**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»  
ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите** Заместитель директора  по УМР \_\_\_\_\_\_\_\_\_\_\_\_ Е.Г. Конакина\_  (Подпись) (ФИО)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  |

**курсовой проект**Тема: «Приложение для работы с базой данных «Магазин Теплоснобжения»

специальность 09.02.07 группа 32919/2

Студент (ка) Зайцева Д.А.  
 (подпись) (ФИО)  
Преподаватель Девятко Н.С.

(подпись) (ФИО)

Санкт-Петербург  
2024

Содержание

[ВВЕДЕНИЕ 2](#_Toc167696370)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ 4](#_Toc167696371)

[1.1. Описание предметной области 4](#_Toc167696372)

[1.2. Анализ методов решения 5](#_Toc167696373)

[1.3. Обзор средств программирования 6](#_Toc167696374)

[1.4. Описание языка C# 8](#_Toc167696375)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc167696376)

[2.1. Постановка задачи 11](#_Toc167696377)

[2.1.1. Основания для разработки 11](#_Toc167696378)

[2.1.2. Назначение программы 11](#_Toc167696379)

[2.2. Проектирование приложения 11](#_Toc167696380)

[2.2.1. Диаграмма прецедентов 12](#_Toc167696381)

[2.2.2. Диаграмма последовательностей 13](#_Toc167696382)

[2.3. Текст программы 15](#_Toc167696383)

[2.4. Описание программы 15](#_Toc167696384)

[2.4.1. Общие сведения 15](#_Toc167696385)

[2.4.2. Функциональное назначение 15](#_Toc167696386)

[2.4.3. Описание логической структуры системы 16](#_Toc167696387)

[2.4.4. Используемые технические и программные средства 16](#_Toc167696388)

[2.4.5. Вызов и загрузка 17](#_Toc167696389)

[2.5. Программа и методика испытаний 17](#_Toc167696390)

[2.5.1. Объект испытаний 17](#_Toc167696391)

[2.5.2. Цель испытаний 17](#_Toc167696392)

[2.5.3. Требования к программе 17](#_Toc167696393)

[2.5.4. Требования к программной документации 19](#_Toc167696394)

[2.5.5. Средства и порядок испытаний 19](#_Toc167696395)

[2.5.6. Методы испытаний 20](#_Toc167696396)

[2.7. Протокол испытаний 23](#_Toc167696397)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc167696398)

[ПРИЛОЖЕНИЕ А 2](#_Toc167696399)

[ПРИЛОЖЕНИЕ Б 10](#_Toc167696400)

[ПРИЛОЖЕНИЕ В 19](#_Toc167696401)

# ВВЕДЕНИЕ

В настоящее время широкой огласке поддается перевод всех сфер жизнедеятельности в цифровой формат. Таким образом можно наблюдать, как компьютеры широко применяются в таких областях как: медицина, промышленность, образование, кораблестроение и др. Применение компьютеров и цифровых технологий в промышленности значительно облегчает труд рабочих и других лиц, привлеченных к этой сфере, так как позволяет быстро доносить информацию между сотрудниками, делегировать обязанности, проверять ход работы и затрат, отслеживать структуру и цели проектов.

Большое количество магазинов так же предпочитает цифровой формат ведения бухгалтерии, что позволяет ускорить процесс обслуживания клиентов, составления отчетов, исключить вероятность появления ошибки в расчетах продаж. Так же такой формат ведения документации позволяет облегчить порядок отслеживания работы сотрудников, их должность. Помимо этого, такой формат позволяет обеспечить ограничение доступа для различных должностей.

Разработанная в данном курсовом проекте программа может быть использована сотрудниками магазина теплоснабжения, для удобного хранения, редактирования и поиска информации, а также разграничения доступа к данным разных видов пользователей: секретарь, администратор, продавец и директор предприятия.

# 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ

## 1.1. Описание предметной области

Использование компьютерных средств предоставляет многочисленные преимущества во многих сферах деятельности:

1. **Автоматизация задач:** Компьютеры могут выполнять повторяющиеся или сложные задачи с высокой точностью и скоростью, освобождая время и ресурсы для более важной работы;
2. **Улучшение эффективности:** Компьютерные системы могут оптимизировать рабочие процессы, сокращать время выполнения задач и повышать производительность;
3. **Хранение и обработка данных:** Компьютеры предоставляют надежные и доступные решения для хранения и обработки огромных объемов данных;
4. **Анализ и визуализация данных:** Компьютерные инструменты позволяют анализировать данные, идентифицировать тенденции и создавать визуальные представления сложных данных для облегчения принятия решений;
5. **Обмен информацией и общение:** Компьютерные сети упрощают обмен информацией и совместную работу между коллегами, клиентами и другими заинтересованными сторонами;
6. **Повышение качества:** Компьютерные системы могут применять правила и стандарты для обеспечения точности и согласованности в процессах, что приводит к повышению качества результатов;
7. **Инновации и исследования**: Компьютеры используются для разработки новых продуктов, моделирования и тестирования концепций, а также проведения исследований и экспериментов;
8. **Улучшение обслуживания клиентов:** Компьютерные технологии, такие как чат-боты и порталы самообслуживания, позволяют компаниям предоставлять более эффективную и удобную поддержку клиентам;
9. **Персонализация:** Компьютерные системы могут отслеживать поведение пользователей и предоставлять персонализированный опыт, соответствующий их индивидуальным потребностям и предпочтениям;
10. **Доступность и мобильность:** Компьютеры и мобильные устройства предоставляют круглосуточный доступ к информации и инструментам, обеспечивая удобство и гибкость.

Включение компьютерных средств в различные сферы деятельности может значительно повысить эффективность, производительность и качество, приводя к улучшению результатов бизнеса и повышению удовлетворенности клиентов.

Приложение для работы с базой данных представляет собой программное обеспечение, предназначенное для управления и обработки данных в магазине по продаже оборудования и материалов для систем теплоснабжения.

Основные задачи приложения:

1. Хранение и управление сведениями о товарах, включая наименование, описание, характеристики и наличие на складе;
2. Ведение учета клиентов, их заказов и истории покупок;
3. Обеспечение контроля над складскими запасами, включая отслеживание прихода и расхода товаров, расчет остатков;
4. Генерация различных отчетов, таких как отчеты о продажах, наличию товаров и заказах клиентов;
5. Проведение анализа данных для выявления тенденций и принятия обоснованных решений в области управления бизнесом.

Разделы приложения:

1. Управление товарами: включает каталогизацию товаров, управление ценами и скидками, а также настройку параметров фильтрации и сортировки;
2. Управление клиентами: хранит данные о контактной информации, истории заказов и предпочтениях клиентов;
3. Управление заказами: позволяет обрабатывать заказы клиентов, отслеживать статус их выполнения и генерировать подтверждения заказов;
4. Управление складом: обеспечивает отслеживание остатков товаров на складе, а также выдачу и приемку товаров;
5. Отчеты: предоставляет широкий спектр отчетов, таких как отчеты о продажах, наличии товаров, заказах клиентов и т.д;
6. Анализ данных: включает инструменты для анализа данных о продажах, складских запасах и предпочтениях клиентов для выявления тенденций и улучшения бизнес-процессов.

Преимущества использования приложения:

1. Упрощение управления ассортиментом и данными о товарах;
2. Улучшение обслуживания клиентов за счет хранения истории заказов и предпочтений;
3. Оптимизация складских запасов и снижение издержек;
4. Повышение эффективности бизнес-процессов за счет автоматизации ручных операций;
5. Возможность принятия обоснованных решений на основе анализа данных.

## 1.2. Анализ методов решения

Для приложения "Магазин Теплоснабжения" подойдет автоматизированный метод решения, который не требует ручного ввода данных с внешнего устройства. Этот метод:

1. Упрощает управление данными и снижает вероятность ошибок, связанных с ручным вводом;
2. Позволяет быстро обрабатывать большие объемы данных, что особенно важно для магазинов с широким ассортиментом товаров и большим количеством клиентов;
3. Предоставляет возможность автоматического расчета результатов и генерации отчетов, экономя время и повышая эффективность.

Математические вычислительные методы, используемые в приложении:

1. Расчет остатков на складе: используется простая математическая операция вычитания, чтобы рассчитать остаток на складе путем вычитания количества выданных товаров из начального количества;
2. Генерация отчетов о продажах: для расчета общей суммы продаж и других показателей в отчетах о продажах используются простые арифметические операции, такие как сложение, вычитание и умножение;
3. Анализ данных: в модуле анализа данных используются статистические методы, такие как среднее значение, медиана и стандартное отклонение, для выявления тенденций и закономерностей в данных о продажах, складских запасах и предпочтениях клиентов.

Точность вычислений:

Математические вычисления, используемые в приложении, имеют высокую точность, поскольку они основаны на простых арифметических операциях и проверенных статистических методах. Для обеспечения дополнительной точности приложение использует программное обеспечение с плавающей запятой, которое позволяет обрабатывать большие числа с высокой степенью точности.

Выбор метода решения:

Автоматизированный метод решения был выбран, потому что он:

1. Соответствует требованиям приложения и обеспечивает необходимую функциональность;
2. Эффективен и позволяет быстро обрабатывать большие объемы данных;
3. Удобен для пользователей и не требует специальных знаний или навыков;
4. Имеет высокую точность и обеспечивает достоверные результаты.

## 1.3. Обзор средств программирования

Для разработки приложения "Магазин Теплоснабжения" с базой данных было необходимо выбрать подходящие средства программирования. Были проведены сравнения различных языков и сред разработки по следующим критериям:

1. Функциональность;
2. Удобство использования;
3. Простота конструирования пользовательского интерфейса;
4. Требуемые дополнительные библиотеки и компоненты.

После анализа было принято решение использовать следующие средства:

1. Язык программирования: C#;
2. Среда разработки: Visual Studio 2017;
3. База данных: Microsoft SQL Server.

Выбор языка программирования (C#)

C# был выбран благодаря его обширной функциональности, которая включает поддержку объектно-ориентированного программирования, широкий спектр типов данных и управляющие структуры. Он также хорошо интегрируется с другими технологиями Microsoft, такими как .NET Framework и Windows Forms.

Выбор среды разработки (Visual Studio 2017)

Visual Studio 2017 была выбрана как среда разработки для проекта по следующим причинам:

1. Поддержка множества языков программирования, включая C#;
2. Удобный и интуитивно понятный интерфейс;
3. Набор встроенных инструментов для разработки и отладки;
4. Интеграция с Microsoft SQL Server для упрощения работы с базами данных.

Выбор базы данных (Microsoft SQL Server)

Microsoft SQL Server была выбрана в качестве системы управления базами данных (СУБД) по следующим причинам:

1. Высокая производительность и надежность;
2. Поддержка сложных запросов и операций с данными;
3. Широкий набор функций для управления и администрирования баз данных;
4. Интеграция с C# и Visual Studio 2017 обеспечивает плавную связь между приложением и базой данных.

Требуемые дополнительные библиотеки и компоненты

Для расширения функциональности приложения были задействованы следующие дополнительные библиотеки и компоненты:

1. Entity Framework Core: Для упрощения работы с данными и создания запросов к базе данных;
2. Fluent API: Для настройки моделей данных Entity Framework Core;
3. Windows Forms: Для создания пользовательского интерфейса с элементами управления, такими как кнопки, текстовые поля и списки.

Язык программирования C#, среда разработки Visual Studio 2017 и база данных Microsoft SQL Server были выбраны в качестве оптимальных средств программирования для создания приложения "Магазин Теплоснабжения". Эта комбинация предоставляет мощную и удобную платформу для разработки и развертывания приложения.

