Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

К защите допустить:

Заведующий кафедрой ИСиТ

_____А.И. Парамонов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ РЕАЛИЗАЦИИ КОМПЬЮТЕРНЫХ КОМПЛЕКТУЮЩИХ

БГУИР ДП 1-40 01 01 01 041 ПЗ

Студент	Е.А. Гроднов
Руководитель	А.С. Шелягович
Консультанты:	
от кафедры ИСиТ	А.С. Шелягович
по экономической части	В.Г. Горовой
Нормоконтролер	О.А. Шведова
Рецензент	

Минск 2025

Р Е Ф Е Р А Т

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ РЕАЛИЗАЦИИ КОМПЬЮТЕРНЫХ КОМПЛЕКТУЮЩИХ: дипломный проект / Е.А. Гроднов – Минск: БГУИР, 2025, – п.з. – 117 с., чертежей (плакатов) – 6 л. формата А1.

Объектом проектирования является программное средство для реализации компьютерных комплектующих.

Целью данного дипломного проекта является разработка программного средства, которое поможет решить проблему реализации компьютерных комплектующих.

Для решения данной проблемы был разработан комплексный подход, который включает в себя: механизм подготовки, обработки и загрузки данных о товарах, категориях товаров, сотрудниках, алгоритмы ввода данных по накладным, оформления реализации, формирования отчетов и товарного чека.

В процессе работы над проектом были изучены различные подходы к реализации программных средств, проведен анализ уже имеющихся программных средств, которые в той или иной степени реализуют требуемый функционал. Разработана и реализована программная архитектура с возможностью дальнейшего масштабирования, были подобраны подходящие технологии, такие как база данных, а также язык программирования, на котором быстро и легко использовать и разрабатывать на данных технологиях.

Помимо этого, были разработаны диаграммы компонентов и развертывания, построены use-case диаграмма, разработана er-модель базы данных и функциональная модель (в нотации IDEF0) предметной области.

В разделе технико-экономического обоснования были произведены расчеты затрат, связанных с реализацией проекта, а также рентабельности разработки проекта. Произведенные расчеты показали экономическую целесообразность проекта.

Разработанная программа дает возможность обрабатывать большие объемы информации, вести учет необходимой информации, работать с этой информацией, а также строить необходимые отчеты. Программное средство предоставит удобный интерфейс для работы с данными.

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий

Факультет КТ Кафедра ИСиТ
Специальность 1-40 01 01 Специализация 01

УТВЕРЖДАЮ
А.И.Парамонов
« 24 » сентября 2024 г.

ЗАДАНИЕ
по дипломному проекту студента

Гроднова Егора Андреевича
(фамилия, имя, отчество)

1. Тема проекта: **Программное средство для реализации компьютерных комплектующих**

утверждена приказом по университету от « 24 » сентября 2024 г. № 112-и

2. Срок сдачи студентом законченной работы 23 декабря 2024 года

3. Исходные данные к проекту Тип операционной системы – ОС Windows 7 и выше; Язык программирования – C#; клиент-серверная архитектура. Перечень выполняемых Функций: назначение разработки: автоматизация процессов подготовки и документооборота отчетно-справочной документации малого предприятия

4. Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение
1 Анализ предметной области
2 Моделирование предметной области
3 Проектирование и разработка программного средства
4 Тестирование программного средства
5 Руководство по эксплуатации программного средства
6 Технико-экономическое обоснование разработки программного средства для
Заключение
Список использованных источников
Приложение А. Исходный код программного средства

5. Перечень графического материала (с точным указанием наименования) и обозначения вида и типа материала):

Диаграмма вариантов использования. Плакат – формат А1, лист 1.

Диаграмма деятельности. Плакат – формат А1, лист 1.

Диаграмма классов программного средства. Плакат – формат А1, лист 1.

Обработка данных. Схема алгоритма – формат А1, лист 1.

Формирование заявки. Схема алгоритма – формат А1, лист 1.

Шифрование данных. Схема алгоритма – формат А1, лист 1.

6. Технико-экономическое обоснование разработки программного средства

Консультанты по дипломному проекту (с указанием разделов, по которому они консультируют):

Шелягович А.С. консультанта от кафедры – разделы 1-5;

Горовой В.Г. – раздел 6 (технико-экономическое обоснование);

Шведова О.А. – нормоконтроль.

ПРИМЕРНЫЙ КАЛЕНДАРНЫЙ ГРАФИК ВЫПОЛНЕНИЯ ДИПЛОМНОГО ПРОЕКТА

Наименование этапов дипломного проекта	Объём готовности проекта в %	Срок выполнения этапа	Примечание
Первая опроцентовка (введение, разделы 1-3)	30%	16.09.24 – 25.10.24	Консультант от кафедры
Вторая опроцентовка (разделы 3-5)	60%	26.10.24 – 30.11.24	Руководитель проекта
Третья опроцентовка (разделы 5-6, заключение, список использованных источников)	90%	01.12.24 – 18.12.24	Руководитель проекта
Консультации по оформлению графического материала и пояснительной записки, нормоконтроль	–	С 01.10.24 (согласно графику)	Нормоконтролёр
Итоговая проверка готовности дипломного проекта на заседании рабочей комиссии кафедры ИСиТ и допуск к защите в ГЭК	100%	С 23.12.24 (согласно графику)	Председатель рабочей комиссии
Рецензирование дипломного проекта	100%	С 26.12.24 (согласно распоряжению)	Рецензент
Защита дипломного проекта	100%	С 18.01.25 (согласно приказу)	ГЭК

Дата выдачи задания 16 сентября 2024 г. Руководитель _____ /А.С. Шелягович/

Задание принял к исполнению _____ / Е.А. Гроднов /

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире информационные технологии прочно обосновались, заняв свою нишу в повседневной жизни. Многократно увеличились потоки информации. Автоматизированные средства помогают, а в чём-то и заменяют людские ресурсы. Удобство и эффективность таких средств, сложно переоценить. Сейчас использование электронных вычислительных машин ЭВМ) уже стало необходимостью.

Крупные и мелкие предприятия, различные организации успешно используют компьютеры во всех сферах своей деятельности (сфере управления, производства, обучения и др.).

Использование ЭВМ является универсальным решением: повышается общая скорость работы, при более низких затратах, по сравнению с аналогичным трудом человека.

Но этого оказалось бы недостаточным условием, так как немало важно правильно организовать взаимодействие человека и ЭВМ. Поэтому параллельно с техникой, развивалось и программное обеспечение.

Правильно разработанная программа упрощает взаимодействие человека с ПК, тем самым минимизируя процент ошибок и повышая производительность.

В настоящее время существует множество программ для конкретных видов деятельности; универсальных автоматизированных систем, включающих в себя общий набор инструментов для функционирования организаций; пакетов прикладных программ.

Но так как к каждому виду деятельности должен существовать свой подход, бывает достаточно сложно охватить современными средствами цели и задачи какого-либо определённой организации.

Целью дипломного проекта является разработка программного средства для реализации компьютерных комплектующих. Разработанная программа даст возможность обрабатывать большие объемы информации, вести учет необходимой информации, работать с этой информацией, а также строить необходимые отчеты. Программное средство (ПС) предоставит удобный интерфейс для работы с данными.

Разработка проекта будет выполнена с использование среды разработки Visual Studio, как одного из самых распространенных средств разработки, уникальной в своем роде программы.

В первой главе дипломного проекта производится анализ предметной области.

Во второй главе осуществляется моделирование предметной области, а также разрабатываются функциональные требования для разрабатываемого

программного обеспечения (ПО).

Третья глава раскрывает материал разработки приложения и структур данных, используемых в приложении.

В четвертой главе приведено тестирование основных модулей программы, настройка программы. Для облегчения понимания руководство снабжено иллюстрациями интерфейса программы.

В пятой главе приводится методика работы с разрабатываемым ПО.

В шестой главе приводится технико-экономическое обоснование разработки программного средства.

Дипломный проект проверен в системе антиплагиат. Уникальность работы составляет 84%.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Объектом исследования является продажа компьютерных комплектующих. Обычно это происходит на торговых точках или онлайн-платформах с пунктом самовывоза. Однако перечень документов и способ организации однотипен, однако имеет свои различия. Охватить оба вида продаж сложно, поэтому для рассмотрения выберем продажу компьютерных комплектующих на в торговых точках.

Работа торговой организации – сложная система. Каждый товар, который попал на полки магазина, поставляется в соответствии с товарно-транспортной накладной (ТТН).

Товарно-транспортная накладная предназначена для учета движения товарно-материальных ценностей (ТМЦ) при их перемещении с участием транспортных средств и является основанием для списания ТМЦ у грузоотправителя и оприходования их у грузополучателя. ТТН является бланком строгой отчетности [1]. Стандартный вид ТТН изображен на рисунках 1.1 и 1.2.


1-й экз. — грузополучателю
2-й экз. — грузоотправителю
3-й и 4-й экз. — перевозчику

Грузоотправитель | Грузополучатель | Заказчик автомобильной перевозки (Плательщик)

УНП

Типовая форма ТТН-1
УТВЕРЖДЕНО
Постановление Министерства финансов Республики Беларусь 18.12.2008 № 192

Серия СХ 1

ТОВАРНО-ТРАНСПОРТНАЯ НАКЛАДНАЯ

"___" _____ 20___ г.

0000000

Автомобиль _____ (марка, государственный номер) Прицеп _____ (марка, государственный номер) К путевому листу № _____

Владелец автомобиля _____ (наименование) Водитель _____ (фамилия и инициалы)

Заказчик автомобильной перевозки (Плательщик) _____ (наименование, адрес)

Грузоотправитель _____ (наименование, адрес)

Грузополучатель _____ (наименование, адрес)

Основание отпуска _____ (дата и номер договора или другого документа) Пункт погрузки _____ (адрес) Пункт разгрузки _____ (адрес)

Переадресовка _____ (наименование, адрес нового грузополучателя, фамилия, инициалы, подпись уполномоченного должностного лица)

I. ТОВАРНЫЙ РАЗДЕЛ

Наименование товара	Единица измерения	Количество	Цена, руб.	Стоимость, руб.	Ставка НДС, %	Сумма НДС, руб.	Стоимость с НДС, руб.	Количество грузовых мест	Масса груза	Примечание
1	2	3	4	5	6	7	8	9	10	11
Итого	X		X		X					

ОБРАЗЕЦ

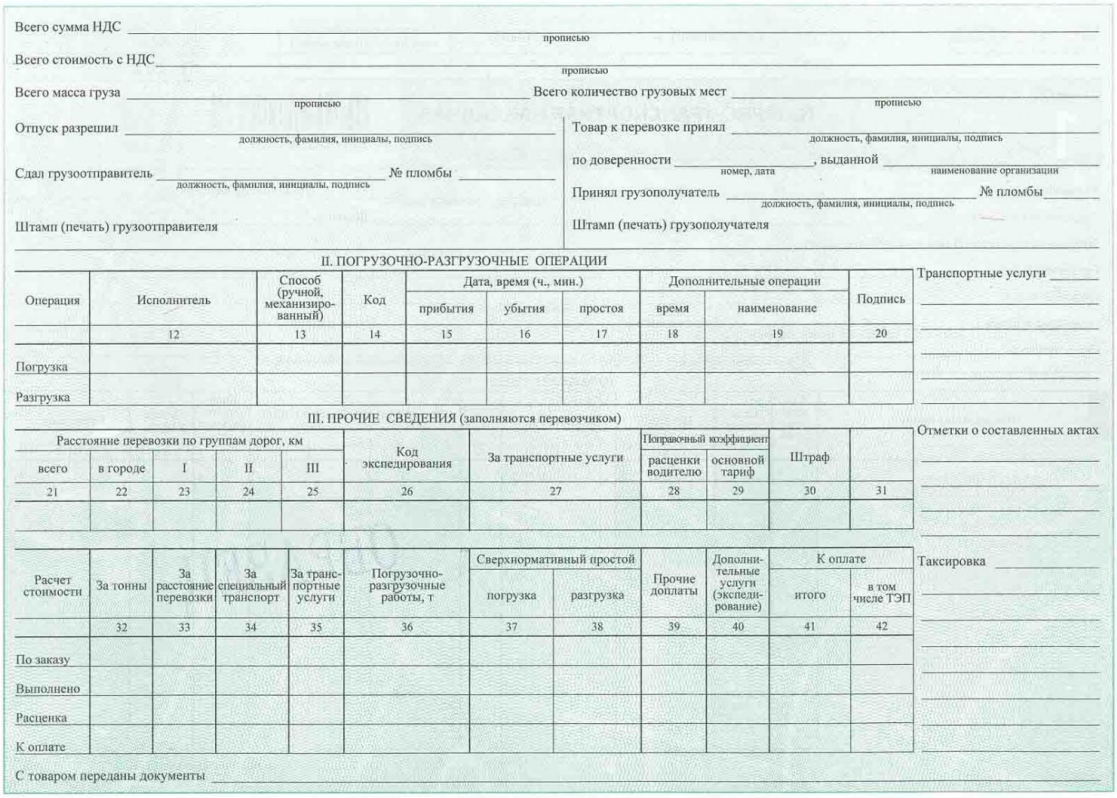
Количество ездок (заездов) _____

Заполняется шариковой ручкой с давлением, обеспечивающим читаемость всех экземпляров.

РУП "Издательство "Белбланкавыд", а4, 1344ц, VIII-2010 г., т. 4000х25х4. УПП «ВОТ», зак.Ц5671-10.

Внимание! Не подлежит размножению и передаче третьим лицам. Кол формы 860. Введена в обращение с 01.05.2009. Телефон доверия МНС РБ (017) 328-53-53.

Рисунок 1.1 – Товарно-транспортная накладная (сторона 1)



Всего сумма НДС _____ прописью

Всего стоимость с НДС _____ прописью

Всего масса груза _____ прописью Всего количество грузовых мест _____ прописью

Отпуск разрешил _____ должность, фамилия, инициалы, подпись

Сдал грузоотправитель _____ должность, фамилия, инициалы, подпись № пломбы _____

Штамп (печать) грузоотправителя

Товар к перевозке принял _____ должность, фамилия, инициалы, подпись

по доверенности _____ номер, дата, выданной _____ наименование организации

Принял грузополучатель _____ должность, фамилия, инициалы, подпись № пломбы _____

Штамп (печать) грузополучателя

II. ПОГРУЗОЧНО-РАЗГРУЗОЧНЫЕ ОПЕРАЦИИ

Операция	Исполнитель	Способ (ручной, механизированный)	Код	Дата, время (ч., мин.) прибытия	убытия	простоя	Дополнительные операции время	наименование	Подпись
	12	13	14	15	16	17	18	19	20
Погрузка									
Разгрузка									

Транспортные услуги _____

III. ПРОЧИЕ СВЕДЕНИЯ (заполняются перевозчиком)

Расстояние перевозки по группам дорог, км всего	в городе	I	II	III	Код экспедирования	За транспортные услуги	Поправочный коэффициент расценки водителю	основной тариф	Штраф	
21	22	23	24	25	26	27	28	29	30	31

Отметки о составленных актах

Расчет стоимости	За тонны	За расстояние перевозки	За специальный транспорт	За транспортные услуги	Погрузочно-разгрузочные работы, т	Сверхнормативный простой погрузка	разгрузка	Прочие доплаты	Дополнительные услуги (экспедирование)	К оплате итого	в том числе ТЭП
	32	33	34	35	36	37	38	39	40	41	42
По заказу											
Выполнено											
Расценка											
К оплате											

Таксировка _____

С товаром переданы документы _____

Рисунок 1.2 – Товарно-транспортная накладная (сторона 2)

Товар, который отпускается покупателю, сопровождается фискальным документом, который служит договором купли-продажи. Это гарантия покупателя в том, что в случае предоставления некачественного товара или услуги, он может обратится по месту покупки товара или предоставления услуги для решения возникших трудностей. Вид чека строго не регламентирован и в зависимости от магазина может выглядеть по-разному. На рисунке 1.3 представлен вид предоставляемого чека. Кроме этого, ежемесячно (а если потребуется, то чаще) проводится инвентаризация. Для этого необходимо четко знать, сколько товара находится в магазине, чтобы своевременно выявить недостачу.

Большинство программ, которые предназначены для управления торговлей строятся на базе 1С предприятие.

Ниже рассмотрены некоторые из них.

«1С: Розница» – это универсальная система для автоматизации розничной торговли [2]. Программа универсальна, ее функционал подходит как крупным торговым точкам, так и небольшим магазинам.

Данное ПО предлагает следующие возможности:

– эффективное ценообразование: учёт продаж позволяет анализировать спрос покупателей, определять их потребности, формировать оптимальные предложения и акции;

– обработка справочной информации: инструменты системы учёта розничной торговли наполняют базу данных из общероссийских классификаторов, справочных и нормативных документов;

– управление системой лояльности: система для автоматизации торговли включает более 20 программ, предусматривающих бонусы и скидки для покупателей;

– эффективное управление запасами товаров: ПО отслеживает продукцию на складе, ведет учёт скоропортящихся единиц;

– контроль закупок: в программе можно создавать заявки для поставщиков, формировать ценники, распределять поступающую в магазин продукцию по выбранным категориям;

– настройка работу кассы: у ПО понятный интерфейс, который поддерживает больше 40 кассовых операций;

– эффективное управление персоналом точки: система составляет список сотрудников, заводит для каждого регистрационные карты с отдельными штрихкодами;

– отчётность и аналитика: ПО помогает анализировать эффективность работы розничного магазина с помощью различных отчётов.

«1С: Управление торговлей» – данное обеспечение больше подойдёт для крупной розничной торговли. Но его функционал поддерживает работу и в других направлениях: онлайн-магазины, импорт, оптовая торговля.

Система предлагает пользователям следующие возможности:

– контроль запасов и управление товаром на складах;

– контроль и учёт продаж, инструменты системы позволяют сопровождать товар на каждом этапе;

– работа с закупками: система определяет ходовые товары и поддерживает их наличие на складе.

– клиентоориентированность: инструменты программного продукта позволяют владельцам магазинов эффективно выстраивать отношения с покупателями.

– финансовый анализ;

– хранилища данных.

Система учёта розничной торговли «МойСклад» – ещё одно программное обеспечение для розничной торговли, которое больше подойдёт субъектам малого и среднего предпринимательства. В отличие от прочих систем, его можно установить не только на компьютер или ноутбук, но и на планшет и телефон.

Как возможности системы помогают владельцам небольших магазинов эффективно вести торговлю:

– учёт товаров;

– аналитика: программа учёта товара в розничной торговле отслеживает рентабельность отдельных единиц продукции;

– финансовые операции: сервис поддерживает ключевые функции, необходимые магазинам: вносит информацию о доходах и расходах, рассчитывает прибыль, фиксирует информацию о расчётах с покупателями, формирует простые отчёты.

– централизованность: к одному ПО можно подключить несколько точек;

– CRM-система;

– телефония: ПО интегрируется с телефонией, покупателям можно звонить прямо с сайта.

Программа для учёта товара в торговой точке «LiteBox» – эта кассовая программа учёта товара в розничной торговле работает на компьютерах, ноутбуках, планшетах и телефонах. Интерфейс понятный, в нём легко разобраться. А функционал программы рассчитан на основные розничные операции:

– продажи товаров (в том числе, подакцизных);

– работа с отложенными чеками;

– работа с кассой, фиксирование прихода и выдачи наличных;

– безналичный расчёт;

– открытие и завершение смен в магазине;

– автоматический учёт скидок;

– формирование и печать чеков.

Программа учёта товара в розничной торговле «Моё дело» – комплексное решение для автоматизации торгового бизнеса. Кроме розничных предприятий, сервис подходит для оптовой торговли, онлайн-магазинов, производственных компаний.

К системе можно подключить до 50 торговых точек. Программой предусмотрен широкий функционал, позволяющий делегировать большинство рутинных процессов:

– товароучёт;

– онлайн-касса;

– работа с широкой номенклатурой;

– эффективное управление финансами;

– сбор и хранение данных;

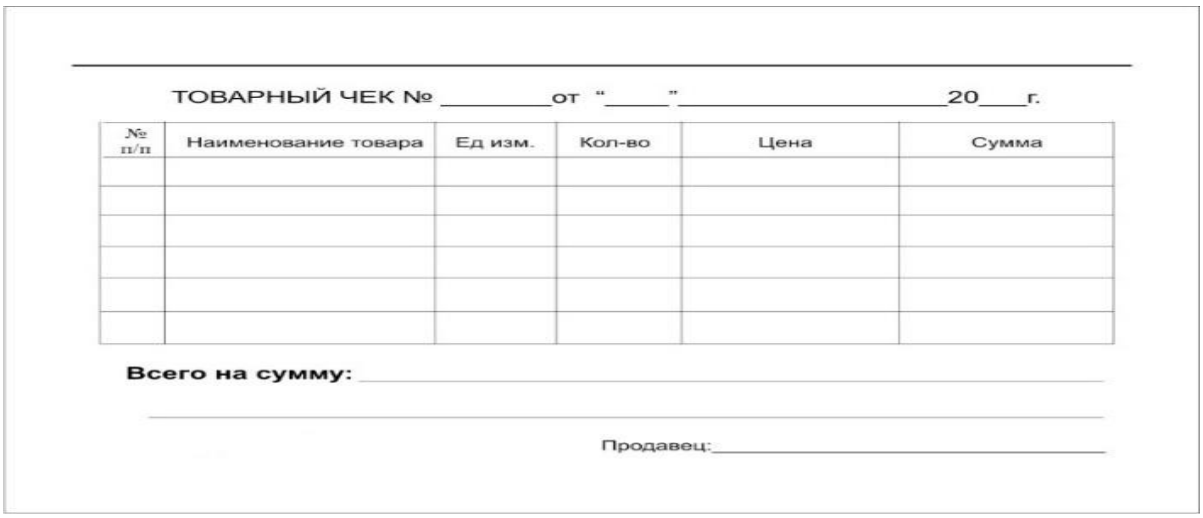
– формирование торговых документов;

– онлайн-бухгалтерия;

– доступ по должностям.

После анализа имеющихся решений было принято решение создать собственное программное средство, так как все имеющиеся аналоги либо

только частично удовлетворяют требованиям, либо наоборот имеют слишком большой излишний функционал, который не будет использоваться, однако платить за него необходимо.



ТОВАРНЫЙ ЧЕК № _____от "_____"_____20____г.

№ п/п	Наименование товара	Ед изм.	Кол-во	Цена	Сумма

Всего на сумму: _____

_____

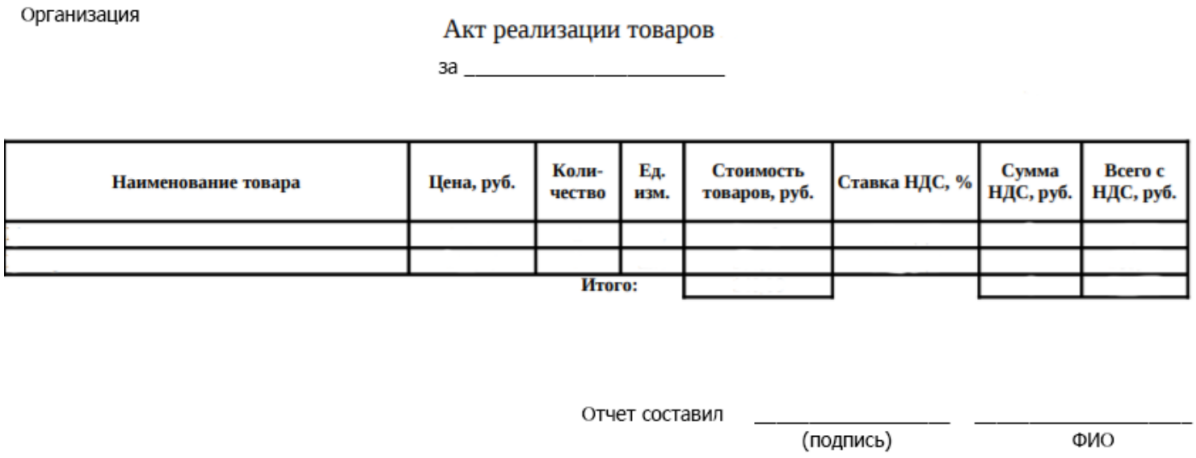
Продавец:_____

Рисунок 1.3 – Чек

ТТН и чек – это основные документы, которые являются базисом для ряда других. Так на основе этих данных формируются всевозможные отчеты:

– отчет по реализации (рисунок 1.4);

– выгрузка остатков.



Организация

Акт реализации товаров

за _____

Наименование товара	Цена, руб.	Коли-чество	Ед. изм.	Стоимость товаров, руб.	Ставка НДС, %	Сумма НДС, руб.	Всего с НДС, руб.
			Итого:				

Отчет составил _____ _____

(подпись) ФИО

Рисунок 1.4 – Акт реализации

1.2 Обзор существующих аналогов

Обзор существующих аналогов программных средств для реализации компьютерных комплектующих В рамках темы диплома «Программное

средство для реализации компьютерных комплектующих» следует рассмотреть несколько существующих аналогов, их функциональность, а также плюсы и минусы каждого из них. Программные решения в этой области могут варьироваться от специализированных систем управления запасами до программного обеспечения для сборки и модификации компьютерных комплектующих.

1. PCPartPicker Описание: это веб-сервис, позволяющий пользователям собирать свои компьютерные конфигурации, проверять совместимость компонентов и сравнивать цены у различных онлайн-ретейлеров. Плюсы: - Простота использования: интуитивно понятный интерфейс позволяет легко добавлять и удалять компоненты. - Совместимость компонентов: система автоматически проверяет, совместимы ли выбранные компоненты. - Обширная база данных: огромный выбор компонентов и комплектаций с доступом к ценам и отзывам. Минусы: - Онлайн-зависимость: требует доступа к интернету для функционирования и обновления данных. - Отсутствие функционала для локальных продавцов: ориентирован в основном на крупные онлайн-магазины.

2. iFixit Описание: Платформа, предоставляющая руководства по ремонту и модификации компьютерной техники и запчастей. Плюсы: - Обширная база знаний: множество статей и видеоуроков по ремонту и модификации. - Комьюнити: активное сообщество пользователей, готовых делиться своим опытом и советами. Минусы: - Ограниченный функционал для сборки: не предоставляет возможность собирать конфигурации или проверять совместимость компонентов. - Фокус на ремонте: основное внимание уделяется восстановлению и замене деталей, а не их продаже.

3. Newegg Описание: Один из крупнейших интернет-магазинов, специализирующихся на продаже компьютерных комплектующих и электроники. Плюсы: - Широкий ассортимент: огромное количество компонентов и электроники. - Обзоры и рейтинги: пользователи могут оставлять свои отзывы и рекомендации. Минусы: - Ограниченные инструменты для сборки: фокус на продаже, нет инструментов для проектирования конфигураций. - Цены: иногда могут быть выше, чем у других онлайн-магазинов.

4. BuildMyPC Описание: Веб-приложение для сборки пользовательских ПК, которое предлагает возможность выбора комплектующих и собирания конфигурации. Плюсы: - Легкость в использовании: простой интерфейс для начинающих пользователей. - Проверка совместимости: система автоматически проверяет, соответствуют ли выбранные компоненты друг другу. Минусы: - Ограниченная база данных: может не включать все новые компоненты на рынке. - Проблемы с актуальностью цен: иногда цены не

обновляются в реальном времени. Заключение Существующие аналоги программных средств для реализации компьютерных комплектующих обладают своими уникальными функциями, которые обеспечивают пользователей необходимыми инструментами для сборки, модификации и ремонта компьютерных систем. Каждый из них имеет свои плюсы и минусы, что определяет их целевую аудиторию и применение. Поэтому при разработке собственного программного обеспечения важно учитывать недостатки существующих решений и предлагать улучшенные функции и возможности для пользователей

5. DNS (ДНС) Описание: Один из крупнейших российских ритейлеров, предлагающий широкий ассортимент компьютерных комплектующих, электроники и бытовой техники. Также предоставляют возможность покупки через интернет и в розницу. Плюсы: - Широкий ассортимент: большой выбор компьютерных комплектующих, включая редкие и уникальные товары. - Физические магазины: возможность проверки товара вживую перед покупкой, что полезно для пользователей, предпочитающих делать покупки оффлайн. - Система лояльности и акции: часто предлагаются скидки и акции для постоянных клиентов. Минусы: - Качество обслуживания: пользовательские отзывы иногда указывают на проблемы с обслуживанием или задержками в доставках.

Существующие аналоги программных средств для реализации компьютерных комплектующих обладают своими уникальными функциями, которые обеспечивают пользователей необходимыми инструментами для сборки, модификации и ремонта компьютерных систем. Каждый из них имеет свои плюсы и минусы, что определяет их целевую аудиторию и применение. Поэтому при разработке собственного программного обеспечения важно учитывать недостатки существующих решений и предлагать улучшенные функции и возможности для пользователей.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Функциональная модель разрабатываемого программного средства

Анализируя предметную область, можно выделить два основных процесса: поступление и продажа товаров. Помимо этого, существует ряд вспомогательных процессов: внесение данных о сотрудниках, поступления товара, продажа товара.

Рассмотрим их подробно.

Внесение данных о сотрудниках – в этом процессе в систему вносятся данные о сотрудниках магазина, которые работают в нем.

Поступление товара – этот процесс фиксирует, когда, что и в каком количестве поступило в магазин.

Продажа товара – это процесс фиксирует что, в каком количестве и когда было продано. При этом, на основе этого представляется возможность фиксировать информацию об остатках в системе.

На основе вышесказанного выделим сущности:

– номенклатура (наименование, категория, баланс, цена единицы, цена продажи);

– сотрудник (полное имя, телефон, должность);

– движение товаров (дата, номенклатура, количество, сумма, цена единицы, номер чека, номер ттн);

– ТТН (номер ТТН, дата, сотрудник);

– чек (номер чека, дата, сотрудник).

Установим для каждого атрибута тип данных.

Номенклатура:

– наименование – текстовое поле;

– категория – текстовое поле;

– баланс – числовое поле;

– цена единицы – числовое поле с плавающей точкой;

– цена продажи – числовое поле с плавающей точкой.

Сотрудник:

– полное имя – текстовое поле;

– телефон – текстовое поле;

– должность – текстовое поле.

Движение товаров:

– номенклатура – текстовое поле;

– дата – поле даты;

– количество – числовое поле;

– сумма – числовое поле с плавающей точкой;
– цена единицы – числовое поле с плавающей точкой;
– номер чека – числовое поле;
– номер ттн – числовое поле.
ТТН:
– номер ТТН – текстовое поле;
– дата – поле даты;
– сотрудник – текстовое поле.
Чек:
– номер чека – числовое поле;
– дата – поле даты;
– сотрудник – текстовое поле.

Функциональное моделирование является важнейшим элементом концептуального анализа, который выполняется на начальном этапе проектирования любой информационной системы. Разработка и анализ функциональной модели задачи состоит в выявлении набора функций, которыми должна быть наделена система, определении информационных потоков и т.д. [3].

Для функционального анализа удобно использовать простые, доступные для широкого понимания, хорошо проработанные методологии. Несмотря на солидный возраст стандартной методологии IDEF0, она по-прежнему весьма популярна среди аналитиков и широко используется на этапе концептуального анализа.

Методология функционального моделирования IDEF0 – это технология описания системы в целом как множества взаимозависимых действий, или функций. Схемы строятся по иерархическому принципу с необходимой степенью подробности и помогают разобраться в том, что происходит в изучаемой системе или процессе, какие функции выполняются и в какие отношения вступают между собой и с окружающей средой ее функциональные блоки. IDEF0-модель принципиально не может ответить на вопросы о том, как протекают процессы в системе во времени и в пространстве.

Первый верхний уровень функциональной модели, разрабатываемого программного средства представлен контекстной диаграммой системы, которая приведена на рисунке 2.1. Она представляет собой самое общее описание системы и ее взаимодействия с внешней средой.

Рисунок 2.1 – Контекстная диаграмма разрабатываемой системы

Для разрабатываемого ПО входными данными будут являться данные о сотрудниках, данные о товарах, данные о расходе, данные о приходе и данные из базы данных. Управляющей информацией будет служить руководство пользователя, которое определяет порядок работы с программой. Механизм – сотрудник, который будет осуществлять работу с программой (например, сотрудник магазина). Исходящей информацией будет являться обновленная информация в базе данных, товарный чек и отчеты.

В декомпозиции приведены основные блоки как взаимодействия пользователя с системой, так и процессов внутри системы.

