result = MessageBox.Show(confirmationText, "Подтверждение редактирования", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

string boxItem = cmbTables.SelectedItem.ToString();

// Получение значения из 1 столбца (индексация начинается с 0)

string condition = selectedRow.Cells[0].Value.ToString();

this.Hide();

EditForm fm = new EditForm(connect, roleForm, boxItem, condition);

fm.ShowDialog();

}

}

}

private void textBoxColumn\_TextChanged(object sender, EventArgs e)

{

}

private void btnDoOrder\_Click(object sender, EventArgs e)

{

if (cmbTables.SelectedItem?.ToString() != "Товары")

{

MessageBox.Show("Заказ можно совершить только выбрав товар", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else {

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Вы не выбрали товар", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

else {

// Получаем выбранную строку

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

// Формируем текст для подтверждения удаления

string confirmationText = $"Вы действительно хотите заказать: ";

string goodID = selectedRow.Cells[0].Value.ToString();

// Получаем значения всех столбцов выбранной строки и добавляем их к тексту подтверждения

for (int i = 0; i < selectedRow.Cells.Count; i++)

{

confirmationText += $"{selectedRow.Cells[i].Value.ToString()}";

if (i < selectedRow.Cells.Count - 1)

confirmationText += " - ";

}

// Показываем окно подтверждения

DialogResult result = MessageBox.Show(confirmationText, "Оформление заказа", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

string labelText = "Введите количество товара:";

using (InputDialog inputDialog = new InputDialog(labelText))

{

if (inputDialog.ShowDialog() == DialogResult.OK)

{

// Получаем значения customerID, goodID и orderAmount

int customerID = userID; // Предполагается, что UID - переменная, содержащая customerID

decimal orderAmount;

if (!decimal.TryParse(inputDialog.EnteredText, out orderAmount))

{

MessageBox.Show("Пожалуйста, введите корректное число.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Подключение к базе данных и выполнение запроса

try

{

using (SqlConnection sqlConnect = new SqlConnection(Properties.Settings.Default.connectionString))

{

sqlConnect.Open();

SqlCommand command = new SqlCommand();

command.Connection = sqlConnect;

// Получаем максимальное значение userID с учетом указанных RoleID

command.CommandText = "SELECT MAX(userID) FROM Users WHERE roles IN (5, 8, 9)";

int maxUserID = (int)command.ExecuteScalar();

// Выбираем случайный employeeID из диапазона от 1 до maxUserID

Random random = new Random();

int employeeID = random.Next(1, maxUserID + 1);

// Вставляем новую запись в таблицу Orders

command.CommandText = "INSERT INTO Orders (employeeID, customerID, goodID, orderAmount) VALUES (@EmployeeID, @CustomerID, @GoodID, @OrderAmount)";

command.Parameters.AddWithValue("@EmployeeID", employeeID);

command.Parameters.AddWithValue("@CustomerID", customerID);

command.Parameters.AddWithValue("@GoodID", goodID);

command.Parameters.AddWithValue("@OrderAmount", orderAmount);

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно сделали заказ");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка добавления заказа: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

}

}

private void btnCheckOrder\_Click(object sender, EventArgs e)

{

this.Hide();

CheckOrder fm = new CheckOrder(userID, connect);

fm.ShowDialog();

}

private void lblCount\_Click(object sender, EventArgs e)

{

}

}

/\*

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

--{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = $"Date > {dateTimePicker1.Value} AND Date < {dateTimePicker2}";

}

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = $"Date > {dateTimePicker1.Value} AND Date < {dateTimePicker2}";

}

\*/

}

## 9.3 Листинг формы добавления

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection.Emit;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace test\_subd

{

public partial class AddForm : Form

{

private SqlConnection connect;

private int roleForm;

private string boxItem;

public AddForm(SqlConnection cnct, int rl, string item)

{

connect = cnct;

roleForm = rl;

boxItem = item;

InitializeComponent();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand logRequest = new SqlCommand();

logRequest.Connection = conn;

if (boxItem == "Товары")

{

logRequest.CommandText = $"INSERT INTO [Goods] (firmGoodID, typeGoodID, goodModel, goodAmount, goodSize, warehouseID) " +

$"VALUES ({cmbFirstSell.SelectedIndex + 1}, {cmbSecondSell.SelectedIndex + 1}, '{txtThirdSell.Text}', {txtFourthSell.Text}, {txtFifthSell.Text}, {cmbFifthSell.SelectedIndex + 1})";

}

else if (boxItem == "Фирмы")

{

logRequest.CommandText = $"INSERT INTO [FirmsGoods] (fgfirms) VALUES ('{txtSecondSell.Text}')";

}

else if (boxItem == "Водители")

{

logRequest.CommandText = $"INSERT INTO [EmployeeDriver] (employeeDriverFIO) VALUES ('{txtSecondSell.Text}')";

}

else if (boxItem == "Контейнер")

{

logRequest.CommandText = $"INSERT INTO [Container] (containerType) VALUES ('{txtSecondSell.Text}')";

}

else if (boxItem == "Доставки")

{

logRequest.CommandText = $"INSERT INTO [Delivery] (orderID, employeesDriverID, transportID, containerID, destination) " +

$"VALUES ({cmbFirstSell.SelectedIndex + 1}, {cmbSecondSell.SelectedIndex + 1}, {cmbThirdSell.SelectedIndex + 1}, {cmbFourthSell.SelectedIndex + 1}, '{txtFifthSell.Text}')";

}

else if (boxItem == "Заказы")

{

int employeeId = (int)cmbFirstSell.SelectedValue;

int customerId = (int)cmbSecondSell.SelectedValue;

int goodId = (int)cmbThirdSell.SelectedValue;

logRequest.CommandText = $"INSERT INTO [Orders] (employeeid, customerid, goodid, orderamount) " +

$"VALUES ({employeeId}, {customerId}, {goodId}, '{txtFourthSell.Text}')";

}

else if (boxItem == "Роли")

{

logRequest.CommandText = $"INSERT INTO [Roles] (RolesName) VALUES ('{txtSecondSell.Text}')";

}

else if (boxItem == "Транспорт")

{

logRequest.CommandText = $"INSERT INTO [Transport] (typeDeliveryID, transport) VALUES ({cmbFirstSell.SelectedIndex + 1},'{txtSecondSell.Text}')";

}

else if (boxItem == "ТипТовара")

{

logRequest.CommandText = $"INSERT INTO [TypeGoods] (typeGood) VALUES ('{txtSecondSell.Text}')";

}

else if (boxItem == "ТипДоставки")

{

logRequest.CommandText = $"INSERT INTO [Typeofdelivery] (typeDeliveryVehicle) VALUES ('{txtSecondSell.Text}')";

}

else if (boxItem == "Пользователи")

{

logRequest.CommandText = $"INSERT INTO [Users] (userFIO, roles, loginRules, passwordRules) VALUES ('{txtFirstSell.Text}'), {cmbSecondSell.SelectedIndex + 1}, '{txtFourthSell.Text}', '{txtFifthSell.Text}'";

}

else if (boxItem == "Склады")

{

logRequest.CommandText = $"INSERT INTO [Warehouses] (warehouseAdres) VALUES ('{txtSecondSell.Text}')";

}

logRequest.ExecuteNonQuery();

MessageBox.Show("Запись успешно добавлена");

}

}

private void Form3\_Load(object sender, EventArgs e)

{

if (connect.State == ConnectionState.Open)

{

connect.Close();

}

if (boxItem == "Товары")

{

txtThirdSell.Visible = true;

txtFourthSell.Visible = true;

txtFifthSell.Visible = true;

txtSixthSell.Visible = true;

cmbSecondSell.Visible = true;

cmbFirstSell.Visible = true;

cmbFifthSell.Visible = true;

lblFirstSell.Visible = true;

lblSecondSell.Visible = true;

lblThirdSell.Visible = true;

lblFourthSell.Visible = true;

lblFifthSell.Visible = true;

lblSixthSell.Visible = true;

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand logRequest = new SqlCommand("SELECT fgid, fgfirms FROM FirmsGoods", conn);

using (SqlDataReader rdr = logRequest.ExecuteReader())

{

List<Categories> lstCategories = new List<Categories>();

while (rdr.Read())

{

lstCategories.Add(new Categories(Convert.ToInt32(rdr["fgid"]), rdr["fgfirms"].ToString()));

}

cmbFirstSell.DataSource = lstCategories;

cmbFirstSell.DisplayMember = "name";

cmbFirstSell.ValueMember = "id";

cmbFirstSell.Text = "";