## 1.4. Описание языка C#

C# — современный, объектно-ориентированный язык программирования, предназначенный для создания различных типов приложений. Он отличается простотой изучения и использования, а также широкими возможностями и поддержкой со стороны Microsoft. C# используется в следующих сферах:

* 1. Разработка программного обеспечения
  2. Веб-разработка
  3. Разработка приложений для мобильных устройств
  4. Разработка игр
  5. Облачные вычисления

C# имеет ряд преимуществ, в том числе:

* 1. Удобный и гибкий синтаксис
  2. Эффективная система управления памятью
  3. Поддержка различных парадигм программирования
  4. Широкая экосистема библиотек и инструментов
  5. Активное сообщество разработчиков

**Ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | extern | null | struct |
| as | false | object | switch |
| base | finally | operator | this |
| bool | fixed | out | throw |
| break | float | override | true |
| byte | for | params | try |
| case | foreach | private | typeof |
| catch | goto | protected | uint |
| const | if | public | ulong |
| continue | implicit | readonly | unchecked |
| decimal | in | ref | unsafe |
| default | int | return | ushort |
| delegate | interface | sbyte | using |
| do | internal | sealed | virtual |
| double | is | short | void |
| else | lock | sizeof | while |
| enum | long | stackalloc |  |
| event | namespace | static |  |
| explicit | new | string |  |

**Контекстные ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| add | equals | join | set |
| ascending | from | let | value |
| async | get | on | var |
| await | global | orderby | where |
| by | group | partial | yield |
| descending | in | remove |  |
| dynamic | into | select |  |

**Основные алгоритмические конструкции C#:**

|  |  |
| --- | --- |
| Оператор присваивания | string hello = "hello " + "world"; //результат равен "hello world"  int x1 = 2 + 4; // результат равен 6  int x2 = 10 - 6; //результат равен 4  int x3 = 10 \* 6; //результат равен 60  double x4 = 10.0 / 4.0; //результат равен 2.5  double x5 = 10.0 % 4.0; //результат равен 2  int y1 = 5;  int z1 = ++y1; // z1=6; y1=6  int y2 = 5;  int z2 = y2++; // z2=5; y2=6  int y3 = 5;  int z3 = --y3; // z3=4; y3=4  int y4 = 5;  int z4 = y4--; // z4=5; y4=4 |
| Условный оператор | *if (условие) {(действие) } else {(альтернатива)} ;*  int num1 = 8;  int num2 = 6;  if(num1 > num2)  {      Console.WriteLine("Число {0} больше числа {1}", num1, num2);  }  else  {      Console.WriteLine("Число {0} меньше числа {1}", num1, num2);  } |
| Оператор множественного ветвления | *Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:*  Console.WriteLine("Нажмите Y или N");  string selection = Console.ReadLine();  switch (selection)  {      case "Y":          Console.WriteLine("Вы нажали букву Y");          break;      case "N":          Console.WriteLine("Вы нажали букву N");          break;      default:          Console.WriteLine("Вы нажали неизвестную букву");          break;  }  *После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора сase. И если совпадение будет найдено, то будет выполняться определенный блок сase.*  *В конце блока сase ставится оператор break, чтобы избежать выполнения других блоков.*  *Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default,.* |
| Арифметический цикл  **(**применяется, когда известно количество повторений цикла) | for *([инициализация счетчика]; [условие]; [изменение счетчика])*  for (int i = 0; i < 10; i++)  {      Console.WriteLine("Квадрат числа {0} равен {1}", i, i \* i);  } |
| Цикл с предусловием  **(**применяется, когда неизвестно количество повторений цикла) | *while (условие) { тело цикла };*  Этот цикл будет выполняться до тех пор, пока истинно *условие* (логическое выражение, возвращающее значение типа **Boolean**). При этом если это выражение сразу равно **false**, *тело цикла* не будет выполнено ни разу. Нужно очень внимательно следить за написанием *условия* и контролем завершения цикла, так как в результате ошибки цикл **while** будет повторяться бесконечное количество раз, что приведёт к "зацикливанию" и "зависанию" программы. |
| Цикл с постусловием  **(**применяется, когда неизвестно количество повторений цикла) | *do { тело цикла } while (условие);*  Повторения сначала выполняет *тело цикла*, а затем уже проверяет выполнение *условия*. Таким образом, этот вариант цикла гарантирует, что *тело цикла* будет выполнено по крайней мере один раз. И будет выполняться до тех пор, пока *условие* не станет истинным (**true**). |

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Постановка задачи

### 2.1.1. Основания для разработки

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и утверждена Институтом среднего профессионального образования.

### 2.1.2. Назначение программы

Основное назначение программного продукта: информационная система для средних учебных заведений.

Эксплуатационное назначение программного продукта: приложение предназначено для работников средних специальных учебных заведений, предназначена удобного хранения и редактирования данных, формирования отчетности.

## 2.2. Проектирование приложения

На этапе проектирования приложения важную роль играет разработка UML-диаграмм для определения логического построения функционирования программы, а также необходимых связей базы данных и её компонентов. Для определения всех необходимых пунктов было разработано 4 UML-диаграммы:

### 2.2.1. Диаграмма прецедентов

Диаграммы прецедентов частично описывает use case – прецедент использования проектируемой системы, давая частичное описание частичного применения системы с точки зрения условного внешнего обозревателя. При этом описание фокусируется на том, что должна делать система по отношению к своему внешнему окружению (периферии), а не то на том, как она эта делает, то есть диаграмма прецедентов есть частичная спецификация (рисунок 1).



Рисунок 1 Диаграмма прецедентов

### 2.2.2. Диаграмма последовательностей

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.



Рисунок 2 Диаграмма последовательностей

## 2.3. Текст программы

Текст программы в соответствии с ГОСТ 19.101-77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении В.

## 2.4. Описание программы

### 2.4.1. Общие сведения

Полное наименование программной разработки: «ПРИЛОЖЕНИЕ ДЛЯ РАБОТЫ С БД «МАГАЗИН ТЕПЛОСНОБЖЕНИЯ»».

Приложение для работы с БД «Магазин теплоснабжения»: секретарь ООО использует приложение для внесения данных о сотрудниках предприятия, поставляемых товарах и услугах; приложение позволяет добавлять, редактировать, удалять, просматривать, искать записи о сотрудниках, товарах, услугах, формировать отчеты о продажах за месяца, кварталы и года; продавец имеет ограниченные права и может только просматривать и искать информацию, а также выводить сводные отчеты; администратор системы работает с учетными записями пользователей, может менять статус сотрудников, добавлять новых, удалять старых; директор ООО имеет полный доступ к таблицам товаров и услуг, к ценовой части БД, с возможностью корректировки и отслеживания.

### 2.4.2. Функциональное назначение

Основное назначение программного продукта: компьютерная игра в жанре «симулятор игрового автомата».

Эксплуатационное назначение программного продукта: приложение предназначено для работников «Магазина Теплоснабжения», предназначена для удобного хранения и редактирования данных, формирования отчетности.

### 2.4.3. Описание логической структуры системы

Программа содержит несколько основных модулей:

1) Для авторизации:

Авторизация.cs – форма предназначена для регулирования входящих должностей, дальнейшего их направления на нужную форму по их должности.

2) Для работы админа:

Админ.cs – содержит список пользователей, которые используют приложение, данные которых вносятся админом. Форма так же содержит методы для добавления, удаления, изменения и поиска сотрудников.

3) Для работы пользователя:

Главная.cs – содержит список товаров, услуг, продаж, поставок и методы для записи, отображения, редактирования товаров, услуг, продаж, поставок, а также выборку исходя из выбранных пользователем данных.

Информация\_о\_пользователе.cs – содержит краткие сведения о пользователе, который авторизовался.

Отчет.cs – форма предназначена для отображения созданного в «ОтчетПомощь» отчета.

ОтчетПомощь.cs – форма предназначена для выбора периода для формирования отчета.

### 2.4.4. Используемые технические и программные средства

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~500 Кб;
* объем необходимой памяти на жестком диске ~3Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь»;

### 2.4.5. Вызов и загрузка

Программа может быть загружена как с диска, так и с жесткого диска. В последнем случае требуется предварительно переписать программу с диска на жесткий диск.

Исполняемым файлом программы является файл Курсовая\_Зайцева.exe. Для его запуска необходимо дважды щелкнуть по исполняемому файлу левой кнопкой мышки.

## 2.5. Программа и методика испытаний

### 2.5.1. Объект испытаний

Объектом испытаний является программа .exe. Администратор добавляет пользователей ролей «Секретарь» и «Продавец» для дальнейшей работы с данным сотрудников: создания их контингента, должности, личных данных и даты приема на работу.

### 2.5.2. Цель испытаний

Целью испытаний является проверка соответствия программы требованиям Технического Задания.

### 2.5.3. Требования к программе

В процессе испытаний подлежат проверке следующие требования к программе:

#### 2.5.3.1. Требования к функциональным характеристикам

**1 версия:**

- программа должна обеспечивать просмотр, добавление, удаление, редактирование данных о сотрудниках, услугах, товарах, продажах;

- программа должна отображать специальную форму для удобного добавления данных о сотрудниках, услугах, товарах, продажах;

- программа должна автоматически подгружать данные редактируемой записи в специальную форму для их удобной корректировки;

- программа должна обеспечивать поиск информации по основным критериям в каждой таблице;

- программа должна иметь удобный пользовательский интерфейс.

**2 версия:**

- программа должна иметь возможность авторизации пользователя;

- после авторизации программа должна отображать в верхней части окна данные о роли пользователя в системе и его ФИО;

- программа должна предоставлять возможность смены пользователя без перезапуска приложения;

- программа должна обеспечивать разграничение прав доступа для администратора, секретаря, продавца и директора предприятия, исходя из введённого логина: секретарь ООО использует приложение для внесения данных о сотрудниках предприятия, поставляемых товарах и услугах; приложение позволяет добавлять, редактировать, удалять, просматривать, искать записи о сотрудниках, товарах, услугах, формировать отчеты о продажах за месяца, кварталы и года; продавец имеет ограниченные права и может только просматривать и искать информацию, а также выводить сводные отчеты; администратор системы работает с учетными записями пользователей, может менять статус сотрудников, добавлять новых, удалять старых; директор ООО имеет полный доступ к таблицам товаров и услуг, к ценовой части БД, с возможностью корректировки и отслеживания;

- программа должна иметь возможность импортировать данные о поставляемых товарах и оказываемых услугах из внешних источников;

- программа должна уметь автоматически создавать отчеты по требованию пользователя (роль секретарь), содержащий сводную информацию по продажам/услугам/сотрудникам на заданный период, и экспортировать отчеты в формате word/excel/pdf с возможностью предварительного просмотра и вывода на печать.

#### 2.5.3.2. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 10 или совместимой. Язык интерфейса – русский.

#### 2.5.3.3. Требования к маркировке и упаковке

Программа должна поставляться в виде проекта, исполняемого (еxе) файла, установщика и документации.

#### 2.5.3.4. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

### 2.5.4. Требования к программной документации

На Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

### 2.5.5. Средства и порядок испытаний

Для проведения испытаний необходимы следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~20Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- стандартный офисный принтер;

- манипулятор «мышь».

Для проведения испытаний необходимы следующие технические средства: операционная система Windows 10 или выше.

Испытания проводятся в следующем порядке:

1) проверяется наличие и комплектность программной документации (п.2.6.4);

2) проверяется соответствие требованиям к маркировке и упаковке (п.2.6.3.3);

3) проверяется соответствие требованиям к функциональным характеристикам (п.2.6.3.1);

4) проверяется соответствие требованиям к информационной и программной совместимости (п.2.6.3.2).

### 2.5.6. Методы испытаний

#### 2.5.6.1. Для проверки способности программы осуществлять авторизацию пользователя по введённому логину и паролю, необходимо:

* запустить программу;
* ввести действующий логин и пароль в поля авторизации;
* нажать кнопку «Войти»;
* убедиться, появилось сообщение об успешной авторизации, осуществлён переход на следующую страницу, исходя из роли пользователя (авторизация произошла).

#### 2.5.6.2. Для проверки способности программы обеспечивать добавление новых пользователей (регистрацию), необходимо:

* запустить программу;
* ввести логин и пароль в поля авторизации, соответствующие роли админа;
* нажать кнопку «Войти»;
* убедиться, что осуществлён переход на страницу авторизации пользователей;
* нажать кнопку «Добавить»;
* выбрать роль нового пользователя, ввести новый логин и пароль;
* нажать кнопку «Добавить»;
* убедиться, что добавленный пользователь отображается в главной панели.

#### 2.5.6.3. Для проверки способности программы контролировать правильность введённого логина и пароля в окне авторизации, необходимо:

* запустить программу;
* ввести неверный логин и пароль в поля авторизации;
* нажать кнопку «Войти»;
* убедиться, что на экран вывелось сообщение об ошибке.