В зависимости от входящей информации пользователь должен совершить одно из действий: оформление прихода, оформление расхода, изменение содержимого базы данных, формирование отчетов.

На этапе «Оформление прихода» входящей информацией служит данные о приходе, а исходящей «Данные по приходу для внесения в БД».

На этапе «Оформление расхода» входящей информацией будет «Данные о расходе», и «Данные о сотрудниках», а исходящей «Товарный чек» и «Данные о реализации для внесения в БД». После всех трех этапов программа переходит на следующий этап «Изменение содержимого в БД». После обработки информации программа либо завершит свою работу (в данном случае исходящая информацией будет «Обновленные данные в БД») или перейдет на следующий этап «Формирование отчетов». Исходящей

информацией на данном этапе будут «Данные из базы данных». К этапу «Формирование отчетов» можно перейти в самом начале работы программы. Входящей информацией в данном случае также будут являться «Данные из БД». Исходящей информацией в обоих случаях будет «Отчеты». Декомпозиция функциональной модели представлена на рисунке 2.2. Для более глубокого понимания процессов полноценной работы ПС необходимо описание последовательности происхождения и обработки информации. Наиболее удобной формой представления является диаграмма потоков данных (DFD – Data Flow Diagram). Она является графической иерархической спецификацией, описывающей систему с позиций потоков данных.

Процессами программы являются: «Удалить данные из БД», «Изменить данные в БД», «Добавить данные в БД», «Оформить отчет», «Оформить приход» и «Оформить расход». Информационная взаимосвязь между перечисленными выше процессами, последовательность обработки и преобразования входящих данных отображены на диаграмме потоков данных.

Хранилище данных «БД «Store» заполняется информацией, поступающей от внешних сущностей «Данные для ввода и изменения данных в БД» и «Покупатель» посредством процессов «Изменить данные в БД», «Удалить данные из БД», «Добавить данные в БД», «Оформить приход» и «Оформить расход».

Процесс «Оформить приход» преобразует данные, поступающие из хранилища данных «БД «Store». Преобразованные данные поступают обратно в хранилище данных «БД «Store».

Процесс «Сформировать отчет» обрабатывает данные, поступающие от хранилища данных «БД «Store». Обработанные данные поступают во внешнюю сущность «Отчет по реализации» и «Выгрузка остатков».

Процесс «Оформить расход» обрабатывает данные, поступающие от хранилища данных «БД «Store» и от внешней сущности «Покупатель». Обработанные данные поступают обратно в хранилище данных «БД «Store», а также во внешнюю сущность «Товарный чек».

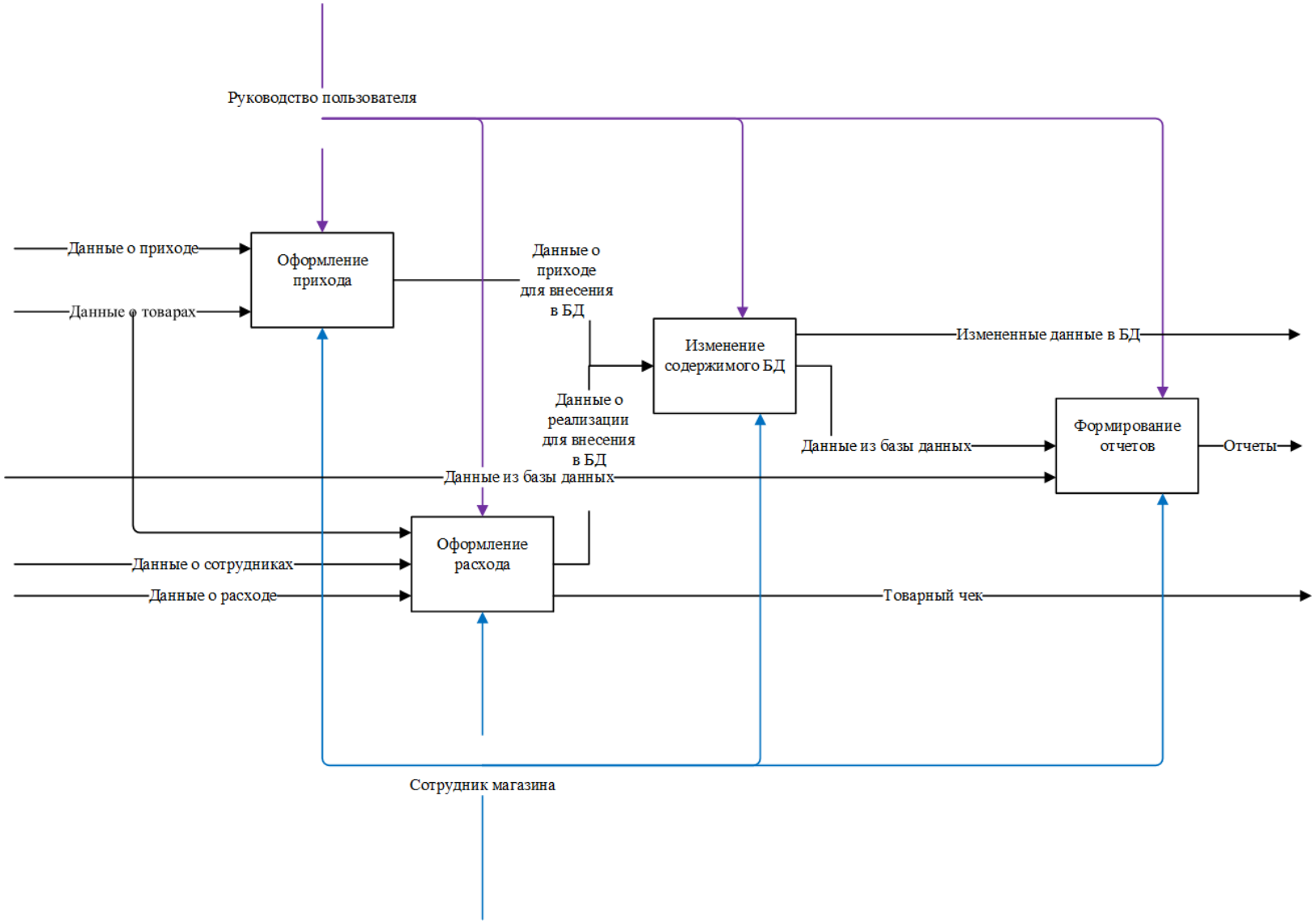


Рисунок 2.2 – Диаграмма декомпозиции разрабатываемой системы

2.2 Разработка функциональной спецификации требований к программному средству

Один из наиболее ответственных этапов создания программного продукта – этап постановки задачи [4]. На этом этапе принимаются важные решения относительно функций создаваемого ПО, эксплуатационных ограничений, накладываемых на него.

Производится выбор архитектуры, среды разработки ПО, интерфейса пользователя и др. От этого выбора будет зависеть качество и стоимость конечного программного продукта.

Функциональные требования описывают сервисы, предоставляемые программной системой, ее поведение в определенных ситуациях, реакцию на те или иные входные данные и действия, которые система позволит выполнять пользователям. Иногда сюда добавляются сведения о том, чего система делать не должна.

Каждый программный продукт предназначен для выполнения определенных функций. Для того чтобы определить, подходит та или иная программа для решения задач, необходимо иметь четкий набор критериев, на основании которого можно сделать правильный выбор.

При написании функциональных требований необходимо учитывать, что чем они будут подробнее, тем более точная оценка работ по срокам и стоимости будет произведена перед разработкой технического задания на создание программного обеспечения. Если на дальнейших этапах разработки ПО не возникнет дополнений к изначально сформулированным функциональным требованиям, то эта оценка будет достаточно точной. В то же время при описании требований не надо углубляться в какие-то мелкие детали. Необходимо описывать именно функции программы, а не то, какую кнопку надо нажать в верхнем левом углу окна программы, чтобы получить результат. Такие детали должны быть подробно проработаны уже в процессе разработки технического задания.

Таким образом разрабатываемое программное средство должно удовлетворять характеристикам, перечисленным ниже.

Программный продукт должен обеспечивать работу пяти категориям пользователей: оператор, продавец, старший продавец, заведующий, бухгалтер.

Оператор должен иметь следующие возможности: ввод ТТН.

Продавец должен иметь следующие возможности:

– просматривать таблицу «Номенклатура»;

– оформлять реализацию.

Старший продавец должен иметь следующие возможности:

– просматривать таблицу «Номенклатура»;

– оформлять реализацию;

– осуществлять построение отчета реализации.

Заведующий должен иметь следующие возможности:

– просматривать все таблицы (кроме таблицы, которая содержит данные о логинах, правах и паролях всех пользователей);

– вводить ТТН;

– оформлять реализацию;

– формировать отчет реализации;

– осуществлять выгрузку остатков.

Администратор должен иметь следующие возможности: получает доступ ко всему функционалу системы.

Кроме того, разрабатываемый программный продукт должен обладать следующими нефункциональными требованиями:

1. Простота эксплуатации и удобство использования: система должна быть простой в эксплуатации для опытного оператора.

2. Устойчивость к сбоям: процент ситуаций, приводящих к сбоям, должен быть минимальным.

3. Требования производительности: разрабатываемое ПС должно иметь минимальные требования к производительности быть в рамках минимальных требований для операционной системы:

– процессор с тактовой частотой 1000 МГц;

– оперативная память не менее 1024 MB;

– свободное место на жестком диске не менее 20 MB;

– «Windows 7» и выше.

Требования распространения: по заказу, развертываемая версия.

Более детально все функциональные возможности разрабатываемой программной системы представлено на диаграмме вариантов использования.

Актерами разрабатываемой системы будут обозначенные ранее роли. У каждого актера есть свой набор прецедентов.

Оператор может только вводить ТТН.

Продавец может:

– просматривать таблицу «Товар»;

– оформлять продажу.

Старший продавец может:

– просматривать таблицу «Товар»;

– оформлять продажу;

– осуществлять построение отчета реализации.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Разработка архитектуры программного средства

Архитектура программного обеспечения (*software(application) architecture*) − это структура программы или вычислительной системы, которая включает программные компоненты, видимые снаружи свойства этих компонентов, а также отношения между ними [5].

Необходимо определить внутренние и внешние интерфейсы каждой программной составной части.

Для представления архитектуры используется Диаграмма компонентов и диаграмма развертывания.

Диаграмма компонентов (*component diagram*) − статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами [6].

В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Диаграмма компонентов представлена на рисунке 3.1.

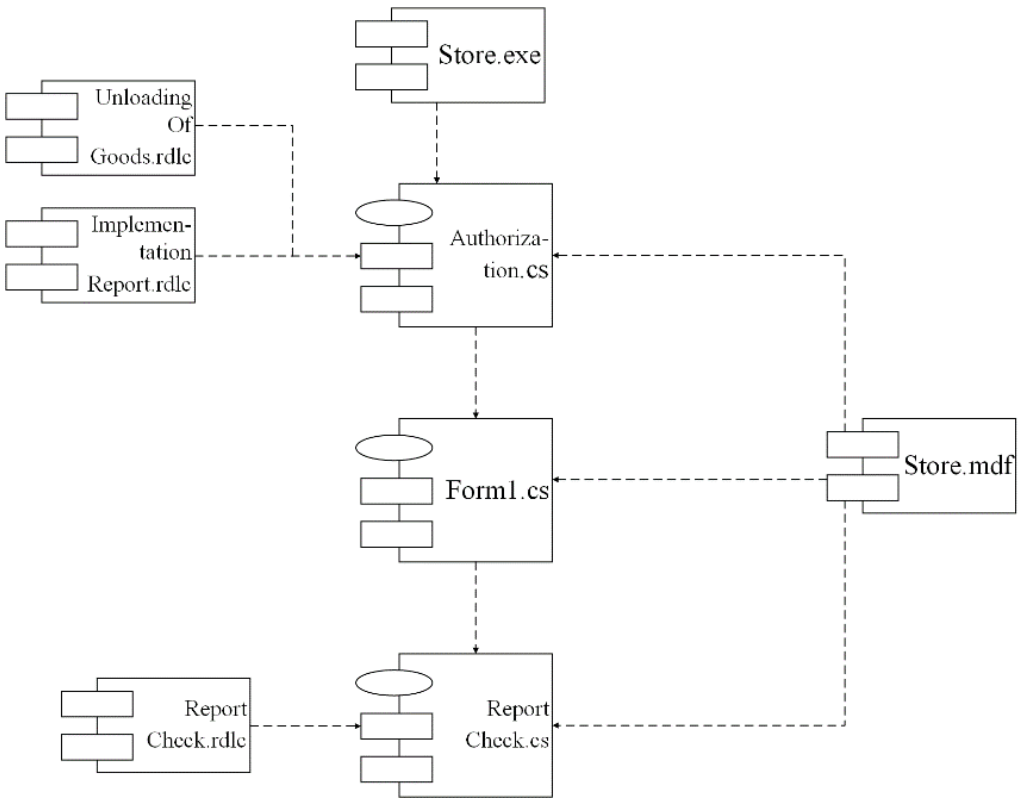


Рисунок 3.1 – Диаграмма компонентов

Диаграмма компонентов состоит из следующих составляющих:

– файлы: Store.mdf, reportCheck.rdlc, UnloadingOfGoods.rdlc,

ImplementationReport.rdlc.

– компонентов: Authorization.cs, Form1.cs, ReportCheck.cs, Store.exe

Компонент «Store.exe» связан с компонентом «Authorization.cs» - формой авторизации.

Форма авторизации «Authorization.cs» связана с компонентом «Form1.cs», а также связана с файлом «Store.mdf».

Главная форма программы «Form1.cs» связана с компонентом «ReportCheck.cs», а также связана с файлами «Store.mdf», «UnloadingOfGoods.rdlc», «ImplementationReport.rdlc».

Форма с чеком «ReportCheck.cs» связана с файлами «Store.mdf», «reportCheck.rdlc».

Диаграмма развертывания (синоним − диаграмма размещения) – применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы [7]. Кроме того, диаграмма развертывания показывает наличие физических соединений−маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения.

На диаграмме развертывания узлами системы являются программа на компьютере сотрудника, файл базы данных, который находится на сервере. Указана зависимость компонентов «Store.exe» на стационарном компьютере сотрудника и «База данных MS SQL Server «store». Примечание указывает на необходимость использования определенных версий программных средств и конфигураций компьютера.

3.2 Разработка модели базы данных

Опираясь на ранее проведенный анализ, предполагается организация информационного обеспечения автоматизированной системы в виде реляционной базы данных, которая должна быть приведена к третьей нормальной форме (3НФ).

Реляционная база данных (БД) – это совокупность отношений, содержащих всю информацию, которая должна храниться в базе данных [8]. Однако, пользователи могут воспринимать такую базу данных как совокупность таблиц. Таким образом, реляционную базу данных можно рассматривать как хранилище данных, содержащее набор двумерных связанных таблиц.

В реляционной базе данных таблицы связаны между собой; это

позволяет с помощью единственного запроса найти все необходимые данные (которые могут находиться в нескольких таблицах). Будучи связанной посредством общих ключевых полей, информация в реляционной базе данных может объединяться из множества таблиц в единый результирующий набор.

При установлении связи между двумя таблицами одна из них будет являться главной, а вторая – подчиненной. Различие между ними несколько упрощенно можно пояснить следующим образом. В главной таблице всегда доступны все содержащиеся в ней записи. В подчиненной же таблице доступны только те записи, у которых значение атрибутов внешнего ключа совпадает со значением соответствующих атрибутов текущей записи главной таблицы. Причем изменение текущей записи главной таблицы приведет к изменению множества доступных записей подчиненной таблицы, а изменение текущей записи в подчиненной таблице не вызовет никаких изменений ни в одной из таблиц.

Различают четыре типа связей между таблицами базы данных:

– «один-к-одному» – каждой записи одной таблицы соответствует только одна запись другой таблицы;

– «один-ко-многим» – одной записи главной таблицы могут соответствовать несколько записей подчиненной таблицы;

– «многие-к-одному» – нескольким записям главной таблицы может соответствовать одна и та же запись подчиненной таблицы;

– «многие-ко-многим» – одна запись главной таблицы связана с несколькими записями подчиненной таблицы, а одна запись подчиненной таблицы связана с несколькими записями главной таблицы.

Нормальная форма – это способ избавиться от избыточности информации и некоторых проблем, связанных с обработкой данных.

Переменная отношения находится в ЗНФ тогда и только тогда, когда ее не ключевые атрибуты (если они вообще существуют) являются одновременно:

– взаимно независимыми;

– неприводимо зависимыми от первичного ключа.

Неключевой атрибут – это атрибут, который не входит в состав первичного ключа рассматриваемой переменной отношения. Два или больше атрибутов называются взаимно независимыми, если ни один из них функционально не зависит от какой-либо комбинации остальных атрибутов. Подобная независимость подразумевает, что каждый такой атрибут может обновляться независимо от остальных атрибутов [9].

В состав проектируемой БД входят справочные и операционные таблицы.

Справочные таблицы содержат информацию справочного характера,

обладающую невысокой степенью изменчивости по сравнению с таблицами БД других видов; как правило, находятся с операционными таблицами в отношении «один-ко-многим», являясь при этом родительскими таблицами.

Под операционными таблицами понимаются таблицы БД, в которых происходит устойчивое во времени непрерывное или периодическое обновление, или добавление информации. Операционные таблицы находятся, как правило, в подчиненном отношении со справочными таблицами.

К справочным таблицам будут относиться таблицы «Users», «Employe», «Roles», «Position», «Nomenclature» и «Category». К операционным же относятся таблицы «TTN», «Check» и «GoodsMovement».

Для реализации заданных свойств БД необходимо организовать связь между справочными и операционными таблицами в отношении «один-ко-многим».

Таблица «Roles» связана с таблицей «Users» по полям «Code» и «CodeRole» в отношении «один-ко-многим».

Таблица «Employe» связана с таблицей «Users» по полям «Code» и «CodeEmploye» в отношении «один-ко-многим».

Таблица «Employe» связана с таблицей «TTN» по полям «Code» и «CodeEmploye» в отношении «один-ко-многим».

Таблица «Employe» связана с таблицей «Check» по полям «Code» и «CodeEmploye» в отношении «один-ко-многим».

Таблица «Positions» связана с таблицей «Employe» по полям «Code» и «CodePosition» в отношении «один-ко-многим».

Таблица «Nomenclature» связана с таблицей «GoodsMovement» по полям «Code» и «CodeNomenclature» в отношении «один-ко-многим».

Таблица «Category» связана с таблицей «Nomenclature» по полям «Code» и «Code Category» в отношении «один-ко-многим».

Таблица «TTN» связана с таблицей «GoodsMovement» по полям «Code» и «CodeTTN» в отношении «один-ко-многим».

Таблица «Check» связана с таблицей «GoodsMovement» по полям «Code» и «CodeCheck» в отношении «один-ко-многим».

Состав информации для операционных и справочных таблиц приведен далее в таблицах 3.1-3.9.

Таблица 3.1 – Структура таблицы «Category»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код категории
Name	nchar(10)	Нет / Да	Наименование категории

Таблица 3.2 – Структура таблицы «Employe»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код сотрудника
FullName	varchar(50)	Нет / Да	ФИО сотрудника
DOB	date	Нет / Да	День рождения
Address	varchar(50)	Нет / Да	Адрес
Phone	varchar(15)	Нет / Да	Номер телефона
Email	varchar(30)	Нет / Да	Электронный адрес
CodePosition	int	Нет / Нет	Код должности

Таблица 3.3 – Структура таблицы «GoodsMovement»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код движения товара
Data	date	Нет / Да	Дата операции
CodeNomenclature	int	Нет / Нет	Код номенклатуры
Quantity	real	Нет / Да	Количество товара
Sum	float	Нет / Да	Сумма товара
CodeTTN	int	Нет / Нет	Код ТТН
CodeCheck	int	Нет / Нет	Код чека

Таблица 3.4 – Структура таблицы «Nomenclature»

Наименование поля	Тип данных	Ключевое поле/ Обязательное	Описание
Code	int	Да / Да	Код товара
Name	varchar(50)	Нет / Да	Наименование товара
CodeCategory	int	Нет / Нет	Код категории товара
Balance	real	Нет / Да	Текущий остаток
UnitPrice	real	Нет / Да	Цена единицы
SellingPrice	real	Нет / Да	Розничная цена единицы

Таблица 3.5– Структура таблицы «Positions»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код должности
Name	varchar(30)	Нет / Да	Наименование должности

Таблица 3.6 – Структура таблицы «Check»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код
NumberCheck	varchar(10)	Нет / Да	Номер чека
Data	date	Нет / Да	Дата
CodeEmploye	int	Нет / Нет	Код сотрудника, который продал товар

Таблица 3.7 – Структура таблицы «Roles»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код роли
Name	varchar(50)	Нет / Да	Наименование роли

Таблица 3.8 – Структура таблицы «TTN»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код
NumberTTN	varchar(20)	Нет / Да	Номер ТТН
Data	date	Нет / Да	Дата
CodeEmploye	int	Нет / Нет	ФИО принявшего

Таблица 3.9 – Структура таблицы «Users»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код
Login	varchar(50)	Нет / Да	Логин пользователя
Password	varchar(20)	Нет / Да	Пароль пользователя
CodeEmploye	int	Нет / Нет	Код сотрудника
CodeRole	int	Нет / Нет	Код роли

3.3 Разработка алгоритма программного средства

Схема программы отображает последовательность операций в программе, описывая алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями. Схемы программы для каждой роли представлены на рисунках 3.2-3.5.

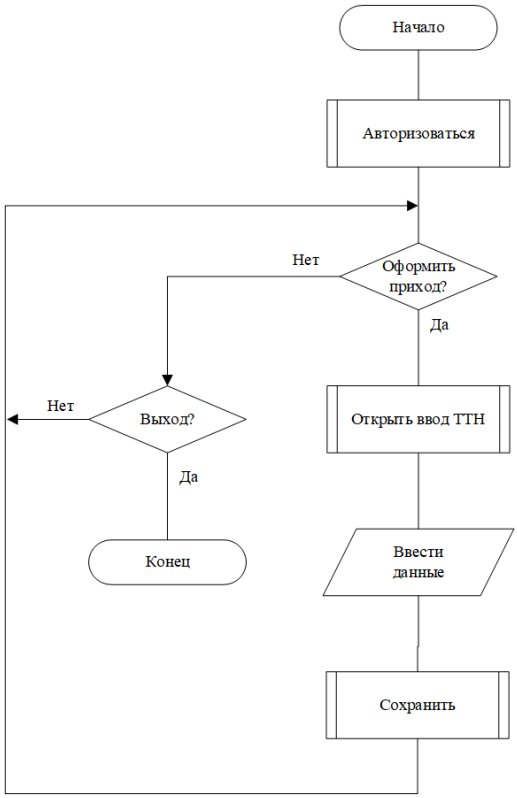


Рисунок 3.2 – Общая схема алгоритма (роль «Оператор»)

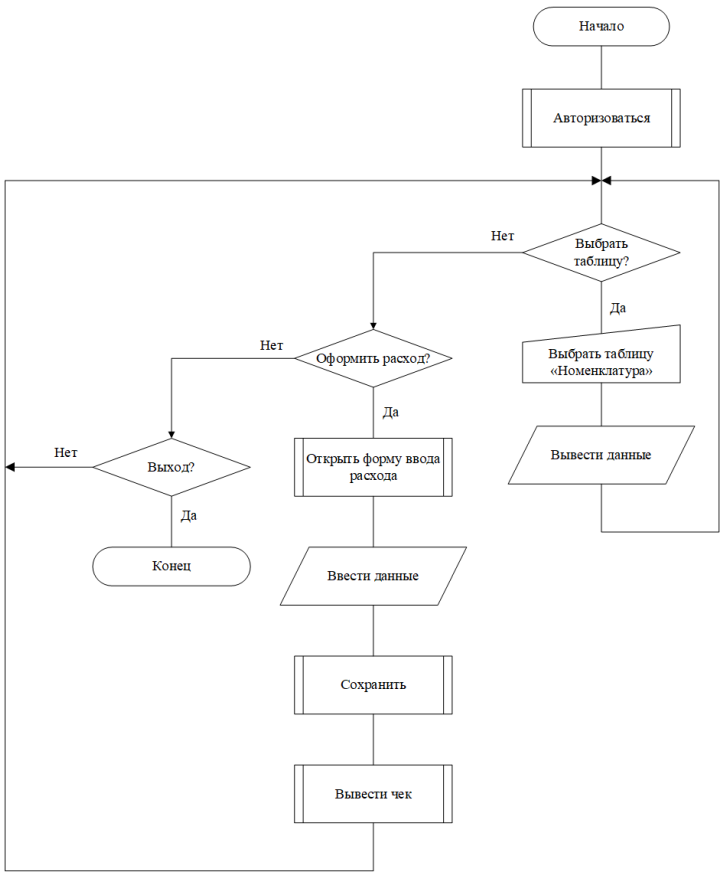


Рисунок 3.3 – Общая схема алгоритма (роль «Продавец»)

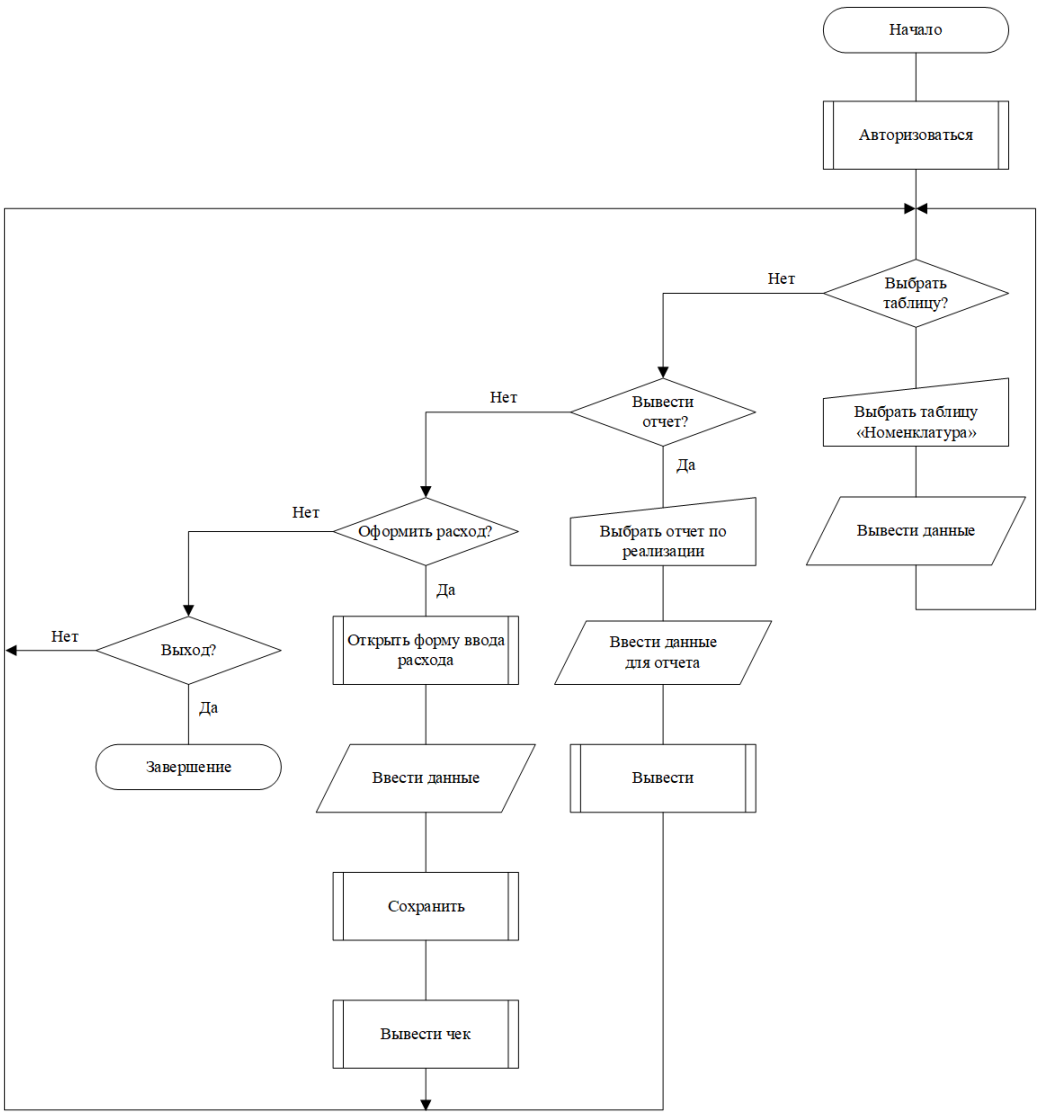


Рисунок 3.4 – Общая схема алгоритма (роль «Старший продавец»)

Опишем схема работы программы от имени администратора.

Началом схемы будет являться запущенная программа. После этого поток переходит к блоку выбора пункта меню. Пользователь выбирает пункт меню, который ему необходим.

Если пользователю необходимо работать с информационными таблицами, то поток переходит к выбору таблицы, затем поток направляется к блоку вывода данных. После этого у пользователя есть три варианта: добавить, редактировать или удалить данные.

В случае добавления данных поток переходит в соответствующий пункт меню, затем к вводу данных и сохранению их. Ели пользователю необходимо редактировать данные, то поток переходит к выбору записи, вводу новых данные и сохранению их. В случае удаления данных поток переходит к выбору записи, затем к удалению. После этого пользователь может продолжить работу с таблицей или вернутся к главному меню выбора.

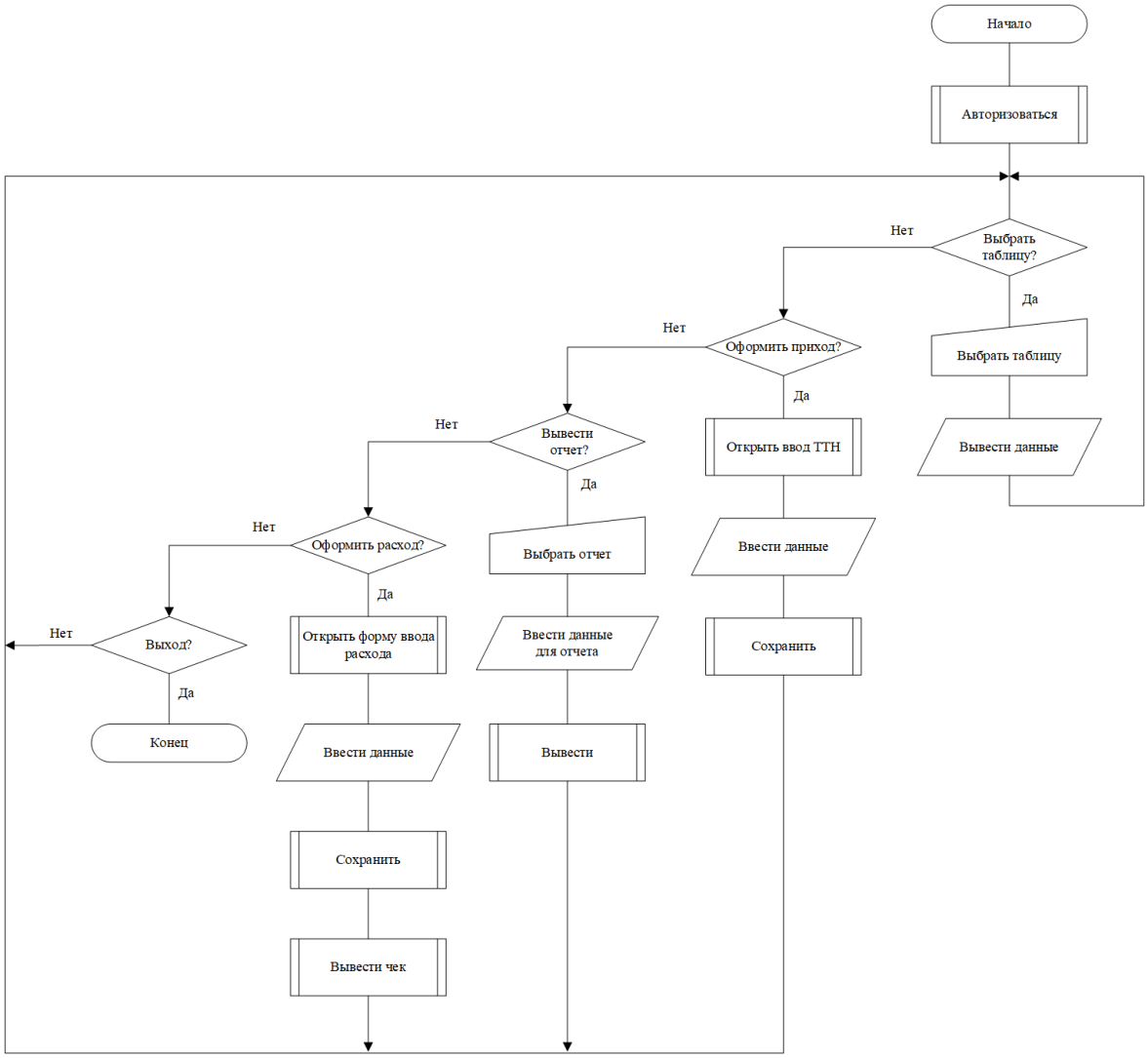


Рисунок 3.5 – Общая схема алгоритма (роль «Заведующий»)

Если пользователю необходимо оформить приход, то поток переходит к вводу данных по ТТН и сохранению приход.