}

SqlCommand typeGoodCommand = new SqlCommand("SELECT typeGoodsID, typeGood FROM TypeGoods", conn);

SqlDataReader reader1 = typeGoodCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["typeGoodsID"]), reader1["typeGood"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

cmbSecondSell.Text = "";

reader1.Close();

SqlCommand warehousesCommand = new SqlCommand("SELECT warehouseID, warehouseAdres FROM Warehouses", conn);

reader1 = warehousesCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["warehouseID"]), reader1["warehouseAdres"].ToString()));

}

cmbFifthSell.DataSource = lstCategories1; // Changed from comboBox2 to comboBox5

cmbFifthSell.DisplayMember = "name"; // Changed from "name" to "warehouseAdres"

cmbFifthSell.ValueMember = "id"; // Changed from "id" to "warehouseID"

cmbFifthSell.Text = "";

reader1.Close();

}

}

else if (boxItem == "Фирмы")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 20;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

btnAdd.Left = 20;

btnAdd.Top = 65;

btnExit.Left = 20;

btnExit.Top = 90;

lblSecondSell.Text = "Фирма";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

}

else if (boxItem == "Водители")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 170;

btnAdd.Left = 20;

btnAdd.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "ФИО";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

}

else if (boxItem == "Контейнер")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 170;

btnAdd.Left = 20;

btnAdd.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "Контейнер";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

}

else if (boxItem == "Доставки")

{

lblFirstSell.Text = "Доставка";

lblSecondSell.Text = "Водитель";

lblThirdSell.Text = "Транспорт";

lblFourthSell.Text = "Контейнер";

lblFifthSell.Text = "Пункт назначения";

txtFifthSell.Visible = true;

cmbFirstSell.Visible = true;

cmbSecondSell.Visible = true;

cmbThirdSell.Visible = true;

cmbFourthSell.Visible = true;

lblFirstSell.Visible = true;

lblSecondSell.Visible = true;

lblThirdSell.Visible = true;

lblFourthSell.Visible = true;

lblFifthSell.Visible = true;

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

// order

SqlCommand orderCommand = new SqlCommand("Select orderid from Orders", conn);

SqlDataReader reader1 = orderCommand.ExecuteReader();

if (reader1.HasRows)

{

while (reader1.Read())

{

cmbFirstSell.Items.Add(Convert.ToInt32(reader1["orderid"]));

}

}

reader1.Close();

// driver

SqlCommand employeeDriverCommand = new SqlCommand("Select employeesDriverID, employeeDriverFIO from EmployeeDriver", conn);

reader1 = employeeDriverCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["employeesDriverID"]), reader1["employeeDriverFIO"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

cmbSecondSell.Text = "";

reader1.Close();

// transport

SqlCommand transportCommand = new SqlCommand("Select transportID, transport from Transport", conn);

reader1 = transportCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["transportID"]), reader1["transport"].ToString()));

}

cmbThirdSell.DataSource = lstCategories1;

cmbThirdSell.DisplayMember = "name";

cmbThirdSell.ValueMember = "id";

cmbThirdSell.Text = "";

reader1.Close();

// container

SqlCommand containerCommand = new SqlCommand("Select containerID, containerType from Container", conn);

reader1 = containerCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["containerID"]), reader1["containerType"].ToString()));

}

cmbFourthSell.DataSource = lstCategories1;

cmbFourthSell.DisplayMember = "name";

cmbFourthSell.ValueMember = "id";

cmbFourthSell.Text = "";

}

}

else if (boxItem == "Заказы")

{

lblFirstSell.Text = "Менеджер";

lblSecondSell.Text = "Клиент";

lblThirdSell.Text = "Товар";

lblFourthSell.Text = "Количество";

lblFirstSell.Visible = true;

lblSecondSell.Visible = true;

lblThirdSell.Visible = true;

lblFourthSell.Visible = true;

txtFourthSell.Visible = true;

cmbFirstSell.Visible = true;

cmbSecondSell.Visible = true;

cmbThirdSell.Visible = true;

// employee

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand employeeCommand = new SqlCommand("Select userID, userFIO from Users where roles in (1,5,8,9)", conn);

SqlDataReader reader1 = employeeCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["userID"]), reader1["userFIO"].ToString()));

}

cmbFirstSell.DataSource = lstCategories1;

cmbFirstSell.DisplayMember = "name";

cmbFirstSell.ValueMember = "id";

cmbFirstSell.Text = "";

reader1.Close();

// customer

SqlCommand customerCommand = new SqlCommand("Select userID, userFIO from Users where roles in (2,3,4,6,7,10)", conn);

reader1 = customerCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["userID"]), reader1["userFIO"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

cmbSecondSell.Text = "";

reader1.Close();

// good

SqlCommand goodCommand = new SqlCommand("Select goodID, goodModel from Goods", conn);

reader1 = goodCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["goodID"]), reader1["goodModel"].ToString()));

}

cmbThirdSell.DataSource = lstCategories1;

cmbThirdSell.DisplayMember = "name";

cmbThirdSell.ValueMember = "id";

cmbThirdSell.Text = "";

reader1.Close();

}

}

else if (boxItem == "Роли")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 170;

btnAdd.Left = 20;

btnAdd.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "Роли";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

}

else if (boxItem == "Транспорт")

{

this.Height = 200;

btnAdd.Left = 150;

btnAdd.Top = 90;

btnExit.Left = 150;

btnExit.Top = 115;

lblFirstSell.Text = "Тип доставки";

lblSecondSell.Text = "Транспорт";

lblFirstSell.Visible = true;

lblSecondSell.Visible = true;

cmbFirstSell.Visible = true;

txtSecondSell.Visible = true;

// type

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand typeCommand = new SqlCommand("Select typeDeliveryID, typeDeliveryVehicle from Typeofdelivery", conn);

SqlDataReader reader1 = typeCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["typeDeliveryID"]), reader1["typeDeliveryVehicle"].ToString()));

}

cmbFirstSell.DataSource = lstCategories1;

cmbFirstSell.DisplayMember = "name";

cmbFirstSell.ValueMember = "id";

cmbFirstSell.Text = "";

reader1.Close();

}

}

else if (boxItem == "ТипТовара")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 10;

lblSecondSell.Top = 20;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

cmbFirstSell.Width = 150;

btnAdd.Left = 20;

btnAdd.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "Тип товара";

lblSecondSell.Visible = true;

cmbFirstSell.Visible = true;

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

// Параметризованный запрос для безопасности и читаемости кода

SqlCommand typeCommand = new SqlCommand("SELECT typeGoodsID, typeGood FROM TypeGoods", conn);

SqlDataReader reader1 = typeCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["typeGoodsID"]), reader1["typeGood"].ToString())); // Исправлено название столбца

}

reader1.Close();

conn.Close();

cmbFirstSell.DataSource = lstCategories1;

cmbFirstSell.DisplayMember = "name"; // Установка отображаемого поля

cmbFirstSell.ValueMember = "id"; // Установка поля значения

cmbFirstSell.Text = "";

}

else if (boxItem == "ТипДоставки")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 20;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

btnAdd.Left = 20;

btnAdd.Top = 65;

btnExit.Left = 20;

btnExit.Top = 90;

lblSecondSell.Text = "Тип Доставки";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

}

else if (boxItem == "Пользователи")

{

lblFirstSell.Text = "ФИО";

lblSecondSell.Text = "Роли";

lblThirdSell.Text = "Логин";

lblFourthSell.Text = "Пароль";

lblFirstSell.Visible = true;

lblSecondSell.Visible = true;

lblThirdSell.Visible = true;

lblFourthSell.Visible = true;

txtFirstSell.Visible = true;

txtThirdSell.Visible = true;

txtFourthSell.Visible = true;

cmbSecondSell.Visible = true;

// roles

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand rolesCommand = new SqlCommand("Select RolesID, RolesName from Roles", conn);

SqlDataReader reader1 = rolesCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["RolesID"]), reader1["RolesName"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

cmbSecondSell.Text = "";

reader1.Close();

}

}

else if (boxItem == "Склады")

{

this.Width = 350;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 300;

btnAdd.Left = 20;

btnAdd.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "Адрес";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

}

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

// Получаем ссылку на форму 2

MainForm fm = Application.OpenForms["MainForm"] as MainForm;

if (fm != null)

{

// Показываем форму 2

fm.Show();

// Вызываем метод обновления данных на форме 2

fm.RefreshData();

}

}

private void comboBox5\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

public class Categories

{

public int id { get; set; }

public string name { get; set; }

public Categories(int id, string name)

{

this.id = id;

this.name = name;