#### 2.5.6.4. Для проверки способности программы обеспечивать добавление новых товаров, необходимо:

* запустить программу;
* ввести логин и пароль в поля авторизации, соответствующие роли секретаря;
* нажать кнопку «Войти»;
* убедиться, что осуществлён переход на страницу главного меню
* нажать меню «Таблицы», выбрать «Товары»;
* убедиться, что осуществлён вывод таблицы Товары;
* нажать меню «Действия с таблицей», выбрать «Добавить»;
* заполнить данные товара;
* убедиться, что добавленный товар отображается в главной панели.

#### 2.5.6.5. Для проверки способности программы обеспечивать изменение товаров, необходимо:

* запустить программу;
* ввести логин и пароль в поля авторизации, соответствующие роли секретаря;
* нажать кнопку «Войти»;
* убедиться, что осуществлён переход на страницу главного меню
* нажать меню «Таблицы», выбрать «Товары»;
* убедиться, что осуществлён вывод таблицы Товары;
* нажать меню «Действия с таблицей», выбрать «Изменить»;
* выбрать товар для изменения, нажав на соответствующее ему поле таблицы;
* изменить данные о товаре;
* убедиться, что измененный товар отображается в главной панели.

#### 2.5.6.6. Для проверки способности программы обеспечивать удаление товаров, необходимо:

* запустить программу;
* ввести логин и пароль в поля авторизации, соответствующие роли секретаря;
* нажать кнопку «Войти»;
* убедиться, что осуществлён переход на страницу главного меню
* нажать меню «Таблицы», выбрать «Товары»;
* убедиться, что осуществлён вывод таблицы Товары;
* выделить товар для удаления
* нажать меню «Действия с таблицей», выбрать «Удалить»;
* убедиться, что удаленный товар не отображается в главной панели.

#### 2.5.6.7. Для проверки способности программы отображать данные о пользователе, необходимо:

* запустить программу;
* ввести логин и пароль в поля авторизации, соответствующие роли секретаря;
* нажать кнопку «Войти»;
* убедиться, что осуществлён переход на страницу главного меню
* нажать меню «Вход», выбрать «Информация о пользователе»;
* убедиться, что выведенная информация корректна.

#### 2.5.6.8. Для проверки способности программы формировать отчет о продажах, необходимо:

* запустить программу;
* ввести логин и пароль в поля авторизации, соответствующие роли секретаря;
* нажать кнопку «Войти»;
* убедиться, что осуществлён переход на страницу главного меню
* нажать меню «Таблицы», выбрать «Отчет»;
* ввести промежуток дат для формирования отчета;
* убедиться, что отобразилась форма отчета с корректными данными.

## 2.7. Протокол испытаний

Результаты испытаний программы представлены в таблице 1, рисунки приведены в приложении Б.

**Результаты испытаний программы**

*Таблица 1*

| **№** | **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результат** |
| --- | --- | --- | --- | --- |
| 1 | Способность программы осуществлять авторизацию пользователя по введённому логину и паролю | * запустить программу; * ввести действующий логин и пароль в поля авторизации; * нажать кнопку «Войти». | Появилось сообщение об успешной авторизации, осуществлён переход на следующую страницу, исходя из роли пользователя (авторизация произошла). | Пройден  Рисунок Б1 |
| 2 | Способность программы обеспечивать добавление новых пользователей (регистрацию) | запустить программу;  ввести логин и пароль в поля авторизации, соответствующие роли администратора;  нажать кнопку «Войти»;  убедиться, что осуществлён переход на страницу авторизации пользователей;  нажать меню «Действия с таблицей», «Добавить»;  ввести данные нового сотрудника;  нажать кнопку «Добавить»; | Убедиться, что добавленный пользователь отображается в таблице.  (авторизация произошла) | Пройден  Рисунок Б2-Б3 |
| 3 | Способность программы контролировать правильность введённого логина и пароля в окне авторизации | * запустить программу; * ввести неверный логин и пароль в поля авторизации; * нажать кнопку «Войти»; | * Убедиться, что на экран вывелось сообщение об ошибке. | Пройден  Рисунок Б4 |
| 4 | Способность программы обеспечивать добавление новых товаров | * запустить программу; * ввести логин и пароль в поля авторизации, соответствующие роли секретаря; * нажать кнопку «Войти»; * убедиться, что осуществлён переход на страницу главного меню. Нажать в меню «Таблицы», нажать кнопку «Товары», нажать в меню «Действия с таблицей», нажать кнопку «Добавить»; * заполнить данные товара; | * Убедиться, что осуществлён вывод таблицы «Товары»; убедиться, что произвелся вывод панели «Добавить Товар»; убедиться, что новый товар добавлен в таблицу «Товары» | Пройден  Рисунок Б5 – Б6 |
| 5 | Способность программы обеспечивать отображение пароля при авторизации | * запустить программу; * ввести логин и пароль в поля авторизации; * нажать кнопку «Показать пароль» | * Убедиться, что введённый в соответствующее поле пароль отображается символами, а не звёздочками | Пройден  Рисунок Б7 |
| 6 | Способность программы очищать поля ввода при авторизации | * запустить программу; * ввести логин и пароль в поля авторизации; * нажать кнопку «Очистить» | * Убедиться, что поля для введения данных после нажатия стали пустыми | Пройден  Рисунок Б8-Б9 |
| 7 | Способность программы обеспечивать поиск | * запустить программу; * ввести действующий логин и пароль в поля авторизации; * нажать кнопку «Войти»; * убедиться, что осуществлён переход на страницу главного меню; * Нажать в меню «Таблицы», нажать кнопку «Товары»,»; * убедиться, что осуществлён вывод таблицы «Товары»; * ввести в строку поиска значение «Лён», нажать на кнопку «Найти»; | * Убедиться, что в главной панели отображается товар, в названии которого содержится введённое в строке поиска значение. | Пройден  Рисунок Б10 |
| 8 | Способность программы экспортировать созданные отчёты в формате Excel | * запустить программу; * ввести действующий логин и пароль в поля авторизации; * нажать кнопку «Войти»; * убедиться, что осуществлён переход на страницу главного меню; * Нажать в меню «Таблицы», нажать кнопку «Отчет»; убедиться, что осуществлён переход на вкладку с формирование отчета; * выбрать даты «12.12.2000» и «12.12.2025», нажать на кнопку * «Сформировать отчёт»; * убедиться, что отчёт сформирован исходя из выбранных параметров; * нажать кнопку «Сохранить» для экспорта, выбрать формат для экспорта «Excel»; * убедиться, что выбранные для отчёта данные перенесены в файл разрешения .xlsx. | * Убедиться, что отчёт сформирован исходя из выбранных параметров; * Убедиться, что выбранные для отчёта данные перенесены в файл разрешения .xlsx. | Пройден  Пройден  Рисунок Б11-Б12 |
| 9 | Способность программы редактировать существующие записи | * запустить программу; * ввести логин и пароль в поля авторизации, соответствующие роли секретаря; * нажать кнопку «Войти»; * убедиться, что осуществлён переход на страницу главного меню. Нажать в меню «Таблицы», нажать кнопку «Товары», нажать в меню «Действия с таблицей», нажать кнопку «Изменить», выбрать товар для изменения, * изменить данные товара, нажать на кнопку «Изменить»; | * Убедиться, что при нажатии на кнопку появляется панель для изменения; * убедиться, что изменённые данные отображаются в существующей записи товаров. | Пройден  Рисунок Б13-Б15 |
| 10 | Способность программы удалять существующие записи | * запустить программу; * ввести логин и пароль в поля авторизации, соответствующие роли секретаря; * нажать кнопку «Войти»; * убедиться, что осуществлён переход на страницу главного меню. Нажать в меню «Таблицы», нажать кнопку «Товары», выбрать строку товара для удаления, нажать в меню «Действия с таблицей», нажать кнопку «Удалить»; | * Убедиться, что после нажатия на кнопку появляется информационное сообщение о подтверждении удаления записи; * Убедиться, что после нажатия на кнопку появляется информационное сообщение об удалении записи, а также запись удаляется из главной панели. | Пройден  Рисунок Б16-Б18 |

ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний.

Разработанная программа может быть использована в развлекательных целях, для отдыха и релаксации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд ., Язык программирования C#, 4-е издание, СПб:-ПИТЕР, 2020
2. Фленов М., Библия C#, 3-е издание, СПб:-БХВ, 2019
3. Культин Н., Microsoft Visual C# в задачах и примерах, 2-е издание, СПб:-БХВ, 2015
4. Жанры игр и их особенности [Электронный ресурс] — URL: [https://sreda.temadnya.com/ 1043184354218215685/zhanry-igr-i-ih-osobennosti/](https://sreda.temadnya.com/%201043184354218215685/zhanry-igr-i-ih-osobennosti/) (дата обращения: 12.09.2020)
5. Описание языка C# [Электронный ресурс] — URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/ (дата обращения: 03.12.2019)
6. Стандарты Единой Системы Программной Документации:

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению

ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

ПРИЛОЖЕНИЕ А

Министерство ОБРАЗОВАНИЯ И науки Российской Федерации  
Федеральное автономное образовательное учреждение высшего профессионального образования  
«Санкт-Петербургский государственный политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ)  
**Институт среднего профессионального образования**

УТВЕРЖДАЮ  
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_ Андреев В.А.  
\_\_. \_\_. 2024

Приложение для работы с БД «Магазин Теплоснобжения»  
**Техническое задание**  
Листов 7

ПРОВЕРИЛ  
Преподаватель   
\_\_\_\_\_\_\_\_\_\_\_ Девятко Н.С.  
22.01.2024

ВЫПОЛНИЛ  
Студент группы 32919/2  
\_\_\_\_\_\_\_\_\_\_\_ Зайцева Д.А.  
22.01.2024

2024

1. ВВЕДЕНИЕ
   1. Полное наименование программной разработки: «ПРИЛОЖЕНИЕ ДЛЯ РАБОТЫ С БД «МАГАЗИН ТЕПЛОСНОБЖЕНИЯ»».
   2. Приложение для работы с БД «Магазин теплоснабжения»: секретарь ООО использует приложение для внесения данных о сотрудниках предприятия, поставляемых товарах и услугах; приложение позволяет добавлять, редактировать, удалять, просматривать, искать записи о сотрудниках, товарах, услугах, формировать отчеты о продажах за месяца, кварталы и года; продавец имеет ограниченные права и может только просматривать и искать информацию, а также выводить сводные отчеты; администратор системы работает с учетными записями пользователей, может менять статус сотрудников, добавлять новых, удалять старых; директор ООО имеет полный доступ к таблицам товаров и услуг, к ценовой части БД, с возможностью корректировки и отслеживания.
   3. В соответствии с заданием программный продукт состоит из теоретической и практической частей. Теоретическая часть включает описание предметной области задачи, анализ методов её решения, обзор и обоснование выбора средств программирования. Практическая часть включает подробное описание проектирования программного продукта, разработку и реализацию на языке программирования C# с использованием среды программирования Visual Studio программных модулей программного продукта, и СУБД Microsoft SQL Server для разработки и управления базой данных.
2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ
   1. Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и утверждена Институтом среднего профессионального образования.
3. НАЗНАЧЕНИЕ РАЗРАБОТКИ
   1. Основное назначение программного продукта: информационная система по работе с базой данных магазина теплоснабжения.
   2. Эксплуатационное назначение программного продукта: программа предназначена для сотрудников магазина теплоснабжения, для удобного хранения, редактирования и поиска информации, а также разграничения доступа к данным разных видов пользователей: секретарь, администратор, продавец и директор предприятия.

4. ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1. Требования к функциональным характеристикам:

**1 версия:**

* программа должна обеспечивать просмотр, добавление, удаление, редактирование данных о сотрудниках, услугах, товарах, продажах;
* программа должна отображать специальную форму для удобного добавления данных о сотрудниках, услугах, товарах, продажах;
* программа должна автоматически подгружать данные редактируемой записи в специальную форму для их удобной корректировки;
* программа должна обеспечивать поиск информации по основным критериям в каждой таблице;
* программа должна иметь удобный пользовательский интерфейс.