Если пользователю необходимо найти ТТН, то поток переходит к выбору критерия для поиска, вводу данных для поиска и выводу данные.

Если пользователю необходимо вывести отчет, то поток переходит к выбору отчета, вводу исходных данных и выводу отчета.

Если пользователю необходимо ввести реализацию, то пользователь выбирает соответствующий пункт меню, вводит данные по расходу, сохранят данные и получает товарный чек.

Если ничего из перечисленного не подходит, то поток переходит к завершению.

Для корректной работы программы необходимо наличие базы данных на сервере. Если база данных была утеряна, то необходимо создать новую базу с именем «store». В своем составе она должна иметь 9 таблиц, структура

которых представлена в таблицах 3.10-3.18. Таблицы между собой должна быть обязательно связаны (рисунок 3.6).

Таблица 3.10 – Структура таблицы «Category»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код категории
Name	nchar(10)	Нет / Да	Наименование категории

Таблица 3.11 – Структура таблицы «Employe»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код сотрудника
FullName	varchar(50)	Нет / Да	ФИО сотрудника
DOB	date	Нет / Да	День рождения
Address	varchar(50)	Нет / Да	Адрес
Phone	varchar(15)	Нет / Да	Номер телефона
Email	varchar(30)	Нет / Да	Электронный адрес
CodePosition	int	Нет / Нет	Код должности

Таблица 3.12 – Структура таблицы «GoodsMovement»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код движения товара
Data	date	Нет / Да	Дата операции
CodeNomenclature	int	Нет / Нет	Код номенклатуры
Quantity	real	Нет / Да	Количество товара
Sum	float	Нет / Да	Сумма товара
CodeTTN	int	Нет / Нет	Код ТТН
CodeCheck	int	Нет / Нет	Код чека

Таблица 3.13 – Структура таблицы «Check»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код
NumberCheck	varchar(10)	Нет / Да	Номер чека
Data	date	Нет / Да	Дата
CodeEmploye	int	Нет / Нет	Код сотрудника

Таблица 3.14 – Структура таблицы «Nomenclature»

Наименование поля	Тип данных	Ключевое поле/ Обязательное	Описание
Code	int	Да / Да	Код товара
Name	varchar(50)	Нет / Да	Наименование товара
CodeCategory	int	Нет / Нет	Код категории товара
Balance	real	Нет / Да	Текущий остаток
UnitPrice	real	Нет / Да	Цена единицы
SellingPrice	real	Нет / Да	Розничная цена единицы

Таблица 3.15 – Структура таблицы «Positions»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код должности
Name	varchar(30)	Нет / Да	Наименование должности

Таблица 3.16 – Структура таблицы «Roles»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код роли
Name	varchar(50)	Нет / Да	Наименование роли

Таблица 3.17 – Структура таблицы «TTN»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код
NumberTTN	varchar(20)	Нет / Да	Номер ТТН
Data	date	Нет / Да	Дата
CodeEmploye	int	Нет / Нет	ФИО принявшего

Таблица 3.18 – Структура таблицы «Users»

Наименование поля	Тип данных	Ключевое поле / Обязательное	Описание
Code	int	Да / Да	Код
Login	varchar(50)	Нет / Да	Логин пользователя
Password	varchar(20)	Нет / Да	Пароль пользователя
CodeEmploye	int	Нет / Нет	Код сотрудника
CodeRole	int	Нет / Нет	Код роли

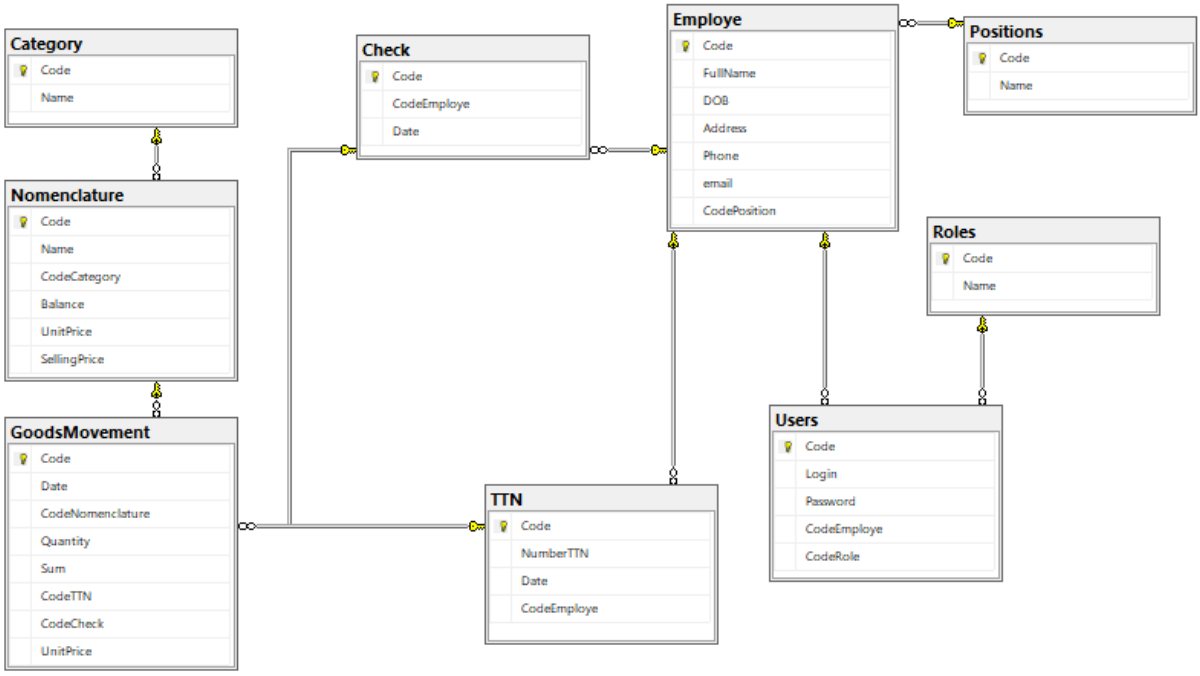


Рисунок 3.6 – Схема БД

Если при запуске программы база данных не была обнаружена, то программа запустится в режиме ограниченной функциональности, т.е. пользователю будут запрещены все действия, которые касаются базы данных. Для исправления данной ситуации необходимо восстановить базу данных.

3.4 Проектирование лингвистического обеспечения программного средства

Лингвистическое обеспечение – совокупность средств и правил для формализации естественного языка, используемых при общении пользователей и эксплуатационного персонала системы с комплексом средств автоматизации при функционировании программного средства [10].

Формализованным средством общения пользователей с компьютерной системой является пользовательский интерфейс, который должен быть не только интуитивно понятным, но и предупреждать пользователя о наступлении какого-либо события или предостерегать от ошибок.

К лингвистическому обеспечению можно отнести различного типа сообщения, информирующие пользователя о наступлении того или иного события. Данный механизм позволит избежать ошибок, связанных с использованием в вычислениях посторонних либо некорректных данных.

Поскольку программа предоставляет пользователю возможность дополнения базы данных новыми сведениями, а также изменения уже существующей информации, то также необходимо наличие механизма,

осуществляющего проверку вводимых пользователем значений.

Оповещение пользователя о наступлении какого-либо события будет происходить посредством стандартного диалогового окна, которые представлены на рисунках 3.6-3.8.

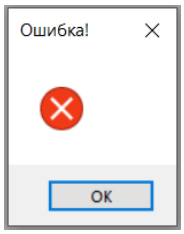


Рисунок 3.7 – Сообщение о наступлении события «Ошибка»

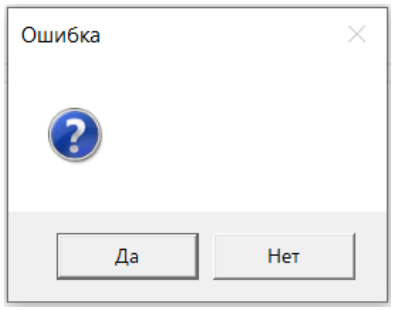


Рисунок 3.8 – Сообщение о наступлении события «Вопрос»

Рисунок 3.9 – Сообщение о наступлении события «Внимание»

Таким образом, интерфейс навязывает пользователю свои правила общения, что позволяет ему не допускать наиболее часто встречающиеся ошибки, обращать внимание на события, которые могут привести к ограничению функционала программы или ее некорректной работе.

3.5 Проектирование графического интерфейса

Графический интерфейс пользователя – это разновидность пользовательского интерфейса, в котором все элементы представлены на дисплее в виде графических объектов.

При запуске программы, пользователю необходимо пройти авторизацию. Окно входа в систему представлено на рисунке 3.9.

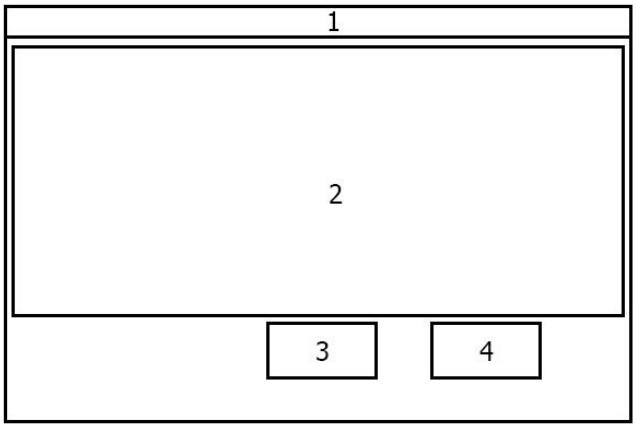


Рисунок 3.10 – Общий вид окна авторизации

На рисунке 3.9: 1 – строка заголовка программы; 2 – область ввода логина и пароля; 3 – кнопка принятия; 4 – кнопка отмены

Главное окно программы представлено областью вкладок, отвечающих за категории для группировки функций. Каждая категория имеет свою панель навигации, в виде группы вкладок для доступа к таблицам БД, а также построению отчетов. Количество вкладок категорий и доступных элементов управления изменяется в зависимости от роли пользователя. Общий вид главного окна представлен на рисунке 3.10.

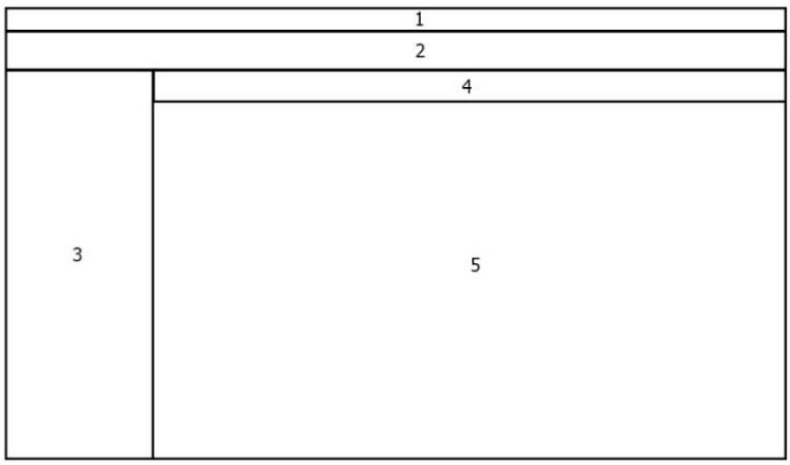


Рисунок 3.11 – Общий вид главного окна

На рисунке 3.10: 1 – строка заголовка программы; 2 – панель с вкладками категорий; 3 – панель с вкладками подкатегорий; 4 – управляющие кнопки; 5 – рабочая область

Посредством главного окна программы пользователь может

просматривать содержимое таблиц БД, производить редактирование, добавление и удаление данных, построение отчетов. На рисунке 3.11 представлен общий вид окна редактирования и добавления записей.

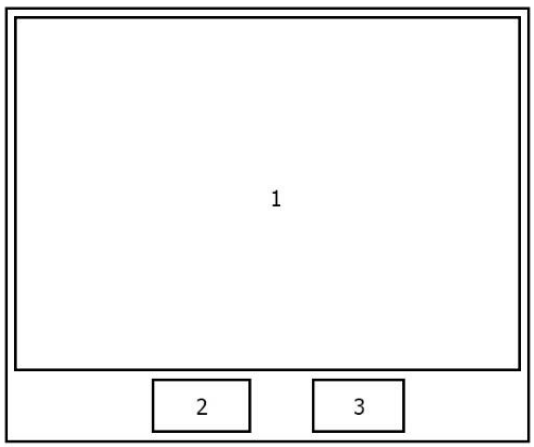


Рисунок 3.12 – Общий вид окна редактирования и добавления данных

На рисунке 3.11: 1 – поля со сведениями о редактируемом или добавляемом объекте; 2 – кнопка сохранения изменения; 3 – кнопка отмены изменений

Таким образом, спроектированный графический интерфейс будет поддерживать многооконный режим, что позволит не перегружать информацией главное окно программы. Главное окно программы, в свою очередь, позволит выполнять предусмотренные программным алгоритмом функции в полном объеме.

3.6 Обзор инструментария проектирования

Современные средства разработки программного обеспечения имеют большое разнообразие критериев, которые при использовании дают разработчику возможность автоматизации процесса разработки приложений.

На данный момент с помощью инструментальных средства можно:

– создавать интерфейс с использованием стандартных компонентов;

– передавать управление различным процессам, в зависимости от состояния системы;

– создавать как оболочки для баз данных, так и сами базы данных;

– разрабатывать более надежное программное обеспечение, с помощью обработки исключительных ситуаций, которые возникают из-за некорректной работы программного обеспечения.

Современные средства разработки характеризуются следующими параметрами:

– поддержка объектно-ориентированного стиля программирования;

– возможность использования CASE-технологий, как для проектирования разрабатываемой системы, так и для разработки моделей реляционных баз данных;

– использование визуальных компонент для наглядного проектирования интерфейса; – поддержка баз данных;

– возможность использовать алгоритмы реляционной алгебры для управления реляционными базами данных.

При создании программного средства «Store», использовались следующие критерии выбора разработки программного средства:

– быстрота разработки приложений;

– возможность создания приложения для Windows;

– перспективность платформы, разрабатываемого приложения;

– легкость создания дружественного интерфейса, причем как стандартного, так и не стандартного;

– простота и удобство, эффективность работы при создании форм представления данных;

– надежность работы среды разработки.

Программа эксплуатируется на персональном компьютере (ПК) типа IBM PC/AT. Для работы в диалоговом режиме используется экран дисплея, клавиатура и манипулятор типа «мышь», а для печати отчетов и чека - принтер.

В качестве языка для создания автоматизированной системы был выбран язык программирования «C#».

C# – объектно-ориентированный язык программирования (ОО-язык). Разработан как язык разработки приложений для платформы Microsoft .NET Framework. [11]

Преимуществом создания нового объектно-ориентированного языка программирования (ОО-языка) по сравнению с расширением существующих заключается в том, что при создании нового языка нет необходимости заботиться о проблемах обратной совместимости, которые обычно заметно затрудняют исправление застарелых проблем и даже внесение новых свойств в стандарт языка [12].

Многие существующие языки программирования обладают весьма запутанным синтаксисом и конструкциями с неочевидной семантикой. C# занимает некоторую промежуточную позицию: из стандарта языка убраны наиболее неприятные и неоднозначные особенности языка программирования C++, но в то же время язык сохранил мощные выразительные возможности, присущие для таких ОО-языков, как C++, Java или VB. По умолчанию, C# запрещает прямое манипулирование памятью, предоставляя взамен богатую

систему типов и сборку мусора. Непосредственная работа с памятью по-прежнему доступна в специальном режиме «опасного» кода, но требует явного декларирования. Как следствие, в C# активно используется всего один оператор доступа «.». Преобразования типов в C# значительно строже, чем в C++, большинство преобразований может быть совершено только явным образом. Кроме того, все приведения должны быть безопасными (т.е. запрещены неявные преобразования с переполнением, использование целых переменных как указателей и т.п.).

Возможно, самое важное, что необходимо сказать про язык C#, – это то, что он генерирует код, предназначенный для выполнения только в среде выполнения .NET.

.NET Framework – это новая и революционная платформа, созданная компанией Microsoft для разработки приложений [13].

Один из основных мотивов создания данной технологии – то, что она предназначается для объединения разнородных операционных систем (ОС).

Более того, приведенное выше определение .NET Framework не содержит никаких ограничений относительно типов приложений, создание которых она поддерживает.

Технология .NET Framework разрабатывалась таким образом, чтобы ее можно было использовать из любого языка программирования: C#, C++, Visual Basic, JScript и даже более старых языков, таких как COBOL.

.NET Framework состоит из огромной библиотеки программ, к которой можно обращаться из различных языков программирования с помощью различных технологий объектно-ориентированного программирования (ООП). Эта библиотека разбита на несколько различных модулей таким образом, что имеется возможность использовать ту или иную ее часть в зависимости от требуемых результатов.

Основная идея в данном случае заключается в том, что различные ОС могут поддерживать некоторые или все из этих модулей.

Приложениям может потребоваться доступ к базам данных, что осуществляется с помощью раздела .NET Framework, называемого Active Data Objects.NET (ADO.NET). Также можно использовать и многие другие ресурсы, например, инструменты для создания сетевых компонентов, графического вывода, выполнения сложных математических вычислений в т. д.

Для информационного обеспечения разрабатываемой системы необходимо использование базы данных, которая будет создана с помощью системы управления базами данных (СУБД) Microsoft SQL Server.

Microsoft SQL Server характеризуется такими особенностями как:

– производительность – Microsoft SQL Server работает очень быстро;

– надежность и безопасность – Microsoft SQL Server предоставляет шифрование данных;

– простота – с данной СУБД относительно легко работать и вести администрирование.

Для организации взаимодействия приложения с БД целесообразно использование языка T-SQL (Transact Structured Query Language) или сокращенно SQL.

SQL – это специализированный непроцедурный язык, позволяющий описывать данные, осуществлять выборку и разработку информации из реляционных СУБД [14]. Сам по себе SQL не является языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые увеличивают его функциональность до полноценного языка программирования.

Подключение базы данных к программе осуществляется с помощью технологии ADO.NET.

Модель ADO.NET – большой шаг вперед в области технологий доступа к данным Microsoft. Она предоставляет разработчику беспрецедентные возможности для управления взаимодействием кода и данных [15].

ADO.NET – это набор библиотек, поставляемый с Microsoft .NET Framework и предназначенный для взаимодействия с различными хранилищами данных из .NET-приложений. Библиотеки ADO.NET включают классы для соединения с источником данных, выполнения запросов и обработки их результатов. Кроме того, ADO.NET можно использовать в качестве надежного, иерархически организованного, отсоединенного кэша данных для автономной работы с данными. Главный отсоединенный объект, DataSet, позволяет сортировать, осуществлять поиск, фильтровать, сохранять отложенные изменения и перемещаться по иерархичным данным.

В Microsoft Visual Studio .NET есть ряд поставщиков для взаимодействия с той или иной базой данных.

Поставщик данных .NET – это набор классов, предназначенных для взаимодействия с хранилищем данных определенного типа. .NET Framework включает два поставщика – SQL Client .NET Data Provider и OLE DB .NET Data Provider [16]. Поставщик OLE DB .NET Data Provider позволяет взаимодействовать с различными хранилищами данных посредством поставщика OLE DB. Поставщик SQL Client .NET Data Provider рассчитан исключительно на взаимодействие с БД «SQL Server» версии 7 или более поздней.

Таким образом, использование описанных средств разработки позволит в полном объеме осуществить реализацию поставленной задачи разработки ПС.

3.7 Описание программной реализации программного средства

Разрабатываемая АИС обеспечивает функционал, который в наиболее полном объеме позволит автоматизировать работу по управлению магазином.

Она включает в себя следующий перечень функций:

– ведение таблицы номенклатуры;

– учёт прихода и расхода;

– ведение справочных таблиц (ТТН, категорий, сотрудников, должностей, пользователей и ролей);

– формирование отчета реализации за различные периоды;

– формирование товарного чека;

– осуществление выгрузки остатков.

Доступ к системе осуществляется исключительно по логину и паролю. Любые действия пользователя (вход в систему; удаление, добавление и редактирование записей) регистрируются.

Каждый пользователь имеет свой набор функций, который доступен для его учетной записи. Роль пользователя в системе зависит от занимаемой должности. Так, в программе предусмотрено пять уровней прав:

– оператор – пользователь, который обладает данным уровнем прав, может только вводить ТТН;

– продавец – пользователь, который обладает данным уровнем прав, может просматривать таблицу «Номенклатура» и оформлять реализацию;

– старший продавец – пользователь, который обладает данным уровнем прав, может просматривать таблицу «Номенклатура», оформлять реализацию и осуществлять построение отчета реализации;

– заведующий – пользователь, который обладает данным уровнем прав, может просматривать все таблицы (кроме таблицы, которая содержит данные о логинах, правах и паролях всех пользователей), вводить ТТН, оформлять реализацию, формировать отчет реализации, осуществлять выгрузку остатков;

– администратор – пользователь, который обладает данным уровнем прав, получает доступ ко всему функционалу системы.

При работе с системой не должно возникать моментов, когда возникает необработанная исключительная ситуация, которая приводит к закрытию приложения. Все ошибки фиксируются и обрабатываются с помощью блоков, приведённых на рисунке 3.12:

```
try //начало блока отлова
{
    command.ExecuteNonQuery();
}
catch //обработка ошибки
{
    MessageBox.Show("Ошибка редактирования записи. Запись не добавлена", "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
```

Рисунок 3.13 – Пример блока отлова ошибок

Кроме того, при работе со вводом или редактированием ТТН, а также при списании товара, должен обеспечиваться корректный ввод данных в таблицу, т.е. при возникновении ошибки обработки информации на каком-либо этапе, изменения должны быть откачены к предыдущему состоянию. Это осуществляется с помощью сохранения и отката транзакции. На рисунке 3.13 представлен пример сохранения транзакции при успешном добавлении данных при вводе ТТН или отката транзакции при возникновении ошибки.

```
try
{
    command = new SqlCommand(query, myConnection, tran); //связываем запрос с подключением
    tran.Save("save1"); //сохраняем точку отката
    command.ExecuteNonQuery(); //выполняем запрос
}
catch
{
    MessageBox.Show("Ошибка сохранения данных по чеку. Оформление чека Отменено. Обратитесь к администратору.",
                    "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    tran.Rollback("save1"); //откатываеемся к точке сохранения
    tran.Commit(); //фиксируем
    return;
}
tran.Commit(); //фиксируем
```

Рисунок 3.14 – Пример блока отката транзакций

Ввод новых данных в БД и редактирование существующей информации осуществляется при отображенной таблице БД, в которую необходимо добавить запись либо ее изменить. Для этого необходимо нажать кнопку «Добавить» («Изменить») рядом с таблицей. Для внесения информации откроется окно с полями, в которые необходимо внести данные. Для сохранения внесенных данных необходимо нажать кнопку «Ок». Если хотя бы одно обязательно поле будет не введено, кнопка сохранения изменений будет недоступна. Добавление данных в таблицу и редактирование данных в таблице осуществляется посредством SQL-запросов, которые приведен на рисунке 3.14.

```
query = "INSERT INTO Employe(FullName, DOB, Address, Phone, email, CodePosition) " +
            "SELECT '" + employeesFullName.Text + "', '" +
                         employeesDOB.Value.Year + "-" +
                         employeesDOB.Value.Month + "-" +
                         employeesDOB.Value.Day + "', '" +
                         employeesAddress.Text + "', '" +
                         employeesPhone.Text + "', '" +
                         employeesEmail.Text + "', Code FROM Positions WHERE Name = '" +
                         employeesPosition.Text + "'";

query = "UPDATE Employe " +
            " SET FullName = '" + employeesFullName.Text + "'," +
              " DOB = '" + employeesDOB.Value.Year + "-" +
                           employeesDOB.Value.Month + "-" +
                           employeesDOB.Value.Day + "'," +
              " Address = '" + employeesAddress.Text + "'," +
              " Phone = '" + employeesPhone.Text + "'," +
              " email='" + employeesEmail.Text + "'," +
              " CodePosition=(SELECT Code " +
                             "FROM Positions " +
                             "WHERE Name='" + employeesPosition.Text + "') " +
            " WHERE Code =" + employeesCode.Text;
```

Рисунок 3.15 – SQL-запрос добавления и редактирования данных

Для удаления записи из таблицы, необходимо выделить нужную запись и нажать кнопку «Удалить», а затем согласиться с удалением. Удаление записи из таблицы также осуществляется посредством SQL-запроса, приведенного на рисунке 3.15.

```
string query = "DELETE FROM " + table +
               " Where Code = " + code;
```

Рисунок 3.16 – SQL-запрос удаления данных

Формирование отчетов осуществляется на основе представлений. Для корректного формирования отчета с заданными параметрами, старое представление удаляется из базы данных, а затем создается новое с заданными параметрами. Код создания представления приведен на рисунке 3.16.

```
string query = "IF OBJECT_ID('ReportUnloadingOfResidues') IS NOT NULL " +
                          "DROP VIEW ReportUnloadingOfResidues;";
query = "CREATE VIEW ReportUnloadingOfResidues AS " +
        "SELECT Category.Name AS Category, " +
                "Nomenclature.Name AS Nomenclature, " +
                "Nomenclature.Balance AS ProductBalance, " +
                "Nomenclature.SellingPrice AS SellingPrice,  " +
                "ROUND(Nomenclature.Balance * Nomenclature.SellingPrice, 2) AS Cost " +
        "FROM Nomenclature, Category " +
        "WHERE Nomenclature.CodeCategory = Category.Code and Nomenclature.Balance > 0";
```

Рисунок 3.17 – SQL-запрос создания представлений

Полный листинг программы приведен в приложении А.

В процессе разработки осуществлялась работа с большим количеством переменных, т.к. не все значения можно было занести в поля компонентов. В таблице 3.10 приведен список переменных, которые использовались для тех или иных целей в процессе разработки программы.

Таблица 3.10 – Список используемых переменных

Наименование	Тип	Описание
1	**2**	**3**
role1	String	Роль пользователя
user1	String	Логин пользователя
connectionString	String	Строка подключения к БД
fl	String	Флаг действия с БД
query	String	Текст запроса к БД
myConnection	SqlConnection	Объект, предоставляющий сеанс связи с источником данных SQL Server
comboBoxName	List<ComboBox>	Ссылки на динамически созданные «ComboBox» для хранения имен номенклатуры
comboBoxNameCheck		
textBoxUnitPrice	List<TextBox>	Ссылки на динамически созданные «TextBox» для хранения стоимости за единицу, общей стоимости соответственно
textBoxUnitPriceCheck		
textBoxTotalCost		
textBoxTotalCostCheck		
codeNom	List<int>	Хранение кодов номенклатуры, сотрудников соответственно
codeEmploye		
countNomenclature	int	Количество введенных данных в ТТН, Чек и сотрудников
countNomenclatureCheck		
countEmp		
y	int	Переменные-счетчики
I		
count		
command	SqlCommand	Экземпляр класса, который инкапсулирует sql-выражение, которое должно быть выполнено
reader	SqlDataReader	Переменная для получения данных

Продолжение таблицы 3.10

sum	Double	Подсчет суммы
CB	ComboBox	Создание ComboBox
TB	TextBox	Создание TextBox
NUD	NumericUpDown	Создание NumericUpDown
DTP	DateTimePicker	Создание DateTimePicker
rand	Random	Генерация случайного числа

Для обработки тех или иных данных в программе помимо переменных используются также события компонентов. В таблице 3.11 приведен перечень используемых событий и их назначение.

Таблица 3.11 – Используемые события компонентов

Наименование события	Назначение
FormClosed	Выполнение запрограммированных действий при закрытии формы
SelectedIndexChanged	Выполнение запрограммированных действий при выборе пункта в «ComboBox»
DrawItem	Отрисовка имен горизонтальных вкладок
Load	Выполнение запрограммированных действий при загрузке формы
ChangeCheck	Выполнение запрограммированных действий при установке или удалении флажка у компонента «CheckBox»
TextChange	Выполнение действий при изменении текста в «ComboBox» или «TextBox»
MouseClick	При нажатии на вкладку выполняются запрограммированные на ней операции
OnClick	При нажатии на кнопку выполняются запрограммированные на ней операции

Таким образом, описанная выше совокупность переменных, свойств компонентов и функций в полном объеме осуществляет решение задачи.

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Для вызова программы нужно скачать на свой ПК папку с программой, которая находится на диске под названием «Store» (рисунок 4.1)

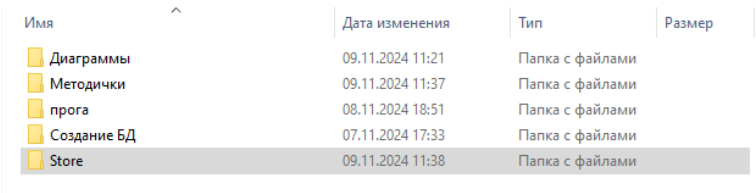


Рисунок 4.1 – Папка с программой

Запуск приложения можно осуществить при двойном клике на файле под названием «Store.exe» (рисунок 4.2).

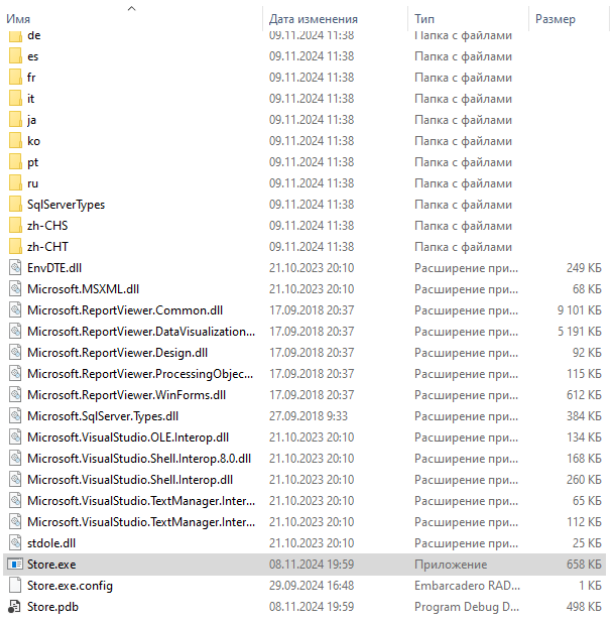


Рисунок 4.2 – Файл с программой

После запуска компьютерная система автоматически подключится к БД. Если по каким-либо причинам база данных недоступна, то пользователю будет выдано сообщение, изображенное на рисунке 4.3. Если же доступ к БД есть, то пользователь должен пройти авторизацию (рисунок 4.4), путем ввода логина и пароля. Если логин или пароль введен неверно, то пользователю будет выдано сообщение, изображенное на рисунке 4.5. Если же логин и пароль введен верно, то пользователь получит доступ к следующему окну.

Набор доступных вкладок и действий зависит от роли пользователя. На рисунках (рисунок 4.6) изображено главное окно программы под правами администратора.