}

}

}

## 9.4 Листинг формы редактирования

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Menu;

using System.Runtime.Remoting.Contexts;

using test\_subd.Properties;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Reflection.Emit;

using System.Runtime.Remoting.Messaging;

namespace test\_subd

{

public partial class EditForm : Form

{

SqlConnection connect;

int roleForm;

string boxItem;

string condition;

public EditForm(SqlConnection cnct, int rl, string item, string editID)

{

connect = cnct;

roleForm = rl;

boxItem = item;

condition = editID;

InitializeComponent();

}

private void Form4\_Load(object sender, EventArgs e)

{

lblNameTabel.Text = boxItem;

int user = 0;

if (connect.State == ConnectionState.Open)

{

connect.Close();

}

if (boxItem == "Товары")

{

txtThirdSell.Visible = true;

txtFourthSell.Visible = true;

txtFifthSell.Visible = true;

txtSixthSell.Visible = true;

cmbSecondSell.Visible = true;

cmbFirstSell.Visible = true;

cmbSixthSell.Visible = true;

lblFirstCell.Visible = true;

lblSecondSell.Visible = true;

lblThirdCell.Visible = true;

lblFourthCell.Visible = true;

lblFifthCell.Visible = true;

lblSixthCell.Visible = true;

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand logRequest = new SqlCommand("SELECT fgid, fgfirms FROM FirmsGoods", conn);

SqlDataReader rdr = logRequest.ExecuteReader();

List<Categories> lstCategories = new List<Categories>();

while (rdr.Read())

{

lstCategories.Add(new Categories(Convert.ToInt32(rdr["fgid"]), rdr["fgfirms"].ToString()));

}

cmbFirstSell.DataSource = lstCategories;

cmbFirstSell.DisplayMember = "name";

cmbFirstSell.ValueMember = "id";

rdr.Close();

SqlCommand typeGoodCommand = new SqlCommand("SELECT typeGoodsID, typeGood FROM TypeGoods", conn);

SqlDataReader reader1 = typeGoodCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["typeGoodsID"]), reader1["typeGood"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

reader1.Close();

SqlCommand warehousesCommand = new SqlCommand("SELECT warehouseID, warehouseAdres FROM Warehouses", conn);

reader1 = warehousesCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["warehouseID"]), reader1["warehouseAdres"].ToString()));

}

cmbSixthSell.DataSource = lstCategories1;

cmbSixthSell.DisplayMember = "name";

cmbSixthSell.ValueMember = "id";

reader1.Close();

string query = $"SELECT \* FROM Goods WHERE goodID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader5 = command.ExecuteReader();

if (reader5.Read())

{

textBox1.Text = reader5["goodID"].ToString();

cmbFirstSell.SelectedValue = reader5["firmGoodID"];

cmbSecondSell.SelectedValue = reader5["typeGoodID"];

cmbSixthSell.SelectedValue = reader5["warehouseID"];

txtThirdSell.Text = reader5["goodModel"].ToString();

txtFourthSell.Text = reader5["goodAmount"].ToString();

txtFifthSell.Text = reader5["goodSize"].ToString();

}

reader5.Close();

conn.Close();

}

}

else if (boxItem == "Фирмы")

{

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 20;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

btnEdit.Left = 20;

btnEdit.Top = 65;

btnExit.Left = 20;

btnExit.Top = 90;

lblSecondSell.Text = "Фирма";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

string query = $"SELECT \* FROM FirmsGoods WHERE fgid = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["fgid"].ToString();

txtSecondSell.Text = reader["fgfirms"].ToString();

}

reader.Close();

conn.Close();

}

else if (boxItem == "Водители")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 170;

btnEdit.Left = 20;

btnEdit.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

lblSecondSell.Text = "ФИО";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

string query = $"SELECT \* FROM EmployeeDriver WHERE employeesDriverID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["employeesDriverID"].ToString();

txtSecondSell.Text = reader["employeeDriverFIO"].ToString();

}

reader.Close();

conn.Close();

}

else if (boxItem == "Контейнер")

{

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 170;

btnEdit.Left = 20;

btnEdit.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "Контейнер";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

string query = $"SELECT \* FROM Container WHERE containerID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["containerID"].ToString();

txtSecondSell.Text = reader["containerType"].ToString();

}

reader.Close();

conn.Close();

}

else if (boxItem == "Доставки")

{

lblFirstCell.Text = "Заказ";

lblSecondSell.Text = "Водитель";

lblThirdCell.Text = "Транспорт";

lblFourthCell.Text = "Контейнер";

lblFifthCell.Text = "Пункт назначения";

txtFifthSell.Visible = true;

cmbFirstSell.Visible = true;

cmbSecondSell.Visible = true;

cmbThirdSell.Visible = true;

cmbFourthSell.Visible = true;

lblFirstCell.Visible = true;

lblSecondSell.Visible = true;

lblThirdCell.Visible = true;

lblFourthCell.Visible = true;

lblFifthCell.Visible = true;

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

// order

SqlCommand orderCommand = new SqlCommand("SELECT orderid FROM Orders", conn);

SqlDataReader reader1 = orderCommand.ExecuteReader();

if (reader1.HasRows)

{

while (reader1.Read())

{

cmbFirstSell.Items.Add(Convert.ToInt32(reader1["orderid"]));

}

}

reader1.Close();

// driver

SqlCommand employeeDriverCommand = new SqlCommand("SELECT employeesDriverID, employeeDriverFIO FROM EmployeeDriver", conn);

SqlDataReader reader2 = employeeDriverCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader2.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader2["employeesDriverID"]), reader2["employeeDriverFIO"].ToString()));

}

reader2.Close();

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

// transport

SqlCommand transportCommand = new SqlCommand("SELECT transportID, transport FROM Transport", conn);

SqlDataReader reader3 = transportCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader3.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader3["transportID"]), reader3["transport"].ToString()));

}

reader3.Close();

cmbThirdSell.DataSource = lstCategories1;

cmbThirdSell.DisplayMember = "name";

cmbThirdSell.ValueMember = "id";

// container

SqlCommand containerCommand = new SqlCommand("SELECT containerID, containerType FROM Container", conn);

SqlDataReader reader4 = containerCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader4.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader4["containerID"]), reader4["containerType"].ToString()));

}

reader4.Close();

cmbFourthSell.DataSource = lstCategories1;

cmbFourthSell.DisplayMember = "name";

cmbFourthSell.ValueMember = "id";

// Выполнение запроса для boxItem

string query = $"SELECT \* FROM Delivery WHERE deliveryID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader5 = command.ExecuteReader();

if (reader5.Read())

{

cmbFirstSell.SelectedItem = reader5["orderid"];

cmbSecondSell.SelectedValue = reader5["employeesDriverID"];

cmbThirdSell.SelectedValue = reader5["transportID"];

cmbFourthSell.SelectedValue = reader5["containerID"];

txtFifthSell.Text = reader5["destination"].ToString();

}

reader5.Close();

conn.Close();

}

}

else if (boxItem == "Заказы")

{

lblFirstCell.Text = "Менеджер";

lblSecondSell.Text = "Клиент";

lblThirdCell.Text = "goodid";

lblFourthCell.Text = "orderamount";

lblFirstCell.Visible = true;

lblSecondSell.Visible = true;

lblThirdCell.Visible = true;

lblFourthCell.Visible = true;

txtFourthSell.Visible = true;

cmbFirstSell.Visible = true;

cmbSecondSell.Visible = true;

cmbThirdSell.Visible = true;

// employee

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand employeeCommand = new SqlCommand("SELECT userID, userFIO FROM Users WHERE roles IN (1,2,5,8,9)", conn);

SqlDataReader reader1 = employeeCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["userID"]), reader1["userFIO"].ToString()));

}

cmbFirstSell.DataSource = lstCategories1;

cmbFirstSell.DisplayMember = "name"; // Отображаем имя пользователя

cmbFirstSell.ValueMember = "id"; // Устанавливаем значение пользователя как идентификатор

reader1.Close();

// customer

SqlCommand customerCommand = new SqlCommand("Select userID, userFIO from Users where roles in (3,4,6,7,10)", conn);

reader1 = customerCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["userID"]), reader1["userFIO"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

reader1.Close();

// good

SqlCommand goodCommand = new SqlCommand("Select goodID, goodModel from Goods", conn);

reader1 = goodCommand.ExecuteReader();

lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["goodID"]), reader1["goodModel"].ToString()));

}

cmbThirdSell.DataSource = lstCategories1;

cmbThirdSell.DisplayMember = "name";

cmbThirdSell.ValueMember = "id";

reader1.Close();

// Выполнение запроса для boxItem

string query = $"SELECT \* FROM Orders WHERE orderid = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader5 = command.ExecuteReader();

if (reader5.Read())