**2 версия:**

* программа должна иметь возможность авторизации пользователя;
* после авторизации программа должна отображать в верхней части окна данные о роли пользователя в системе и его ФИО;
* программа должна предоставлять возможность смены пользователя без перезапуска приложения;
* программа должна обеспечивать разграничение прав доступа для администратора, секретаря, продавца и директора предприятия, исходя из введённого логина: секретарь ООО использует приложение для внесения данных о сотрудниках предприятия, поставляемых товарах и услугах; приложение позволяет добавлять, редактировать, удалять, просматривать, искать записи о сотрудниках, товарах, услугах, формировать отчеты о продажах за месяца, кварталы и года; продавец имеет ограниченные права и может только просматривать и искать информацию, а также выводить сводные отчеты; администратор системы работает с учетными записями пользователей, может менять статус сотрудников, добавлять новых, удалять старых; директор ООО имеет полный доступ к таблицам товаров и услуг, к ценовой части БД, с возможностью корректировки и отслеживания;
* программа должна иметь возможность импортировать данные о поставляемых товарах и оказываемых услугах из внешних источников;
* программа должна уметь автоматически создавать отчеты по требованию пользователя (роль секретарь), содержащий сводную информацию по продажам/услугам/сотрудникам на заданный период, и экспортировать отчеты в формате word/excel/pdf с возможностью предварительного просмотра и вывода на печать.

4.2. Требования к надежности:

* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* организация бесперебойного питания.

4.3. Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~4 Мб;
* объем необходимой памяти на жестком диске ~100 Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь»;
* стандартный офисный принтер.

.

4.4. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 10 или совместимой, и СУБД Microsoft SQL Server для ведения и управления базой данных. Язык интерфейса – русский.

4.5. Требования к маркировке и упаковке

Программа должна поставляться в виде проекта, исполняемого (еxе) файла, установщика и документации.

4.6. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

4.7. Специальные требования

Теоретическая часть включает описание предметной области задачи, анализ методов её решения, обзор и обоснование выбора средств программирования. Практическая часть включает подробное описание проектирования программного продукта, разработку и реализацию на языке программирования C# с использованием среды программирования Visual Studio программных модулей программного продукта, и СУБД Microsoft SQL Server для разработки и управления базой данных.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

5.2. Перечень материалов пояснительной записки



Рисунок 1 – Перечень материалов пояснительной записки

6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание стадии** | **Содержание этапа** | **Срок 2024 г.** | | **Форма  отчетности** |
| **начало** | **конец** |
| Техническое задание | Составление технического задания | 15.01 | 21.01 | Техническое задание |
| Эскизный проект | Проектирование программы | 22.01 | 18.02 | UML-диаграммы |
| Рабочий проект | Разработка спецификаций | 19.02 | 03.03 | Спецификации отдельных компонентов |
| Составление программы | 04.03 | 31.03 | Программная документация |
| Приёмо-сдаточные испытания | 01.04 | 14.04 | Протокол испытаний (п. 2.7 пояснительной записки) |
| Приёмка | Защита курсового проекта | 15.04 | 20.04 | Оценка за курсовой проект |

8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1. Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

8.2. Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

ПРИЛОЖЕНИЕ Б

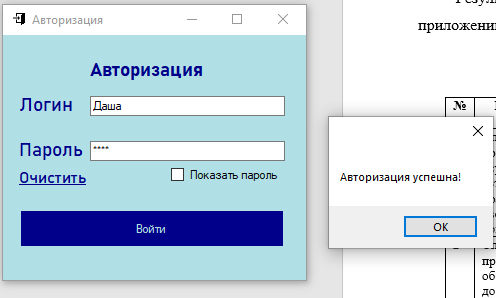


Рисунок Б1 Авторизация

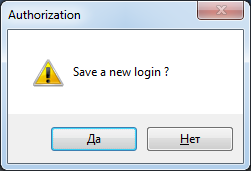
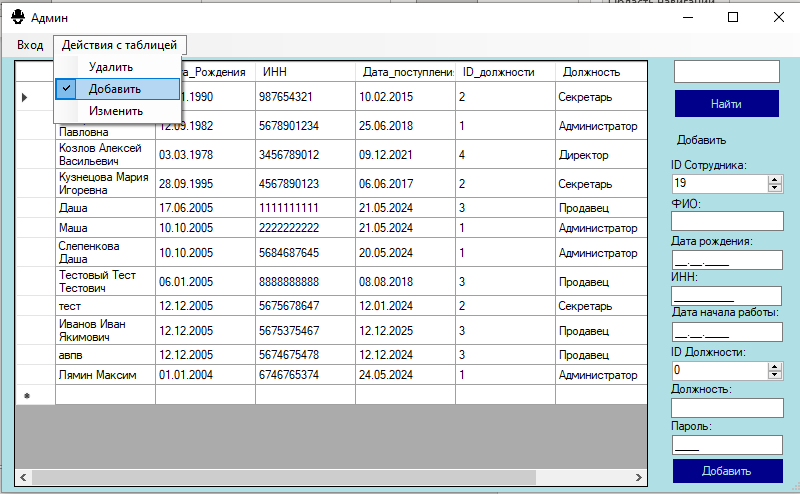
 