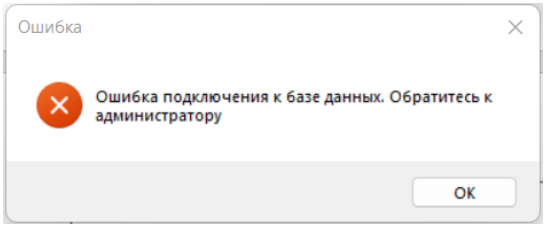


Рисунок 4.3 – Сообщение, выдаваемое, если база данных недоступна

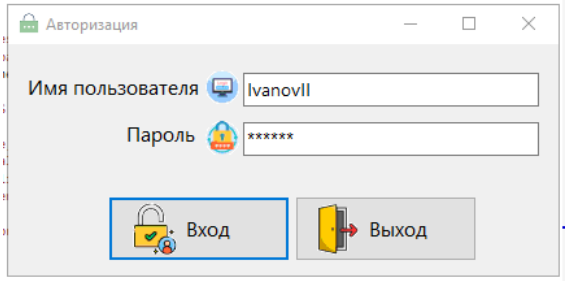


Рисунок 4.4 – Окно авторизации

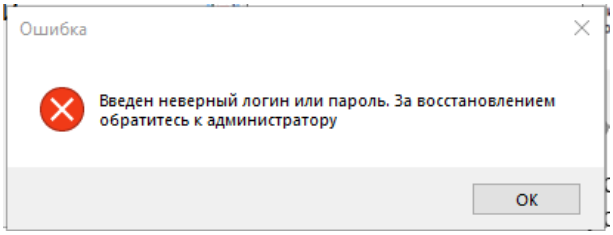


Рисунок 4.5 – Сообщение, выдаваемое, если введены неверные логин или пароль

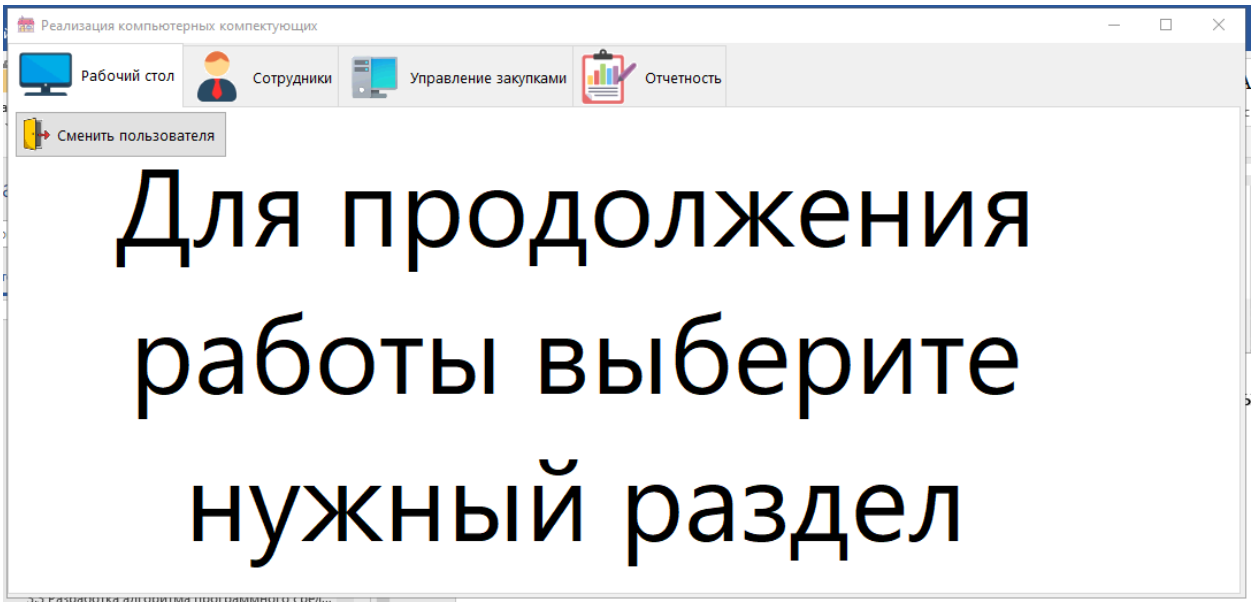


Рисунок 4.6 – Главное окно программы для роли «Администратор»

Вкладка «Сотрудники» предназначена для управления сотрудниками магазина.

На рисунке 4.7 изображено содержание данного пункта.

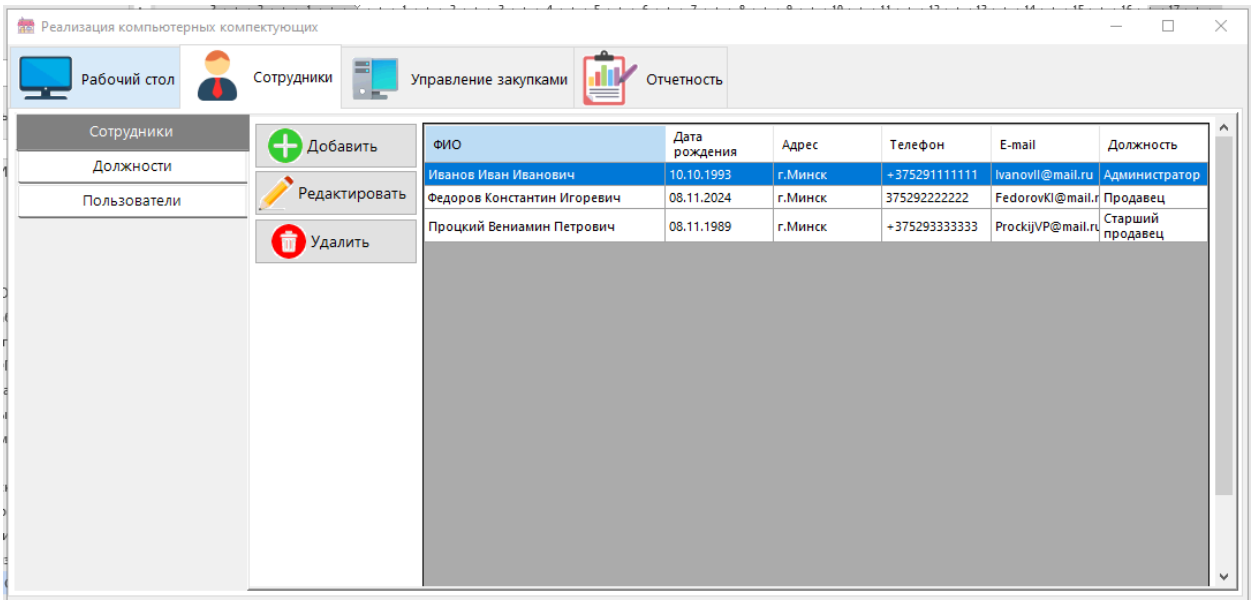


Рисунок 4.7 – Раздел «Сотрудники»

Для добавления записи в таблицу «Сотрудники» необходимо нажать кнопку «Добавить», после чего ввести нужные данные (рисунок 4.8).

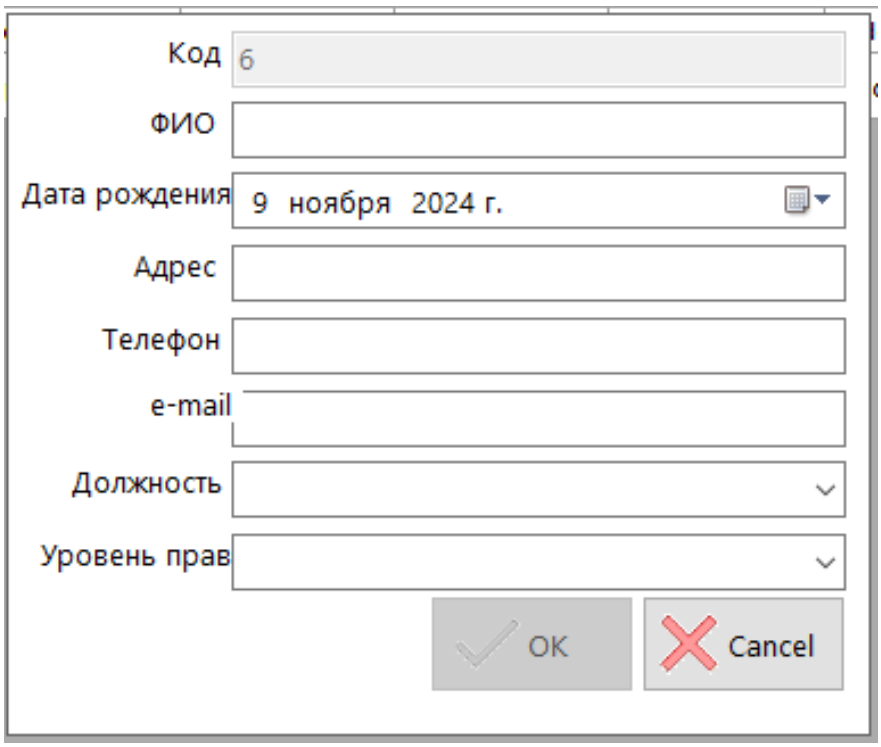


Рисунок 4.8 – Ввод нового сотрудника

После нажатия кнопки «Ок», будет автоматически сгенерирован логин и пароль для входа в систему (рисунок 4.9). Если же логин совпал с логином другого сотрудника, то будет предложено ввести логин самостоятельно (рисунок 4.10).

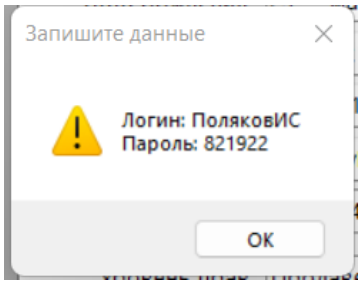


Рисунок 4.9 – Сообщение со сгенерированным логином и паролем

Рисунок 4.10 – Панель ввода логина

Если же при добавлении записи произошла ошибка, то пользователю будет выдано сообщение, изображенное на рисунке 4.11.

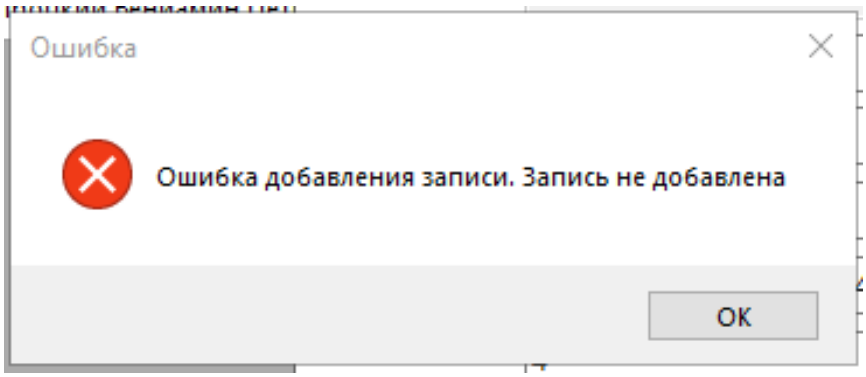


Рисунок 4.11 – Ошибка при добавлении записи в таблицу

Для редактирования таблицы необходимо нажать кнопку «Редактировать». Данные редактируемой записи автоматически внесутся в соответствующие поля на панели редактирования. Панель редактирования выглядит аналогично панели добавления, изображенной на рисунке 4.8.

Для удаления записи необходимо выделить нужную запись, затем нажать кнопку «Удалить» и согласиться с удалением (рисунок 4.12).

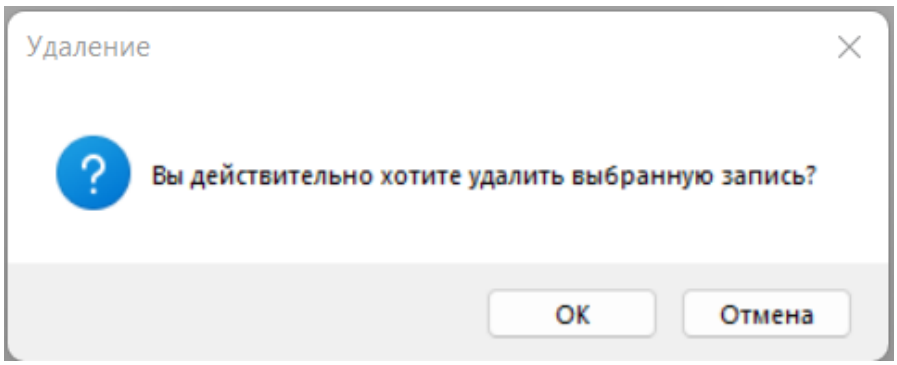


Рисунок 4.12 – Запрос согласия на удаление записи

Аналогично редактируются, изменяются и удаляются записи из других таблиц базы данных.

Вкладка «Управления закупками» предназначена для работы с товаром. На рисунке 4.13 изображено ее содержание.

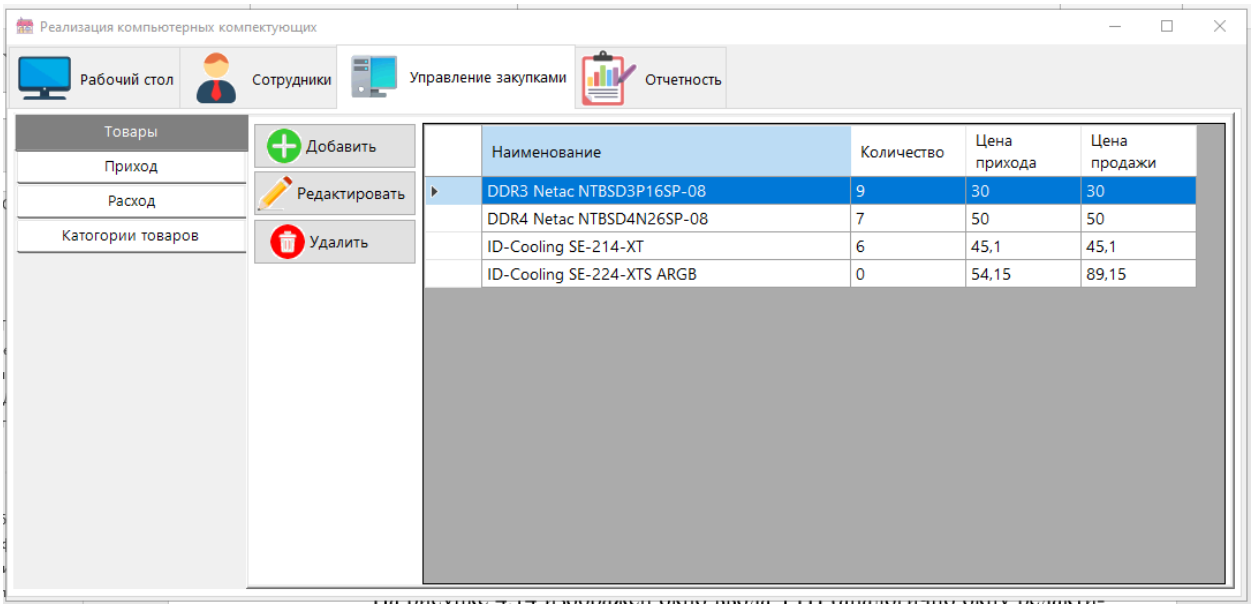


Рисунок 4.13 – Раздел «Управление закупками»

Основными пунктами в данной вкладке является приход и расход.

На рисунке 4.14 изображен окно ввода ТТН (аналогично окну редактирования ТТН).

Ввод ТТН необходим для: учета движения товара, контроля за грузом, формирование отчетности, упрощение документооборота и подтверждение сделки.

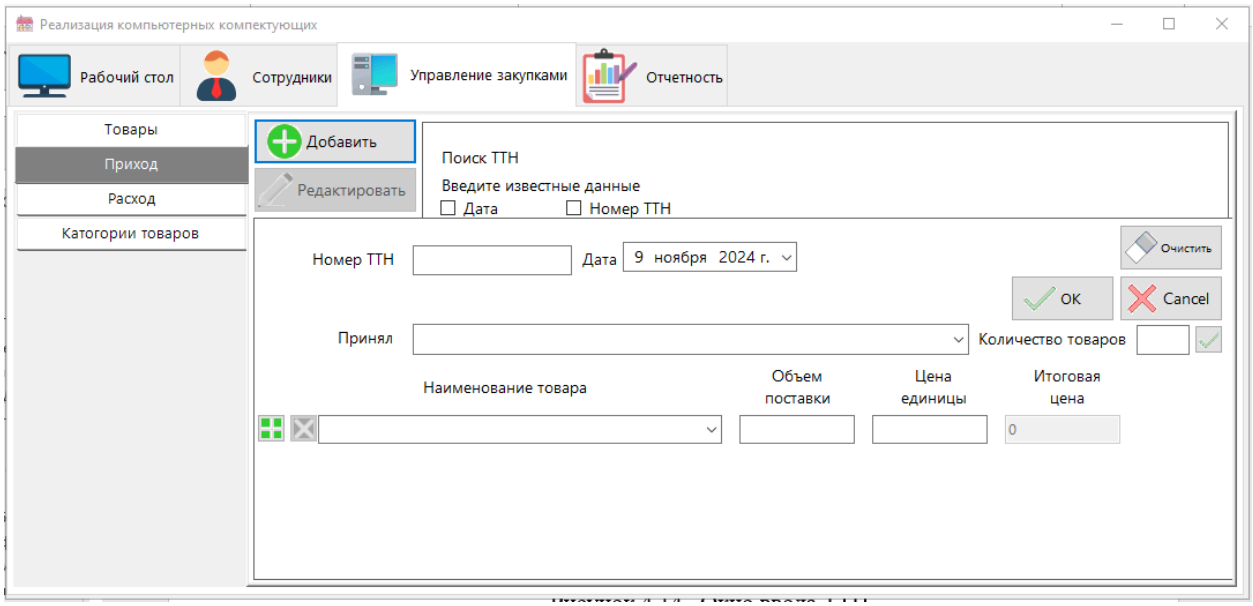


Рисунок 4.14 – Окно ввода ТТН

Для просмотра или редактирования ТТН, необходимо ее найти. На рисунке 4.15 изображено окно поиска ТТН.

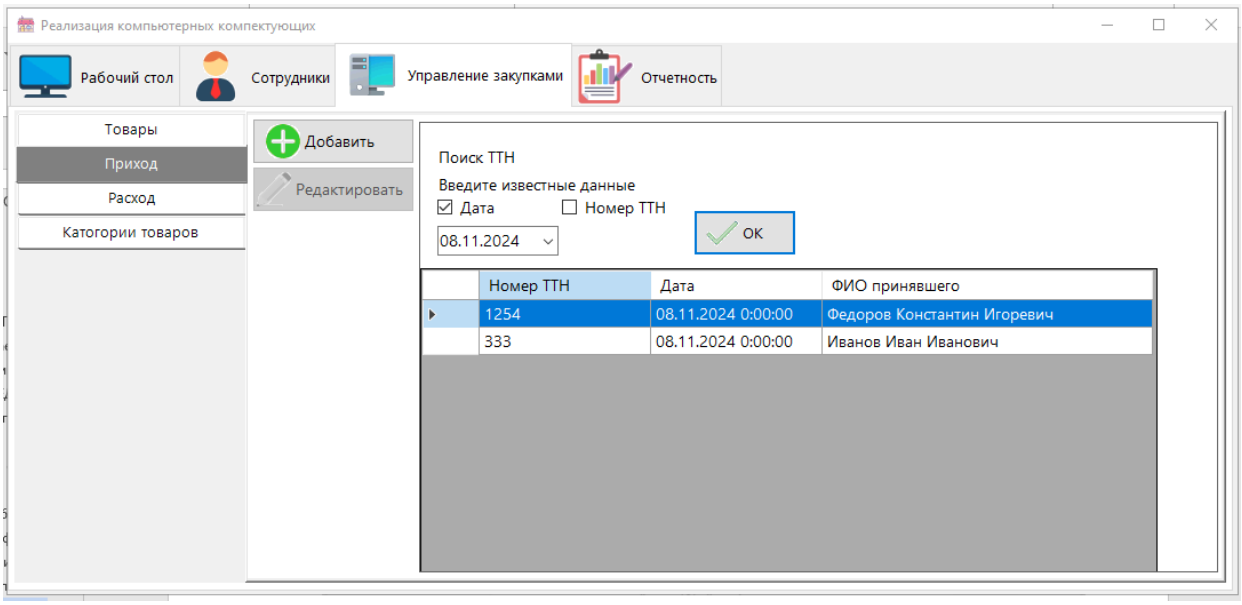


Рисунок 4.15 – Окно поиска ТТН

На рисунке 4.16 изображено окно продажи (списания средств), а на рисунке 4.17 сформированный товарный чек.

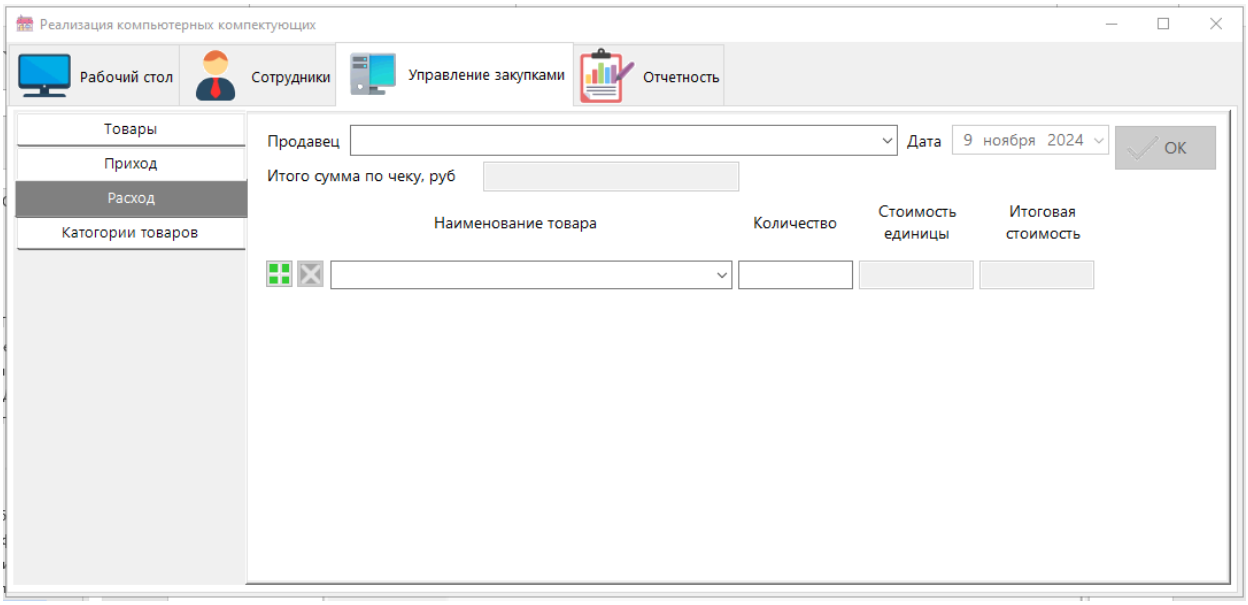


Рисунок 4.16 – Окно оформления продажи

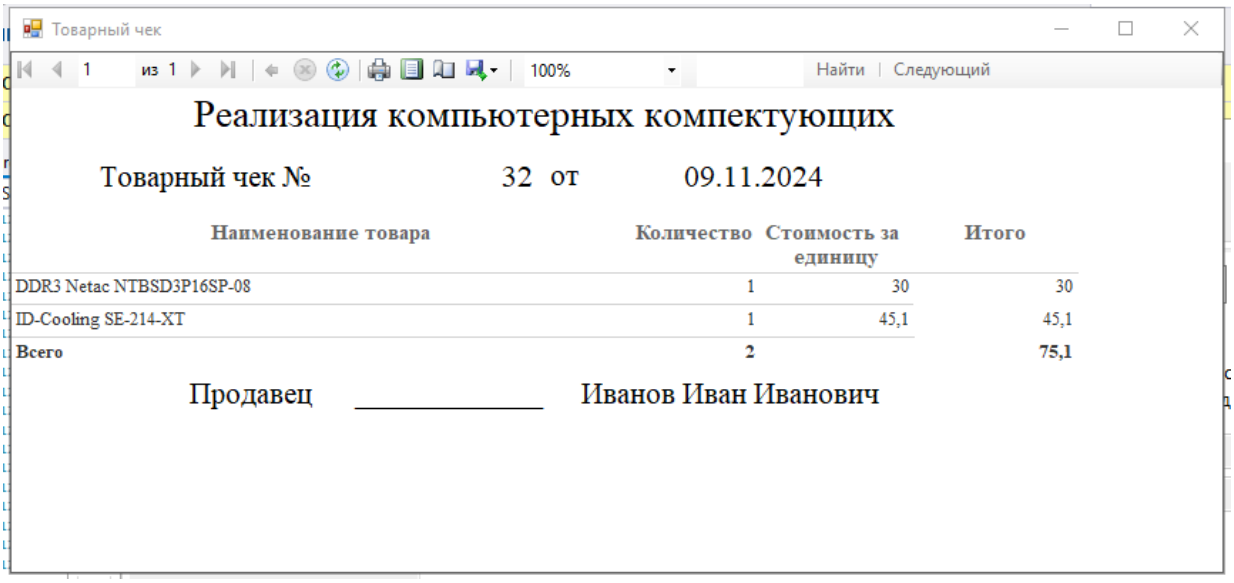


Рисунок 4.17 – Сформированный товарный чек

Вкладка «Отчетность» предназначена для построения отчета по реализации и выгрузки остатков.

Выгрузка остатков осуществляется автоматически по выбору вкладки «Выгрузка остатков».

На рисунке 4.18 изображен отчет «Выгрузка остатков».

Она нужна для того чтобы анализировать запасы и планировать закупки. А также отслеживать продажи и упрощать взаимодействие.

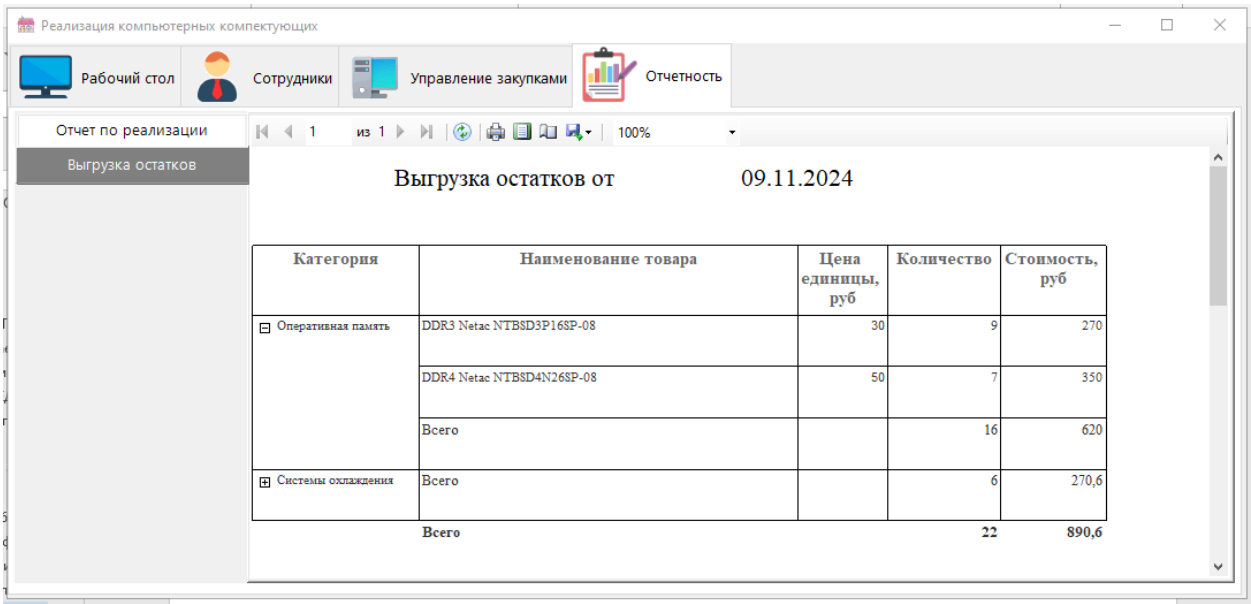


Рисунок 4.18 – Сформированный отчет «Выгрузка остатков»

На рисунке 4.19 изображен построенный отчет по реализации.

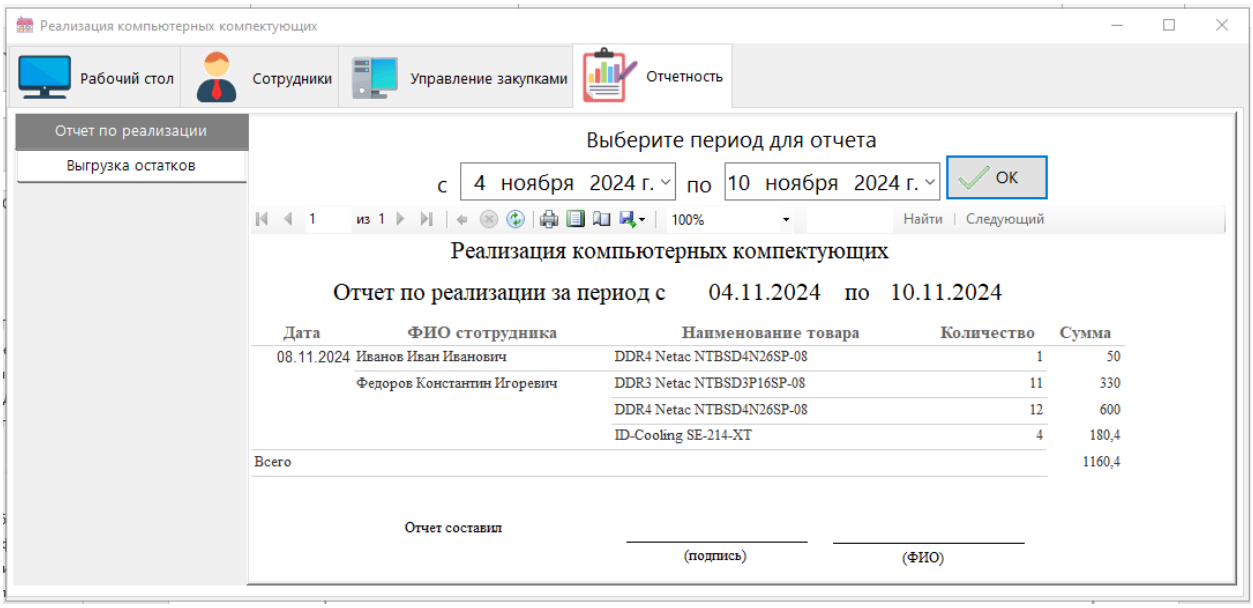


Рисунок 4.19 – Отчет по реализации

Таким образом, произведя пробную эксплуатацию ПО, подтвердилась его корректная работа, а также то, что спроектированное ПС позволяет решить поставленную задачу.

5 РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ ПРОГРАММНОГО СРЕДСТВА

5.1 Введение

Программное средство «Store» предназначена для облегчения работы сотрудника, который продает компьютерные комплектующие. Основные его функции:

– ведение таблицы номенклатуры;

– учёт прихода и расхода;

– ведение справочных таблиц (ТТН, категорий, сотрудников, должностей, пользователей и ролей);

– формирование отчета реализации за различные периоды;

– формирование товарного чека;

– осуществление выгрузки остатков.

ПС не требует от пользователя глубоких знаний персонального компьютера, проста в освоении. Для работы с ней пользователю достаточно ознакомиться с руководством пользователя.

ПС используется в работе сотрудника, который продает компьютерные комплектующие, и предназначено для облегчения работы с документацией и повседневными выполняемыми функциями.

Для корректной работы программы компьютер должен обладать следующими минимальными характеристиками:

– процессор с тактовой частотой 1000 МГц;

– оперативная память не менее 1024 MB;

– свободное место на жестком диске не менее 20 MB;

– «Windows 7» и выше.

А также иметь следующие аппаратные и программные обеспечения:

– манипулятор мыши;

– клавиатура.

Для корректного запуска программы база данных (структура описана ниже) должна находиться на сервере со следующей конфигурацией:

– MS SQL Server 2014 и выше;

– название сервера: SQLEXPRESS;

– название базы данных: Store;

– режим аутентификации: проверка подлинности Windows.

5.3 Описание операций

Главное окно программы (в зависимости от роли количество отображаемых пунктов меняется):

1. Пункт «Рабочий стол» – с рабочего стола можно выйти в окно авторизации с помощью кнопки «Сменить пользователя» (рисунок 5.2).