{

textBox1.Text = reader5["orderid"].ToString();

cmbFirstSell.SelectedValue = reader5["employeeid"];

cmbSecondSell.SelectedValue = reader5["customerid"];

cmbThirdSell.SelectedValue = reader5["goodid"];

txtFourthSell.Text = reader5["orderamount"].ToString();

}

reader5.Close();

conn.Close();

}

}

else if (boxItem == "Роли")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 170;

btnEdit.Left = 20;

btnEdit.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

lblSecondSell.Text = "Роли";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

string query = $"SELECT \* FROM Roles WHERE RolesID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["RolesID"].ToString();

txtSecondSell.Text = reader["RolesName"].ToString();

}

reader.Close();

conn.Close();

}

else if (boxItem == "Транспорт")

{

this.Height = 200;

btnEdit.Left = 150;

btnEdit.Top = 90;

btnExit.Left = 150;

btnExit.Top = 115;

lblFirstCell.Text = "Тип доставки";

lblSecondSell.Text = "Транспорт";

lblFirstCell.Visible = true;

lblSecondSell.Visible = true;

cmbFirstSell.Visible = true;

txtSecondSell.Visible = true;

// type

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand typeCommand = new SqlCommand("Select typeDeliveryID, typeDeliveryVehicle from Typeofdelivery", conn);

SqlDataReader reader1 = typeCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["typeDeliveryID"]), reader1["typeDeliveryVehicle"].ToString()));

}

cmbFirstSell.DataSource = lstCategories1;

cmbFirstSell.DisplayMember = "name";

cmbFirstSell.ValueMember = "id";

reader1.Close();

string query = $"SELECT \* FROM Transport WHERE transportID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader5 = command.ExecuteReader();

if (reader5.Read())

{

textBox1.Text = reader5["transportID"].ToString();

cmbFirstSell.SelectedValue = reader5["typeDeliveryID"];

txtSecondSell.Text = reader5["transport"].ToString();

}

reader5.Close();

conn.Close();

}

}

else if (boxItem == "ТипТовара")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 10;

lblSecondSell.Top = 35;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

cmbFirstSell.Width = 150;

btnEdit.Left = 20;

btnEdit.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

lblSecondSell.Text = "Тип товара";

lblSecondSell.Visible = true;

cmbFirstSell.Visible = true;

// Параметризованный запрос для безопасности и читаемости кода

SqlCommand typeCommand = new SqlCommand("SELECT typeGoodsID, typeGood FROM TypeGoods", conn);

SqlDataReader reader = typeCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader["typeGoodsID"]), reader["typeGood"].ToString())); // Исправлено название переменной

}

reader.Close();

cmbFirstSell.DataSource = lstCategories1;

cmbFirstSell.DisplayMember = "name"; // Установка отображаемого поля

cmbFirstSell.ValueMember = "id"; // Установка поля значения

SqlCommand typeCommand2 = new SqlCommand("SELECT \* FROM TypeGoods WHERE typeGoodsID = @condition", conn);

typeCommand2.Parameters.AddWithValue("@condition", condition);

SqlDataReader reader2 = typeCommand2.ExecuteReader();

if (reader2.Read())

{

textBox1.Text = reader2["typeGoodsID"].ToString();

cmbFirstSell.Text = reader2["typeGood"].ToString(); // Установка текста в комбобоксе, так как у вас необходимо выбрать значение по тексту, а не по идентификатору

}

reader2.Close();

conn.Close();

}

else if (boxItem == "ТипДоставки")

{

this.Width = 250;

this.Height = 200;

lblSecondSell.Left = 20;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

btnEdit.Left = 20;

btnEdit.Top = 65;

btnExit.Left = 20;

btnExit.Top = 90;

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

lblSecondSell.Text = "Тип доставки";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

string query = $"SELECT \* FROM Typeofdelivery WHERE typeDeliveryID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["typeDeliveryID"].ToString();

txtSecondSell.Text = reader["typeDeliveryVehicle"].ToString();

}

reader.Close();

conn.Close();

}

else if (boxItem == "Пользователи")

{

lblFirstCell.Text = "ФИО";

lblSecondSell.Text = "Роли";

lblThirdCell.Text = "Логин";

lblFourthCell.Text = "Пароль";

lblFirstCell.Visible = true;

lblSecondSell.Visible = true;

lblThirdCell.Visible = true;

lblFourthCell.Visible = true;

txtFirstCell.Visible = true;

txtThirdSell.Visible = true;

txtFourthSell.Visible = true;

cmbSecondSell.Visible = true;

// roles

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand rolesCommand = new SqlCommand("Select RolesID, RolesName from Roles", conn);

SqlDataReader reader1 = rolesCommand.ExecuteReader();

List<Categories> lstCategories1 = new List<Categories>();

while (reader1.Read())

{

lstCategories1.Add(new Categories(Convert.ToInt32(reader1["RolesID"]), reader1["RolesName"].ToString()));

}

cmbSecondSell.DataSource = lstCategories1;

cmbSecondSell.DisplayMember = "name";

cmbSecondSell.ValueMember = "id";

reader1.Close();

string query = $"SELECT \* FROM Users WHERE userID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader5 = command.ExecuteReader();

if (reader5.Read())

{

textBox1.Text = reader5["userID"].ToString();

cmbSecondSell.SelectedValue = reader5["roles"];

txtThirdSell.Text = reader5["loginRules"].ToString();

txtFourthSell.Text = reader5["passwordRules"].ToString();

txtFirstCell.Text = reader5["userFIO"].ToString();

}

reader5.Close();

conn.Close();

}

}

else if (boxItem == "Склады")

{

SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString);

conn.Open();

this.Width = 350;

this.Height = 200;

lblSecondSell.Left = 15;

lblSecondSell.Top = 25;

txtSecondSell.Left = 20;

txtSecondSell.Top = 40;

txtSecondSell.Width = 300;

btnEdit.Left = 20;

btnEdit.Top = 80;

btnExit.Left = 20;

btnExit.Top = 105;

lblSecondSell.Text = "Адрес";

lblSecondSell.Visible = true;

txtSecondSell.Visible = true;

string query = $"SELECT \* FROM Warehouses WHERE warehouseID = {condition}";

SqlCommand command = new SqlCommand(query, conn);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["warehouseID"].ToString();

txtSecondSell.Text = reader["warehouseAdres"].ToString();

}

reader.Close();

conn.Close();

}

}

private void btnEdit\_Click(object sender, EventArgs e)

{

using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.connectionString))

{

conn.Open();

SqlCommand logRequest = new SqlCommand();

logRequest.Connection = conn;

if (boxItem == "Товары")

{

logRequest.CommandText = $"Update Goods set firmGoodID = {cmbFirstSell.SelectedIndex + 1}, typeGoodID = '{cmbSecondSell.SelectedIndex + 1}'," +

$" goodModel = '{txtThirdSell.Text}', goodAmount = {txtFourthSell.Text}, goodSize = {txtFifthSell.Text}, warehouseID = {cmbSixthSell.SelectedIndex + 1} where goodID = {textBox1.Text}";

}

else if (boxItem == "Фирмы")

{

logRequest.CommandText = $"Update [FirmsGoods] set fgfirms = '{txtSecondSell.Text}' where fgid = {textBox1.Text}";

}

else if (boxItem == "Водители")

{

logRequest.CommandText = $"Update [EmployeeDriver] set employeeDriverFIO = '{txtSecondSell.Text}' where employeeDriverID = {textBox1.Text}";

}

else if (boxItem == "Контейнер")

{

logRequest.CommandText = $"Update [Container] set containerType = '{txtSecondSell.Text}' where containerID = {textBox1.Text}";

}

else if (boxItem == "Доставки")

{

logRequest.CommandText = $"Update [Delivery] set orderID = {cmbFirstSell.SelectedIndex + 1}, employeesDriverID = {cmbSecondSell.SelectedIndex + 1}," +

$" transportID = {cmbThirdSell.SelectedIndex + 1}, containerID = {cmbFourthSell.SelectedIndex + 1}, " +

$" destination = '{txtFifthSell.Text}' where deliveryID = {textBox1.Text})";

}

else if (boxItem == "Заказы")

{

int employeeId = (int)cmbFirstSell.SelectedValue;

int customerId = (int)cmbSecondSell.SelectedValue;

int goodId = (int)cmbThirdSell.SelectedValue;

logRequest.CommandText = $"UPDATE [Orders] SET employeeid = {employeeId}, " +

$"customerid = {customerId}, " +

$"goodid = {goodId}, " +

$"orderamount = {txtFourthSell.Text} " +

$"WHERE orderid = {textBox1.Text}";