Рисунок Б2 Регистрация нового пользователя

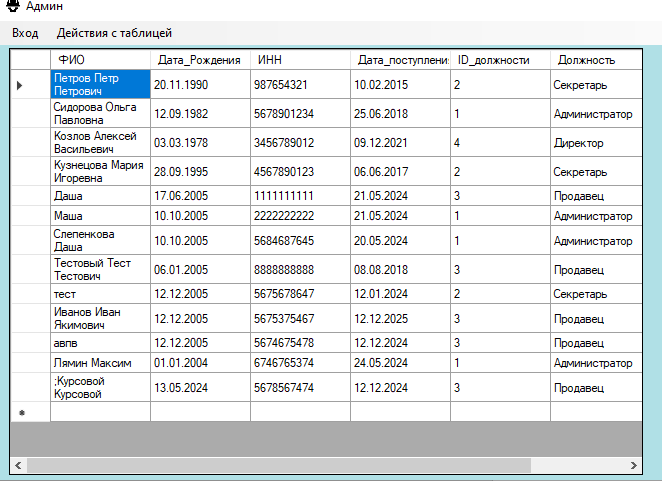


Рисунок Б3 Добавленный новый пользователь

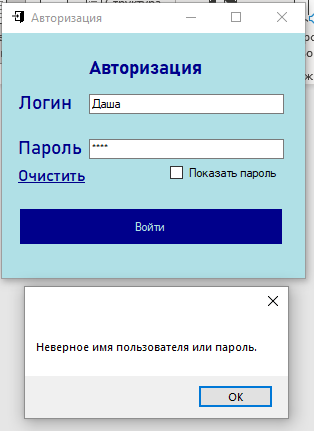


Рисунок Б4 Вывод ошибки при регистрации

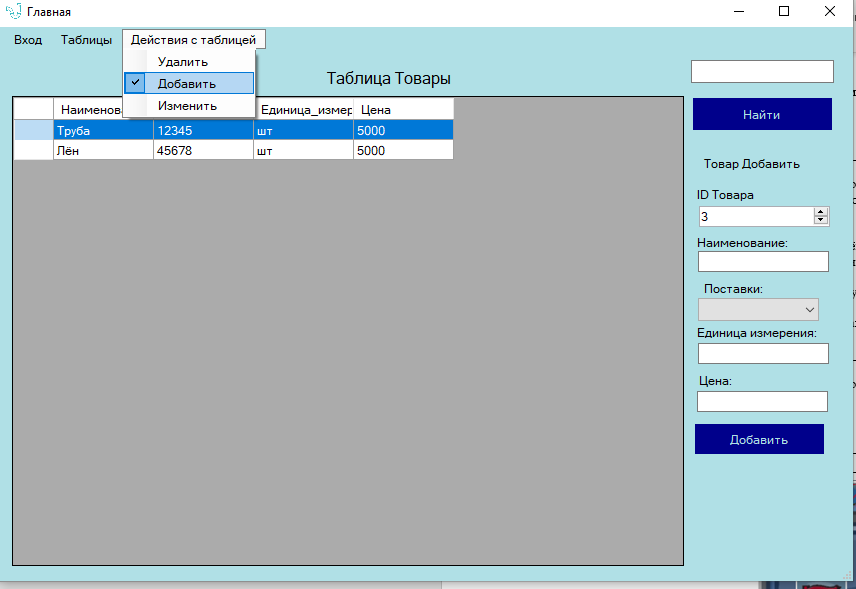


Рисунок Б5 Добавление нового товара

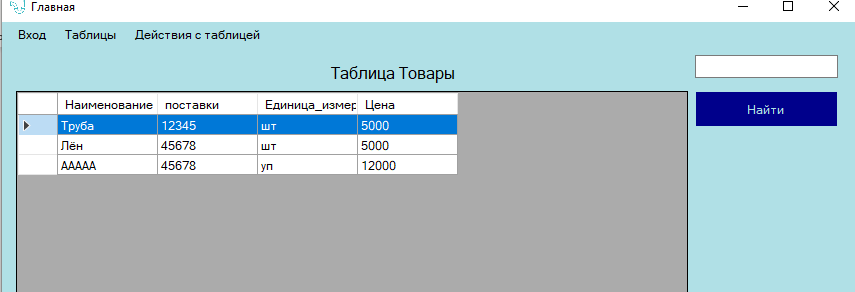


Рисунок Б6 Новый добавленный товар

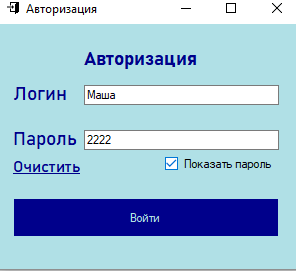


Рисунок Б7 Отображение пароля

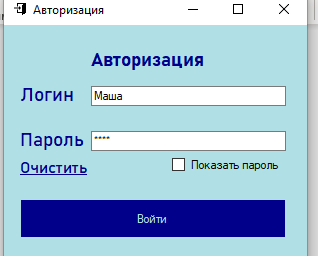


Рисунок Б8 Ввод данных

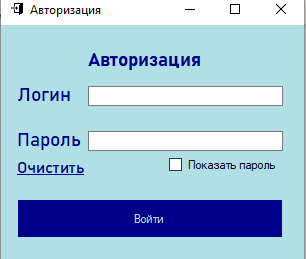


Рисунок Б9 Очистка данных

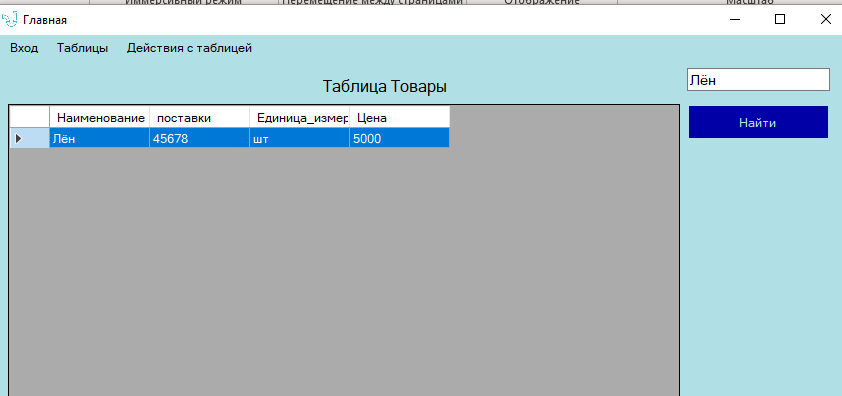


Рисунок Б10 Поиск данных

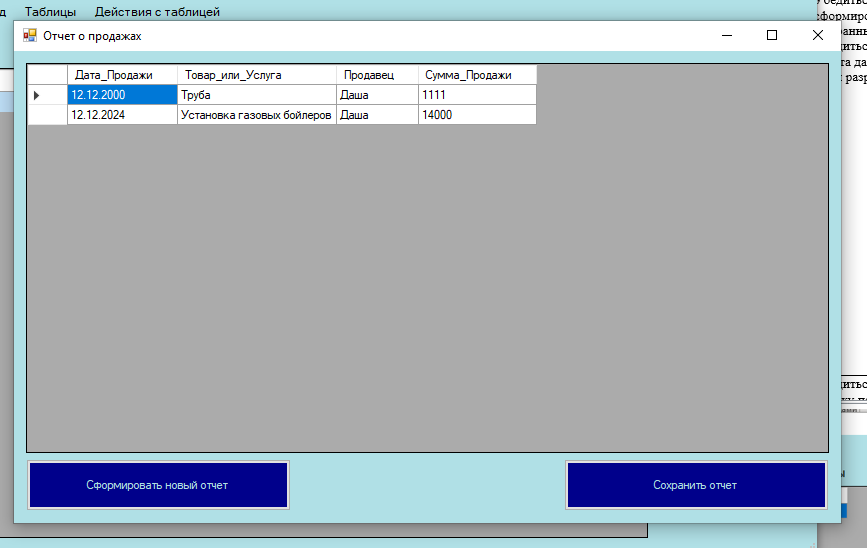


Рисунок Б11 Формирование отчета

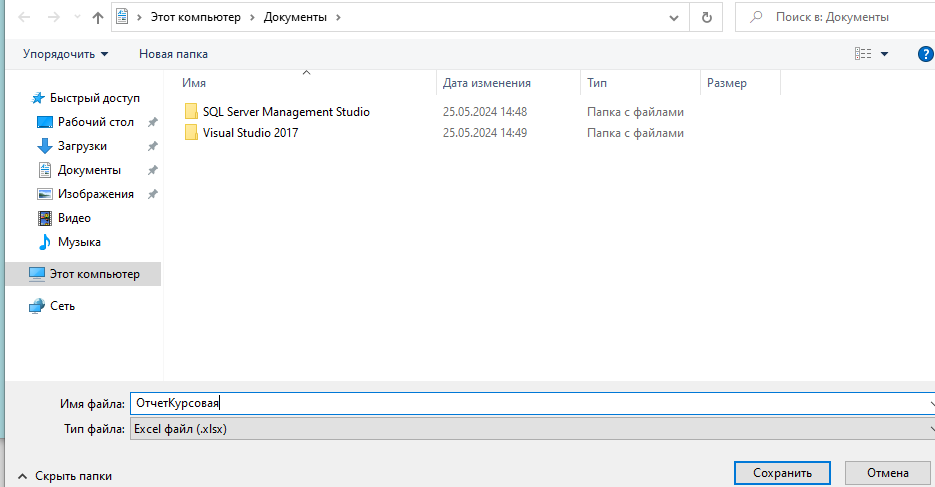


Рисунок Б12 Сохраненный отчет

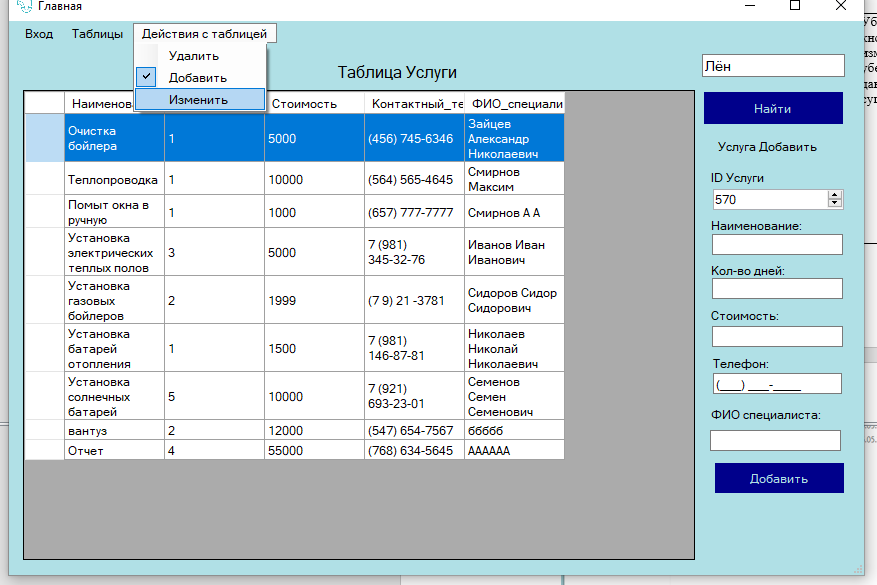


Рисунок Б13 Изменение данных

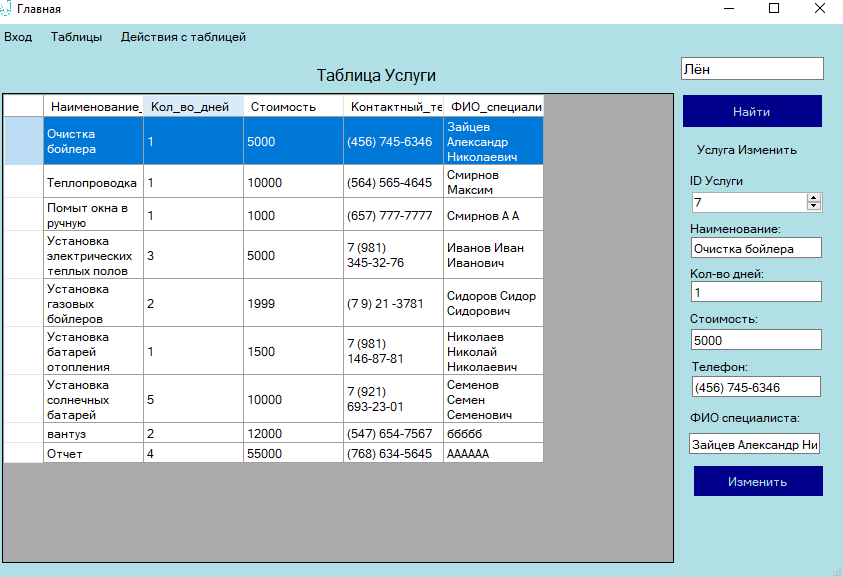


Рисунок Б14 Автоматический ввод данных

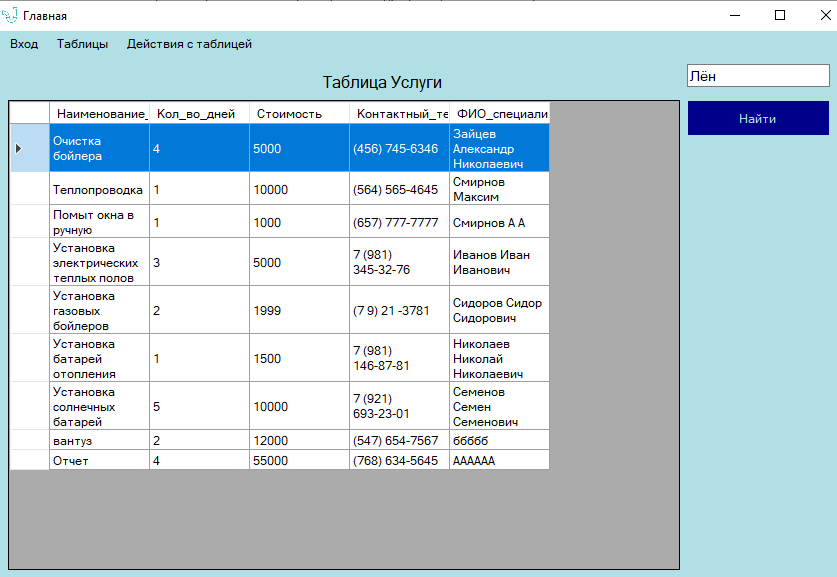


Рисунок Б15 Измененные данные

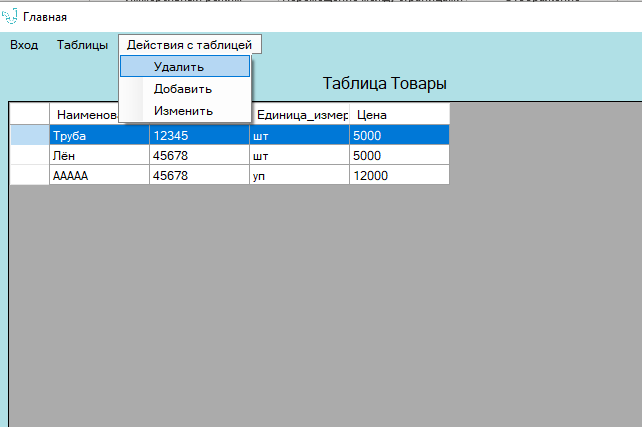


Рисунок Б16 Удаление данных

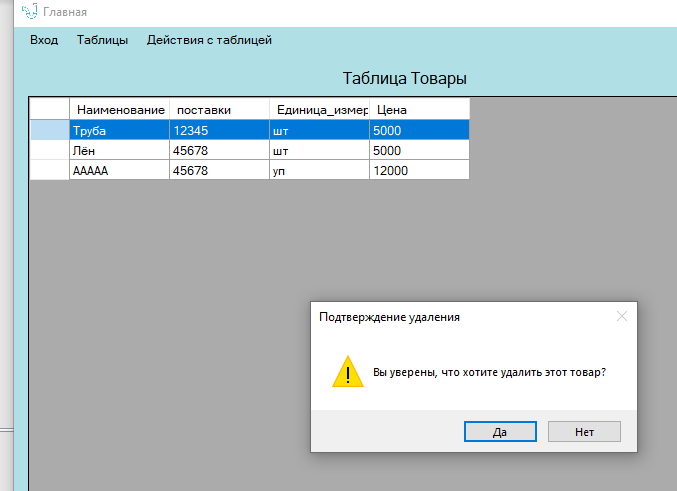


Рисунок Б17 Выбор удаления

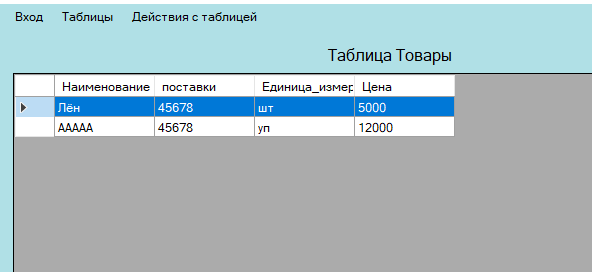


Рисунок Б18 Отображение после удаления

ПРИЛОЖЕНИЕ В

**Текст программы**

**Модуль Авторизация.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Курсовая\_Зайцева;

namespace Курсовая\_Зайцева

{

public partial class Авторизация : Form

{

private readonly БД db;

public Авторизация()

{

InitializeComponent();

db = new БД();

}

private void button1\_Click(object sender, EventArgs e)

{

string username = textBox1.Text;

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Введите имя пользователя и пароль.");

return;

}

string connectionString = db.GetConnectionString();

int userid = GetUserID(username, connectionString); // Получаем userid из базы данных

if (AuthenticateUser(username, password, connectionString))

{

string userRole = GetUserRole(username, connectionString); // Получаем роль пользователя

MessageBox.Show("Авторизация успешна!");

this.Hide();

if (userRole == "Администратор")

{

Админ adminForm = new Админ(userid);

adminForm.ShowDialog();

}

else

{

Главная maimMuni = new Главная(userid); // Передаем userid в конструктор формы MaimMuni

maimMuni.ShowDialog();

}

try

{

this.Show();

}

catch

{

}

}

else

{

MessageBox.Show("Неверное имя пользователя или пароль.");

}

}

private string GetUserRole(string username, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Должность FROM Сотрудники WHERE ФИО = @ФИО";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

}

}

return "Unknown";

}

private int GetUserID(string username, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT ID\_сотрудника FROM Сотрудники WHERE ФИО = @ФИО";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

object result = command.ExecuteScalar();

if (result != null)

{

return (int)result;

}

}

}

return -1;

}

private bool AuthenticateUser(string username, string password, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Сотрудники WHERE ФИО = @ФИО AND Пароль = @Пароль";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

command.Parameters.AddWithValue("@Пароль", password);

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

return count > 0;

}

}

}

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

{

textBox1.Clear();

textBox2.Clear();

}

private void chkShowPassword\_CheckedChanged(object sender, EventArgs e)

{

textBox2.PasswordChar = chkShowPassword.Checked ? '\0' : '\*';