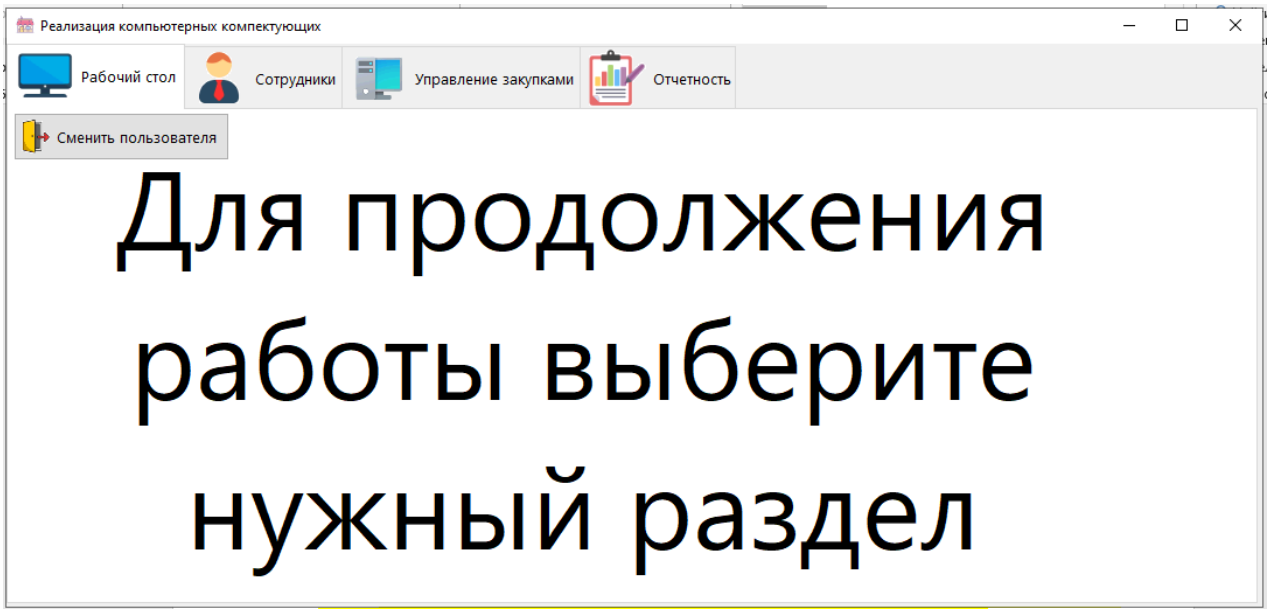


Рисунок 5.1 – Пункт меню «Рабочий стол»

2. Пункт меню «Сотрудники»» – предназначен для доступа к подменю «Сотрудники», «Должности» и «Пользователи» (рисунок 5.3).

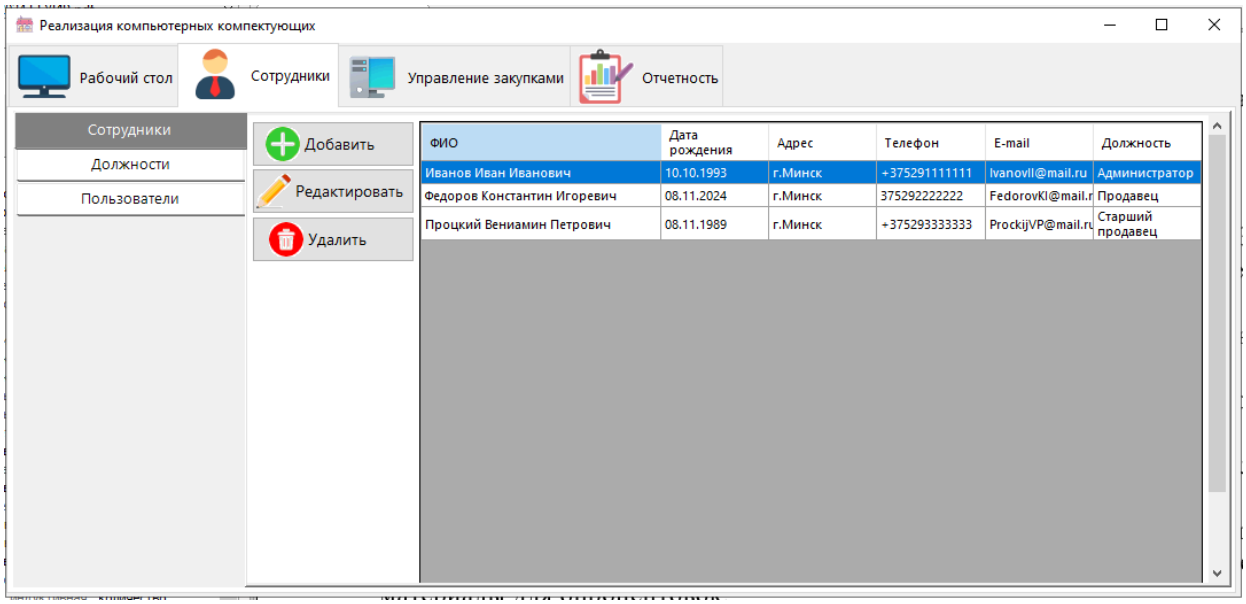


Рисунок 5.2 – Пункт меню «Сотрудники»

Подменю «Сотрудники» – позволяет работать с таблицей «Сотрудники». Кроме области вывода данных, оно имеет три кнопки:

– «Добавление» – открывает панель добавления данных;

– «Редактирование» – открывает панель редактирования данных;

– «Удаление» – выдает запрос на согласие на удаление. Пользователь может согласиться, и тогда запись будет удалена, или отказаться – тогда запись будет оставлена.

Окно редактирования и добавления данных (рисунок 5.4):

– «ОК» – подтверждает добавление записи или внесенные изменения;

– «Cancel» – отменяет внесение изменений.

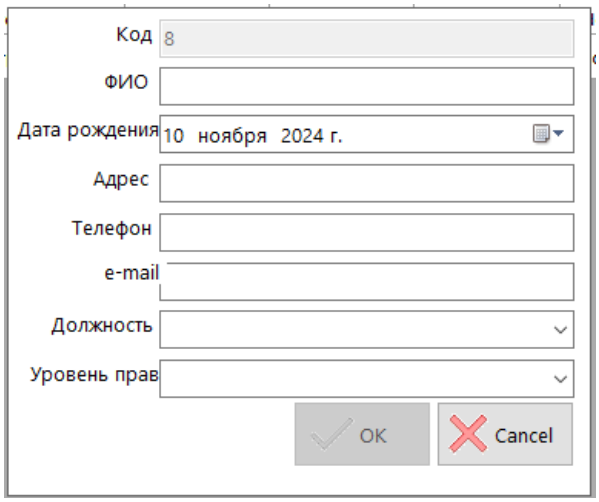


Рисунок 5.3 – Панель добавления и редактирования

Подменю «Должности» – позволяет работать с таблицей «Должности». Кроме области вывода данных, оно имеет три кнопки:

– «Добавление» – открывает панель добавления данных;

– «Редактирование» – открывает панель редактирования данных;

– «Удаление» – выдает запрос на согласие на удаление. Пользователь может согласиться, и тогда запись будет удалена, или отказаться – тогда запись будет оставлена.

Окно редактирования и добавления данных (рисунок 5.5):

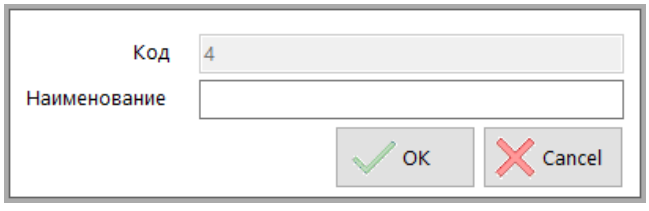


Рисунок 5.4 – Панель добавления и редактирования

Подменю «Пользователи» – позволяет работать с таблицей «Пользователи». Кроме области вывода данных, оно имеет три кнопки:

– «Добавление» – открывает панель добавления данных;

– «Редактирование» – открывает панель редактирования данных;

– «Удаление» – выдает запрос на согласие на удаление. Пользователь может согласиться, и тогда запись будет удалена, или отказаться – тогда запись будет оставлена.

Окно редактирования и добавления данных (рисунок 5.6):

– «ОК» – подтверждает добавление записи или внесенные изменения;

– «Cancel» – отменяет внесение изменений;

– «Сбросить пароль» – сбрасывает пароль пользователя и генерирует новый.

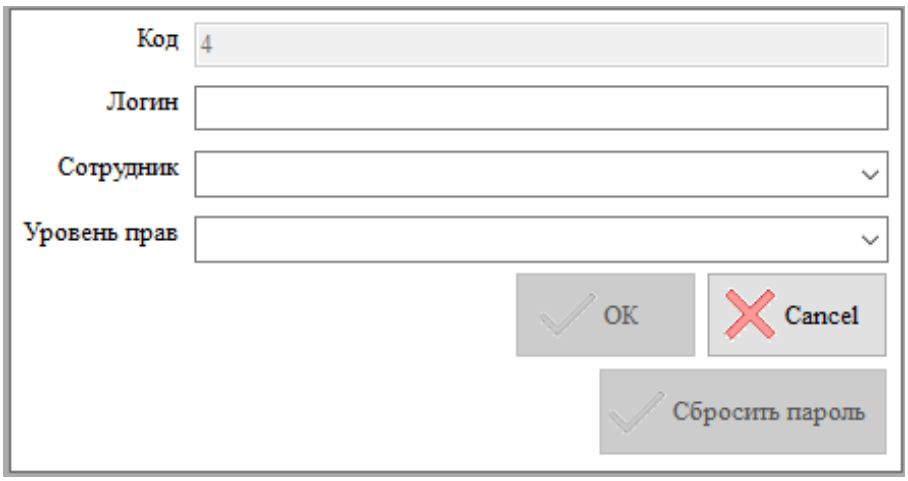


Рисунок 5.5 – Панель добавления и редактирования

3. Пункт меню «Управление закупками»» – предназначен для доступа к подменю «Товары», «Приход», «Расход» и «Категории товаров» (рисунок 5.7).

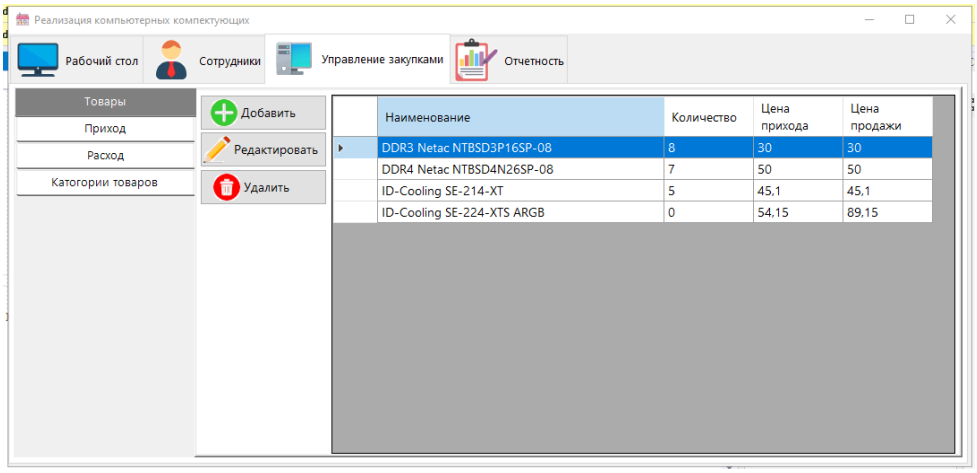


Рисунок 5.7 – Пункт меню «Управление закупками»

Подменю «Товары» – позволяет работать с таблицей «Товары». Кроме области вывода данных, оно имеет три кнопки:

– «Добавление» – открывает панель добавления данных;

– «Редактирование» – открывает панель редактирования данных;

– «Удаление» – выдает запрос на согласие на удаление. Пользователь может согласиться, и тогда запись будет удалена, или отказаться – тогда запись будет оставлена.

Окно редактирования и добавления данных (рисунок 5.8):

– «ОК» – подтверждает добавление записи или внесенные изменения;

– «Cancel» – отменяет внесение изменений.

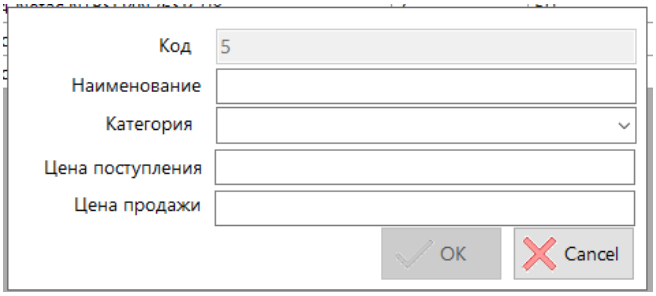


Рисунок 5.8 – Панель добавления и редактирования

Подменю «Приход» – позволяет оформлять приход товара. Из данного подменю можно получить доступ к трем функциям: добавление, поиск и редактирование ТТН.

Для поиска ТТН необходимо выбрать критерий поиска (дата или номер ТТН), а затем ввести данные для поиска и нажать кнопку «ОК» (рисунок 5.9).

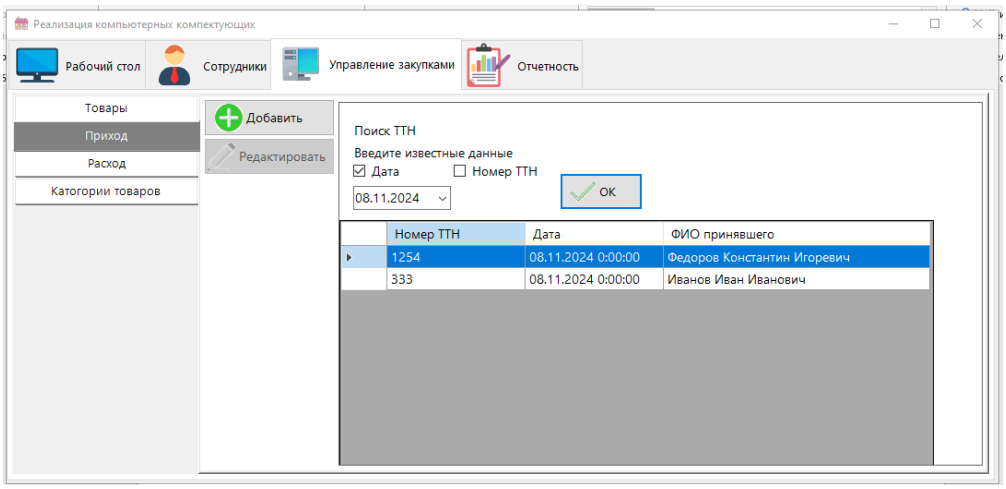


Рисунок 5.9 – Поиск ТТН

Для редактирования ТТН необходимо сперва ее найти, затем выделить нужную в списке и нажать кнопку «Редактировать».

Для добавления данных по ТТН – необходимо нажать кнопку «Добавить».

Окно редактирования и добавления данных (рисунок 5.10):

– «Назад» – возвращает назад к поиску;

– «Очистить» – очищает панель добавления/редактирования;

– «Ок» – подтверждает сохранение данных;

– «Cancel» – отменяет внесенные изменения;

– кнопка « ✓ » – вводит количество товаров по накладной;

– кнопка «+» – добавляет строку для ввода товара;

– кнопка «х» – удаляет строку для ввода товара.

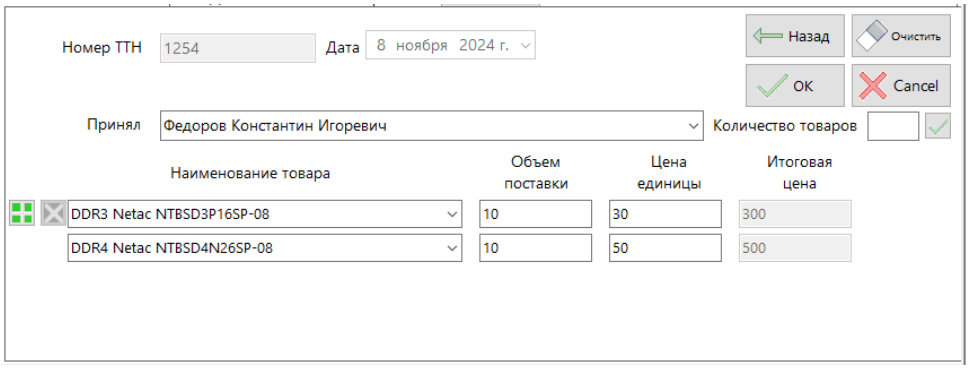


Рисунок 5.10 – Панель добавления и редактирования данных по ТТН

Подменю «Расход» – вводит реализацию товара (рисунок 5.11):

– «Ок» – подтверждает сохранение данных;

– кнопка «+» – добавляет строку для ввода товара;

– кнопка «х» – удаляет строку для ввода товара.

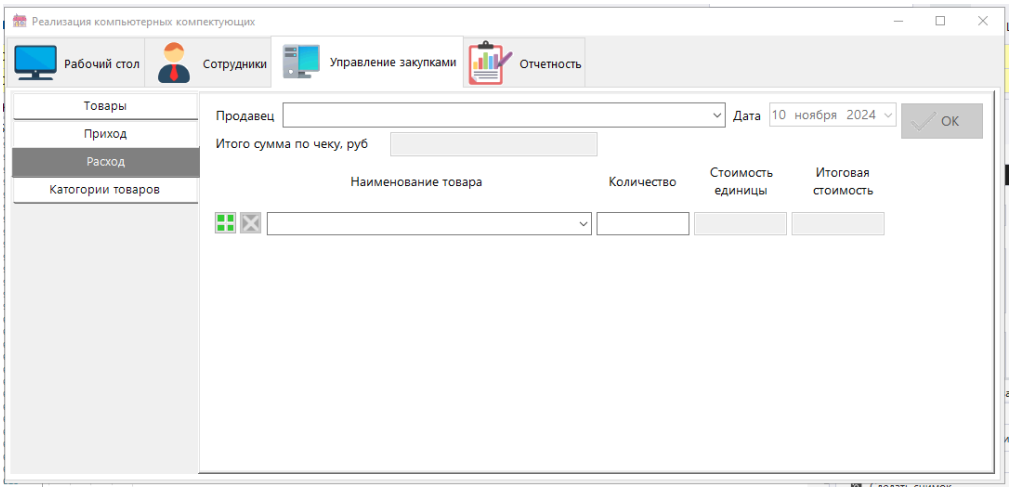


Рисунок 5.11 – Панель добавления расхода

Подменю «Категории товаров» – позволяет работать с таблицей

«Категории товаров». Кроме области вывода данных, оно имеет три кнопки:

– «Добавление» – открывает панель добавления данных;

– «Редактирование» – открывает панель редактирования данных;

– «Удаление» – выдает запрос на согласие на удаление. Пользователь может согласиться, и тогда запись будет удалена, или отказаться – тогда запись будет оставлена.

Окно редактирования и добавления данных (рисунок 5.12):

– «ОК» – подтверждает добавление записи или внесенные изменения;

– «Cancel» – отменяет внесение изменений.

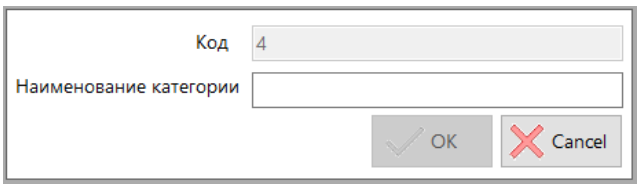


Рисунок 5.12 – Панель добавления и редактирования

4. Пункт меню «Отчетность» – позволяет строить отчет по реализации и выгрузку остатков.

Выгрузка остатков происходит автоматически при выборе советующего подменю.

6 ТЕХНИКО – ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА

6.1 Описание функций, назначения и потенциальных пользователей программного средства

«Store» – это программный продукт для продажи компьютерных комплектующих.

Программное средство используется в работе сотрудника магазина для облегчения работы с документацией и повседневными выполняемыми функциями.

Затраты на основную заработную плату команды разработчиков определяются исходя из состава и численности команды, размеров месячной заработной платы каждого из участников команды, а также общей трудоемкости разработки программного средства.

Расчет величины основной заработной платы участников команды осуществляется по формуле:

$$З_о = \sum_{i-1}^{n} З_{ч_i} t_i \qquad (6.1)$$

где n – количество исполнителей, занятых разработкой конкретного ПС;

$З_{ч_i}$ – часовая заработная плата i-го исполнителя (руб.);

t_i – трудоемкость работ, выполняемых i-м исполнителем (ч).

Данные по заработной плате команды разработчиков предоставлены на 01.11.2024 г.

Расчет основной заработной платы команды разработчиков представлен в таблице 6.1.

Таблица 6.1 – Расчет основной заработной платы

№	Участник команды	Вид выполняемой работы	Месячная заработная плата, руб.	Часовая заработная плата, руб.	Трудоемкость работ, ч.	Зарплата по тарифу, руб.
1	**2**	**3**	**4**	**5**	**6**	**7**
2	Руководитель	Анализ предметной области и актуальности	2750	16,37	16	261,92
3	Тестировщик	Тестирование ПС	1250	7,44	8	59,52
Премия (50%)						1160,88

Продолжение таблицы 6.1

Итого затраты на основную заработную плату, руб	3482,64

Затраты на дополнительную заработную плату команды разработчиков включает выплаты, предусмотренные законодательством о труде (оплата трудовых отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по формуле:

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (6.2)$$

где $З_о$ – затраты на основную заработную плату, (руб.);

$Н_д$ – норматив дополнительной заработной платы (10%).

Дополнительная заработная плата составит:

$$З_д = \frac{3482{,}64 * 10}{100} = 348{,}26 \text{ (руб.)}$$

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

$$Р_{соц} = \frac{(З_о + З_д) \cdot Н_{соц}}{100}, \quad (6.3)$$

где $Н_{соц}$ – норматив отчислений на социальные нужды, % (согласно действующему законодательству, принимает равным 34,6% (34% – социальное страхование, 0,6% – обязательное страхование)).

$$Р_{соц} = \frac{(3482{,}64 + 348{,}26) \cdot 34{,}6}{100} = 1325{,}49 \text{ (руб.)}$$

Прочие затраты включаются в себестоимость разработки ПС в процентах от затрат на основную заработную плату команды разработчиков (таблица 6.1) по формуле:

$$З_{пз} = \frac{(З_о \cdot Н_{пз})}{100}, \quad (6.4)$$

где $Н_{пз}$ – норматив прочих затрат (120%).

$$З_{пз} = \frac{(3482{,}64 \cdot 120)}{100} = 4179{,}17 \text{ (руб.)}$$

Полная сумма затрат на разработку программного средства находится путем суммирования всех рассчитанных статей затрат (таблица 6.2).

Таблица 6.2 – Затраты на разработку программного средства

Статья затрат	Сумма, руб.
Основная заработная плата команды разработчиков	3482,64
Дополнительная заработная плата команды разработчиков	348,26
Отчисления на социальные нужды	1325,49
Прочие затраты	4179,17
Общая сумма затрат на разработку	9335,56

6.2 Экономический эффект от разработки ПС по индивидуальному заказу

На рынке существует множество специализированных организаций, способных предложить схожее ПС.

В данном случае рассчитаем цену продукта по формуле:

$$Ц = З_р + П + НДС \quad (6.5)$$

Прибыль, включаемая в цену, рассчитывается по формуле:

$$П = \frac{З_р \cdot У_р}{100\%}, \quad (6.6)$$

где $З_р$ – затраты на разработку и реализацию ПС;

$У_р$ – запланированный норматив рентабельности, % Примем $У_р$ равным 27% .

$$П = \frac{9335{,}56 * 27}{100} = 2520{,}60 \text{ (руб.)}$$

Налог на добавленную стоимость при таком подходе к ценообразованию рассчитывается по формуле:

$$НДС = \frac{(З_р + П) \cdot Н_{дс}}{100}, \quad (6.7)$$

где $Н_{дс}$ – ставка налога на добавленную стоимость согласно действующему законодательству (20%).

$$НДС = \frac{(9335{,}56 + 27) \cdot 20}{100} = 1872{,}51 \text{ (руб.)}$$

Тогда цена продукта равна:

$$Ц = 9335{,}56 + 2520{,}60 + 1872{,}51 = 13\,728{,}67 \text{ (руб.)}$$

Организация является плательщиком налога на прибыль, экономический эффект рассчитывается по формуле

$$П_ч = П - \frac{П \cdot Н_п}{100}, \quad (6.8)$$

где $П_ч$ – это чистая прибыль, получаемая организацией-разработчиком от реализации данного ПС;

$Н_п$ – ставка налога на прибыль (20%)

$$П_ч = 2520{,}60 - \frac{2520{,}60 \cdot 20}{100} = 2016{,}6$$

Проект будет экономически эффективным, если рентабельность затрат на разработку программного средства будет не меньше средней процентной ставки по банковским депозитным вкладам. Рентабельность затрат на разработку ПС рассчитывается по формуле

$$Р = \frac{П_ч}{З_р} * 100\%, \quad (6.9)$$

где $П_ч$ – чистая прибыль, получаемая организацией-разработчиком от реализации данного ПС (руб.);

$З_р$ – общая сумма затрат на разработку ПС (руб.)

Рентабельность составит:

$$Р = \frac{2016{,}6}{9335{,}56} * 100\% = 21{,}6 \text{ (\%)}$$

По результатам проведённых расчётов, затраты на разработку программного средства «Store» составят 9335,56 рублей, при этом чистая прибыль составит 2016,89, а рентабельность разработки ПС составит 21,6%.

ЗАКЛЮЧЕНИЕ

Прогресс, достигнутый за последние несколько лет во всех аспектах вычислительной техники, включая теорию, технологию и приложения, привели к значительному расширению области применения компьютеров и росту числа их пользователей. Существенной частью современного общества являются разнообразные системы доступа и хранения информации, которые являются неотъемлемой составляющей современного научно-технического прогресса. Существует много веских причин перевода существующей информации на компьютерную основу, так как более быстрая обработка данных и централизация их хранения позволяют сберечь значительные средства, а главное и время для получения необходимой информации, а также упрощает доступ и ведение.

В результате выполнения выпускной квалификационной работы было спроектировано и разработано программное средство для реализации компьютерных комплектующих.

Проведен анализ предметной области комплекса задач, к которому относится задача реализации компьютерных комплектующих. Изучен документооборот организации. Определен состав и структура информации. Построена концептуальная модель предметной области. Рассмотрены программы-аналоги, изучены их функциональные возможности. В итоге было принято решение о необходимости создания нового программного средства.

В качестве платформы для разработки выбран язык C# и среда программирования Visual Studio, так как она обладает всеми необходимыми инструментами

Актуальность и целесообразность создания программного средства определяется необходимостью создания эффективной системы для реализации компьютерных комплектующих. Внедрение информационной системы позволит снизить трудозатраты при анализе и формирования необходимой отчетности.

За разработанным ПС остается возможность расширяемости. Так, например, можно добавить дополнительные отчеты, ведение табеля смены и т.д.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Гуринович, Н. Заполнение первичных учетных документов: детали и особенности / Н. Гуринович // Главный бухгалтер. – 2015. – № 20 (884). – С. 42 – 45.

[2] Программы для учёта товаров в розничной торговле [Электронный ресурс]. – Режим доступа : https://www.moedelo.org/club/tovarnyy-uchet/programmy-dlya-ucheta-tovarov-v-roznichnoj-torgovle#subject-h2-1.

[3] Овчаренко, А.Г. Построение моделей процессов с помощью IDEF-технологии / А.Г. Овчаренко, А.Ю. Козлюк, А.Г. Лапынина // Учеб-метод. пособие для учащихся специальности 220501.65 / А.Г. Овчаренко, А.Ю. Козлюк, А.Г. Лапынина. – Бийск : АлтГТУ, 2010. – 57 с.

[4] Анализ требований и определение спецификаций программного обеспечения [Электронный ресурс]. – Режим доступа : http://www.starik2222.narod.ru/trpp/lec2/17.htm.

[5] Архитектура программного обеспечения [Электронный ресурс]. – Режим доступа : https://dic.academic.ru/dic.nsf/ruwiki/146913.

[6] Простое руководство по диаграммы компонентов [Электронный ресурс]. – Режим доступа : https://clck.ru/3EWsdG

[7] Бугай, О.В. САПР программного обеспечения издательско-полиграфического комплекса: учеб. пособие для студентов специальностей «Программное обеспечение информационных технологий», «Информационные системы и технологии / О.В. Бугай, В. С. Юденков – Мн.: БГТУ, 2008. – 174 с.

[8] Избачков, Ю.С. Информационные системы: Учебник для вузов / Ю.С. Избачков, В.Н. Петров, А.А. Васильев, И.С. Телина – 3-е изд. – СПб.: Питер, 2011. – 544 с.

[9] Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт – 8-е изд. – М.: Издательский дом «Вильяме», 2005. – 1328 с.

[10] Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения : ГОСТ 34.003-90. – Введ. 01.01.1992. – 20 с.

[11] Язык программирования C#: краткая история, возможности и перспективы [Электронный ресурс]. – Режим доступа : https://timeweb.com/ru/community/articles/chto-takoe-csharp.

[12] Арчер, Т. Основы C#. Новейшие технологии / Т.Арчер. – Пер. с англ. – М.: Издательско-торговый дом «Русская Редакция», 2001. – 448 с.

[13] Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – 6-е изд., пер. с англ. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.: ил.

[14] Куликов, С.С. Работа с MySQL, MS SQL Server и Oracle в примерах / С.С. Куликов – Минск: БОФФ, 2016. – 556 с.

[15] Сеппа, Д. Microsoft ADO.NET / Д. Сеппа. – Пер. с англ. – М.: Издательско-торговый дом «Русская Редакция», 2003. – 640 с.

[16] Уоткинз, Д. Программирование на платформе .NET. / Д. Уоткинз, М. Хаммонд, Б. Эйбрамз. – Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 368 с.

[17] Антиплагиат – онлайн проверка текстов на заимствования [Электронный ресурс]. – Режим доступа: https://antiplagiat.ru/.

[18] Соловей, Л. В. Программирование на языке C#: учеб. пособие / Л. В. Соловей, Н. Н. Мирошниченко, Н. Г. Пономарёв. – Х.: НТУ «ХПИ», 2016. – 356 с.