}

else if (boxItem == "Роли")

{

logRequest.CommandText = $"Update [Roles] set RolesName = '{txtSecondSell.Text}' where RolesID = {textBox1.Text}";

}

else if (boxItem == "Транспорт")

{

logRequest.CommandText = $"Update [Transport] set typeDeliveryID = {cmbFirstSell.SelectedIndex + 1}, transport = '{txtSecondSell.Text}'" +

$"where transportID = {textBox1.Text}";

}

else if (boxItem == "ТипТовара")

{

logRequest.CommandText = $"Update [TypeGoods] set typeGood = '{cmbSecondSell.SelectedIndex + 1}'" +

$"where typeGoodsID = {textBox1.Text}";

}

else if (boxItem == "ТипДоставки")

{

logRequest.CommandText = $"Update [Typeofdelivery] set typeDeliveryVehicle) = '{txtSecondSell.Text}'" +

$"where typeDeliveryID = {textBox1.Text}";

}

else if (boxItem == "Пользователи")

{

logRequest.CommandText = $"Update [Users] set userFIO = '{txtFirstCell.Text}'," +

$" roles = {cmbSecondSell.SelectedIndex + 1}, loginRules = '{txtFourthSell.Text}', passwordRules = '{txtFifthSell.Text}' " +

$"where userID = {textBox1.Text}";

}

else if (boxItem == "Склады")

{

logRequest.CommandText = $"Update [Warehouses] set warehouseAdres = '{txtSecondSell.Text}'" +

$"where warehouseID = {textBox1.Text}";

}

logRequest.ExecuteNonQuery();

MessageBox.Show("Запись успешно отредактирована");

}

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

// Получаем ссылку на форму 2

MainForm fm = Application.OpenForms["MainForm"] as MainForm;

if (fm != null)

{

// Показываем форму 2

fm.Show();

// Вызываем метод обновления данных на форме 2

fm.RefreshData();

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void comboBox5\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

## 9.5 Листинг формы диалогового окна добавления заказа

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Menu;

using System.Runtime.Remoting.Contexts;

using test\_subd.Properties;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Reflection.Emit;

using System.Runtime.Remoting.Messaging;

using System.Xml;

namespace test\_subd

{

public partial class InputDialog : Form

{

public string EnteredText { get; private set; }

public InputDialog(string labelText)

{

InitializeComponent();

label3.Text = labelText;

}

private void btnOK\_Click(object sender, EventArgs e)

{

EnteredText = txtAmount.Text;

DialogResult = DialogResult.OK;

}

private void InitializeComponent()

{

this.txtAmount = new System.Windows.Forms.TextBox();

this.label3 = new System.Windows.Forms.Label();

this.btnOK = new System.Windows.Forms.Button();

this.btnCancel = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// txtAmount

//

this.txtAmount.Location = new System.Drawing.Point(13, 28);

this.txtAmount.Name = "txtAmount";

this.txtAmount.Size = new System.Drawing.Size(155, 20);

this.txtAmount.TabIndex = 2;

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(10, 12);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(66, 13);

this.label3.TabIndex = 5;

this.label3.Text = "Количество";

//

// btnOK

//

this.btnOK.Location = new System.Drawing.Point(13, 54);

this.btnOK.Name = "btnOK";

this.btnOK.Size = new System.Drawing.Size(75, 23);

this.btnOK.TabIndex = 6;

this.btnOK.Text = "Внести";

this.btnOK.UseVisualStyleBackColor = true;

this.btnOK.Click += new System.EventHandler(this.btnOK\_Click);

//

// btnCancel

//

this.btnCancel.Location = new System.Drawing.Point(93, 54);

this.btnCancel.Name = "btnCancel";

this.btnCancel.Size = new System.Drawing.Size(75, 23);

this.btnCancel.TabIndex = 7;

this.btnCancel.Text = "Отмена";

this.btnCancel.UseVisualStyleBackColor = true;

this.btnCancel.Click += new System.EventHandler(this.btnCancel\_Click);

//

// InputDialog

//

this.ClientSize = new System.Drawing.Size(187, 123);

this.Controls.Add(this.btnCancel);

this.Controls.Add(this.btnOK);

this.Controls.Add(this.label3);

this.Controls.Add(this.txtAmount);

this.Name = "InputDialog";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Оформление заказа";

this.Load += new System.EventHandler(this.DoOrder\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

private void DoOrder\_Load(object sender, EventArgs e)

{

}

private void btnCancel\_Click(object sender, EventArgs e)

{

{

DialogResult = DialogResult.Cancel;

}

}

// Здесь может быть ваш код

}

}

## 9.6 Листинг формы просмотра заказов пользователя

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Menu;

using System.Runtime.Remoting.Contexts;

using test\_subd.Properties;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Reflection.Emit;

using System.Runtime.Remoting.Messaging;

namespace test\_subd

{

public partial class CheckOrder : Form

{

int userID;

private SqlConnection connect;

public CheckOrder(int uID, SqlConnection cnct)

{

InitializeComponent();

userID = uID;

connect = cnct;

}

private void InitializeComponent()

{

this.dataGridView1 = new System.Windows.Forms.DataGridView();

this.btnCancel = new System.Windows.Forms.Button();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();

this.SuspendLayout();

//

// dataGridView1

//

this.dataGridView1.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.dataGridView1.Location = new System.Drawing.Point(12, 41);

this.dataGridView1.Name = "dataGridView1";

this.dataGridView1.Size = new System.Drawing.Size(407, 251);

this.dataGridView1.TabIndex = 0;

this.dataGridView1.CellContentClick += new System.Windows.Forms.DataGridViewCellEventHandler(this.dataGridView1\_CellContentClick);

//

// btnCancel

//

this.btnCancel.Location = new System.Drawing.Point(344, 298);

this.btnCancel.Name = "btnCancel";

this.btnCancel.Size = new System.Drawing.Size(75, 23);

this.btnCancel.TabIndex = 1;

this.btnCancel.Text = "Назад";

this.btnCancel.UseVisualStyleBackColor = true;

this.btnCancel.Click += new System.EventHandler(this.btnCancel\_Click);

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 14.25F, ((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Italic))), System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label1.Location = new System.Drawing.Point(224, 14);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(125, 24);

this.label1.TabIndex = 2;

this.label1.Text = "Мои заказы";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 14.25F, ((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Italic))), System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label2.Location = new System.Drawing.Point(12, 14);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(217, 24);

this.label2.TabIndex = 3;

this.label2.Text = "Количество заказов:";

//

// CheckOrder

//

this.ClientSize = new System.Drawing.Size(433, 327);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.btnCancel);

this.Controls.Add(this.dataGridView1);

this.Name = "CheckOrder";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Мои заказы";

this.Load += new System.EventHandler(this.CheckOrder\_Load);

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

private void CheckOrder\_Load(object sender, EventArgs e)

{

using (SqlConnection sqlConnect = new SqlConnection(Properties.Settings.Default.connectionString))

{

SqlCommand command = new SqlCommand("GetOrdersByCustomerID", sqlConnect);

command.CommandType = CommandType.StoredProcedure;

// Добавляем параметр для передачи customerID в процедуру

command.Parameters.AddWithValue("@CustomerID", userID);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet dataSet = new DataSet();

try

{

sqlConnect.Open();

adapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

label1.Text = Convert.ToString(dataSet.Tables[0].Rows.Count); //считаем строчки

}

catch (Exception ex)

{

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

}

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

this.Close();

Form fm = Application.OpenForms["MainForm"];

fm.Show();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

# **Приложение В (обязательное). Программный код создания БД**

USE [master]

GO

/\*\*\*\*\*\* Object: Database [plswwww] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

CREATE DATABASE [plswwww]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'plswwww', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\plswwww.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'plswwww\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\plswwww\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

WITH CATALOG\_COLLATION = DATABASE\_DEFAULT, LEDGER = OFF

GO

ALTER DATABASE [plswwww] SET COMPATIBILITY\_LEVEL = 160

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

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

end

GO

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

GO

ALTER DATABASE [plswwww] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [plswwww] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [plswwww] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [plswwww] SET ARITHABORT OFF

GO

ALTER DATABASE [plswwww] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [plswwww] SET AUTO\_SHRINK OFF

GO

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

GO

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

GO

ALTER DATABASE [plswwww] SET CURSOR\_DEFAULT GLOBAL

GO

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

GO

ALTER DATABASE [plswwww] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [plswwww] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [plswwww] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [plswwww] SET ENABLE\_BROKER