}

}

}

**Модуль Админ.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Курсовая\_Зайцева;

namespace Курсовая\_Зайцева

{

public partial class Авторизация : Form

{

private readonly БД db;

public Авторизация()

{

InitializeComponent();

db = new БД();

}

private void button1\_Click(object sender, EventArgs e)

{

string username = textBox1.Text;

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Введите имя пользователя и пароль.");

return;

}

string connectionString = db.GetConnectionString();

int userid = GetUserID(username, connectionString); // Получаем userid из базы данных

if (AuthenticateUser(username, password, connectionString))

{

string userRole = GetUserRole(username, connectionString); // Получаем роль пользователя

MessageBox.Show("Авторизация успешна!");

this.Hide();

if (userRole == "Администратор")

{

Админ adminForm = new Админ(userid);

adminForm.ShowDialog();

}

else

{

Главная maimMuni = new Главная(userid); // Передаем userid в конструктор формы MaimMuni

maimMuni.ShowDialog();

}

try

{

this.Show();

}

catch

{

}

}

else

{

MessageBox.Show("Неверное имя пользователя или пароль.");

}

}

private string GetUserRole(string username, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Должность FROM Сотрудники WHERE ФИО = @ФИО";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

}

}

return "Unknown";

}

private int GetUserID(string username, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT ID\_сотрудника FROM Сотрудники WHERE ФИО = @ФИО";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

object result = command.ExecuteScalar();

if (result != null)

{

return (int)result;

}

}

}

return -1;

}

private bool AuthenticateUser(string username, string password, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Сотрудники WHERE ФИО = @ФИО AND Пароль = @Пароль";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

command.Parameters.AddWithValue("@Пароль", password);

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

return count > 0;

}

}

}

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

{

textBox1.Clear();

textBox2.Clear();

}

private void chkShowPassword\_CheckedChanged(object sender, EventArgs e)

{

textBox2.PasswordChar = chkShowPassword.Checked ? '\0' : '\*';

}

}

}

**Модуль Главная.cs**

private void chkShowPassword\_CheckedChanged(object sender, EventArgs e)

{

textBox2.PasswordChar = chkShowPassword.Checked ? '\0' : '\*';

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using Курсовая\_Зайцева;

namespace Курсовая\_Зайцева

{

public partial class Главная : Form

{

private int userid;

private readonly БД db;

private readonly string connectionString;

public Главная(int userid)

{

InitializeComponent();

this.userid = userid;

db = new БД();

connectionString = db.GetConnectionString();

string userRole = GetUserRole(userid, db.GetConnectionString());

dataGridView1.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

}

private string GetUserRole(int ID\_сотрудника, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Должность FROM Сотрудники WHERE ID\_сотрудника = @ID\_сотрудника";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ID\_сотрудника", ID\_сотрудника);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

}

}

return "Unknown"; // Если роль не найдена

}

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

{

Application.Restart();

}

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

{

foreach(ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Товары();

}

private void Отобразить\_Товары()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_Поставки.Visible = false;

panel\_Продажи.Visible = false;

Panel\_Товар.Visible = false;

panel\_Услуги.Visible = false;

label26.Text = "Таблица Товары";

connection.Open();

string query = @"SELECT t.ID\_Товара, t.Наименование, t.поставки, t.Единица\_измерения, t.Цена

FROM Товары t ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Товары = new DataTable();

adapter.Fill(Товары);

dataGridView1.DataSource = Товары;

dataGridView1.Columns["ID\_Товара"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о товарах: " + ex.Message);

}

}

private void Найти\_Товар(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT t.ID\_Товара, t.Наименование, t.поставки, t.Единица\_измерения, t.Цена

FROM Товары t

WHERE t.ID\_Товара LIKE @searchValue OR t.Наименование LIKE @searchValue OR t.поставки

LIKE @searchValue OR t.Единица\_измерения LIKE @searchValue OR t.Цена LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Товары = new DataTable();

adapter.Fill(Товары);

dataGridView1.DataSource = Товары;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_Товар(int ID\_Товара)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Товары WHERE ID\_Товара = @ID\_Товара";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ID\_Товара", ID\_Товара);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

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

}

else

{

MessageBox.Show("Произошла ошибка при удалении товара.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении товара: " + ex.Message);

}

}

private void Добавить\_Товар()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string ID\_товара = numericUpDown1.Text;

string Наименование\_Товара = textBox1.Text;

string Поставки = comboBox1.Text;

string Единица\_Измерения = textBox2.Text;

string Цена = textBox3.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Insert into Товары(ID\_Товара,Наименование,поставки,Единица\_измерения,Цена)

VALUES(@ID\_товара,@Наименование\_Товара,@Поставки,@Единица\_Измерения,@Цена);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@ID\_товара", ID\_товара);

cmd1.Parameters.AddWithValue("@Наименование\_Товара", Наименование\_Товара);

cmd1.Parameters.AddWithValue("@Поставки", Поставки);

cmd1.Parameters.AddWithValue("@Единица\_Измерения", Единица\_Измерения);

cmd1.Parameters.AddWithValue("@Цена", Цена);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

textBox1.Clear();

comboBox1.Text = "";

textBox2.Clear();

textBox3.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при добавлении товара: " + ex.Message);

}

}

private void Изменить\_Товар()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string ID\_товара = numericUpDown1.Text;

string Наименование\_Товара = textBox1.Text;

string Поставки = comboBox1.Text;

string Единица\_Измерения = textBox2.Text;

string Цена = textBox3.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Update Товары set ID\_товара=@ID\_товара, Наименование=@Наименование\_Товара, поставки=@Поставки, Единица\_Измерения=@Единица\_Измерения, Цена=@Цена

Where ID\_товара=@ID\_товара;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@ID\_товара", ID\_товара);

cmd1.Parameters.AddWithValue("@Наименование\_Товара", Наименование\_Товара);

cmd1.Parameters.AddWithValue("@Поставки", Поставки);

cmd1.Parameters.AddWithValue("@Единица\_Измерения", Единица\_Измерения);

cmd1.Parameters.AddWithValue("@Цена", Цена);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

textBox1.Clear();

comboBox1.Text = "";

textBox2.Clear();

textBox3.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при изменении товара: " + ex.Message);

}

}

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

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Услугу();

}

private void Отобразить\_Услугу()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_Поставки.Visible = false;

panel\_Продажи.Visible = false;

Panel\_Товар.Visible = false;

panel\_Услуги.Visible = false;

label26.Text = "Таблица Услуги";

connection.Open();

string query = @"SELECT l.ID\_Услуги, l.Наименование\_услуги, l.Кол\_во\_дней, l.Стоимость, l.Контактный\_телефон, l.ФИО\_специалиста

FROM Услуги l ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Услуги = new DataTable();

adapter.Fill(Услуги);

dataGridView1.DataSource = Услуги;

dataGridView1.Columns["ID\_Услуги"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных об услугах: " + ex.Message);

}

}

private void Найти\_Услугу(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT l.ID\_Услуги, l.Наименование\_услуги, l.Кол\_во\_дней, l.Стоимость, l.Контактный\_телефон, l.ФИО\_специалиста

FROM Услуги l

WHERE l.ID\_Услуги LIKE @searchValue OR l.Наименование\_услуги LIKE @searchValue OR l.Кол\_во\_дней LIKE @searchValue

OR l.Стоимость LIKE @searchValue OR l.Контактный\_телефон LIKE @searchValue OR l.ФИО\_специалиста LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Услуги = new DataTable();

adapter.Fill(Услуги);

dataGridView1.DataSource = Услуги;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_Услугу(int ID\_Услуги)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Услуги WHERE ID\_Услуги = @ID\_Услуги";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ID\_Услуги", ID\_Услуги);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

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

}

else

{

MessageBox.Show("Произошла ошибка при удалении услуги.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении услуги: " + ex.Message);

}

}

private void Добавить\_Услугу()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string ID\_услуги = numericUpDown2.Text;

string Наименование\_Услуги = textBox6.Text;

string Кол\_во\_дней = textBox5.Text;

string Стоимость = textBox4.Text;

string Телефон\_Услуги = maskedTextBox4.Text;

string ФИО\_специалиста = textBox13.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Insert into Услуги(ID\_Услуги,Наименование\_услуги,Кол\_во\_дней,Стоимость,Контактный\_телефон,ФИО\_специалиста)

VALUES(@ID\_услуги,@Наименование\_Услуги,@Кол\_во\_дней,@Стоимость,@Телефон\_Услуги,@ФИО\_специалиста);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@ID\_услуги", ID\_услуги);

cmd1.Parameters.AddWithValue("@Наименование\_Услуги", Наименование\_Услуги);

cmd1.Parameters.AddWithValue("@Кол\_во\_дней", Кол\_во\_дней);

cmd1.Parameters.AddWithValue("@Стоимость", Стоимость);

cmd1.Parameters.AddWithValue("@Телефон\_Услуги", Телефон\_Услуги);

cmd1.Parameters.AddWithValue("@ФИО\_специалиста", ФИО\_специалиста);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

textBox6.Clear();

textBox5.Clear();

textBox4.Clear();

maskedTextBox4.Clear();

textBox13.Clear();

maskedTextBox4.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при добавлении услуги: " + ex.Message);

}

}

private void Изменить\_Услугу()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string ID\_услуги = numericUpDown2.Text;

string Наименование\_Услуги = textBox6.Text;

string Кол\_во\_дней = textBox5.Text;

string Стоимость = textBox4.Text;

string Телефон\_Услуги = maskedTextBox4.Text;

string ФИО\_специалиста = textBox13.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Update Услуги set ID\_Услуги=@ID\_услуги, Наименование\_услуги=@Наименование\_Услуги, Кол\_во\_дней=@Кол\_во\_дней, Стоимость=@Стоимость, Контактный\_телефон=@Телефон\_Услуги, ФИО\_специалиста=@ФИО\_специалиста

Where ID\_Услуги=@ID\_услуги;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@ID\_услуги", ID\_услуги);

cmd1.Parameters.AddWithValue("@Наименование\_Услуги", Наименование\_Услуги);

cmd1.Parameters.AddWithValue("@Кол\_во\_дней", Кол\_во\_дней);

cmd1.Parameters.AddWithValue("@Стоимость", Стоимость);

cmd1.Parameters.AddWithValue("@Телефон\_Услуги", Телефон\_Услуги);

cmd1.Parameters.AddWithValue("@ФИО\_специалиста", ФИО\_специалиста);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

textBox6.Clear();

textBox5.Clear();

textBox4.Clear();

maskedTextBox4.Clear();

textBox13.Clear();

maskedTextBox4.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при изменении услуги: " + ex.Message);

}

}

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

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Продажу();

}

private void Отобразить\_Продажу ()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_Поставки.Visible = false;

panel\_Продажи.Visible = false;

Panel\_Товар.Visible = false;

panel\_Услуги.Visible = false;

label26.Text = "Таблица Продажи";

connection.Open();

string query = @"SELECT p.ID\_Продажи, p.Дата, p.ID\_Сотрудника, p.[Товар/услуга], p.Сумма\_продажи

FROM Продажи p ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Продажи = new DataTable();

adapter.Fill(Продажи);

dataGridView1.DataSource = Продажи;

dataGridView1.Columns["ID\_Продажи"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о продажах: " + ex.Message);

}

}

private void Найти\_Продажу(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT p.ID\_Продажи, p.Дата, p.ID\_Сотрудника, p.[Товар/услуга], p.Сумма\_продажи

FROM Продажи p

WHERE p.ID\_Продажи LIKE @searchValue OR p.Дата LIKE @searchValue OR p.ID\_Сотрудника LIKE @searchValue

OR p.[Товар/услуга] LIKE @searchValue OR p.Сумма\_продажи LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Продажи = new DataTable();

adapter.Fill(Продажи);

dataGridView1.DataSource = Продажи;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_Продажу(int ID\_Продажи)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Продажи WHERE ID\_Продажи = @ID\_Продажи";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ID\_Продажи", ID\_Продажи);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

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

}

else

{

MessageBox.Show("Произошла ошибка при удалении продажи.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении продажи: " + ex.Message);