[19] Кнут, Д. Искусство программирования. Т. 1–3 / Д. Кнут. – М.: Вильямс, 2020. – 486 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Текст программы

```
using Microsoft.Reporting.WinForms;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Drawing;
using System.Text;
using System.Windows.Forms;



namespace Store
{
   public partial class Shop : Form
   {
      public string role1, user1;
      string connectionString = "Data Source=.\\SQLEXPRESS;" +
            "                    Initial Catalog=Store;" +
            "                    Integrated Security=True;" +
            "                    MultipleActiveResultSets=True";
      SqlConnection myConnection;
      string fl;
      List<ComboBox> comboBoxName = new List<ComboBox>();
      List<TextBox> textBoxScopeOfDelivery = new List<TextBox>();
      List<TextBox> textBoxUnitPrice = new List<TextBox>();
      List<TextBox> textBoxTotalCost = new List<TextBox>();
      List<TextBox> textBoxUnitPriceCheck = new List<TextBox>();
      List<ComboBox> comboBoxNameCheck = new List<ComboBox>();
      List<TextBox> numericUpDownCountCheck = new List<TextBox>();
      List<TextBox> textBoxTotalCostCheck = new List<TextBox>();
      List<int> codeNom = new List<int>();
      List<int> codeEmploye = new List<int>();
      List<int> codeNomenclature = new List<int>();
      int countNomenclature = 0;
      int countNomenclatureCheck = 0;
      int y;
      bool close = true;


      //изменения табеля смены
      List<ComboBox> comboBoxArr = new List<ComboBox>();
      List<Label> Label1Arr = new List<Label>();
      List<Label> Label2Arr = new List<Label>();
      List<DateTimePicker> dateTimePicker1Arr = new List<DateTimePicker>();
      List<DateTimePicker> dateTimePicker2Arr = new List<DateTimePicker>();
      ComboBox CB;
      Label LB;
      DateTimePicker DTP;
      int countEmp = 0;

      //заполнение таблицы "Пользователи"
      void fillUsers()
      {
         usersTable.Rows.Clear();
         string query = "SELECT Users.Code, Users.Login, Users.Password, Employe.FullName, Roles.Name " +
            "FROM Users, Employe, Roles " +
            "WHERE Users.CodeEmploye = Employe.Code and Users.CodeRole = Roles.Code";
         SqlCommand command = new SqlCommand(query, myConnection);
         try
         {
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
               usersTable.Rows.Add(reader[0].ToString(),
                           reader[1].ToString(),
```

```
                        reader[2].ToString(),
                        reader[3].ToString(),
                        reader[4].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        //перевод в транслит для добавлени логина
        private static Dictionary<string, string> transliter = new Dictionary<string, string>();
        private static void prepareTranslit()
        {
            transliter.Add("а", "a"); transliter.Add("б", "b"); transliter.Add("в", "v");
            transliter.Add("г", "g"); transliter.Add("д", "d"); transliter.Add("е", "e");
            transliter.Add("ё", "yo"); transliter.Add("ж", "zh"); transliter.Add("з", "z");
            transliter.Add("и", "i"); transliter.Add("й", "j"); transliter.Add("к", "k");
            transliter.Add("л", "l"); transliter.Add("м", "m"); transliter.Add("н", "n");
            transliter.Add("о", "o"); transliter.Add("п", "p"); transliter.Add("р", "r");
            transliter.Add("с", "s"); transliter.Add("т", "t"); transliter.Add("у", "u");
            transliter.Add("ф", "f"); transliter.Add("х", "h"); transliter.Add("ц", "c");
            transliter.Add("ч", "ch"); transliter.Add("ш", "sh"); transliter.Add("щ", "sch");
            transliter.Add("ъ", "j"); transliter.Add("ы", "i"); transliter.Add("ь", "j");
            transliter.Add("э", "e"); transliter.Add("ю", "yu"); transliter.Add("я", "ya");
            transliter.Add("А", "A"); transliter.Add("Б", "B"); transliter.Add("В", "V");
            transliter.Add("Г", "G"); transliter.Add("Д", "D"); transliter.Add("Е", "E");
            transliter.Add("Ё", "Yo"); transliter.Add("Ж", "Zh"); transliter.Add("З", "Z");
            transliter.Add("И", "I"); transliter.Add("Й", "J"); transliter.Add("К", "K");
            transliter.Add("Л", "L"); transliter.Add("М", "M"); transliter.Add("Н", "N");
            transliter.Add("О", "O"); transliter.Add("П", "P"); transliter.Add("Р", "R");
            transliter.Add("С", "S"); transliter.Add("Т", "T"); transliter.Add("У", "U");
            transliter.Add("Ф", "F"); transliter.Add("Х", "H"); transliter.Add("Ц", "C");
            transliter.Add("Ч", "Ch"); transliter.Add("Ш", "Sh"); transliter.Add("Щ", "Sch");
            transliter.Add("Ъ", "J"); transliter.Add("Ы", "I"); transliter.Add("Ь", "J");
            transliter.Add("Э", "E"); transliter.Add("Ю", "Yu"); transliter.Add("Я", "Ya");
        }
        public static string GetTranslit(string sourceText)
        {
            StringBuilder ans = new StringBuilder();
            for (int i = 0; i < sourceText.Length; i++) { if (transliter.ContainsKey(sourceText[i].ToString())) {
ans.Append(transliter[sourceText[i].ToString()]); } else { ans.Append(sourceText[i].ToString()); } }
            return ans.ToString();
        }

        //Составление логина
        private static string SurnameName(string name)
        {
            string user = "";
            int i = 0;
            int length = name.Length;
            for (i = 0; i < length; i++)
            {
                if (name[i] != ' ')
                {
                    user += name[i];
                }
                else
                {
                    break;
                }
            }

            i++;
            user += name[i];
            for (int j = i; j < length; j++)
            {
                if (name[j] == ' ') { i = j; break; }
```

```
            }

            i++;
            user += name[i];
            return user;
        }

        private void checkEmp(object sender, EventArgs e)
        {
            for (int i = 0; i < countEmp; i++)
            {
                if ((comboBoxArr[i].Text == "") || (dateTimePicker1Arr[i].Value > dateTimePicker2Arr[i].Value) ||
(dateTimePicker1Arr[i].Value.ToString() == "") || (dateTimePicker2Arr[i].Value.ToString() == ""))
                {
                }
            }
        }

        void addEmp()
        {
            CB = new ComboBox();
            CB.Location = new Point(70, y);
            CB.Font = new Font("Segoe UI", 10);
            CB.Width = 250;
            CB.Parent = this;
            if (countEmp != 1)
            {
                for (int i = 0; i < comboBoxArr[0].Items.Count; i++)
                {
                    CB.Items.Add(comboBoxArr[0].Items[i].ToString());
                }
            }
            CB.SelectedIndexChanged += new EventHandler(checkEmp);
            comboBoxArr.Add(CB);

            LB = new Label();
            LB.Location = new Point(325, y);
            LB.Font = new Font("Segoe UI", 10);
            LB.Text = "с";
            LB.Width = 10;
            LB.Parent = this;
            Label1Arr.Add(LB);

            DTP = new DateTimePicker();
            DTP.Format = DateTimePickerFormat.Time;
            DTP.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 09, 00, 00);
            DTP.ShowUpDown = true;
            DTP.Location = new Point(345, y);
            DTP.Font = new Font("Segoe UI", 10);
            DTP.Width = 100;
            DTP.Parent = this;
            DTP.ValueChanged += new EventHandler(checkEmp);
            dateTimePicker1Arr.Add(DTP);

            LB = new Label();
            LB.Location = new Point(450, y);
            LB.Font = new Font("Segoe UI", 10);
            LB.Text = "по";
            LB.Width = 30;
            LB.Parent = this;
            Label2Arr.Add(LB);

            DTP = new DateTimePicker();
            DTP.Format = DateTimePickerFormat.Time;
            DTP.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 20, 00, 00);
            DTP.ShowUpDown = true;
            DTP.Location = new Point(480, y);
            DTP.Font = new Font("Segoe UI", 10);
```

```
            DTP.Width = 100;
            DTP.Parent = this;
            DTP.ValueChanged += new EventHandler(checkEmp);
            dateTimePicker2Arr.Add(DTP);
        }

        private void KeyPressTB(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsDigit(e.KeyChar) && e.KeyChar != 8 && e.KeyChar != Convert.ToChar(","))
                e.Handled = true;
        }
        //выборка кода
        void selectCode(string table, string query, List<int> code, ComboBox comboBox)
        {
            SqlCommand command = new SqlCommand(query, myConnection);
            code.Clear();
            comboBox.Items.Clear();
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    code.Add(int.Parse(reader[0].ToString()));
                    comboBox.Items.Add(reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        //заполнение таблицы "Категория"
        void fillCategory()
        {
            categoryTable.Rows.Clear();
            string query = "SELECT Code, Name FROM Category";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    categoryTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        void fillNomenclature()
        {
            nomenclatureTable.Rows.Clear();
            string query = "SELECT Nomenclature.Code, Nomenclature.Name,  Nomenclature.Balance, Nomenclature.UnitPrice,
Nomenclature.SellingPrice" +
                        " FROM Nomenclature";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    nomenclatureTable.Rows.Add(reader[0].ToString(),
```

```
                        reader[1].ToString(),
                        reader[2].ToString(),
                        reader[3].ToString(),
                        reader[4].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }

    //заполнение таблицы "Должность"
    void fillPositions()
    {
        positionsTable.Rows.Clear();
        string query = "SELECT Code, Name FROM Positions";
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                positionsTable.Rows.Add(reader[0].ToString(),
                        reader[1].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }

    //проверка заполнения всех полей при вводе номенклатуры
    void checkNomenclatureInput()
    {
        if (nomenclatureName.Text != "")
        {
            nomenclatureOkButton.Enabled = true;
        }
        else
        {
            nomenclatureOkButton.Enabled = false;
        }
    }

    //проверка заполнения всех полей при вводе сорудника
    void checkEmployeesInput()
    {
        if ((employeesCode.Text != "") && (employeesFullName.Text != "") && (employeesAddress.Text != "") &&
(employeesPhone.Text != "") && (employeesEmail.Text != "") && (employeesPosition.SelectedIndex != -1))
        {
            employeesOkButton.Enabled = true;
        }
        else
        {
            employeesOkButton.Enabled = false;
        }
    }

    //заполенение comboBox при выборе CheckBox
    void checkTtnSearchComboBox(CheckBox checkBoxSearch, ComboBox comboBoxSearch, string selectFields, string table)
    {
        if (checkBoxSearch.Checked)
        {
            comboBoxSearch.Visible = true;
```

```
            string query = "SELECT " + selectFields + " FROM " + table;
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                comboBoxSearch.Items.Clear();
                while (reader.Read())
                {
                    comboBoxSearch.Items.Add(reader[0].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        else
        {
            comboBoxSearch.SelectedItem = -1;
            comboBoxSearch.Items.Clear();
            comboBoxSearch.Visible = false;
        }
    }

    //удаление записи из таблицы
    void deleteRecords(string code, string table)
    {
        string query = "DELETE FROM " + table +
                    " Where Code = " + code;
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            command.ExecuteNonQuery();
        }
        catch
        {
            MessageBox.Show("Ошибка удаление записи. Запись не удалена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
        }
    }

    //очистка панели добавления/реадктирвоания сотрудника
    void clearEmployeesAddEdit()
    {
        emploeesEditAddPanel.Visible = false;
        employeesCode.Text = "";
        employeesFullName.Text = "";
        employeesDOB.Value = System.DateTime.Today;
        employeesAddress.Text = "";
        employeesPhone.Text = "";
        employeesEmail.Text = "";
        employeesPosition.SelectedIndex = -1;
        employeesPosition.Items.Clear();
        employeesRole.SelectedIndex = -1;
        employeesRole.Items.Clear();
    }

    //выбор товара при вводе ТТН
    private void comboBoxNameChange(object sender, EventArgs e)
    {
        int count = 0;
        for (int i = 0; i < countNomenclature; i++)
            if (comboBoxName[i] == sender)
            {
                count = i;
                break;
            }
```

```
        if (comboBoxName[count].SelectedIndex != -1)
        {
            string query = "SELECT Nomenclature.Name, Nomenclature.UnitPrice " +
                "FROM Nomenclature " +
                "WHERE Nomenclature.Code='" + codeNom[comboBoxName[count].SelectedIndex].ToString() + "'";
            SqlCommand command1 = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader1 = command1.ExecuteReader();
                reader1.Read();
                textBoxUnitPrice[count].Text = reader1[1].ToString();
                reader1.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
    }

    //выбор товара при создании чека
    private void comboBoxNameChangeCheck(object sender, EventArgs e)
    {
        int count = 0;
        for (int i = 0; i < countNomenclatureCheck; i++)
            if (comboBoxNameCheck[i] == sender)
            {
                count = i;
                break;
            }
        if (comboBoxNameCheck[count].SelectedIndex != -1)
        {
            string query = "SELECT Nomenclature.SellingPrice " +
                "FROM Nomenclature " +
                "WHERE Nomenclature.Code=" + codeNom[comboBoxNameCheck[count].SelectedIndex].ToString();
            SqlCommand command1 = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader1 = command1.ExecuteReader();
                reader1.Read();
                textBoxUnitPriceCheck[count].Text = reader1[0].ToString();
                reader1.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            ChangeCheck(count);
        }
    }


    //расчет итоговой стоимости по товару в ТТН
    private void Change(object sender, EventArgs e)
    {
        int count = 0;
        for (int i = 0; i < countNomenclature; i++)
            if ((textBoxScopeOfDelivery[i] == sender) || (comboBoxName[i] == sender) || (textBoxUnitPrice[i] == sender))
            {
                count = i;
                break;
            }
        if ((textBoxUnitPrice[count].Text != "") && (textBoxScopeOfDelivery[count].Text != ""))
        {
            double d = double.Parse(textBoxScopeOfDelivery[count].Text);
            double p = Math.Round(double.Parse(textBoxUnitPrice[count].Text) * d, 2);
```

```
            textBoxTotalCost[count].Text = Math.Round((p), 2).ToString();
        }
    }

    //расчет итоговой стоимости по товару в чеке
    private void ChangeCheck(int count)
    {
        if ((textBoxUnitPriceCheck[count].Text != "") && (numericUpDownCountCheck[count].Text != ""))
        {
            double d = double.Parse(numericUpDownCountCheck[count].Text);
            double p = Math.Round(double.Parse(textBoxUnitPriceCheck[count].Text) * d, 2);
            textBoxTotalCostCheck[count].Text = p.ToString();
        }
        double sum = 0;
        for (int i = 0; i < countNomenclatureCheck; i++)
            if (textBoxTotalCostCheck[i].Text != "") sum += double.Parse(textBoxTotalCostCheck[i].Text);
        textBox1.Text = sum.ToString();
    }

    //изменение отображения вкладок на tabControl
    private static void DrawItem(object sender, DrawItemEventArgs e)
    {
        Graphics g = e.Graphics;
        Brush _textBrush;
        TabPage _tabPage = ((TabControl)sender).TabPages[e.Index];
        System.Drawing.Rectangle _tabBounds = ((TabControl)sender).GetTabRect(e.Index);
        if (e.State == DrawItemState.Selected)
        {
            _textBrush = new SolidBrush(Color.White);
            g.FillRectangle(Brushes.Gray, e.Bounds);
        }
        else
        {
            _textBrush = new System.Drawing.SolidBrush(e.ForeColor);
            e.DrawBackground();
        }
        System.Drawing.Font _tabFont = new System.Drawing.Font("Segoe UI", (float)10.0, FontStyle.Regular,
GraphicsUnit.Point);
        StringFormat _stringFlags = new StringFormat();
        _stringFlags.Alignment = StringAlignment.Center;
        _stringFlags.LineAlignment = StringAlignment.Center;
        g.DrawString(_tabPage.Text, _tabFont, _textBrush, _tabBounds, new StringFormat(_stringFlags));
    }

    private void numericUpDownCountCheckChangeCheck(object sender, EventArgs e)
    {
        int count = 0;
        for (int i = 0; i < countNomenclatureCheck; i++)
            if (numericUpDownCountCheck[i] == sender)
            {
                count = i;
                break;
            }
        ChangeCheck(count);
    }

    //Добавление полей для ввода товара при вводе ТТН
    private void addNomenclatureTTN(int y)
    {
        ComboBox CB = new ComboBox();
        CB.Location = new Point(55, y);
        CB.Font = new Font("Segoe UI", 10);
        CB.Width = 350;
        CB.SelectedIndexChanged += new System.EventHandler(comboBoxNameChange);
        CB.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
        CB.AutoCompleteSource = AutoCompleteSource.ListItems;
        CB.Parent = ttnAddEditPanel;

        if (comboBoxName.Count != 0)
```

```
    {
        int temp = comboBoxName[0].Items.Count;
        CB.Items.Clear();
        for (int i = 0; i < temp; i++)
            CB.Items.Add(comboBoxName[0].Items[i].ToString());
    }
    else
    {
        string query = "SELECT Code, Name FROM Nomenclature WHERE Balance>0";
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            codeNom.Clear();
            CB.Items.Clear();
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                codeNom.Add(int.Parse(reader[0].ToString()));
                CB.Items.Add(reader[1].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }
    comboBoxName.Add(CB);

    TextBox TB = new TextBox();
    TB.Location = new Point(420, y);
    TB.Font = new Font("Segoe UI", 10);
    TB.Width = 100;
    TB.Parent = ttnAddEditPanel;
    TB.TextChanged += new System.EventHandler(Change);
    TB.KeyPress += new System.Windows.Forms.KeyPressEventHandler(KeyPressTB);
    textBoxScopeOfDelivery.Add(TB);

    TextBox TB1 = new TextBox();
    TB1.Location = new Point(535, y);
    TB1.Font = new Font("Segoe UI", 10);
    TB1.Width = 100;
    TB1.Parent = ttnAddEditPanel;
    TB1.TextChanged += new System.EventHandler(Change);
    TB1.KeyPress += new System.Windows.Forms.KeyPressEventHandler(KeyPressTB);
    textBoxUnitPrice.Add(TB1);

    TextBox TB2 = new TextBox();
    TB2.Location = new Point(650, y);
    TB2.Font = new Font("Segoe UI", 10);
    TB2.Width = 100;
    TB2.Parent = ttnAddEditPanel;
    TB2.Enabled = false;
    TB2.Text = "0";
    textBoxTotalCost.Add(TB2);
}

//Добавление полей для ввода товара при вводе чека
private void addNomenclatureCheck(int y)
{
    ComboBox CB;
    TextBox TB;
    CB = new ComboBox();
    CB.Location = new Point(70, y);
    CB.Font = new Font("Segoe UI", 10);
    CB.Width = 350;
    CB.SelectedIndexChanged += new System.EventHandler(comboBoxNameChangeCheck);
    CB.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
    CB.AutoCompleteSource = AutoCompleteSource.ListItems;
```

```
    CB.Parent = panel2;
    if (comboBoxNameCheck.Count != 0)
    {
        int temp = comboBoxNameCheck[0].Items.Count;
        CB.Items.Clear();
        for (int i = 0; i < temp; i++)
            CB.Items.Add(comboBoxNameCheck[0].Items[i].ToString());
    }
    else
    {
        string query = "SELECT Code, Name FROM Nomenclature WHERE Balance>0";
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            codeNom.Clear();
            CB.Items.Clear();
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                codeNom.Add(int.Parse(reader[0].ToString()));
                CB.Items.Add(reader[1].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }
    comboBoxNameCheck.Add(CB);

    TB = new TextBox();
    TB.Location = new Point(425, y);
    TB.Font = new Font("Segoe UI", 10);
    TB.Width = 100;
    TB.Enabled = true;
    TB.Parent = panel2;
    TB.TextChanged += new System.EventHandler(numericUpDownCountCheckChangeCheck);
    TB.KeyPress += new System.Windows.Forms.KeyPressEventHandler(KeyPressTB);
    numericUpDownCountCheck.Add(TB);

    TB = new TextBox();
    TB.Location = new Point(530, y);
    TB.Font = new Font("Segoe UI", 10);
    TB.Width = 100;
    TB.Enabled = false;
    TB.Parent = panel2;
    textBoxUnitPriceCheck.Add(TB);
    TB = new TextBox();
    TB.Location = new Point(635, y);
    TB.Font = new Font("Segoe UI", 10);
    TB.Width = 100;
    TB.Enabled = false;
    TB.Parent = panel2;
    textBoxTotalCostCheck.Add(TB);
}

//Удаление полей для ввода товара при вводе ТТН
private void deleteNomenclatureTTN(int number)
{
    comboBoxName.RemoveAt(number);
    textBoxScopeOfDelivery.RemoveAt(number);
    textBoxTotalCost.RemoveAt(number);
    textBoxUnitPrice.RemoveAt(number);
    int i = ttnAddEditPanel.Controls.Count;
    for (int j = 0; j <= 3; j++)
    {
        i--;
        ttnAddEditPanel.Controls[i].Enabled = false;
```

```
            ttnAddEditPanel.Controls[i].Dispose();
        }
    }

    //Удаление полей для ввода товара при вводе чека
    private void deleteNomenclatureCheck(int number)
    {
        comboBoxNameCheck.RemoveAt(number);
        numericUpDownCountCheck.RemoveAt(number);
        textBoxUnitPriceCheck.RemoveAt(number);
        textBoxTotalCostCheck.RemoveAt(number);
        int i = panel2.Controls.Count;
        for (int j = 0; j <= 3; j++)
        {
            i--;
            panel2.Controls[i].Enabled = false;
            panel2.Controls[i].Dispose();
        }
    }

    //Изменение запятой на точку
    private static string CommaToPoint(string number)
    {
        string temp = "";
        int length = number.Length;
        for (int i = 0; i < length; i++)
            if (number[i] != ',')
            {
                temp += number[i];
            }
            else
            {
                temp += '.';
            }
        return temp;
    }

    private string editTTN(CheckBox checkBox1, CheckBox checkBox2, DateTimePicker dateTimePicker, string comboBox_2)
    {
        string query = "";
        if ((checkBox1.Checked) && (checkBox2.Checked))
        {
            query = "SELECT distinct TTN.Code, TTN.NumberTTN, TTN.Date, Employe.FullName FROM TTN,
Employe,Nomenclature, GoodsMovement WHERE " +
                "Employe.Code = TTN.CodeEmploye and GoodsMovement.CodeTTN=TTN.Code and
GoodsMovement.CodeNomenclature=Nomenclature.Code and TTN.NumberTTN = '" + comboBox_2 + "' and TTN.Date = '" +
dateTimePicker.Value.Year.ToString() + "-" +
                                    dateTimePicker.Value.Month.ToString() + "-" +
                                    dateTimePicker.Value.Day.ToString() + "'";
        }
        else
        if ((checkBox1.Checked == false) && (checkBox2.Checked))
        {
            query = "SELECT distinct TTN.Code, TTN.NumberTTN, TTN.Date, Employe.FullName FROM TTN,
Employe,Nomenclature, GoodsMovement WHERE " +
                "Employe.Code = TTN.CodeEmploye and GoodsMovement.CodeTTN=TTN.Code and
GoodsMovement.CodeNomenclature=Nomenclature.Code and TTN.NumberTTN = '" + comboBox_2 + "'";
        }
        else
        if ((checkBox1.Checked) && (checkBox2.Checked == false))
        {
            query = "SELECT distinct TTN.Code, TTN.NumberTTN, TTN.Date, Employe.FullName FROM TTN, Employe,
Nomenclature, GoodsMovement WHERE " +
                "Employe.Code = TTN.CodeEmploye and GoodsMovement.CodeTTN=TTN.Code and
GoodsMovement.CodeNomenclature=Nomenclature.Code and TTN.Date = '" + dateTimePicker.Value.Year.ToString() + "-" +
                                    dateTimePicker.Value.Month.ToString() + "-" +
                                    dateTimePicker.Value.Day.ToString() + "'";
        }
        else
        if ((checkBox1.Checked == false) && (checkBox2.Checked == false))
```

```
        {
            query = "SELECT distinct TTN.Code, TTN.NumberTTN, TTN.Date, Employe.FullName FROM TTN, Employe, Nomenclature, GoodsMovement WHERE " +
                "Employe.Code = TTN.CodeEmploye and GoodsMovement.CodeTTN=TTN.Code and GoodsMovement.CodeNomenclature=Nomenclature.Code";
        }
        return query;
    }

    public Shop(string role, string user)
    {
        InitializeComponent();
        reportsTabs.DrawItem += new DrawItemEventHandler(tabControl2_DrawItem);
        goodsTabs.DrawItem += new DrawItemEventHandler(tabControl3_DrawItem);
        role1 = role;
        user1 = user;
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу "StoreDataSet.CommodityReport2". При необходимости она может быть перемещена или удалена.
        this.commodityReport2TableAdapter.Fill(this.StoreDataSet.CommodityReport2);
        // TODO: данная строка кода позволяет загрузить данные в таблицу "StoreDataSet.CommodityReport1". При необходимости она может быть перемещена или удалена.
        this.commodityReport1TableAdapter.Fill(this.StoreDataSet.CommodityReport1);
        // TODO: данная строка кода позволяет загрузить данные в таблицу "StoreDataSet.ReportUnloadingOfResidues". При необходимости она может быть перемещена или удалена.

        myConnection = new SqlConnection(connectionString);
        try
        {
            this.reportUnloadingOfResiduesTableAdapter.Fill(this.StoreDataSet.ReportUnloadingOfResidues);
            myConnection.Open();
            goods.Parent = MainTabs;
            reports.Parent = MainTabs;
            employees.Parent = MainTabs;
        }
        catch
        {
            MessageBox.Show("Ошибка подключения к базе данных. Обратитесь к администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            employees.Parent = null;
            goods.Parent = null;
            reports.Parent = null;
            return;
        }
        this.reportViewer5.RefreshReport();
        if (role1 == "Заведующий")
        {
            employeesAddButton.Enabled = false;
            employeesEditButton.Enabled = false;
            employeesDeleteButton.Enabled = false;
            positionsAddButton.Enabled = false;
            positionsEditButton.Enabled = false;
            positionsDeleteButton.Enabled = false;
            employeesTabs.TabPages.Remove(users);
            nomenclatureAddButton.Enabled = false;
            nomenclatureEditButton.Enabled = false;
            nomenclatureDeleteButton.Enabled = false;
            goodsTabs.TabPages.Remove(ttn);
            categoryAddButton.Enabled = false;
            categoryEditButton.Enabled = false;
            categoryDeleteButton.Enabled = false;
        }
        else if (role1 == "Оператор")
        {
            MainTabs.TabPages.Remove(employees);
            MainTabs.TabPages.Remove(reports);
            goodsTabs.TabPages.Remove(nomenclature);
            goodsTabs.TabPages.Remove(check);
```

```
            goodsTabs.TabPages.Remove(category);
            reportsTabs.TabPages.Remove(remnantsOfGoods);
            ttnEditButton.Enabled = false;
            ttnPreviewPanel.Visible = false;
        }
        else if (role1 == "Продавец")
        {
            MainTabs.TabPages.Remove(employees);
            MainTabs.TabPages.Remove(reports);
            goodsTabs.TabPages.Remove(ttn);
            goodsTabs.TabPages.Remove(category);
            reportsTabs.TabPages.Remove(remnantsOfGoods);
            ttnAddButton.Enabled = false;
            ttnEditButton.Enabled = false;
        }
        else if (role1 == "Старший продавец")
        {
            MainTabs.TabPages.Remove(employees);
            goodsTabs.TabPages.Remove(ttn);
            goodsTabs.TabPages.Remove(category);
            reportsTabs.TabPages.Remove(remnantsOfGoods);
            ttnAddButton.Enabled = false;
            ttnEditButton.Enabled = false;
        }
        this.reportViewer2.RefreshReport();
        this.reportViewer5.RefreshReport();
    }

    private void tabControl1_DrawItem(object sender, DrawItemEventArgs e)
    {
        DrawItem(sender, e);
    }

    private void tabControl3_DrawItem(object sender, DrawItemEventArgs e)
    {
        DrawItem(sender, e);
    }

    private void tabControl2_DrawItem(object sender, DrawItemEventArgs e)
    {
      DrawItem(sender, e);
    }

    private void tabControl4_DrawItem(object sender, DrawItemEventArgs e)
    {
        DrawItem(sender, e);
    }

    private void button4_Click_1(object sender, EventArgs e)
    {
        clearEmployeesAddEdit();
    }

    private void tabControl4_MouseClick(object sender, MouseEventArgs e)
    {
        if (employeesTabs.SelectedTab.Name == "employee")
        {
            string query = "SELECT Employe.Code, Employe.FullName, CONVERT (varchar,Employe.DOB,104),
Employe.Address, Employe.Phone, Employe.email, Positions.Name FROM Employe, Positions WHERE
Employe.CodePosition=Positions.Code";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                employeesTable.Rows.Clear();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    employeesTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString(),
                                reader[2].ToString(),
```

```
                        reader[3].ToString(),
                        reader[4].ToString(),
                        reader[5].ToString(),
                        reader[6].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        if (employeesTabs.SelectedTab.Name == "position")
        {
            string query = "SELECT Code, Name FROM Positions";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                positionsTable.Rows.Clear();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    positionsTable.Rows.Add(reader[0].ToString(),
                        reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        if (employeesTabs.SelectedTab.Name == "users")
        {
            string query = "SELECT Users.Code, Users.Login, Users.Password, Employe.FullName, Roles.Name " +
                    "FROM Users, Employe, Roles " +
                    "WHERE Users.CodeEmploye = Employe.Code and Users.CodeRole = Roles.Code";
            SqlCommand command = new SqlCommand(query, myConnection);
            usersTable.Rows.Clear();
            try
            {

                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    usersTable.Rows.Add(reader[0].ToString(),
                        reader[1].ToString(),
                        reader[2].ToString(),
                        reader[3].ToString(),
                        reader[4].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    private void button10_Click(object sender, EventArgs e)
    {
        string query;
        SqlCommand command;
        SqlDataReader reader;
        fl = "add";
        try
        {
            query = "SELECT Name FROM Positions";
            command = new SqlCommand(query, myConnection);
```

```
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                employeesPosition.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        try
        {
            query = "SELECT Name FROM Roles";
            command = new SqlCommand(query, myConnection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                employeesRole.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        try
        {
            query = "SELECT IDENT_CURRENT('Employe')";
            command = new SqlCommand(query, myConnection);
            reader = command.ExecuteReader();
            reader.Read();
            employeesCode.Text = (int.Parse(reader[0].ToString()) + 1).ToString();
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        emploeesEditAddPanel.Visible = true;
        employeesRole.Visible = true;
        label16.Visible = true;
    }

    private void button9_Click(object sender, EventArgs e)
    {
        string query;
        SqlCommand command;
        SqlDataReader reader;
        fl = "edit";
        query = "SELECT Name FROM Positions";
        command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                employeesPosition.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "SELECT Employe.Code, Employe.FullName, Employe.DOB, Employe.Address, Employe.Phone,
```

```
Employe.email, Positions.Name FROM Employe, Positions WHERE Positions.Code=Employe.CodePosition and Employe.Code=" + employeesTable.Rows[employeesTable.CurrentRow.Index].Cells[0].Value.ToString();
            command = new SqlCommand(query, myConnection);
            try
            {
                reader = command.ExecuteReader();
                reader.Read();
                employeesCode.Text = reader[0].ToString();
                employeesFullName.Text = reader[1].ToString();
                employeesDOB.Value = new DateTime(DateTime.Parse(reader[2].ToString()).Year, DateTime.Parse(reader[2].ToString()).Month, DateTime.Parse(reader[2].ToString()).Day, 0, 0, 0);
                employeesAddress.Text = reader[3].ToString();
                employeesPhone.Text = reader[4].ToString();
                employeesEmail.Text = reader[5].ToString();
                employeesPosition.SelectedItem = reader[6].ToString();
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            emploeesEditAddPanel.Visible = true;
            employeesOkButton.Enabled = true;
            employeesRole.Visible = false;
            label16.Visible = false;
        }

        private void button7_Click_1(object sender, EventArgs e)
        {
            string query;
            SqlCommand command;
            if (fl == "add")
            {
                query = "INSERT INTO Employe(FullName, Phone, CodePosition) " +
                        "SELECT '" + employeesFullName.Text + "', '" +
                                employeesPhone.Text + "', (SELECT Code " +
                                    "FROM Positions " +
                                    "WHERE Name='" + employeesPosition.Text + "')";
                command = new SqlCommand(query, myConnection);
                try
                {
                    command.ExecuteNonQuery();
                }
                catch
                {
                    MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
            }
            if (fl == "edit")
            {
                query = "UPDATE Employe " +
                        " SET FullName = '" + employeesFullName.Text + "'," +
                            " Phone = '" + employeesPhone.Text + "'," +
                            " CodePosition=(SELECT Code " +
                                    "FROM Positions " +
                                    "WHERE Name='" + employeesPosition.Text + "') WHERE Code =" + employeesCode.Text;
                command = new SqlCommand(query, myConnection);
                try
                {
                    command.ExecuteNonQuery();
                }
                catch
                {
                    MessageBox.Show("Ошибка редактировании записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
```

```
        }
        clearEmployeesAddEdit();
        employeesTable.Rows.Clear();
        query = "SELECT Employe.Code, Employe.FullName, Employe.Phone, Positions.Name FROM Employe, Positions WHERE Employe.CodePosition=Positions.Code";
        command = new SqlCommand(query, myConnection);
        try
        {
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                employeesTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString(),
                                reader[2].ToString(),
                                reader[3].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }

    private void button8_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Вы действительно хотите удалить выбранную запись?", "Удаление", MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)
        {
            deleteRecords(employeesTable.Rows[employeesTable.CurrentRow.Index].Cells[0].Value.ToString(), "Employe");
            employeesTable.Rows.Clear();
            string query = "SELECT Employe.Code, Employe.FullName, CONVERT (varchar,Employe.DOB,104), Employe.Address, Employe.Phone, Employe.email, Positions.Name FROM Employe, Positions WHERE Employe.CodePosition=Positions.Code";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    employeesTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString(),
                                reader[2].ToString(),
                                reader[3].ToString(),
                                reader[4].ToString(),
                                reader[5].ToString(),
                                reader[6].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
    }

    private void button16_Click(object sender, EventArgs e)