GO

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

GO

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

GO

ALTER DATABASE [plswwww] SET TRUSTWORTHY OFF

GO

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

GO

ALTER DATABASE [plswwww] SET PARAMETERIZATION SIMPLE

GO

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

GO

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

GO

ALTER DATABASE [plswwww] SET RECOVERY FULL

GO

ALTER DATABASE [plswwww] SET MULTI\_USER

GO

ALTER DATABASE [plswwww] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [plswwww] SET DB\_CHAINING OFF

GO

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

GO

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

GO

ALTER DATABASE [plswwww] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [plswwww] SET ACCELERATED\_DATABASE\_RECOVERY = OFF

GO

EXEC sys.sp\_db\_vardecimal\_storage\_format N'plswwww', N'ON'

GO

ALTER DATABASE [plswwww] SET QUERY\_STORE = ON

GO

ALTER DATABASE [plswwww] SET QUERY\_STORE (OPERATION\_MODE = READ\_WRITE, CLEANUP\_POLICY = (STALE\_QUERY\_THRESHOLD\_DAYS = 30), DATA\_FLUSH\_INTERVAL\_SECONDS = 900, INTERVAL\_LENGTH\_MINUTES = 60, MAX\_STORAGE\_SIZE\_MB = 1000, QUERY\_CAPTURE\_MODE = AUTO, SIZE\_BASED\_CLEANUP\_MODE = AUTO, MAX\_PLANS\_PER\_QUERY = 200, WAIT\_STATS\_CAPTURE\_MODE = ON)

GO

USE [plswwww]

GO

/\*\*\*\*\*\* Object: Table [dbo].[EmployeeDriver] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[EmployeeDriver](

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

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

CONSTRAINT [PK\_\_Employee\_\_39F40C22E521ED6C] PRIMARY KEY CLUSTERED

(

[employeesDriverID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Container] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Container](

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

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

CONSTRAINT [PK\_\_Containe\_\_F10B29B0F8E8D574] PRIMARY KEY CLUSTERED

(

[containerID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Delivery] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Delivery](

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

[orderID] [int] NOT NULL,

[employeesDriverID] [int] NOT NULL,

[transportID] [int] NOT NULL,

[containerID] [int] NOT NULL,

[destination] [varchar](150) NOT NULL,

CONSTRAINT [PK\_\_Delivery\_\_CDDCBC2A2BDD3ECA] PRIMARY KEY CLUSTERED

(

[deliveryID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Transport] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Transport](

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

[typeDeliveryID] [int] NOT NULL,

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

CONSTRAINT [PK\_\_Transpor\_\_59165AC79C19E90D] PRIMARY KEY CLUSTERED

(

[transportID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Доставки] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE view [dbo].[\_Доставки] as

select deliveryID as "ID", orderID as "ID заказа", employeeDriverFIO as "Водитель",

t.transport as "Транспорт", c.containerType as "Контейнер", destination as "Пункт назначения"

from Delivery d

Join EmployeeDriver e on d.employeesDriverID = e.employeesDriverID

JOIN Transport t on d.transportID = t.transportID

JOIN Container c on d.containerID = c.containerID

GO

/\*\*\*\*\*\* Object: Table [dbo].[FirmsGoods] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[FirmsGoods](

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

[fgfirms] [varchar](25) NOT NULL,

CONSTRAINT [PK\_\_FirmsGoo\_\_37F1A865780B73CC] PRIMARY KEY CLUSTERED

(

[fgid] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Goods] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Goods](

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

[firmGoodID] [int] NOT NULL,

[typeGoodID] [int] NOT NULL,

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

[goodAmount] [int] NOT NULL,

[goodSize] [float] NOT NULL,

[warehouseID] [int] NOT NULL,

CONSTRAINT [PK\_\_Goods\_\_66339DED055BD00A] PRIMARY KEY CLUSTERED

(

[goodID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_goodss] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create VIEW [dbo].[\_goodss] AS

SELECT goodModel as "Наименование", goodSize as "Размер", fgfirms as "Фирма"

FROM Goods g

JOIN FirmsGoods f

ON g.firmGoodID = f.fgid

GO

/\*\*\*\*\*\* Object: Table [dbo].[Orders] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Orders](

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

[employeeid] [int] NOT NULL,

[customerid] [int] NOT NULL,

[goodid] [int] NOT NULL,

[orderamount] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[orderid] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Users](

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

[userFIO] [varchar](50) NULL,

[roles] [int] NULL,

[loginRules] [varchar](50) NULL,

[passwordRules] [varchar](50) NULL,

CONSTRAINT [PK\_\_Users\_\_CB9A1CDFE6CE9A22] PRIMARY KEY CLUSTERED

(

[userID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Заказы] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE view [dbo].[\_Заказы] as

select orderID as "ID", u.userFIO as "Сотрудник", s.userFIO as "Клиент",

goodModel as "Товар", orderamount as "Количество товара"

from Orders o

Join Users u on o.employeeid = u.userID

Join Users s on o.customerid = s.userID

JOIN Goods g on o.goodid = g.goodID

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Водители] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE view [dbo].[\_Водители] as

select employeesDriverID as "ID", employeeDriverFIO as "ФИО" from EmployeeDriver

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Фирмы] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_Фирмы] as

select fgid as "ID", fgfirms as "Фирма" from FirmsGoods

GO

/\*\*\*\*\*\* Object: Table [dbo].[Roles] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Roles](

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

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

CONSTRAINT [PK\_\_Employee\_\_C4B278205646A5AA] PRIMARY KEY CLUSTERED

(

[RolesID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Роли] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_Роли] as

select RolesID as "ID", RolesName as "Роль" from Roles

GO

/\*\*\*\*\*\* Object: Table [dbo].[TypeGoods] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[TypeGoods](

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

[typeGood] [varchar](100) NOT NULL,

CONSTRAINT [PK\_\_TypeGood\_\_E7779FDC002DDD95] PRIMARY KEY CLUSTERED

(

[typeGoodsID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_ТипТовара] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_ТипТовара] as

select typeGoodsID as "ID", typeGood as "Роль" from TypeGoods

GO

/\*\*\*\*\*\* Object: Table [dbo].[Typeofdelivery] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Typeofdelivery](

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

[typeDeliveryVehicle] [varchar](100) NOT NULL,

CONSTRAINT [PK\_\_Typeofde\_\_CC6F5679DBEE88B1] PRIMARY KEY CLUSTERED

(

[typeDeliveryID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_ТипДоставки] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_ТипДоставки] as

select typeDeliveryID as "ID", typeDeliveryVehicle as "Тип техники" from Typeofdelivery

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Пользователи] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_Пользователи] as

select userID as "ID", userFIO as "ФИО", RolesName as "Роль",

loginRules as "Логин", passwordRules as "Пароль"

from Users u

JOIN Roles r on u.roles = r.RolesID

GO

/\*\*\*\*\*\* Object: Table [dbo].[Warehouses] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Warehouses](

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

[warehouseAdres] [varchar](250) NOT NULL,

CONSTRAINT [PK\_\_Warehous\_\_86BA77FA373482E0] PRIMARY KEY CLUSTERED

(

[warehouseID] ASC

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

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Склады] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_Склады] as

select warehouseID as "ID", warehouseAdres as "Адрес" from Warehouses

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Товары] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_Товары] as

select goodID as "ID", fgfirms as "Фирма", typeGood "Тип товара",

goodModel as "Модель", goodAmount as "Количество", goodSize as "Размер", warehouseAdres as "Адрес хранения"

from Goods g

JOIN FirmsGoods f on g.firmGoodID = f.fgid

JOIN TypeGoods t on g.typeGoodID = t.typeGoodsID

JOIN Warehouses w on g.warehouseID = w.warehouseID

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Контейнер] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[\_Контейнер] as

select containerID as "ID", containerType as "Тип" from Container

GO

/\*\*\*\*\*\* Object: View [dbo].[\_Транспорт] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE view [dbo].[\_Транспорт] as

select t.transportID as "ID", tofd.typeDeliveryVehicle as "Тип доставки", t.transport as "Транспорт" from Transport t

JOIN Typeofdelivery tofd ON

t.typeDeliveryID = tofd.typeDeliveryID

GO

SET IDENTITY\_INSERT [dbo].[Container] ON

INSERT [dbo].[Container] ([containerID], [containerType]) VALUES (1, N'Малый')

INSERT [dbo].[Container] ([containerID], [containerType]) VALUES (2, N'Средний')

INSERT [dbo].[Container] ([containerID], [containerType]) VALUES (3, N'Большой')