}

}

private void Добавить\_Продажу()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string ID\_Продажи = numericUpDown4.Text;

string Дата\_продажи = maskedTextBox7.Text;

string ID\_Сотрудника = numericUpDown5.Text;

string Товар\_Услуга = textBox11.Text;

string Сумма\_Продажи = textBox10.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Insert into Продажи(ID\_Продажи,Дата,ID\_Сотрудника,[Товар/услуга],Сумма\_продажи)

VALUES(@ID\_Продажи,@Дата\_продажи,@ID\_Сотрудника,@Товар\_Услуга,@Сумма\_Продажи);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@ID\_Продажи", ID\_Продажи);

cmd1.Parameters.AddWithValue("@Дата\_продажи", Дата\_продажи);

cmd1.Parameters.AddWithValue("@ID\_Сотрудника", ID\_Сотрудника);

cmd1.Parameters.AddWithValue("@Товар\_Услуга", Товар\_Услуга);

cmd1.Parameters.AddWithValue("@Сумма\_Продажи", Сумма\_Продажи);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

maskedTextBox7.Clear();

textBox11.Clear();

textBox10.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при добавлении продажи: " + ex.Message);

}

}

private void Изменить\_Продажу()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string ID\_Продажи = numericUpDown4.Text;

string Дата\_продажи = maskedTextBox7.Text;

string ID\_Сотрудника = numericUpDown5.Text;

string Товар\_Услуга = textBox11.Text;

string Сумма\_Продажи = textBox10.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Update Продажи set ID\_Продажи=@ID\_Продажи, Дата=@Дата\_продажи, ID\_Сотрудника=@ID\_Сотрудника, [Товар/услуга]=@Товар\_Услуга, Сумма\_продажи=@Сумма\_Продажи

Where ID\_Продажи=@ID\_Продажи;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@ID\_Продажи", ID\_Продажи);

cmd1.Parameters.AddWithValue("@Дата\_продажи", Дата\_продажи);

cmd1.Parameters.AddWithValue("@ID\_Сотрудника", ID\_Сотрудника);

cmd1.Parameters.AddWithValue("@Товар\_Услуга", Товар\_Услуга);

cmd1.Parameters.AddWithValue("@Сумма\_Продажи", Сумма\_Продажи);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

maskedTextBox7.Clear();

textBox11.Clear();

textBox10.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при изменении продажи: " + ex.Message);

}

}

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

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Поставку();

}

private void Отобразить\_Поставку()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_Поставки.Visible = false;

panel\_Продажи.Visible = false;

Panel\_Товар.Visible = false;

panel\_Услуги.Visible = false;

label26.Text = "Таблица Поставки";

connection.Open();

string query = @"SELECT po.поставки, po.Поставщик, po.Дата\_поставки, po.Дата\_приема\_поставки, po.ID\_Сотрудника

FROM Поставки po ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Поставки = new DataTable();

adapter.Fill(Поставки);

dataGridView1.DataSource = Поставки;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о поставке: " + ex.Message);

}

}

private void Найти\_Поставку(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT po.поставки, po.Поставщик, po.Дата\_поставки, po.Дата\_приема\_поставки, po.ID\_Сотрудника

FROM Поставки po

WHERE po.поставки LIKE @searchValue OR po.Поставщик LIKE @searchValue OR po.Дата\_поставки LIKE @searchValue

OR po.Дата\_приема\_поставки LIKE @searchValue OR po.ID\_Сотрудника LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Поставки = new DataTable();

adapter.Fill(Поставки);

dataGridView1.DataSource = Поставки;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_Поставку(int поставки)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Поставки WHERE поставки = @поставки";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@поставки", поставки);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

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

}

else

{

MessageBox.Show("Произошла ошибка при удалении поставки.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении поставки: " + ex.Message);

}

}

private void Добавить\_Поставку()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string Номер\_Поставки = numericUpDown6.ToString();

string Поставщик = textBox9.Text;

string Дата\_Поставки = maskedTextBox5.Text;

string Дата\_Приема\_Поставки = maskedTextBox2.Text;

string ID\_Сотрудника = numericUpDown3.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Insert into Поставки(поставки,Поставщик,Дата\_поставки,Дата\_приема\_поставки,ID\_Сотрудника)

VALUES(@Номер\_Поставки,@Поставщик,@Дата\_Поставки,@Дата\_Приема\_Поставки,@ID\_Сотрудника);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@Номер\_Поставки", Номер\_Поставки);

cmd1.Parameters.AddWithValue("@Поставщик", Поставщик);

cmd1.Parameters.AddWithValue("@Дата\_Поставки", Дата\_Поставки);

cmd1.Parameters.AddWithValue("@Дата\_Приема\_Поставки", Дата\_Приема\_Поставки);

cmd1.Parameters.AddWithValue("@ID\_Сотрудника", ID\_Сотрудника);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

numericUpDown6.Value = 0;

textBox9.Clear();

maskedTextBox5.Clear();

maskedTextBox2.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при добавлении поставки: " + ex.Message);

}

}

private void Изменить\_Поставку()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string Номер\_Поставки = numericUpDown6.ToString();

string Поставщик = textBox9.Text;

string Дата\_Поставки = maskedTextBox5.Text;

string Дата\_Приема\_Поставки = maskedTextBox2.Text;

string ID\_Сотрудника = numericUpDown3.Text;

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = @"

Update Поставки set поставки=@Номер\_Поставки, Поставщик=@Поставщик, Дата\_поставки=@Дата\_Поставки, Дата\_приема\_поставки=@Дата\_Приема\_Поставки, ID\_Сотрудника=@ID\_Сотрудника

Where поставки=@Номер\_Поставки;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@Номер\_Поставки", Номер\_Поставки);

cmd1.Parameters.AddWithValue("@Поставщик", Поставщик);

cmd1.Parameters.AddWithValue("@Дата\_Поставки", Дата\_Поставки);

cmd1.Parameters.AddWithValue("@Дата\_Приема\_Поставки", Дата\_Приема\_Поставки);

cmd1.Parameters.AddWithValue("@ID\_Сотрудника", ID\_Сотрудника);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

numericUpDown6.Value = 0;

textBox9.Clear();

maskedTextBox5.Clear();

maskedTextBox2.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при изменении поставки: " + ex.Message);

}

}

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

{

Отчет otchot = new Отчет();

otchot.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

if (товарыToolStripMenuItem.Checked)

{

string searchValue = textBoxSearch.Text;

Найти\_Товар(searchValue);

}

else if (услугиToolStripMenuItem.Checked)

{

string searchValue = textBoxSearch.Text;

Найти\_Услугу(searchValue);

}

else if (поставкиToolStripMenuItem.Checked)

{

string searchValue = textBoxSearch.Text;

Найти\_Поставку(searchValue);

}

else if (продажиToolStripMenuItem.Checked)

{

string searchValue = textBoxSearch.Text;

Найти\_Продажу(searchValue);

}

}

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

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

if (товарыToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int ID\_Товара = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ID\_Товара"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить этот товар?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_Товар(ID\_Товара);

Отобразить\_Товары();

}

}

else

{

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

}

}

else if (услугиToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int ID\_Услуги = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ID\_Услуги"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить эту услугу?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_Услугу(ID\_Услуги);

Отобразить\_Услугу();

}

}

else

{

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

}

}

else if (поставкиToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int поставки = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ID\_Поставки"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить эту поставку?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_Поставку(поставки);

Отобразить\_Поставку();

}

}

else

{

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

}

}

else if (продажиToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int ID\_Продажи = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ID\_Продажи"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить эту продажу?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_Продажу(ID\_Продажи);

Отобразить\_Продажу();

}

}

else

{

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

}

}

}

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

{

Информация\_о\_пользователе maimMuni = new Информация\_о\_пользователе(userid); // Передаем userid в конструктор формы MaimMuni

maimMuni.Show();

}

private void UpdatePanelForTableИзменить(string tableName)

{

switch (tableName)

{

case "Товары":

Panel\_Товар.Visible = true;

label1.Text = "Товар Изменить";

button1.Text = "Изменить";

break;

case "Услуги":

panel\_Услуги.Visible = true;

label12.Text = "Услуга Изменить";

button2.Text = "Изменить";

break;

case "Поставки":

panel\_Поставки.Visible = true;

label18.Text = "Поставка Изменить";

button3.Text = "Изменить";

break;

case "Продажи":

panel\_Продажи.Visible = true;

label24.Text = "Продажа Изменить";

button5.Text = "Изменить";

break;

default:

break;

}

}

private void UpdatePanelForTableДобавить(string tableName)

{

switch (tableName)

{

case "Товары":

Panel\_Товар.Visible = true;

label1.Text = "Товар Добавить";

button1.Text = "Добавить";

break;

case "Услуги":

panel\_Услуги.Visible = true;

label12.Text = "Услуга Добавить";

button2.Text = "Добавить";

break;

case "Поставки":

panel\_Поставки.Visible = true;

label18.Text = "Поставка Добавить";

button3.Text = "Добавить";

break;

case "Продажи":

panel\_Продажи.Visible = true;

label24.Text = "Продажа Добавить";

button5.Text = "Добавить";

break;

default:

break;

}

}

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

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Изменить();

try

{

dataGridView1\_CellClick(dataGridView1, new DataGridViewCellEventArgs(0, dataGridView1.SelectedRows[0].Index));

}

catch

{

}

}

private void Изменить()

{

try

{

// Определяем, какую таблицу выбрал пользователь и вызываем метод обновления панели

if (товарыToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("Товары");

}

else if (услугиToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("Услуги");

}

else if (поставкиToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("Поставки");

}

else if (продажиToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("Продажи");

}

else

{

// Если ни один пункт не выбран, скрываем все панели

UpdatePanelForTableИзменить("");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при реализации: " + ex.Message);

}

}

private void Добавить()

{

try

{

// Определяем, какую таблицу выбрал пользователь и вызываем метод обновления панели

if (товарыToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("Товары");

}

else if (услугиToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("Услуги");

}

else if (поставкиToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("Поставки");

}

else if (продажиToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("Продажи");

}

else

{

// Если ни один пункт не выбран, скрываем все панели

UpdatePanelForTableДобавить("");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при реализации: " + ex.Message);

}

}

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

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Добавить();

if (товарыToolStripMenuItem.Checked)

{

comboBox1.Items.Clear();

int oleg = 100;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT MAX(ID\_Товара) FROM Товары";

using (SqlCommand command = new SqlCommand(query, connection))

{

oleg = int.Parse(command.ExecuteScalar().ToString()) + 1;

}

query = $"SELECT поставки FROM Поставки";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(((int)reader[0]).ToString());

}

}

}

}

catch

{

}

numericUpDown1.Value = oleg;

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

else if (услугиToolStripMenuItem.Checked)

{

// заполняем поля текущими значениями выбранной строки

int oleg = 999;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT MAX(ID\_Услуги) FROM Услуги";

using (SqlCommand command = new SqlCommand(query, connection))

{

oleg = int.Parse(command.ExecuteScalar().ToString()) + 1;

}

}

}

catch

{

}

numericUpDown2.Value = oleg;

textBox6.Text = "";

textBox5.Text = "";

textBox4.Text = "";

maskedTextBox4.Text = "";

textBox13.Text = "";

}

else if (поставкиToolStripMenuItem.Checked)

{

// заполняем поля текущими значениями выбранной строки

numericUpDown6.Value = 0;

textBox9.Text = "";

maskedTextBox5.Text = "";

maskedTextBox2.Text = "";

numericUpDown3.Value = 0;

}

else if (продажиToolStripMenuItem.Checked)

{

// заполняем поля текущими значениями выбранной строки

int oleg = 100;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT MAX(ID\_Продажи) FROM Продажи";

using (SqlCommand command = new SqlCommand(query, connection))

{

oleg = int.Parse(command.ExecuteScalar().ToString()) + 1;

}

}

}

catch

{

}

numericUpDown4.Value = oleg;

maskedTextBox7.Text = "";

numericUpDown5.Value = 0;

textBox11.Text = "";

textBox10.Text = "";

}

}

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

{

if (!добавитьToolStripMenuItem.Checked)

{

if (товарыToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

comboBox1.Items.Clear();