    {
        fl = "add";
        string query = "SELECT IDENT_CURRENT('Positions')";
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();
            positionsCode.Text = (int.Parse(reader[0].ToString()) + 1).ToString();
            reader.Close();
```

```
            }
            catch
            {
                MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            positionsEditAddPanel.Visible = true;
        }

        private void button15_Click(object sender, EventArgs e)
        {
            fl = "edit";
            string query = "SELECT Code, Name FROM Positions WHERE Code=" +
positionsTable.Rows[positionsTable.CurrentRow.Index].Cells[0].Value.ToString();
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                reader.Read();
                positionsCode.Text = reader[0].ToString();
                positionsName.Text = reader[1].ToString();
                positionsEditAddPanel.Visible = true;
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        private void button14_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Вы действительно хотите удалить выбранную запись?", "Удаление",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)
            {
                deleteRecords(positionsTable.Rows[positionsTable.CurrentRow.Index].Cells[0].Value.ToString(), "Positions");
                fillPositions();
            }
        }

        private void button13_Click(object sender, EventArgs e)
        {
            string query;
            SqlCommand command;
            if (fl == "add")
            {
                query = "INSERT INTO Positions (Name) " +
                    "VALUES ('" + positionsName.Text + "')";
                command = new SqlCommand(query, myConnection);
                try
                {
                    command.ExecuteNonQuery();
                }
                catch
                {
                    MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
            }
            if (fl == "edit")
            {
                query = "UPDATE Positions " +
                    " SET Name = '" + positionsName.Text + "'" +
                    " WHERE Code =" + positionsCode.Text + "";
                command = new SqlCommand(query, myConnection);
                try
                {
                    command.ExecuteNonQuery();
                }
```

```
            catch
            {
                MessageBox.Show("Ошибка редактирования записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
        }
        positionsEditAddPanel.Visible = false;
        positionsCode.Text = "";
        positionsName.Text = "";
        fillPositions();
    }

    private void button11_Click(object sender, EventArgs e)
    {
        positionsEditAddPanel.Visible = false;
        positionsCode.Text = "";
        positionsName.Text = "";
    }

    private void Tabs_MouseClick(object sender, MouseEventArgs e)
    {
        tabControl4_MouseClick(this, e);
        tabControl3_MouseClick(this, e);
    }

    private void tabControl3_MouseClick(object sender, MouseEventArgs e)
    {
        if (goodsTabs.SelectedTab.Name == "nomenclature")
        {
            string query = "SELECT Nomenclature.Code, Nomenclature.Name, Nomenclature.Balance,  Nomenclature.UnitPrice,
Nomenclature.SellingPrice " +
                    "FROM Nomenclature";
            SqlCommand command = new SqlCommand(query, myConnection);
            nomenclatureTable.Rows.Clear();
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    nomenclatureTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString(),
                                reader[2].ToString(),
                                reader[3].ToString(),
                                reader[4].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        if (goodsTabs.SelectedTab.Name == "check")
        {
            countNomenclatureCheck = 1;
            y = 125;
            addNomenclatureCheck(y);
            codeEmploye.Clear();
            comboBox1.Items.Clear();
            string query = "SELECT Employe.Code, FullName FROM Employe";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    codeEmploye.Add(int.Parse(reader[0].ToString()));
                    comboBox1.Items.Add(reader[1].ToString());
```

```
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        dateTimePicker1.Value = DateTime.Today;
    }
    if (goodsTabs.SelectedTab.Name == "category")
    {
        fillCategory();
    }
}

private void button19_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Вы действительно хотите удалить выбранную запись?", "Удаление",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)
    {
        deleteRecords(nomenclatureTable.Rows[nomenclatureTable.CurrentRow.Index].Cells[0].Value.ToString(),
"Nomenclature");
        fillNomenclature();
    }
}

private void button21_Click(object sender, EventArgs e)
{
    fl = "add";
    string query = "SELECT Name FROM Category";
    SqlCommand command = new SqlCommand(query, myConnection);
    SqlDataReader reader;
    command = new SqlCommand(query, myConnection);
    try
    {
        reader = command.ExecuteReader();
        nomenclatureCategory.Items.Clear();
        while (reader.Read())
        {
            nomenclatureCategory.Items.Add(reader[0].ToString());
        }
        reader.Close();
    }
    catch
    {
        MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    query = "SELECT IDENT_CURRENT('Nomenclature')";
    command = new SqlCommand(query, myConnection);
    try
    {
        reader = command.ExecuteReader();
        reader.Read();
        nomenclatureCode.Text = (int.Parse(reader[0].ToString()) + 1).ToString();
        reader.Close();
    }
    catch
    {
        MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    nomenclatureEditAddPanel.Visible = true;
}

private void button20_Click(object sender, EventArgs e)
{
    fl = "edit";
    SqlDataReader reader;
```

```
            string query = "SELECT Name FROM Category";
            SqlCommand command = new SqlCommand(query, myConnection);
            command = new SqlCommand(query, myConnection);
            try
            {
               reader = command.ExecuteReader();
               nomenclatureCategory.Items.Clear();
               while (reader.Read())
               {
                  nomenclatureCategory.Items.Add(reader[0].ToString());
               }
               reader.Close();
            }
            catch
            {
               MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
               return;
            }
            query = "SELECT Nomenclature.Code, Nomenclature.Name,  Category.Name, UnitPrice, SellingPrice " +
               " FROM Nomenclature, Category" +
               "  WHERE Nomenclature.CodeCategory = Category.Code and Nomenclature.Code = " +
nomenclatureTable.Rows[nomenclatureTable.CurrentRow.Index].Cells[0].Value.ToString();
            command = new SqlCommand(query, myConnection);
            try
            {
               reader = command.ExecuteReader();
               reader.Read();
               nomenclatureCode.Text = reader[0].ToString();
               nomenclatureName.Text = reader[1].ToString();
               nomenclatureCategory.SelectedItem = reader[2].ToString();
               nomenclatureUnitPrice.Text = reader[3].ToString();
               nomenclatureSellingPrice.Text = reader[4].ToString();
               reader.Close();
            }
            catch
            {
               MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
               return;
            }
            nomenclatureEditAddPanel.Visible = true;
         }

         private void button17_Click(object sender, EventArgs e)
         {
            nomenclatureEditAddPanel.Visible = false;
            nomenclatureCode.Text = "";
            nomenclatureName.Text = "";
            nomenclatureCategory.Items.Clear();
            nomenclatureCategory.Text = "";
         }


         private void button18_Click(object sender, EventArgs e)
         {
            string query;
            SqlCommand command;
            nomenclatureUnitPrice.Text = CommaToPoint(nomenclatureUnitPrice.Text);
            nomenclatureSellingPrice.Text = CommaToPoint(nomenclatureSellingPrice.Text);
            if (fl == "add")
            {
               query = "INSERT INTO Nomenclature (Name, Balance, CodeCategory, UnitPrice, SellingPrice) " +
                        "Select '" + nomenclatureName.Text.ToString() + "', 0, Category.Code, " + nomenclatureUnitPrice.Text + ", " +
nomenclatureSellingPrice.Text +
                        " From Category " +
                        "Where Category.Name='" + nomenclatureCategory.Text.ToString() + "';";
               command = new SqlCommand(query, myConnection);
               try
               {
                  command.ExecuteNonQuery();
               }
```

```
            catch
            {
                MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
        }
        if (fl == "edit")
        {
            query = "UPDATE Nomenclature " +
                    "SET Name='" + nomenclatureName.Text.ToString() + "', " +
                    " UnitPrice=" + nomenclatureUnitPrice.Text.ToString() + ", " +
                    " SellingPrice=" + nomenclatureSellingPrice.Text.ToString() + ", " +
                        "CodeCategory = (SELECT Code FROM Category WHERE Name='" +
nomenclatureCategory.Text.ToString() + "') " +
                    " WHERE Code=" + nomenclatureCode.Text.ToString();
            command = new SqlCommand(query, myConnection);
            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Ошибка редактирования записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
        }
        nomenclatureEditAddPanel.Visible = false;
        nomenclatureCode.Text = "";
        nomenclatureName.Text = "";
        nomenclatureUnitPrice.Text = "";
        nomenclatureSellingPrice.Text = "";
        nomenclatureCategory.Items.Clear();
        nomenclatureCategory.Text = "";
        fillNomenclature();
    }

    private void button37_Click(object sender, EventArgs e)
    {
        fl = "add";
        SqlDataReader reader;
        string query = "SELECT IDENT_CURRENT('Category')";
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            reader.Read();
            categoryCode.Text = (int.Parse(reader[0].ToString()) + 1).ToString();
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        panel8.Visible = true;
    }

    private void button36_Click(object sender, EventArgs e)
    {
        fl = "edit";
        SqlDataReader reader;
        string query = "SELECT Code, Name " +
            "FROM Category " +
            "WHERE Code =" + categoryTable.Rows[categoryTable.CurrentRow.Index].Cells[0].Value.ToString();
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
```

```
            reader.Read();
            categoryCode.Text = reader[0].ToString();
            categoryName.Text = reader[1].ToString();
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        panel8.Visible = true;
    }

    private void button35_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Вы действительно хотите удалить выбранную запись?", "Удаление",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)
        {
            deleteRecords(categoryTable.Rows[categoryTable.CurrentRow.Index].Cells[0].Value.ToString(), "Category");
            fillCategory();
        }
    }

    private void button33_Click(object sender, EventArgs e)
    {
        categoryName.Text = "";
        categoryCode.Text = "";
        panel8.Visible = false;
    }

    private void button34_Click(object sender, EventArgs e)
    {
        string query;
        SqlCommand command;
        if (fl == "add")
        {
            query = "INSERT INTO Category (Name) " +
                    "VALUES ('" + categoryName.Text.ToString() + "')";
            command = new SqlCommand(query, myConnection);
            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
        }
        if (fl == "edit")
        {
            query = "UPDATE Category " +
                    "SET Name='" + categoryName.Text.ToString() + "'" +
                    " WHERE Code=" + categoryCode.Text.ToString(); ;
            command = new SqlCommand(query, myConnection);
            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Ошибка редактирования записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
        }
        categoryName.Text = "";
        categoryCode.Text = "";
        panel8.Visible = false;
```

```
    fillCategory();
}

private void button51_Click(object sender, EventArgs e)
{
    ttnDeleteNomenclatureButton.Enabled = true;
    countNomenclature++;
    y += 30;
    addNomenclatureTTN(y);
}

private void button50_Click(object sender, EventArgs e)
{
    deleteNomenclatureTTN(countNomenclature - 1);
    countNomenclature--;
    y -= 30;
    if (countNomenclature == 1)
    {
        ttnDeleteNomenclatureButton.Enabled = false;
    }
}

private void button53_Click(object sender, EventArgs e)
{
    int count = int.Parse(ttnCountNomenclature.Text.ToString());
    if (count > 1) ttnDeleteNomenclatureButton.Enabled = true;
    else ttnDeleteNomenclatureButton.Enabled = false;
    if (count >= countNomenclature)
    {
        int k = countNomenclature;
        for (int j = k + 1; j <= count; j++)
        {
            countNomenclature++;
            y += 30;
            addNomenclatureTTN(y);
        }
    }
    else
    {
        int k = countNomenclature;
        for (int j = k - 1; j >= count; j--)
        {
            countNomenclature--;
            y -= 30;
            if (countNomenclature == 1)
            {
                ttnDeleteNomenclatureButton.Enabled = false;
            }
            deleteNomenclatureTTN(j);
        }
    }
}

private void button52_Click(object sender, EventArgs e)
{
    string query;
    SqlCommand command;
    SqlDataReader reader;
    SqlTransaction tran = null;
    List<double> SellingPrice = new List<double>();
    List<double> NewSellingPrice = new List<double>();
    List<int> codeRevaluationOfGoods = new List<int>();
    List<int> codeGoodsMovement = new List<int>();
    tran = myConnection.BeginTransaction("Transaction1");
    int codeTTN = 0;
    SellingPrice.Clear();
    NewSellingPrice.Clear();
    for (int i = 0; i < countNomenclature; i++)
    {
        query = "SELECT Nomenclature.SellingPrice " +
```

```
                "FROM Nomenclature " +
                 "WHERE Nomenclature.Code = " + codeNom[comboBoxName[i].SelectedIndex].ToString();
            command = new SqlCommand(query, myConnection, tran);
            try
            {
               reader = command.ExecuteReader();
               reader.Read();
               SellingPrice.Add(double.Parse(reader[0].ToString()));
               NewSellingPrice.Add(Math.Round(double.Parse(textBoxTotalCost[i].Text.ToString())                    /
double.Parse(textBoxScopeOfDelivery[i].Text.ToString()), 2));
               reader.Close();
            }
            catch
            {
               MessageBox.Show("Ошибка сохранения данных по ТТН. Оформление ТТН Отменено. Обратитесь к
администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
               return;
            }
         }
         if (fl == "add")
         {
            query = "SELECT IDENT_CURRENT('TTN')";
            command = new SqlCommand(query, myConnection, tran);
            try
            {
               reader = command.ExecuteReader();
               reader.Read();
               codeTTN = int.Parse(reader[0].ToString()) + 1;
               reader.Close();
            }
            catch
            {
               MessageBox.Show("Ошибка сохранения данных по ТТН. Оформление ТТН Отменено. Обратитесь к
администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
               return;
            }
            query = "";
            try
            {
               query += "INSERT INTO TTN (NumberTTN, Date, CodeEmploye) " +
                     "SELECT '" + ttnNumber.Text + "', '" + ttnDate.Value.Year + "-" +
                                          ttnDate.Value.Month + "-" +
                                          ttnDate.Value.Day + "', " +
                              codeEmploye[ttnEmployee.SelectedIndex] + "; ";
               for (int i = 0; i < countNomenclature; i++)
               {
                  int codeProduct = codeNom[comboBoxName[i].SelectedIndex];
                  string up = CommaToPoint(textBoxUnitPrice[i].Text);
                  string tp = CommaToPoint(textBoxTotalCost[i].Text);
                  string sod = CommaToPoint(textBoxScopeOfDelivery[i].Text.ToString());
                  query += " UPDATE Nomenclature " +
                                    "SET UnitPrice=" + up + ", " +
                                       "SellingPrice="   +   CommaToPoint(Math.Round(Double.Parse(textBoxTotalCost[i].Text)   /
Double.Parse(textBoxScopeOfDelivery[i].Text), 2).ToString()) + " " +
                                    " WHERE Code=" + codeNom[comboBoxName[i].SelectedIndex].ToString() + ";";
                  string Weight = CommaToPoint(textBoxScopeOfDelivery[i].Text.ToString());
                  query += " INSERT INTO GoodsMovement(Date, " +
                                          "CodeNomenclature, " +
                                          "Quantity, " +
                                          "Sum, " +
                                          "CodeTTN," +
                                          "UnitPrice) " +
                              "VALUES('" + ttnDate.Value.Year.ToString() + "-" +
                                    ttnDate.Value.Month.ToString() + "-" +
                                    ttnDate.Value.Day.ToString() + "', " +
                                    codeNom[comboBoxName[i].SelectedIndex].ToString() + ", " +
                                    Weight + ", " +
                                    tp + ", " + codeTTN.ToString() + ", " +
                                    up + "); ";
               }
```

```
                command = new SqlCommand(query, myConnection, tran);
                tran.Save("save1");
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Ошибка сохранения данных по ТТН. Оформление ТТН Отменено. Обратитесь к администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                tran.Rollback("save1");
                tran.Commit();
                return;
            }
            tran.Commit();
        }
        else
        {
            codeTTN = int.Parse(ttnTable.Rows[ttnTable.CurrentRow.Index].Cells[0].Value.ToString());
            DateTime date = DateTime.Parse(ttnTable.Rows[ttnTable.CurrentRow.Index].Cells[2].Value.ToString());
            query = "UPDATE TTN " +
                "SET NumberTTN = '" + ttnNumber.Text + "', " +
                "Date = '" + ttnDate.Value.Year + "-" +
                        ttnDate.Value.Month + "-" +
                        ttnDate.Value.Day + "', " +
                "CodeEmploye = " + codeEmploye[ttnEmployee.SelectedIndex].ToString() + " " +
                "WHERE Code = " + codeTTN.ToString() + "; ";
            int ia = 0, ib = 0, ic = 0;
            ia = countNomenclature;
            ic = countNomenclature;
            for (int i = 0; i < countNomenclature; i++)
            {
                string up = CommaToPoint(textBoxUnitPrice[i].Text);
                string tp = CommaToPoint(textBoxTotalCost[i].Text);
                string sod = textBoxScopeOfDelivery[i].Text.ToString();
                query += " UPDATE Nomenclature " +
                                "SET UnitPrice=" + up + ", " +
                                    "SellingPrice="    +    CommaToPoint(Math.Round(Double.Parse(textBoxTotalCost[i].Text)    /
Double.Parse(textBoxScopeOfDelivery[i].Text), 2).ToString()) + " " +
                                " WHERE Code=" + codeNom[comboBoxName[i].SelectedIndex].ToString() + "; ";

            }
            string query2 = "";
            query2 = "SELECT code " +
                "FROM GoodsMovement " +
                "WHERE date = '" + ttnDate.Value.Year.ToString() + "-" +
                                ttnDate.Value.Month.ToString() + "-" +
                                ttnDate.Value.Day.ToString() + "' and CodeTTN = " + codeTTN.ToString() + "; ";
            command = new SqlCommand(query2, myConnection, tran);
            try
            {
                reader = command.ExecuteReader();
                codeGoodsMovement.Clear();
                while (reader.Read())
                    codeGoodsMovement.Add(int.Parse(reader[0].ToString()));
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка сохранения данных по ТТН. Оформление ТТН Отменено. Обратитесь к администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            for (int i = 0; i < ia; i++)
            {
                string TotalCost = CommaToPoint(textBoxTotalCost[i].Text);
                string UnitPrice = CommaToPoint(textBoxUnitPrice[i].Text);
                string Weight = CommaToPoint(textBoxScopeOfDelivery[i].Text.ToString());
                query += "UPDATE GoodsMovement " +
                    "SET Date = '" + ttnDate.Value.Year.ToString() + "-" + ttnDate.Value.Month.ToString() + "-" +
ttnDate.Value.Day.ToString() + "', " +
                    "CodeNomenclature =" + codeNom[comboBoxName[i].SelectedIndex].ToString() + ", " +
```

```
                "Quantity =" + Weight + ", " +
                "SUM =" + TotalCost + ", " +
                "UnitPrice=" + UnitPrice + " " +
                "WHERE Code = " + codeGoodsMovement[i] + "; ";
            }
            for (int i = ia; i < ib; i++)
            {
                query += "DELETE FROM GoodsMovement WHERE code = " + codeGoodsMovement[i] + "; ";
            }
            for (int i = ia; i < ic; i++)
            {
                string TotalCost = CommaToPoint(textBoxTotalCost[i].Text);
                string UnitPrice = CommaToPoint(textBoxUnitPrice[i].Text);
                string Weight = CommaToPoint(textBoxScopeOfDelivery[i].Text.ToString());
                query += " INSERT INTO GoodsMovement(Date, " +
                                "CodeNomrnclature, " +
                                "Quantity, " +
                                "Sum, " +
                                "CodeTTN," +
                                "UnitPrice) " +
                "VALUES('" + ttnDate.Value.Year.ToString() + "-" +
                        ttnDate.Value.Month.ToString() + "-" +
                        ttnDate.Value.Day.ToString() + "', " +
                        codeNom[comboBoxName[i].SelectedIndex].ToString() + ", " +
                        Weight + ", " +
                        TotalCost + ", " + codeTTN.ToString() + ", " + UnitPrice + "); ";
            }
            try
            {
                command = new SqlCommand(query, myConnection, tran);
                tran.Save("save1");
                command.ExecuteNonQuery();
                tran.Commit();
            }
            catch
            {
                MessageBox.Show("Ошибка сохранения данных по ТТН. Оформление ТТН Отменено. Обратитесь к администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                tran.Rollback("save1");
                tran.Commit();
                return;
            }
            button55_Click(sender, e);
        }
        ttnNumber.Text = "";
        ttnEmployee.Items.Clear();
        ttnEmployee.Text = "";
        ttnCountNomenclature.Text = "";
        ttnDate.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month, System.DateTime.Today.Day, 09, 00, 00);
        ttnAddEditPanel.Visible = false;
        for (int i = countNomenclature - 1; i >= 0; i--)
        {
            deleteNomenclatureTTN(i);
        }
        countNomenclature = 0;
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        if (ttnCheckDateSearch.Checked)
        {
            ttnDateSearch.Visible = true;
            ttnOkSearchButton.Visible = true;
        }
        else
        {
            ttnDateSearch.Visible = false;
        }
    }
```

```
        private void checkBox2_CheckedChanged(object sender, EventArgs e)
        {
            checkTtnSearchComboBox(ttnCheckNumberSearch, ttnNumberSearch, "NumberTTN", "TTN");
        }

        private void button55_Click(object sender, EventArgs e)
        {

        }

        private void Shop_FormClosed(object sender, FormClosedEventArgs e)
        {
            if (close)
            {
                System.Windows.Forms.Application.Exit();
            }

        }

        private void button57_Click(object sender, EventArgs e)
        {
            ttnNumber.Text = "";
            ttnEmployee.SelectedIndex = -1;
            ttnEmployee.Items.Clear();
            ttnCountNomenclature.Text = "";
            ttnDate.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 09, 00, 00);
            for (int i = 0; i < countNomenclature; i++)
            {
                comboBoxName[i].Items.Clear();
                comboBoxName[i].Text = "";
                comboBoxName[i].Enabled = false;
                textBoxScopeOfDelivery[i].Text = "";
                textBoxTotalCost[i].Text = "";
                textBoxUnitPrice[i].Text = "";
            }
            for (int i = countNomenclature - 1; i >= 0; i--)
            {
                deleteNomenclatureTTN(i);

            }
            countNomenclature = 0;
            ttnPreviewPanel.Visible = true;
            ttnAddEditPanel.Visible = false;
        }

        private void ttnAddButton_Click(object sender, EventArgs e)
        {
            fl = "add";
            ttnBackButton.Visible = false;
            ttnDate.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 09, 00, 00);
            ttnAddEditPanel.Visible = true;
            ttnSaveButton.Visible = true;
            ttnBackButton.Visible = false;
            ttnAddNomenclatureButton.Visible = true;
            ttnDeleteNomenclatureButton.Visible = true;
            ttnCountNomenclatureLabel.Visible = true;
            ttnCountNomenclature.Visible = true;
            ttnCountNomenclatureOkButton.Visible = true;
            countNomenclature++;
            y = 170;
            addNomenclatureTTN(y);
            string query = "SELECT Employe.Code, FullName FROM Employe";
            SqlCommand command = new SqlCommand(query, myConnection);
            command = new SqlCommand(query, myConnection);
            codeEmploye.Clear();
            ttnEmployee.Items.Clear();
            SqlDataReader reader;
```

```
            try
            {
                reader = command.ExecuteReader();
                while (reader.Read())
                {
                    codeEmploye.Add(int.Parse(reader[0].ToString()));
                    ttnEmployee.Items.Add(reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            for (int i = 0; i < countNomenclature; i++)
            {
                comboBoxName[i].Enabled = true;
                comboBoxName[i].Items.Clear();
            }
            query = "SELECT Code, Name FROM Nomenclature";
            command = new SqlCommand(query, myConnection);
            try
            {
                reader = command.ExecuteReader();
                codeNom.Clear();
                for (int i = 0; i < countNomenclature; i++)
                    comboBoxName[i].Items.Clear();
                while (reader.Read())
                {
                    codeNom.Add(int.Parse(reader[0].ToString()));
                    for (int i = 0; i < countNomenclature; i++)
                        comboBoxName[i].Items.Add(reader[1].ToString());

                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        private void ttnEditButton_Click(object sender, EventArgs e)
        {
            ttnBackButton.Visible = true;
            button7.Enabled = true;
            ttnSaveButton.Enabled = true;
            ttnNumber.Enabled = false;
            ttnDate.Enabled = false;
            string query;
            SqlCommand command;
            SqlDataReader reader;
            ttnBackButton.Visible = true;
            ttnNumber.Text = ttnTable.Rows[ttnTable.CurrentRow.Index].Cells[1].Value.ToString();
            DateTime dd = DateTime.Parse(ttnTable.Rows[ttnTable.CurrentRow.Index].Cells[2].Value.ToString());
            ttnDate.Value = new DateTime(dd.Year, dd.Month, dd.Day);
            query = "SELECT Code, FullName FROM Employe";
            command = new SqlCommand(query, myConnection);
            codeEmploye.Clear();
            ttnEmployee.Items.Clear();
            try
            {
                reader = command.ExecuteReader();
                while (reader.Read())
                {
                    codeEmploye.Add(int.Parse(reader[0].ToString()));
```

```
            ttnEmployee.Items.Add(reader[1].ToString());
        }
        reader.Close();
      }
      catch
      {
        MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
      }
      ttnEmployee.SelectedItem = ttnTable.Rows[ttnTable.CurrentRow.Index].Cells[3].Value.ToString();
      ComboBox cbName = new ComboBox();
      query = "SELECT Code, Name FROM Nomenclature";
      selectCode("Nomenclature", query, codeNom, cbName);
      query = "SELECT Nomenclature.Name, " +
                "GoodsMovement.Quantity, " +
                "GoodsMovement.UnitPrice, " +
                "GoodsMovement.Sum, " +
                "GoodsMovement.Code " +
            "FROM GoodsMovement, " +
                "Nomenclature, " +
                "Category, " +
                "TTN " +
            "WHERE GoodsMovement.CodeNomenclature = Nomenclature.Code and " +
                 "Nomenclature.CodeCategory = Category.Code and " +
                 "GoodsMovement.CodeTTN = TTN.Code and " +
                 "TTN.NumberTTN = '" + ttnNumber.Text + "'";
      command = new SqlCommand(query, myConnection);
      y = 140;
      int i = -1;
      countNomenclature = 0;
      try
      {
        reader = command.ExecuteReader();

        while (reader.Read())
        {
          y += 30;
          addNomenclatureTTN(y);
          countNomenclature++;
          i++;
          int count1 = cbName.Items.Count;
          if (i == 0)
          {
            comboBoxName[i].Items.Clear();
            for (int k = 0; k < count1; k++)
              comboBoxName[i].Items.Add(cbName.Items[k].ToString());
          }
          comboBoxName[i].SelectedItem = reader[0].ToString();
          comboBoxName[i].Enabled = true;
          textBoxScopeOfDelivery[i].Text = reader[1].ToString();
          textBoxUnitPrice[i].Text = reader[2].ToString();
          textBoxTotalCost[i].Text = reader[3].ToString();

        }
        reader.Close();
      }
      catch
      {
        MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
      }
      ttnAddEditPanel.Visible = true;
      fl = "edit";
    }

    private void ttnClearButton_Click(object sender, EventArgs e)
    {
      ttnNumber.Text = "";
```

```
        ttnEmployee.SelectedIndex = -1;
        ttnCountNomenclature.Text = "";
        ttnDate.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 09, 00, 00);
        for (int i = countNomenclature - 1; i > 0; i--)
        {
            deleteNomenclatureTTN(i);
        }
        countNomenclature = 1;
        y = 170;
        comboBoxName[countNomenclature - 1].Items.Clear();
        comboBoxName[countNomenclature - 1].Text = "";
        comboBoxName[countNomenclature - 1].Enabled = false;
        textBoxScopeOfDelivery[countNomenclature - 1].Text = "";
        textBoxTotalCost[countNomenclature - 1].Text = "";
        textBoxUnitPrice[countNomenclature - 1].Text = "";
    }

    private void button4_Click(object sender, EventArgs e)
    {
        string codeCheck = "";
        SqlTransaction tran = null;
        tran = myConnection.BeginTransaction("Transaction1");
        string query = "SELECT IDENT_CURRENT('Check')";
        SqlCommand command = new SqlCommand(query, myConnection, tran);
        try
        {
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();
            codeCheck = (int.Parse(reader[0].ToString()) + 1).ToString();
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка сохранения данных по продаже. Оформление продажи Отменено. Обратитесь к
администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "INSERT INTO [Check] (CodeEmploye, Date) " +
                "VALUES(" + codeEmploye[comboBox1.SelectedIndex].ToString() + ", '" + DateTime.Today.Year + "-" +
DateTime.Today.Month + "-" + DateTime.Today.Day + "'); ";
        for (int i = 0; i < countNomenclatureCheck; i++)
            query += "INSERT INTO GoodsMovement(Date, CodeNomenclature, Quantity, Sum, CodeCheck, UnitPrice) " +
                    "VALUES('" + DateTime.Today.Year + "-" + DateTime.Today.Month + "-" + DateTime.Today.Day + "', " +
codeNom[comboBoxNameCheck[i].SelectedIndex] + ", -" + numericUpDownCountCheck[i].Text + ", " +
CommaToPoint(textBoxTotalCostCheck[i].Text) + ", " + codeCheck + ", (SELECT UnitPrice FROM Nomenclature WHERE
Code = " + codeNom[comboBoxNameCheck[i].SelectedIndex] + ")); ";
        try
        {
            command = new SqlCommand(query, myConnection, tran);
            tran.Save("save1");
            command.ExecuteNonQuery();
        }
        catch
        {
            MessageBox.Show("Ошибка сохранения данных по чеку. Оформление чека Отменено. Обратитесь к
администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            tran.Rollback("save1");
            tran.Commit();
            return;
        }
        tran.Commit();
        query = "IF OBJECT_ID('ReportCheck') IS NOT NULL DROP VIEW ReportCheck";
        try
        {
            command = new SqlCommand(query, myConnection, tran);
            command.ExecuteNonQuery();
        }
        catch
        {
```

```
            MessageBox.Show("Ошибка вывода чека. Обратитесь к администратору.", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
        }
        query = "CREATE VIEW ReportCheck AS " +
                "SELECT [Check].code AS NumberCheck, " +
                    "[Check].Date AS Date, " +
                    "Nomenclature.Name AS Nomenclature, " +
                    "-1 * GoodsMovement.Quantity AS Quantity, " +
                    "Round(GoodsMovement.Sum/(-1 * GoodsMovement.Quantity),2) AS UnitPrice, " +
                    "GoodsMovement.Sum AS Cost, " +
                    "Employe.FullName As Employees " +
                "FROM Nomenclature, GoodsMovement, [Check], Employe " +
                "WHERE GoodsMovement.CodeCheck = [Check].Code and " +
                    "Employe.Code=[Check].CodeEmploye and " +
                    "GoodsMovement.CodeNomenclature=Nomenclature.Code and " +
                    "[Check].Code=" + codeCheck + ";";
        try
        {
            command = new SqlCommand(query, myConnection, tran);
            command.ExecuteNonQuery();
            ReportCheck f = new ReportCheck();
            f.reportViewer1.Visible = true;
            f.Show();
        }
        catch
        {
            MessageBox.Show("Ошибка вывода чека. Обратитесь к администратору.", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
        }
    }

    private void button2_Click_1(object sender, EventArgs e)
    {
        button1.Enabled = true;
        countNomenclatureCheck++;
        y += 30;
        addNomenclatureCheck(y);
    }

    private void button1_Click_1(object sender, EventArgs e)
    {
        deleteNomenclatureCheck(countNomenclatureCheck - 1);
        countNomenclatureCheck--;
        y -= 30;
        if (countNomenclatureCheck == 1)
        {
            button1.Enabled = false;
        }
    }

    private void textBox5_TextChanged(object sender, EventArgs e)
    {
        if (textBox5.Text != "")
        {
            foreach (Control cl in panel2.Controls)
                if (cl.Focused) cl.Text = textBox5.Text;
            textBox5.Text = "";
        }
    }

    private void button5_Click_1(object sender, EventArgs e)
    {

    }

    private void button6_Click_1(object sender, EventArgs e)
    {
        try
```

```
        {
            string query = "IF OBJECT_ID('ImplementationReport') IS NOT NULL DROP VIEW ImplementationReport; ";
            SqlCommand command = new SqlCommand(query, myConnection);
            command.ExecuteNonQuery();
            query = "CREATE VIEW ImplementationReport AS " +
                    "SELECT [Check].Date AS Date, " +
                        "Nomenclature.Name AS Nomenclature, " +
                        "GoodsMovement.Quantity * (-1) AS Quantity, " +
                        "GoodsMovement.Sum AS Sum, " +
                        "Employe.FullName AS Employe " +
                     "FROM GoodsMovement, " +
                        "[Check], " +
                        "Nomenclature, " +
                        "Employe " +
                    "WHERE GoodsMovement.CodeCheck = [Check].Code and " +
                        "Nomenclature.Code=GoodsMovement.CodeNomenclature and " +
                        "[Check].CodeEmploye=Employe.Code and " +
                        "GoodsMovement.Date between '" + dateTimePicker5.Value.Year + "-" +
                         dateTimePicker5.Value.Month + "-" +
                         dateTimePicker5.Value.Day + "' and '" + dateTimePicker4.Value.Year + "-" +