SET IDENTITY\_INSERT [dbo].[Container] OFF

GO

SET IDENTITY\_INSERT [dbo].[Delivery] ON

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (1, 2, 3, 5, 3, N'Тульская область, город Мытищи, спуск Ломоносова, 63')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (2, 3, 2, 6, 3, N'Курская область, город Раменское, бульвар Ленина, 21')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (3, 6, 7, 4, 3, N'Московская область, город Павловский Посад, ул. Славы, 61')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (4, 7, 5, 3, 3, N'Кемеровская область, город Егорьевск, ул. Бухарестская, 27')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (5, 1, 7, 2, 3, N'Орловская область, город Одинцово, въезд Ладыгина, 98')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (6, 9, 6, 1, 2, N'Волгоградская область, город Сергиев Посад, ул. Будапештсткая, 89')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (7, 8, 3, 7, 3, N'Архангельская область, город Луховицы, бульвар Чехова, 28')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (8, 10, 1, 8, 1, N'Владимирская область, город Серпухов, ул. Ленина, 03')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (9, 4, 4, 9, 2, N'Свердловская область, город Сергиев Посад, шоссе 1905 года, 03')

INSERT [dbo].[Delivery] ([deliveryID], [orderID], [employeesDriverID], [transportID], [containerID], [destination]) VALUES (10, 5, 3, 10, 3, N'Калининградская область, город Лотошино, наб. Ломоносова, 93')

SET IDENTITY\_INSERT [dbo].[Delivery] OFF

GO

SET IDENTITY\_INSERT [dbo].[EmployeeDriver] ON

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (1, N'Хохлов Адам Максович')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (2, N'Рябов Виталий Адольфович')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (3, N'Воронцов Иосиф Игнатьевич')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (4, N'Мышкин Геннадий Давидович')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (5, N'Иванов Руслан Ефимович')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (6, N'Комаров Игорь Михаилович')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (7, N'Мухин Леонтий Иринеевич')

INSERT [dbo].[EmployeeDriver] ([employeesDriverID], [employeeDriverFIO]) VALUES (8, N'Селиверстов Гордий Семёнович')

SET IDENTITY\_INSERT [dbo].[EmployeeDriver] OFF

GO

SET IDENTITY\_INSERT [dbo].[FirmsGoods] ON

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (1, N'Bosch')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (2, N'Aceline')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (3, N'Zanussi')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (4, N'Samsung ')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (5, N'Сварог')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (6, N'MSI')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (7, N'DEXP')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (8, N'Philips')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (9, N'Xiaom')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (10, N'Marshall')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (11, N'Сокол')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (12, N'Бюрократ ')

INSERT [dbo].[FirmsGoods] ([fgid], [fgfirms]) VALUES (13, N'Edge')

SET IDENTITY\_INSERT [dbo].[FirmsGoods] OFF

GO

SET IDENTITY\_INSERT [dbo].[Goods] ON

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (1, 1, 1, N'Холодильник A', 15, 0, 1)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (2, 1, 1, N'Холодильник B', 13, 1, 1)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (3, 3, 3, N'Снегоуборщик электрический', 20, 1, 9)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (4, 3, 1, N'Водонагреватель', 40, 0, 6)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (5, 4, 5, N'Саундбар', 30, 0, 2)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (6, 5, 4, N'Сварочный аппарат', 40, 0, 2)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (7, 6, 6, N'Ноутбук GF76', 10, 0, 5)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (8, 6, 5, N'Проводная гарнитура Immerse', 6, 0, 7)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (9, 8, 2, N'Проектор', 4, 0, 8)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (10, 7, 2, N'Экран для проектора', 7, 0, 4)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (11, 13, 5, N'Автосабвуфер активный', 2, 2, 8)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (12, 8, 1, N'Кофемашина автоматическая', 15, 1, 9)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (13, 4, 6, N'Телевизор LED', 1, 3, 7)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (14, 9, 6, N'Телевизор LED', 40, 1, 3)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (15, 7, 2, N'Стойка с кронштейном', 50, 1, 6)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (16, 10, 5, N'Домашняя аудиосистема', 37, 1, 1)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (17, 2, 5, N'Микрофон', 60, 0, 5)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (18, 7, 2, N'Кресло офисное', 100, 0, 3)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (19, 11, 2, N'Стол компьютерный', 88, 0, 6)

INSERT [dbo].[Goods] ([goodID], [firmGoodID], [typeGoodID], [goodModel], [goodAmount], [goodSize], [warehouseID]) VALUES (20, 12, 2, N'Кресло офисное', 97, 0, 1)

SET IDENTITY\_INSERT [dbo].[Goods] OFF

GO

SET IDENTITY\_INSERT [dbo].[Orders] ON

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (1, 2, 4, 16, 15)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (2, 2, 10, 19, 15)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (3, 15, 19, 14, 13)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (4, 10, 20, 9, 15)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (5, 10, 20, 10, 15)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (6, 10, 29, 14, 30)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (7, 15, 28, 11, 7)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (8, 14, 35, 1, 20)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (9, 15, 17, 4, 17)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (10, 14, 15, 8, 24)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (11, 14, 15, 8, 1)

INSERT [dbo].[Orders] ([orderid], [employeeid], [customerid], [goodid], [orderamount]) VALUES (12, 8, 12, 5, 150)

SET IDENTITY\_INSERT [dbo].[Orders] OFF

GO

SET IDENTITY\_INSERT [dbo].[Roles] ON

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (1, N'Руководитель')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (2, N'Менеджер')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (3, N'Грузчик')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (4, N'Уборщик')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (5, N'Сисадмин')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (6, N'Специалист по охране труда')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (7, N'Бухгалтер')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (8, N'Заместитель руководителя по работе с сотрудниками')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (9, N'Заместитель руководителя по закупкам')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (10, N'Пользователь')

INSERT [dbo].[Roles] ([RolesID], [RolesName]) VALUES (12, N'Типчик')

SET IDENTITY\_INSERT [dbo].[Roles] OFF

GO

SET IDENTITY\_INSERT [dbo].[Transport] ON

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (1, 1, N'Ford Transit')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (2, 1, N'Ford Transit')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (3, 1, N'Самолет')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (4, 3, N'Самолет')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (5, 2, N'Корабль')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (6, 2, N'Корабль')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (7, 2, N'Корабль')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (8, 1, N'Fiat')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (9, 1, N'KAMAZ')

INSERT [dbo].[Transport] ([transportID], [typeDeliveryID], [transport]) VALUES (10, 1, N'KAMAZ')

SET IDENTITY\_INSERT [dbo].[Transport] OFF

GO

SET IDENTITY\_INSERT [dbo].[TypeGoods] ON

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (1, N'Бытовая техника')

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (2, N'В офис')

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (3, N'К зиме')

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (4, N'Инструменты')

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (5, N'Звук')

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (6, N'Цифровая техника')

INSERT [dbo].[TypeGoods] ([typeGoodsID], [typeGood]) VALUES (7, N'Лето')

SET IDENTITY\_INSERT [dbo].[TypeGoods] OFF

GO

SET IDENTITY\_INSERT [dbo].[Typeofdelivery] ON

INSERT [dbo].[Typeofdelivery] ([typeDeliveryID], [typeDeliveryVehicle]) VALUES (1, N'Наземный')

INSERT [dbo].[Typeofdelivery] ([typeDeliveryID], [typeDeliveryVehicle]) VALUES (2, N'Морской')

INSERT [dbo].[Typeofdelivery] ([typeDeliveryID], [typeDeliveryVehicle]) VALUES (3, N'Воздушный')