// заполняем поля текущими значениями выбранной строки

numericUpDown1.Value = int.Parse(row.Cells["ID\_Товара"].Value.ToString());

textBox1.Text = row.Cells["Наименование"].Value.ToString();

textBox2.Text = row.Cells["Единица\_измерения"].Value.ToString();

textBox3.Text = row.Cells["Цена"].Value.ToString();

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = $"SELECT поставки FROM Поставки";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(((int)reader[0]).ToString());

}

}

comboBox1.Text = row.Cells["поставки"].Value.ToString();

}

}

catch

{

}

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

else if (услугиToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

// заполняем поля текущими значениями выбранной строки

numericUpDown2.Value = int.Parse(row.Cells["ID\_Услуги"].Value.ToString());

textBox6.Text = row.Cells["Наименование\_услуги"].Value.ToString();

textBox5.Text = row.Cells["Кол\_во\_дней"].Value.ToString();

textBox4.Text = row.Cells["Стоимость"].Value.ToString();

maskedTextBox4.Text = row.Cells["Контактный\_телефон"].Value.ToString();

textBox13.Text = row.Cells["ФИО\_специалиста"].Value.ToString();

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

else if (поставкиToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

// заполняем поля текущими значениями выбранной строки

numericUpDown6.Value = int.Parse(row.Cells["поставки"].Value.ToString());

textBox9.Text = row.Cells["Поставщик"].Value.ToString();

maskedTextBox5.Text = row.Cells["Дата\_поставки"].Value.ToString();

maskedTextBox2.Text = row.Cells["Дата\_приема\_поставки"].Value.ToString();

numericUpDown3.Value = (int)row.Cells["ID\_Сотрудника"].Value;

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

else if (продажиToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

// заполняем поля текущими значениями выбранной строки

numericUpDown4.Value = int.Parse(row.Cells["ID\_Продажи"].Value.ToString());

maskedTextBox7.Text = row.Cells["Дата"].Value.ToString();

numericUpDown5.Value = (int)row.Cells["ID\_Сотрудника"].Value;

textBox11.Text = row.Cells["Товар/услуга"].Value.ToString();

textBox10.Text = row.Cells["Сумма\_продажи"].Value.ToString();

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_Товар();

Отобразить\_Товары();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_Товар();

Отобразить\_Товары();

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_Услугу();

Отобразить\_Услугу();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_Услугу();

Отобразить\_Услугу();

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_Поставку();

Отобразить\_Поставку();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_Поставку();

Отобразить\_Поставку();

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_Продажу();

Отобразить\_Продажу();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_Продажу();

Отобразить\_Продажу();

}

}

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

{

}

}

}

**Модуль Информация\_о\_пользователе.cs**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using Курсовая\_Зайцева;

namespace Курсовая\_Зайцева

{

public partial class Информация\_о\_пользователе : Form

{

private int userid;

private readonly БД db;

private readonly string connectionString;

public Информация\_о\_пользователе(int userid)

{

InitializeComponent();

this.userid = userid;

db = new БД();

connectionString = db.GetConnectionString();

}

private void Информация\_о\_пользователе\_Load(object sender, EventArgs e)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Сотрудники WHERE ID\_сотрудника = @userid";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@userid", userid);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

label1.Text = reader["ФИО"].ToString();

label2.Text = reader["Должность"].ToString();

label3.Text = reader["Дата\_Рождения"].ToString();

label4.Text = reader["Дата\_поступления\_на\_работу"].ToString();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных специализации: " + ex.Message);

}

}

}

}

**Модуль Отчет.cs**

using NPOI.SS.UserModel;

using NPOI.XSSF.UserModel;

using PdfSharpCore.Drawing;

using PdfSharpCore.Pdf;

using System;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Windows.Forms;

using Xceed.Document.NET;

using Xceed.Words.NET;

namespace Курсовая\_Зайцева

{

public partial class Отчет : Form

{

private DataTable dataTable;

private DateTime date1;

private DateTime date2;

private readonly БД db = new БД();

public Отчет()

{

InitializeComponent();

button1\_Click(null, null);

}

public static void ExportDataTableToWord(DataTable dataTable, string filePath, string period)

{

if (dataTable == null || dataTable.Rows.Count == 0)

{

throw new ArgumentNullException("dataTable", "DataTable is null or empty");

}

using (DocX document = DocX.Create(filePath))

{

document.InsertParagraph($"Отчет за период: {period}").FontSize(14).Bold().Alignment = Alignment.center;

Xceed.Document.NET.Table table = document.AddTable(dataTable.Rows.Count + 1, dataTable.Columns.Count);

for (int i = 0; i < dataTable.Columns.Count; i++)

{

table.Rows[0].Cells[i].InsertParagraph(dataTable.Columns[i].ColumnName).Bold();

}

for (int row = 0; row < dataTable.Rows.Count; row++)

{

for (int col = 0; col < dataTable.Columns.Count; col++)

{

if (dataTable.Columns[col].DataType == typeof(DateTime))

{

table.Rows[row + 1].Cells[col].InsertParagraph(((DateTime)dataTable.Rows[row][col]).ToString("yyyy.MM.dd"));

}

else

{

table.Rows[row + 1].Cells[col].InsertParagraph(dataTable.Rows[row][col].ToString());

}

}

}

document.InsertTable(table);

document.Save();

}

}

public static void ExportDataTableToPdf(DataTable dataTable, string filePath, string period)

{

if (dataTable == null || dataTable.Rows.Count == 0)

{

throw new ArgumentNullException("dataTable", "DataTable is null or empty");

}

PdfDocument document = new PdfDocument();

document.Info.Title = "Отчет";

PdfPage page = document.AddPage();

XGraphics gfx = XGraphics.FromPdfPage(page);

XFont font = new XFont("Arial", 8, XFontStyle.Regular);

int y = 20;

gfx.DrawString($"Отчет за период: {period}", font, XBrushes.Black, new XPoint(20, y));

y += 20;

int x = 20;

for (int i = 0; i < dataTable.Columns.Count; i++)

{

gfx.DrawString(dataTable.Columns[i].ColumnName, font, XBrushes.Black, new XPoint(x, y));

x += 100;

}

y += 20;

for (int row = 0; row < dataTable.Rows.Count; row++)

{

for (int col = 0; col < dataTable.Columns.Count; col++)

{

if (dataTable.Columns[col].DataType == typeof(DateTime))

{

gfx.DrawString(((DateTime)dataTable.Rows[row][col]).ToString("yyyy.MM.dd"), font, XBrushes.Black, new XPoint(20 + col \* 100, y));

}

else

{

gfx.DrawString(dataTable.Rows[row][col].ToString(), font, XBrushes.Black, new XPoint(20 + col \* 100, y));

}

}

y += 20;

}

document.Save(filePath);

}

public static void ExportDataTableToExcel(DataTable dataTable, string filePath, string period)

{

if (dataTable == null || dataTable.Rows.Count == 0)

{

throw new ArgumentNullException("dataTable", "DataTable is null or empty");

}

// Создаем новую книгу Excel

IWorkbook workbook = new XSSFWorkbook();

ISheet sheet = workbook.CreateSheet("Отчет");

// Создаем строку для заголовка с указанием периода

IRow headerRow = sheet.CreateRow(0);

headerRow.CreateCell(0).SetCellValue($"Отчет за период: {period}");

headerRow.Cells[0].CellStyle = GetHeaderCellStyle(workbook);

// Создаем строку для заголовков столбцов

IRow columnsRow = sheet.CreateRow(1);

for (int i = 0; i < dataTable.Columns.Count; i++)

{

columnsRow.CreateCell(i).SetCellValue(dataTable.Columns[i].ColumnName);

columnsRow.Cells[i].CellStyle = GetHeaderCellStyle(workbook);

}

// Заполняем данные из DataTable

for (int row = 0; row < dataTable.Rows.Count; row++)

{

IRow dataRow = sheet.CreateRow(row + 2);

for (int col = 0; col < dataTable.Columns.Count; col++)

{

if (dataTable.Columns[col].DataType == typeof(DateTime))

{

dataRow.CreateCell(col).SetCellValue(((DateTime)dataTable.Rows[row][col]).ToString("yyyy.MM.dd"));

}

else

{

dataRow.CreateCell(col).SetCellValue(dataTable.Rows[row][col].ToString());

}

}

}

// Автоподбор ширины столбцов

for (int i = 0; i < dataTable.Columns.Count; i++)

{

sheet.AutoSizeColumn(i);

}

// Сохраняем книгу Excel в файл

using (FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write))

{

workbook.Write(fileStream);

}

}

// Метод для создания стиля заголовка

private static ICellStyle GetHeaderCellStyle(IWorkbook workbook)

{

ICellStyle cellStyle = workbook.CreateCellStyle();

IFont font = workbook.CreateFont();

font.Boldweight = (short)FontBoldWeight.Bold;

font.FontHeightInPoints = 14;

cellStyle.SetFont(font);

return cellStyle;

}

private DataTable GetSalesReport(DateTime startDate, DateTime endDate)

{

DataTable dataTable = new DataTable();

Console.WriteLine(startDate);

Console.WriteLine(endDate);

string query = @"

SELECT

p.Дата AS Дата\_Продажи,

COALESCE(t.Наименование, u.Наименование\_услуги) AS Товар\_или\_Услуга,

s.ФИО AS Продавец,

p.Сумма\_продажи AS Сумма\_Продажи

FROM

Продажи p

LEFT JOIN

Товары t ON p.[Товар/услуга] = t.ID\_Товара

LEFT JOIN

Услуги u ON p.[Товар/услуга] = u.ID\_Услуги

JOIN

Сотрудники s ON p.ID\_Сотрудника = s.ID\_сотрудника

WHERE

p.Дата BETWEEN @StartDate AND @EndDate

ORDER BY

p.Дата;";

try

{

using (SqlConnection connection = new SqlConnection(db.GetConnectionString()))

{

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@StartDate", startDate);

command.Parameters.AddWithValue("@EndDate", endDate);

SqlDataAdapter adapter = new SqlDataAdapter(command);

adapter.Fill(dataTable);

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при получении отчета: " + ex.Message);

}

return dataTable;

}

private void button1\_Click(object sender, EventArgs e)

{

ОтчетПомощь otchot = new ОтчетПомощь();

otchot.ShowDialog();

if (otchot.DialogResult == DialogResult.OK)

{

button2.Enabled = true;

date1 = otchot.date1;

date2 = otchot.date2;

dataTable = GetSalesReport(date1, date2);

dataGridView1.DataSource = dataTable;

}

if (dataGridView1.Rows.Count == 0)

{

dataGridView1.Columns.Clear();

MessageBox.Show("Нет продаж за выбранный период!");

button2.Enabled = false;

}

}

private void saveFileDialog1\_FileOk(object sender, CancelEventArgs e)

{

string fileName = saveFileDialog1.FileName;

switch (Path.GetExtension(fileName).ToLower())

{

case ".xlsx": // Если выбран Excel-формат

ExportDataTableToExcel(dataTable, fileName, $"{date1.ToString("yyyy.MM.dd")} - {date2.ToString("yyyy.MM.dd")}");

break;

case ".docx": // Если выбран Word-формат

ExportDataTableToWord(dataTable, fileName, $"{date1.ToString("yyyy.MM.dd")} - {date2.ToString("yyyy.MM.dd")}");

break;

case ".pdf": // Если выбран PDF-формат

ExportDataTableToPdf(dataTable, fileName, $"{date1.ToString("yyyy.MM.dd")} - {date2.ToString("yyyy.MM.dd")}");

break;

default:

MessageBox.Show("Выбран неподдерживаемый формат файла.");

e.Cancel = true;

break;

}

}

private void button2\_Click(object sender, EventArgs e)

{

saveFileDialog1.ShowDialog();

}

}

}

**Модуль ОтчетПомощь.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсовая\_Зайцева

{

public partial class ОтчетПомощь : Form

{

public DateTime date1;

public DateTime date2;

public ОтчетПомощь()

{

this.DialogResult = DialogResult.No;

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (DateTime.TryParse(maskedTextBox1.Text, out date1) &&

DateTime.TryParse(maskedTextBox2.Text, out date2))

{

if (date2 >= date1)

{

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Ошибка! Дата начала периода больше даты конца периода!");

}

}

else

{

MessageBox.Show("Ошибка! Данные введены неверно, проверте правильность ввода данных!");

}

}

}

}