                         dateTimePicker4.Value.Month + "-" +
                         dateTimePicker4.Value.Day + "'";
            command = new SqlCommand(query, myConnection);
            command.ExecuteNonQuery();
            reportViewer5.LocalReport.SetParameters(new                Microsoft.Reporting.WinForms.ReportParameter("date1",
dateTimePicker5.Value.ToString("dd.MM.yyyy")));
            reportViewer5.LocalReport.SetParameters(new                Microsoft.Reporting.WinForms.ReportParameter("date2",
dateTimePicker4.Value.ToString("dd.MM.yyyy")));
            ImplementationReportTableAdapter.Fill(StoreDataSet.ImplementationReport);
            reportViewer5.RefreshReport();
            reportViewer5.Visible = true;
        }
        catch
        {
            MessageBox.Show("Ошибка построения отчета по реализациям. Откройти его вручную или обратитесь к
администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void ttnDateSearch_ValueChanged(object sender, EventArgs e)
    {
        ttnOkSearchButton.Enabled = true;
    }

    private void ttnNumberSearch_SelectedIndexChanged(object sender, EventArgs e)
    {
        ttnOkSearchButton.Visible = true;
    }

    private void employeesCode_TextChanged(object sender, EventArgs e)
    {
        checkEmployeesInput();
    }

    private void employeesFullName_TextChanged(object sender, EventArgs e)
    {
        checkEmployeesInput();
    }

    private void employeesAddress_TextChanged(object sender, EventArgs e)
    {
        checkEmployeesInput();
    }

    private void employeesPhone_TextChanged(object sender, EventArgs e)
    {
        checkEmployeesInput();
    }

    private void employeesEmail_TextChanged(object sender, EventArgs e)
```

```
{
   checkEmployeesInput();
}

private void employeesPosition_SelectedIndexChanged(object sender, EventArgs e)
{
   checkEmployeesInput();
}

private void positionsName_TextChanged(object sender, EventArgs e)
{
   if (positionsName.Text != "")
   {
      positionsOkButton.Enabled = true;
   }
   else
   {
      positionsOkButton.Enabled = false;
   }
}

private void nomenclatureName_TextChanged(object sender, EventArgs e)
{
   checkNomenclatureInput();
}

private void categoryName_TextChanged(object sender, EventArgs e)
{
   if (categoryName.Text != "")
   {
      categoryOkButton.Enabled = true;
   }
   else
   {
      categoryOkButton.Enabled = false;
   }
}

private void comboBox1_SelectedIndexChanged_1(object sender, EventArgs e)
{
   if (comboBox1.SelectedIndex != -1)
   {
      button4.Enabled = true;
   }
   else
   {
      button4.Enabled = false;
   }
}

private void button7_Click(object sender, EventArgs e)
{
   ttnClearButton_Click(sender, e);
   ttnAddEditPanel.Visible = false;
}

private void ttnTable_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{

}

private void TtnCountNomenclature_KeyPress(object sender, KeyPressEventArgs e)
{
   if (!Char.IsDigit(e.KeyChar) && e.KeyChar != 8)
      e.Handled = true;
}

private void Label17_Click(object sender, EventArgs e)
{
```

```
        }

        private void reportsTabs_Click(object sender, EventArgs e)
        {
            if (goodsTabs.SelectedTab.Name == "remnantsOfGoods")
            {
                try
                {
                    string query = "IF OBJECT_ID('ReportUnloadingOfResidues') IS NOT NULL " +
                                   "DROP VIEW ReportUnloadingOfResidues;";
                    SqlCommand command = new SqlCommand(query, myConnection);
                    command.ExecuteNonQuery();
                    query = "CREATE VIEW ReportUnloadingOfResidues AS " +
                        "SELECT Category.Name AS Category, " +
                            "Nomenclature.Name AS Nomenclature, " +
                            "Nomenclature.Balance AS ProductBalance, " +
                            "Nomenclature.SellingPrice AS SellingPrice,   " +
                            "ROUND(Nomenclature.Balance * Nomenclature.SellingPrice, 2) AS Cost " +
                        "FROM Nomenclature, Category " +
                        "WHERE Nomenclature.CodeCategory = Category.Code and Nomenclature.Balance > 0";
                    command = new SqlCommand(query, myConnection);
                    command.ExecuteNonQuery();
                      reportViewer2.RefreshReport();
                }
                catch
                {
                    MessageBox.Show("Ошибка построения отчета \"Выгрузка остатков\". Откройти его вручную или обратитесь
к администратору.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        private void ttnOkSearchButton_Click(object sender, EventArgs e)
        {
            string query;
            ttnTable.Rows.Clear();
            ttnTable.Visible = true;
            query = editTTN(ttnCheckDateSearch, ttnCheckNumberSearch, ttnDateSearch, ttnNumberSearch.Text);
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    ttnTable.Rows.Add(reader[0].ToString(),
                                      reader[1].ToString(),
                                      reader[2].ToString(),
                                      reader[3].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            ttnEditButton.Enabled = false;
        }

        private void employeesOkButton_Click(object sender, EventArgs e)
        {
            string query;
            SqlCommand command;
            if (fl == "add")
            {
                query = "INSERT INTO Employe(FullName, DOB, Address, Phone, email, CodePosition) " +
                        "SELECT '" + employeesFullName.Text + "', '" +
                                 employeesDOB.Value.Year + "-" +
                                 employeesDOB.Value.Month + "-" +
                                 employeesDOB.Value.Day + "', '" +
```

```
                        employeesAddress.Text + "', '" +
                        employeesPhone.Text + "', '" +
                        employeesEmail.Text + "', Code FROM Positions WHERE Name = '" +
                        employeesPosition.Text + "'";
            command = new SqlCommand(query, myConnection);
            try
            {
               command.ExecuteNonQuery();
            }
            catch
            {
               MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
               return;
            }
            string user = GetTranslit(SurnameName(employeesFullName.Text));
            int count1 = 0;
            query = "SELECT Code FROM Users WHERE Login='" + user + "'";
            command = new SqlCommand(query, myConnection);
            try
            {
               SqlDataReader reader = command.ExecuteReader();
               while (reader.Read())
               {
                  count1++;
               }
               reader.Close();
            }
            catch
            {
               MessageBox.Show("Ошибка доступа к БД. Обратитесь к администратору", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
               return;
            }
            if (count1 == 0)
            {
               Random rand = new Random();
               int temp;
               temp = rand.Next(1000000);
               query = "INSERT INTO USERS(Login, Password, CodeEmploye, CodeRole) " +
                     "SELECT '" + user + "', " + "'" + temp.ToString() + "', " + employeesCode.Text + ", Code FROM Roles Where
Name = '" + employeesRole.Text + "'";
               try
               {
                  command = new SqlCommand(query, myConnection);
                  command.ExecuteNonQuery();
               }
               catch
               {
                  MessageBox.Show("Ошибка создания логина пользователя. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                  return;
               }
               MessageBox.Show("Логин: " + user + "\nПароль: " + temp.ToString(), "Запишите данные",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
            else
            {
               LoginE.Text = user;
               employeesLoginPanel.Visible = true;
            }
         }
         if (fl == "edit")
         {
            query = "UPDATE Employe " +
                  " SET FullName = '" + employeesFullName.Text + "'," +
                     " DOB = '" + employeesDOB.Value.Year + "-" +
                           employeesDOB.Value.Month + "-" +
                           employeesDOB.Value.Day + "'," +
                     " Address = '" + employeesAddress.Text + "'," +
```

```
                    " Phone = '" + employeesPhone.Text + "'," +
                    " email='" + employeesEmail.Text + "'," +
                    " CodePosition=(SELECT Code " +
                            "FROM Positions " +
                            "WHERE Name='" + employeesPosition.Text + "') " +
                "  WHERE Code =" + employeesCode.Text;
            command = new SqlCommand(query, myConnection);
            try
            {
               command.ExecuteNonQuery();
            }
            catch
            {
               MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
               return;
            }
         }
         clearEmployeesAddEdit();
         employeesTable.Rows.Clear();
         query = "SELECT Employe.Code, Employe.FullName, CONVERT (varchar,Employe.DOB,104), Employe.Address,
Employe.Phone, Employe.email, Positions.Name FROM Employe, Positions WHERE Employe.CodePosition=Positions.Code";
         command = new SqlCommand(query, myConnection);
         try
         {
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
               employeesTable.Rows.Add(reader[0].ToString(),
                           reader[1].ToString(),
                           reader[2].ToString(),
                           reader[3].ToString(),
                           reader[4].ToString(),
                           reader[5].ToString(),
                           reader[6].ToString());
            }
            reader.Close();
         }
         catch
         {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
         }
      }

      private void employeesCancelButton_Click(object sender, EventArgs e)
      {
         clearEmployeesAddEdit();
      }

      private void employeesLoginOkButton_Click(object sender, EventArgs e)
      {
         string query;
         SqlCommand command;
         int count1 = 0;
         query = "SELECT Code FROM Users WHERE Login='" + LoginE.Text + "'";
         command = new SqlCommand(query, myConnection);
         try
         {
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
               count1++;
            }
            reader.Close();
         }
         catch
         {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
```

```
        }
        if (count1 == 0)
        {
            Random rand = new Random();
            int temp;
            temp = rand.Next(1000000);
            query = "INSERT INTO USERS(Login, Password, CodeEmploye, CodeRole) " +
                  "SELECT '" + LoginE.Text + "', " + "'" + temp.ToString() + "', '" + employeesCode.Text + "', Code FROM Roles
Where Name = '" + employeesRole.Text + "'";
            command = new SqlCommand(query, myConnection);
            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            MessageBox.Show("Логин: " + LoginE.Text + "\nПароль: " + temp.ToString(), "Запишите данные",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            LoginE.Text = "";
            employeesLoginPanel.Visible = false;
        }
        else
        {
            return;
        }
    }

    private void usersAddButton_Click(object sender, EventArgs e)
    {
        fl = "add";
        SqlDataReader reader;
        string query = "SELECT IDENT_CURRENT('Users')";
        SqlCommand command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            reader.Read();
            usersCode.Text = (int.Parse(reader[0].ToString()) + 1).ToString();
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "SELECT FullName FROM Employe WHERE Employe.Code not in (Select Users.CodeEmploye FROM Users)";
        command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                usersEmployee.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "SELECT Name FROM Roles";
        command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            while (reader.Read())
```

```
            {
                usersRole.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "SELECT IDENT_CURRENT('Users')";
        command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            reader.Read();
            employeesCode.Text = (int.Parse(reader[0].ToString()) + 1).ToString();
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка связи с БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        usersEditAddPanel.Visible = true;
        usersResetPasswordButton.Enabled = false;
    }

    private void usersEditButten_Click(object sender, EventArgs e)
    {
        fl = "edit";

        string query = "SELECT FullName FROM Employe";
        SqlCommand command = new SqlCommand(query, myConnection);
        SqlDataReader reader;
        try
        {
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                usersEmployee.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "SELECT Name FROM Roles";
        command = new SqlCommand(query, myConnection);
        try
        {
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                usersRole.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        query = "SELECT Users.Code, Users.Login, Employe.FullName, Roles.Name FROM Users, Employe, Roles WHERE Users.CodeRole=Roles.Code and Users.CodeEmploye=Employe.Code and Users.Code=" + usersTable.Rows[usersTable.CurrentRow.Index].Cells[0].Value.ToString();
        command = new SqlCommand(query, myConnection);
        try
```

```
        {
            reader = command.ExecuteReader();
            reader.Read();
            usersCode.Text = reader[0].ToString();
            usersLogin.Text = reader[1].ToString();
            usersEmployee.SelectedItem = reader[2].ToString();
            usersRole.SelectedItem = reader[3].ToString();
            usersEditAddPanel.Visible = true;
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        usersResetPasswordButton.Enabled = true;
    }

    private void usersDeleteButton_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Вы действительно хотите удалить выбранную запись?", "Удаление",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)
        {
            deleteRecords(usersTable.Rows[usersTable.CurrentRow.Index].Cells[0].Value.ToString(), "Users");
            fillUsers();
        }
    }

    private void usersOkButton_Click(object sender, EventArgs e)
    {
        string query;
        SqlCommand command;
        int count1 = 0;
        if (fl == "add")
        {
            query = "SELECT Code FROM Users WHERE Login='" + usersLogin.Text + "'";
            command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    count1++;
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (count1 == 1)
            {
                MessageBox.Show("Введенный логин занят", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
            }
            Random rand = new Random();
            int temp;
            temp = rand.Next(1000000);
            query = "INSERT INTO USERS(Login, Password, CodeEmploye, CodeRole) " +
                "SELECT '" + usersLogin.Text + "', " + "'" + temp.ToString() + "', Employe.Code, Roles.Code FROM Roles,
Employe Where Roles.Name = '" + usersRole.Text + "' and Employe.FullName = '" + usersEmployee.Text + "'";
            command = new SqlCommand(query, myConnection);
            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
```

```
            MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
          }
          MessageBox.Show("Логин: " + usersLogin.Text + "\nПароль: " + temp.ToString(), "Запишите данные",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
        if (fl == "edit")
        {
          query = "UPDATE Users " +
                " SET Login = '" + usersLogin.Text + "'," +
                  " CodeEmploye = (SELECT Code FROM Employe WHERE FullName='" + usersEmployee.Text + "'), " +
                  " CodeRole = (SELECT Code FROM Roles WHERE Name ='" + usersRole.Text + "')" +
                " WHERE Code =" + usersCode.Text;
          command = new SqlCommand(query, myConnection);
          try
          {
            command.ExecuteNonQuery();
          }
          catch
          {
            MessageBox.Show("Ошибка добавления записи. Запись не добавлена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
          }
        }
        usersEditAddPanel.Visible = false;
        usersCode.Text = "";
        usersLogin.Text = "";
        usersEmployee.Items.Clear();
        usersEmployee.Text = "";
        usersRole.Items.Clear();
        usersRole.Text = "";
        fillUsers();
      }

      private void usersCancelButton_Click(object sender, EventArgs e)
      {
        usersEditAddPanel.Visible = false;
        usersCode.Text = "";
        usersLogin.Text = "";
        usersEmployee.Items.Clear();
        usersEmployee.Text = "";
        usersRole.Items.Clear();
        usersRole.Text = "";
      }

      private void usersResetPasswordButton_Click(object sender, EventArgs e)
      {
        if (MessageBox.Show("Вы действительно хотите сбросить пароль?", "Сброс пароля", MessageBoxButtons.OKCancel,
MessageBoxIcon.Question) == DialogResult.OK)
        {

          Random rand = new Random();
          int temp;
          temp = rand.Next(1000000);
          string query = "UPDATE Users " +
                " SET Password = " + temp.ToString() +
                " WHERE Code =" + usersCode.Text + "";
          SqlCommand command = new SqlCommand(query, myConnection);
          try
          {
            command.ExecuteNonQuery();
          }
          catch
          {
            MessageBox.Show("Ошибка Сброса пароля", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
          }
          MessageBox.Show("Логин: " + usersLogin.Text + "\nПароль: " + temp.ToString(), "Запишите данные",
```

```
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                fillUsers();
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click_1(object sender, EventArgs e)
        {
            Authorization f = new Authorization();
            f.Login.Text = "";
            f.password.Text = "";
            f.Show();
            close = false;
            transliter.Clear();
            this.Close();
        }

        private void TtnTable_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)
        {
            ttnEditButton.Enabled = true;
        }
    }
}
```

```
using Microsoft.Reporting.WinForms;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Drawing;
using System.Text;
using System.Windows.Forms;



namespace Store
{
   public partial class Shop : Form
   {
      public string role1, user1;
      string connectionString = "Data Source=.\\SQLEXPRESS;" +
            "                    Initial Catalog=Store;" +
            "                    Integrated Security=True;" +
            "                    MultipleActiveResultSets=True";
      SqlConnection myConnection;
      string fl;
      List<ComboBox> comboBoxName = new List<ComboBox>();
      List<TextBox> textBoxScopeOfDelivery = new List<TextBox>();
      List<TextBox> textBoxUnitPrice = new List<TextBox>();
      List<TextBox> textBoxTotalCost = new List<TextBox>();
      List<TextBox> textBoxUnitPriceCheck = new List<TextBox>();
      List<ComboBox> comboBoxNameCheck = new List<ComboBox>();
      List<TextBox> numericUpDownCountCheck = new List<TextBox>();
      List<TextBox> textBoxTotalCostCheck = new List<TextBox>();
      List<int> codeNom = new List<int>();
      List<int> codeEmploye = new List<int>();
      List<int> codeNomenclature = new List<int>();
      int countNomenclature = 0;
      int countNomenclatureCheck = 0;
      int y;
      bool close = true;


      //изменения табеля смены
      List<ComboBox> comboBoxArr = new List<ComboBox>();
      List<Label> Label1Arr = new List<Label>();
      List<Label> Label2Arr = new List<Label>();
      List<DateTimePicker> dateTimePicker1Arr = new List<DateTimePicker>();
      List<DateTimePicker> dateTimePicker2Arr = new List<DateTimePicker>();
      ComboBox CB;
      Label LB;
      DateTimePicker DTP;
      int countEmp = 0;

      //заполнение таблицы "Пользователи"
      void fillUsers()
      {
         usersTable.Rows.Clear();
         string query = "SELECT Users.Code, Users.Login, Users.Password, Employe.FullName, Roles.Name " +
            "FROM Users, Employe, Roles " +
            "WHERE Users.CodeEmploye = Employe.Code and Users.CodeRole = Roles.Code";
         SqlCommand command = new SqlCommand(query, myConnection);
         try
         {
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
               usersTable.Rows.Add(reader[0].ToString(),
                           reader[1].ToString(),
                           reader[2].ToString(),
                           reader[3].ToString(),
                           reader[4].ToString());
            }
            reader.Close();
         }
```

```
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        //перевод в транслит для добавлени логина
        private static Dictionary<string, string> transliter = new Dictionary<string, string>();
        private static void prepareTranslit()
        {
            transliter.Add("а", "a"); transliter.Add("б", "b"); transliter.Add("в", "v");
            transliter.Add("г", "g"); transliter.Add("д", "d"); transliter.Add("е", "e");
            transliter.Add("ё", "yo"); transliter.Add("ж", "zh"); transliter.Add("з", "z");
            transliter.Add("и", "i"); transliter.Add("й", "j"); transliter.Add("к", "k");
            transliter.Add("л", "l"); transliter.Add("м", "m"); transliter.Add("н", "n");
            transliter.Add("о", "o"); transliter.Add("п", "p"); transliter.Add("р", "r");
            transliter.Add("с", "s"); transliter.Add("т", "t"); transliter.Add("у", "u");
            transliter.Add("ф", "f"); transliter.Add("х", "h"); transliter.Add("ц", "c");
            transliter.Add("ч", "ch"); transliter.Add("ш", "sh"); transliter.Add("щ", "sch");
            transliter.Add("ъ", "j"); transliter.Add("ы", "i"); transliter.Add("ь", "j");
            transliter.Add("э", "e"); transliter.Add("ю", "yu"); transliter.Add("я", "ya");
            transliter.Add("А", "A"); transliter.Add("Б", "B"); transliter.Add("В", "V");
            transliter.Add("Г", "G"); transliter.Add("Д", "D"); transliter.Add("Е", "E");
            transliter.Add("Ё", "Yo"); transliter.Add("Ж", "Zh"); transliter.Add("З", "Z");
            transliter.Add("И", "I"); transliter.Add("Й", "J"); transliter.Add("К", "K");
            transliter.Add("Л", "L"); transliter.Add("М", "M"); transliter.Add("Н", "N");
            transliter.Add("О", "O"); transliter.Add("П", "P"); transliter.Add("Р", "R");
            transliter.Add("С", "S"); transliter.Add("Т", "T"); transliter.Add("У", "U");
            transliter.Add("Ф", "F"); transliter.Add("Х", "H"); transliter.Add("Ц", "C");
            transliter.Add("Ч", "Ch"); transliter.Add("Ш", "Sh"); transliter.Add("Щ", "Sch");
            transliter.Add("Ъ", "J"); transliter.Add("Ы", "I"); transliter.Add("Ь", "J");
            transliter.Add("Э", "E"); transliter.Add("Ю", "Yu"); transliter.Add("Я", "Ya");
        }
        public static string GetTranslit(string sourceText)
        {
            StringBuilder ans = new StringBuilder();
            for (int i = 0; i < sourceText.Length; i++) { if (transliter.ContainsKey(sourceText[i].ToString())) {
ans.Append(transliter[sourceText[i].ToString()]); } else { ans.Append(sourceText[i].ToString()); } }
            return ans.ToString();
        }

        //Составление логина
        private static string SurnameName(string name)
        {
            string user = "";
            int i = 0;
            int length = name.Length;
            for (i = 0; i < length; i++)
            {
                if (name[i] != ' ')
                {
                    user += name[i];
                }
                else
                {
                    break;
                }
            }

            i++;
            user += name[i];
            for (int j = i; j < length; j++)
            {
                if (name[j] == ' ') { i = j; break; }
            }

            i++;
            user += name[i];
            return user;
        }
```

```
    private void checkEmp(object sender, EventArgs e)
    {
        for (int i = 0; i < countEmp; i++)
        {
            if ((comboBoxArr[i].Text == "") || (dateTimePicker1Arr[i].Value > dateTimePicker2Arr[i].Value) ||
(dateTimePicker1Arr[i].Value.ToString() == "") || (dateTimePicker2Arr[i].Value.ToString() == ""))
            {
            }
        }
    }

    void addEmp()
    {
        CB = new ComboBox();
        CB.Location = new Point(70, y);
        CB.Font = new Font("Segoe UI", 10);
        CB.Width = 250;
        CB.Parent = this;
        if (countEmp != 1)
        {
            for (int i = 0; i < comboBoxArr[0].Items.Count; i++)
            {
                CB.Items.Add(comboBoxArr[0].Items[i].ToString());
            }
        }
        CB.SelectedIndexChanged += new EventHandler(checkEmp);
        comboBoxArr.Add(CB);

        LB = new Label();
        LB.Location = new Point(325, y);
        LB.Font = new Font("Segoe UI", 10);
        LB.Text = "с";
        LB.Width = 10;
        LB.Parent = this;
        Label1Arr.Add(LB);

        DTP = new DateTimePicker();
        DTP.Format = DateTimePickerFormat.Time;
        DTP.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 09, 00, 00);
        DTP.ShowUpDown = true;
        DTP.Location = new Point(345, y);
        DTP.Font = new Font("Segoe UI", 10);
        DTP.Width = 100;
        DTP.Parent = this;
        DTP.ValueChanged += new EventHandler(checkEmp);
        dateTimePicker1Arr.Add(DTP);

        LB = new Label();
        LB.Location = new Point(450, y);
        LB.Font = new Font("Segoe UI", 10);
        LB.Text = "по";
        LB.Width = 30;
        LB.Parent = this;
        Label2Arr.Add(LB);

        DTP = new DateTimePicker();
        DTP.Format = DateTimePickerFormat.Time;
        DTP.Value = new DateTime(System.DateTime.Today.Year, System.DateTime.Today.Month,
System.DateTime.Today.Day, 20, 00, 00);
        DTP.ShowUpDown = true;
        DTP.Location = new Point(480, y);
        DTP.Font = new Font("Segoe UI", 10);
        DTP.Width = 100;
        DTP.Parent = this;
        DTP.ValueChanged += new EventHandler(checkEmp);
        dateTimePicker2Arr.Add(DTP);
    }
```

```
        private void KeyPressTB(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsDigit(e.KeyChar) && e.KeyChar != 8 && e.KeyChar != Convert.ToChar(","))
                e.Handled = true;
        }
        //выборка кода
        void selectCode(string table, string query, List<int> code, ComboBox comboBox)
        {
            SqlCommand command = new SqlCommand(query, myConnection);
            code.Clear();
            comboBox.Items.Clear();
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    code.Add(int.Parse(reader[0].ToString()));
                    comboBox.Items.Add(reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        //заполнение таблицы "Категория"
        void fillCategory()
        {
            categoryTable.Rows.Clear();
            string query = "SELECT Code, Name FROM Category";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    categoryTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        void fillNomenclature()
        {
            nomenclatureTable.Rows.Clear();
            string query = "SELECT Nomenclature.Code, Nomenclature.Name,  Nomenclature.Balance, Nomenclature.UnitPrice,
Nomenclature.SellingPrice" +
                        " FROM Nomenclature";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    nomenclatureTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString(),
                                reader[2].ToString(),
                                reader[3].ToString(),
                                reader[4].ToString());
                }
                reader.Close();
```

```
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        //заполнение таблицы "Должность"
        void fillPositions()
        {
            positionsTable.Rows.Clear();
            string query = "SELECT Code, Name FROM Positions";
            SqlCommand command = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    positionsTable.Rows.Add(reader[0].ToString(),
                                reader[1].ToString());
                }
                reader.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        //проверка заполнения всех полей при вводе номенклатуры
        void checkNomenclatureInput()
        {
            if (nomenclatureName.Text != "")
            {
                nomenclatureOkButton.Enabled = true;
            }
            else
            {
                nomenclatureOkButton.Enabled = false;
            }
        }

        //проверка заполнения всех полей при вводе сорудника
        void checkEmployeesInput()
        {
            if ((employeesCode.Text != "") && (employeesFullName.Text != "") && (employeesAddress.Text != "") &&
(employeesPhone.Text != "") && (employeesEmail.Text != "") && (employeesPosition.SelectedIndex != -1))
            {
                employeesOkButton.Enabled = true;
            }
            else
            {
                employeesOkButton.Enabled = false;
            }
        }

        //заполенение comboBox при выборе CheckBox
        void checkTtnSearchComboBox(CheckBox checkBoxSearch, ComboBox comboBoxSearch, string selectFields, string table)
        {
            if (checkBoxSearch.Checked)
            {
                comboBoxSearch.Visible = true;
                string query = "SELECT " + selectFields + " FROM " + table;
                SqlCommand command = new SqlCommand(query, myConnection);
                try
                {
                    SqlDataReader reader = command.ExecuteReader();
                    comboBoxSearch.Items.Clear();
```

```
            while (reader.Read())
            {
                comboBoxSearch.Items.Add(reader[0].ToString());
            }
            reader.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка чтения данных из БД", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
    }
    else
    {
        comboBoxSearch.SelectedItem = -1;
        comboBoxSearch.Items.Clear();
        comboBoxSearch.Visible = false;
    }
}

//удаление записи из таблицы
void deleteRecords(string code, string table)
{
    string query = "DELETE FROM " + table +
                " Where Code = " + code;
    SqlCommand command = new SqlCommand(query, myConnection);
    try
    {
        command.ExecuteNonQuery();
    }
    catch
    {
        MessageBox.Show("Ошибка удаление записи. Запись не удалена", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        return;
    }
}

//очистка панели добавления/реадктирвоания сотрудника
void clearEmployeesAddEdit()
{
    emploeesEditAddPanel.Visible = false;
    employeesCode.Text = "";
    employeesFullName.Text = "";
    employeesDOB.Value = System.DateTime.Today;
    employeesAddress.Text = "";
    employeesPhone.Text = "";
    employeesEmail.Text = "";
    employeesPosition.SelectedIndex = -1;
    employeesPosition.Items.Clear();
    employeesRole.SelectedIndex = -1;
    employeesRole.Items.Clear();
}

//выбор товара при вводе ТТН
private void comboBoxNameChange(object sender, EventArgs e)
{
    int count = 0;
    for (int i = 0; i < countNomenclature; i++)
        if (comboBoxName[i] == sender)
        {
            count = i;
            break;
        }
    if (comboBoxName[count].SelectedIndex != -1)
    {
        string query = "SELECT Nomenclature.Name, Nomenclature.UnitPrice " +
            "FROM Nomenclature " +
            "WHERE Nomenclature.Code='" + codeNom[comboBoxName[count].SelectedIndex].ToString() + "'";
        SqlCommand command1 = new SqlCommand(query, myConnection);
```

```
            try
            {
                SqlDataReader reader1 = command1.ExecuteReader();
                reader1.Read();
                textBoxUnitPrice[count].Text = reader1[1].ToString();
                reader1.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
    }

    //выбор товара при создании чека
    private void comboBoxNameChangeCheck(object sender, EventArgs e)
    {
        int count = 0;
        for (int i = 0; i < countNomenclatureCheck; i++)
            if (comboBoxNameCheck[i] == sender)
            {
                count = i;
                break;
            }
        if (comboBoxNameCheck[count].SelectedIndex != -1)
        {
            string query = "SELECT Nomenclature.SellingPrice " +
                "FROM Nomenclature " +
                "WHERE Nomenclature.Code=" + codeNom[comboBoxNameCheck[count].SelectedIndex].ToString();
            SqlCommand command1 = new SqlCommand(query, myConnection);
            try
            {
                SqlDataReader reader1 = command1.ExecuteReader();
                reader1.Read();
                textBoxUnitPriceCheck[count].Text = reader1[0].ToString();
                reader1.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка чтения данных из БД. Обратитесь к администратору", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            ChangeCheck(count);
        }
    }


    //расчет итоговой стоимости по товару в ТТН
    private void Change(object sender, EventArgs e)
    {
        int count = 0;
        for (int i = 0; i < countNomenclature; i++)
            if ((textBoxScopeOfDelivery[i] == sender) || (comboBoxName[i] == sender) || (textBoxUnitPrice[i] == sender))
            {
                count = i;
                break;
            }
        if ((textBoxUnitPrice[count].Text != "") && (textBoxScopeOfDelivery[count].Text != ""))
        {
            double d = double.Parse(textBoxScopeOfDelivery[count].Text);
            double p = Math.Round(double.Parse(textBoxUnitPrice[count].Text) * d, 2);

            textBoxTotalCost[count].Text = Math.Round((p), 2).ToString();
        }
    }

    //расчет итоговой стоимости по товару в чеке
```

```
        private void ChangeCheck(int count)
        {
            if ((textBoxUnitPriceCheck[count].Text != "") && (numericUpDownCountCheck[count].Text != ""))
            {
                double d = double.Parse(numericUpDownCountCheck[count].Text);
                double p = Math.Round(double.Parse(textBoxUnitPriceCheck[count].Text) * d, 2);
                textBoxTotalCostCheck[count].Text = p.ToString();
            }
            double sum = 0;
            for (int i = 0; i < countNomenclatureCheck; i++)
                if (textBoxTotalCostCheck[i].Text != "") sum += double.Parse(textBoxTotalCostCheck[i].Text);
            textBox1.Text = sum.ToString();
        }

        //изменение отображения вкладок на tabControl
        private static void DrawItem(object sender, DrawItemEventArgs e)
        {
            Graphics g = e.Graphics;
            Brush _textBrush;
            TabPage _tabPage = ((TabControl)sender).TabPages[e.Index];
            System.Drawing.Rectangle _tabBounds = ((TabControl)sender).GetTabRect(e.Index);
            if (e.State == DrawItemState.Selected)
            {
                _textBrush = new SolidBrush(Color.White);
                g.FillRectangle(Brushes.Gray, e.Bounds);
            }
            else
            {
                _textBrush = new System.Drawing.SolidBrush(e.ForeColor);
                e.DrawBackground();
            }
            System.Drawing.Font _tabFont = new System.Drawing.Font("Segoe UI", (float)10.0, FontStyle.Regular,
GraphicsUnit.Point);
            StringFormat _stringFlags = new StringFormat();
            _stringFlags.Alignment = StringAlignment.Center;
            _stringFlags.LineAlignment = StringAlignment.Center;
            g.DrawString(_tabPage.Text, _tabFont, _textBrush, _tabBounds, new StringFormat(_stringFlags));
        }

        private void numericUpDownCountCheckChangeCheck(object sender, EventArgs e)
        {
            int count = 0;
            for (int i = 0; i < countNomenclatureCheck; i++)
                if (numericUpDownCountCheck[i] == sender)
                {
                    count = i;
                    break;
                }
            ChangeCheck(count);
        }

        //Добавление полей для ввода товара при вводе ТТН
        private void addNomenclatureTTN(int y)
        {
            ComboBox CB = new ComboBox();
            CB.Location = new Point(55, y);
            CB.Font = new Font("Segoe UI", 10);
            CB.Width = 350;
            CB.SelectedIndexChanged += new System.EventHandler(comboBoxNameChange);
            CB.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
            CB.AutoCompleteSource = AutoCompleteSource.ListItems;
            CB.Parent = ttnAddEditPanel;

            if (comboBoxName.Count != 0)
            {
                int temp = comboBoxName[0].Items.Count;
                CB.Items.Clear();
                for (int i = 0; i < temp; i++)
                    CB.Items.Add(comboBoxName[0].Items[i].ToString());
            }
```