SET IDENTITY\_INSERT [dbo].[Typeofdelivery] OFF

GO

SET IDENTITY\_INSERT [dbo].[Users] ON

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (1, N'Гуляева Нина Митрофановна', 1, N'TopLogin1', N'TopPassword')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (2, N'Волкова Алика Мэлоровна', 2, N'TopLogin2', N'TopPassword2')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (3, N'Родионова Карина Георгиевна', 3, N'TopLogin3', N'TopPassword3')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (4, N'Лебедева Анжела Васильевна', 4, N'TopLogin4', N'TopPassword4')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (5, N'Симонова Юлия Геласьевна', 5, N'TopLogin5', N'TopPassword5')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (6, N'Павлова Ульяна Матвеевнач', 6, N'TopLogin6', N'TopPassword7')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (7, N'Гаврилова Рая Альвиановна', 7, N'TopLogin7', N'TopPassword7')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (8, N'Тарасов Леонид Олегович', 8, N'TopLogin8', N'TopPassword8')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (9, N'Харитонов Юстин Матвеевич', 9, N'TopLogin9', N'TopPassword9')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (10, N'Зайцев Валерий Наумович', 1, N'TopLogin10', N'TopPassword10')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (11, N'Бобров Архип Владленович', 2, N'TopLogin11', N'TopPassword11')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (12, N'Жуков Пантелей Ильяович', 3, N'TopLogin12', N'TopPassword12')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (13, N'Романов Максим Тимурович', 4, N'TopLogin13', N'TopPassword13')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (14, N'Туров Эльдар Михайлович', 5, N'TopLogin14', N'TopPassword14')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (15, N'Мартынов Герасим Давидович', 6, N'TopLogin15', N'TopPassword15')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (16, N'Потапов Эдуард Робертович', 10, N'TopLogin16', N'TopPassword16')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (17, N'Соболев Олег Артёмович', 10, N'TopLogin17', N'TopPassword17')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (18, N'Бобров Артур Артемович', 10, N'TopLogin18', N'TopPassword18')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (19, N'Никифоров Анатолий Константинович', 10, N'TopLogin19', N'TopPassword19')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (20, N'Макаров Альберт Миронович', 10, N'TopLogin20', N'TopPassword20')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (21, N'Фадеев Мечислав Всеволодович', 10, N'TopLogin21', N'TopPassword21')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (22, N'Рожков Влас Пётрович', 10, N'TopLogin22', N'TopPassword22')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (23, N'Гаврилова Рая Альвиановна', 10, N'TopLogin23', N'TopPassword23')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (24, N'Кузьмин Степан Геннадьевич', 10, N'TopLogin24', N'TopPassword24')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (25, N'Дементьев Мартин Эльдарович', 10, N'TopLogin25', N'TopPassword25')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (26, N'Стрелкова Марина Донатовна', 10, N'TopLogin26', N'TopPassword26')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (27, N'Лукина Лилия Владленовна', 10, N'TopLogin27', N'TopPassword27')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (28, N'Родионов Андрей Валерьянович', 10, N'TopLogin28', N'TopPassword28')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (29, N'Журавлёв Богдан Борисович', 10, N'TopLogin29', N'TopPassword29')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (30, N'Григорьев Игнат Федосеевич', 10, N'TopLogin30', N'TopPassword30')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (31, N'Брагина Божена Егоровна', 10, N'TopLogin31', N'TopPassword31')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (32, N'Селезнёва Дарья Евсеевна', 10, N'TopLogin32', N'TopPassword32')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (33, N'Мухина Эрида Евгеньевна', 10, N'TopLogin33', N'TopPassword33')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (34, N'Брагин Владлен Алексеевич', 10, N'TopLogin34', N'TopPassword34')

INSERT [dbo].[Users] ([userID], [userFIO], [roles], [loginRules], [passwordRules]) VALUES (35, N'Суворов Степан Ефимович', 10, N'TopLogin35', N'TopPassword35')

SET IDENTITY\_INSERT [dbo].[Users] OFF

GO

SET IDENTITY\_INSERT [dbo].[Warehouses] ON

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (1, N'Ростовская область, город Воскресенск, пер. Ломоносова, 15')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (2, N'Саратовская область, город Раменское, пр. Ломоносова, 20')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (3, N'Липецкая область, город Луховицы, пер. Славы, 5')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (4, N'Магаданская область, город Солнечногорск, пер. Домодедовская, 59')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (5, N'Тюменская область, город Лотошино, пер. Сталина, 72')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (6, N'Оренбургская область, город Дмитров, ул. Гагарина, 31')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (7, N'Московская область, город Щёлково, ул. Славы, 40')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (8, N'Ивановская область, город Чехов, шоссе Славы, 02')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (9, N'Калининградская область, город Клин, пр. Космонавтов, 43')

INSERT [dbo].[Warehouses] ([warehouseID], [warehouseAdres]) VALUES (10, N'Амурская область, город Красногорск, ул. 1905 года, 67')

SET IDENTITY\_INSERT [dbo].[Warehouses] OFF

GO

ALTER TABLE [dbo].[Delivery] WITH CHECK ADD CONSTRAINT [Delivery\_fk0] FOREIGN KEY([orderID])

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

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Delivery] CHECK CONSTRAINT [Delivery\_fk0]

GO

ALTER TABLE [dbo].[Delivery] WITH CHECK ADD CONSTRAINT [Delivery\_fk2] FOREIGN KEY([transportID])

REFERENCES [dbo].[Transport] ([transportID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Delivery] CHECK CONSTRAINT [Delivery\_fk2]

GO

ALTER TABLE [dbo].[Delivery] WITH CHECK ADD CONSTRAINT [Delivery\_fk3] FOREIGN KEY([containerID])

REFERENCES [dbo].[Container] ([containerID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Delivery] CHECK CONSTRAINT [Delivery\_fk3]

GO

ALTER TABLE [dbo].[Goods] WITH CHECK ADD CONSTRAINT [Goods\_fk0] FOREIGN KEY([firmGoodID])

REFERENCES [dbo].[FirmsGoods] ([fgid])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Goods] CHECK CONSTRAINT [Goods\_fk0]

GO

ALTER TABLE [dbo].[Goods] WITH CHECK ADD CONSTRAINT [Goods\_fk1] FOREIGN KEY([typeGoodID])

REFERENCES [dbo].[TypeGoods] ([typeGoodsID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Goods] CHECK CONSTRAINT [Goods\_fk1]

GO

ALTER TABLE [dbo].[Goods] WITH CHECK ADD CONSTRAINT [Goods\_fk2] FOREIGN KEY([warehouseID])

REFERENCES [dbo].[Warehouses] ([warehouseID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Goods] CHECK CONSTRAINT [Goods\_fk2]

GO

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_Goods] FOREIGN KEY([goodid])

REFERENCES [dbo].[Goods] ([goodID])

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_Goods]

GO

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_Users] FOREIGN KEY([employeeid])

REFERENCES [dbo].[Users] ([userID])

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_Users]

GO

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_Users1] FOREIGN KEY([customerid])

REFERENCES [dbo].[Users] ([userID])

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_Users1]

GO

ALTER TABLE [dbo].[Transport] WITH CHECK ADD CONSTRAINT [Transport\_fk0] FOREIGN KEY([typeDeliveryID])

REFERENCES [dbo].[Typeofdelivery] ([typeDeliveryID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Transport] CHECK CONSTRAINT [Transport\_fk0]

GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT [FK\_Users\_Roles] FOREIGN KEY([roles])

REFERENCES [dbo].[Roles] ([RolesID])

GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK\_Users\_Roles]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[addcustomer] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[addcustomer] @nam char(100), @company char(100) as

declare @nd int

set @nd = (select max(customerid) from dbo.Customers) + 1

insert into dbo.Customers(customerid, customerfio, customercompany)

values (@nd, @nam, @company)

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[addemployee] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[addemployee] @nam char(100), @post char(100) as

declare @nd int

set @nd = (select max(employeeid) from dbo.Employees) + 1

insert into dbo.Employees(employeeid, employeefio, employeepost)

values (@nd, @nam, @post)

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[addfirmgood] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[addfirmgood] @firm char(100) as

declare @nd int

set @nd = (select max(fgid) from dbo.FirmsGoods) + 1

insert into dbo.FirmsGoods(fgid, fgfirms)

values (@nd, @firm)

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[addorder] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[addorder] @employee int, @customer int, @goodid int, @amount int as

declare @nd int

declare @haves int

set @nd = (select max(orderid) from dbo.Orders) + 1

set @haves = (select goodamount from dbo.Goods where goodid = @goodid)

if (@haves - @amount < 0)

begin

return 'Не хватает товара для заказа';

end

else

begin

declare @minus int

set @minus = (@haves - @amount)

update Goods Set goodamount=@minus

where goodid=@goodid

insert into dbo.Orders(orderid, employeeid, customerid, goodid, orderamount)

values (@nd, @employee, @customer, @goodid, @amount)

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[GetOrdersByCustomerID] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[GetOrdersByCustomerID]

@CustomerID INT

AS

BEGIN

SELECT

g.Фирма AS 'Фирма',

g.Модель AS 'Модель',

o.orderamount as 'Количество'

FROM

Orders o

JOIN

Users u ON o.customerid = u.userID

JOIN

\_Товары g ON o.goodid = g.ID

WHERE

o.customerid = @CustomerID;

END;

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[showgoods] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create proc [dbo].[showgoods] as

select \* from Goods

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[updateamountgood] Script Date: 14.05.2024 17:52:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[updateamountgood] @goodid int, @goodamount int as

UPDATE Goods Set goodamount=@goodamount

Where goodid = @goodid

GO

USE [master]

GO

ALTER DATABASE [plswwww] SET READ\_WRITE

GO